www.amegroups.com

1A039

BIG DATA CLINICAL STUDY AND ITS IMPLEMENTATION WITH R

Editors: Zhongheng Zhang
Fionn Murtagh
Sven Van Poucke

Big Data Clinical Study and Its Implementation with R

Honorary Editors: Michael W. Kattan, Cheng Zheng

Editors: Zhongheng Zhang, Fionn Murtagh, Sven Van Poucke

AME Publishing Company

BIG DATA CLINICAL STUDY AND ITS IMPLEMENTATION WITH R

Honorary Editors: Michael W. Kattan, Cheng Zheng

Editors: Zhongheng Zhang, Fionn Murtagh, Sven Van Poucke

AME Publishing Company

AME Publishing Company

Room C 16F, Kings Wing Plaza 1, NO. 3 on Kwan Street, Shatin, NT, Hong Kong

Information on this title: www.amegroups.com
For more information, contact books@amegroups.com

First published in 2018
Printed in China by AME Publishing Company

Editors: Zhongheng Zhang, Fionn Murtagh, Sven Van Poucke
Cover Image Illustrator: KangFu, Shanghai, China

BIG DATA CLINICAL STUDY AND ITS IMPLEMENTATION WITH R

(Hard Cover)
ISBN: 978-988-77840-8-1
AME Publishing Company, Hong Kong

Big Data Clinical Study and Its Implementation with R (FIRST EDITION)

HONORARY EDITORS

Michael W. Kattan
Department of Quantitative Health Sciences, Cleveland Clinic Foundation, Cleveland, Ohio, USA

Cheng Zheng
Joseph. J. Zilber School of Public Health, University of Wisconsin-Milwaukee, Milwaukee, Wisconsin, USA

EDITORS

Zhongheng Zhang
Department of Emergency Medicine, Sir Run-Run Shaw Hospital, Zhejiang University School of Medicine, Hangzhou 310016, China

Fionn Murtagh
Professor of Data Science, University of Huddersfield, Huddersfield, UK

Sven Van Poucke
Department of Anesthesia, Critical Care, Emergency Medicine and Pain Therapy, Ziekenhuis Oost-Limburg, Genk 3600, Belgium

AUTHORS

Michael W. Kattan
Department of Quantitative Health Sciences, Cleveland Clinic Foundation, Cleveland, Ohio, USA

Chanmin Kim
Department of Biostatistics, Harvard T.H. Chan School of Public Health, Boston, MA 02115, USA

Peng Lan
Department of Critical Care Medicine, Sir Run-Run Shaw Hospital, Zhejiang University School of Medicine, Hangzhou 310016, China

Su Lin
Liver Research Center, First Affiliated Hospital of Fujian Medical University, Fuzhou 350005, China

Fionn Murtagh
University of Huddersfield, Huddersfield, UK

Sven Van Poucke
Department of Anesthesiology, Emergency Medicine, Critical Care Medicine and Pain Therapy. Ziekenhuis Oost-Limburg, Genk, Belgium

Zhongheng Zhang
Department of Emergency Medicine, Sir Run-Run Shaw Hospital, Zhejiang University School of Medicine, Hangzhou 310016, China

Cheng Zheng
Joseph. J. Zilber School of Public Health, University of Wisconsin-Milwaukee, Milwaukee, Wisconsin, USA

COVER IMAGE ILLUSTRATOR

KangFu
Shanghai, China

EXECUTIVE TYPESETTING EDITOR

Xiaoting Xu
AME Publishing Company

We are pleased to announce that the "AME Research Time Medical Book Series" launched by AME Publishing Company have been published as scheduled.

Finishing my medical degree after 4 years and 3 months of study, I decided to quit going on to become a doctor only after 3 months of training. After that, I had been muddling through days and nights until I started engaging in medical academic publishing. Even 10 years after graduation, I had not totally lost the affection for being a doctor. Occasionally, that subconscious feeling would inadvertently arise from the bottom of my heart.

In April 2011, Mr. Tiantian Li, the founder of DXY.cn, and I had a business trip to Philadelphia, where we visited the Mütter Museum. As part of The College of Physicians of Philadelphia, the museum was founded in 1858 and has now become an exhibition hall of various diseases, injuries, deformities, as well as ancient medical instruments and the development of biology. It displays more than 20,000 pieces of items including pictures of wounded bodies at sites of battle, remains of conjoined twins, skeletons of dwarfs, and colons with pathological changes. They even exhibited several exclusive collections such as a soap-like female body and the skull of a two-headed child. This museum is widely known as "BIRTHPLACE OF AMERICAN MEDICINE". Entering an auditorium, we were introduced by the narrator that the inauguration ceremony of the Perelman School of Medicine at the University of Pennsylvania would take place there every year. I asked Mr. Li, "If it was at this auditorium that you had the inauguration ceremony, would you give up being a doctor?" "No," he answered.

In May 2013, we attended a meeting of *British Medical Journal* (*BMJ*) and afterwards a gala dinner was held to present awards to a number of outstanding medical teams. The event was hosted annually by the Editor-in-Chief of *BMJ* and a famous BBC host. Surprisingly, during the award presentation, the speeches made by *BMJ* never mentioned any high impact papers the teams had published in whichever prestigious journals over the past years. Instead, they laid emphasis on the contributions they had made on improving medical services in certain fields, alleviating the suffering of patients, and reducing the medical expenses.

Many friends of mine wondered what AME means.

AME is an acronym of "Academic Made Easy, Excellent and Enthusiastic". On September 3, 2014, I posted three pictures to social media feeds and asked my friends to select their favourite version of the AME promotional leaflet. Unexpectedly we obtained a perfect translation of "AME" from Dr. Yaxing Shen, Department of Thoracic Surgery, Zhongshan Hospital, Shanghai, who wrote: enjoy a grander sight by devoting to academia (in Chinese, it was adapted from the verse of a famous Chinese poem).

AME is a young company with a pure dream. Whilst having a clear focus on research, we have been adhering to the core value "Patients come first". On April 24, 2014, we developed a public account on WeChat (a popular Chinese social media) and named it "Research Time". With a passion for clinical work, scientific research and the stories of science, "Research Time" disseminates cutting-edge breakthroughs in scientific research, provides moment-to-moment coverage of academic activities and shares rarely known behind-the-scene stories. With global vision, together we keep abreast of the advances in clinical research; together we meet and join our hands at the Research Time. We are committed to continue developing the AME platform to aid in the continual forward development and dissemination of medical science.

It is said that how one tastes wine indicates one's personality. We would say how one reads gives a better insight to it. The "AME Research Time Medical Books Series" brings together clinical work, scientific research and humanism. Like making a fine dinner, we hope to cook the most delicate cuisine with all the great tastes and aromas that everyone will enjoy.

Stephen Wang
Founder & CEO,
AME Publishing Company

With the increasing availability of big data, the need is urgent for more studies of best practices when dealing with these data. The field is evolving rapidly, making it impossible to specify a definitive course of action to take. However, as with all complex scientific topics, one must start somewhere. In this book, we cover a large variety of issues that arise when dealing with big data. These issues span the description of datasets, processing of data, building prediction models (both traditional regression-based and novel machine learning oriented), and statistical model evaluation. We maintain an applied perspective throughout the book, leveraging the free yet highly powerful statistical platform R. With rich examples, the reader can quickly learn how to apply our approaches to his or her own big datasets. We have carefully organized our book into discretely contained chapters that deal with common issues, so that the reader may skip to the topic that is of most interest. Our goal for this book is that it becomes an easily accessible desktop reference for applied researchers confronting big data, allowing the research to quickly address the task at hand with a path forward. We realize we cannot have answers to all problems related to big data, but hopefully our book will be helpful when dealing with the more common demands upon big data.

Michael W. Kattan, PhD
Department of Quantitative Health Sciences, Cleveland Clinic Foundation,
Cleveland, Ohio, USA

With the rapidly developing techniques for data collection, storage and computation, the era for big data has come. For clinical studies, in particular, there is an increasing need in handling big data issues, as various forms of high dimensional data from clinical trials are rapidly accumulated with massive sample size. These include the ever-expanding electronic medical record (EMR) system, metabolomics and proteomics data from lab tests, and imaging data from MRI. These data provide us great opportunities for improvements in clinical research. However, they also bring big challenges for data analysis, for example, the famous "curse of dimensionality".

There has been substantial researches done in the area of high dimensional statistics and machine learning to handle big data issues, but these research areas are still quite open and actively studied. Furthermore, big data in clinical studies has their own features, providing more challenges, for example, how to handle missing data in high dimensional settings, how to perform high dimensional survival data analysis, how to study structured data and how to extend causal inference analysis to the high dimensional settings. Theoretical efforts have been made in the last several decades in all these areas and there are many useful tools developed that can be potentially applied to clinical studies.

This book addresses the above challenges encountered in clinical studies and review a broad range of methods from modern statistics and machine learning for big clinical data analysis. This book also provides instructions for many useful R packages with detailed example codes so that reader could apply these new statistical methods to their own studies. The organization of the book is as follow.

There are six chapters in this book. Chapter 1 provides an overview of the big data clinical research, including the perspective, the general accessing workflow, a brief review of machine learning methods and data acquisition and management. Chapter 2 discusses about exploratory data analysis and data management. It focuses on the missing data problem that is frequently encountered in clinical studies by introducing a number of methods and their applications. First it discusses about missing data exploration and data reshaping and aggregating. Then it introduces several imputation methods including single imputation, multiple imputation, and multivariate imputation. Chapter 3 discusses methods for variable selection for both parametric and non-parametric models that are commonly used in clinical studies. It also discusses about methods for diagnostic and introduced a useful R package to draw Nomograms. Chapter 4 discusses about the analysis of survival data. In this chapter both the application of parametric and semi-parametric models are illustrated, as well as the competing risk model. Chapter 5 discusses several commonly used unsupervised and supervised machine learning methods including the k nearest neighbor, naïve Bayes classification, decision tree and neural network. Chapter 6 addresses a number of other important statistical areas that has applications in clinical studies, for example, the hierarchical cluster analysis and its visualization with R, causal mediation analysis, structural equation modeling, and case-crossover design.

With this book, we hope to provide reader a comprehensive introduction to big data clinical studies and easy to follow data analysis applications with R examples that will potentially encourage big data analysis in clinical studies. I would like to sincerely thank Dr. Zhongheng Zhang for the opportunity to write the preface for this book.

Cheng Zheng, PhD
Joseph. J. Zilber School of Public Health,
University of Wisconsin-Milwaukee, Milwaukee, Wisconsin, USA

Big Data is data gathered and sourced, that is ambient or contextual data. This is data from the surroundings of the patient, and also including activity and behavior. All here has very practical description using the R open source, and multi-platform (PC, Mac, or Linux) sofware system. Included in R is data processing, visualization and display, and analysis methods and techniques.

Perspectives on Big Data Clinical Research

The first chapter on "Big data and clinical research" counterposes, as research, interventional analysis, which is, in fact, experimental research, relative to observational studies. For the former, typically at issue are randomized controlled trials. Also for the former, selection is required but then there might be bias associated with what is done. An important point made is that patients' treatments are usually complicated by the patients' comorbidities. So it becomes so very important to have and to use ancillary and contextual observations also. This amounts to the practical setting for big data clinical trials. Electronic medical records can be very important. Such big data may very well include also, demographic attributes, microbiology information on the patient, and other data sources. So it is noted that such observational data, encompassing what amounts to big data clinical trials, can be very relevant to the "real-world", i.e. detailed and comprehensive information on the patient.

A repository, entitled Multiparameter Intelligent Monitoring in Intensive Care III, MIMIC-III, with its data on over forty thousand patients is described. Access to that is described.

An interesting statement is that the demographics of China will lead to very high quality big data sources in China. Figure 1 illustrates how a hospital bed is set up for big data recording.

Further description is provided for potential users of such data, who are not well trained in computer science issues. This includes database querying. In "Accessing critical care big data", there is detailed description of accessing the data, and this includes some aspects of the very large data set size, that is potentially obtainable.

Description is provided of an important release at the beginning of 2017 of the "National scientific data sharing platform for population and health (NSDSPPH)" in China, comprising observed and recorded data with 280 million observations or records. This is noted as an "historic leap in clinical research".

Next there is an introduction to the use of the R statistical software environment. This software is open source, available for all computer platforms and used most of all by the data analysis and statistical etc. communities. Furthermore, at issue here, with the title "Data management by using R", this has a lot to do with simulating clinical data. This is so very beneficial for engagement with all that is at issue here, and all the benefits that may follow for clinical practice.

Data Management

In the managing of data, missing values are studied in a very comprehensive way, that is also very practical. All is illustrated and exemplified using the R software environment. This serves as an excellent introduction to many important and practical uses of R. All that is under discussion, and being described, uses simulated data. Such simulated data is both revealing of the general nature of this data, and it also nicely permits the reader to easily reproduce all that is here. The various data processing aspects are covered, then there is a chapter on reformatting data through aggregation, and then a good deal of coverage of imputation for addressing the problem of missing data. In what is characterized as data management, the final chapter is a short introduction to statistical inference, oriented towards univariate and bivariate, common or standard, distributional assumptions.

Model Building Strategy

Preparing the data for logistic regression is the theme of the first chapter relating to Model Building Strategy. This is the statistical determination of effectiveness and impact when the output is a yes/no outcome, in this framework, quite often, alive versus dead. The framework for carrying out such analysis is described. The chapter that follows is in regard to variable selection, a very standard requirement for the analyses being carried out. Following that, there is a chapter on managing the data requirements for standard, statistical, linear regression. Then comes a chapter on further diagnostics that are part and

parcel of the statistical regression analysis, with the focus here on logistic regression. The chapter that follows is related to this: "Propensity score method is employed to solve the problem of imbalance in baseline characteristics between intervention and control groups." Inference of the causal effect is at issue.

The next chapter presents visualization of the data, through a nomogram, which displays relationships between variables.

Survival Analysis

The next four chapters are dealing with survival analysis. This is the statistical analyses of data that are related to, and that express, the time to an event. Such an event can be death, or disease impact. As throughout all chapters, the presentation uses R examples with a very clear and comprehensive description of implementation of the analytical procedures.

Then there is a chapter using regression, using the Weibull statistical distribution, that is much used for reliability analysis, and, here, what can be referred to as lifetime distributions, and lifetime data analysis.

A chapter has visualization, using what is named the Cox proportional hazards model.

The last chapter for (statistical) survival analysis addresses competing risks, such as data for gender, age, and virus invasion.

Machine Learning

While all analytics have been based on statistics, it can be relevant to also consider machine learning. Generally the manner for using machine learning will be to have a known set of outcomes, termed the training set, and then to make use of that for the test set and generalization set, to predict the outcomes.

The first of these chapters on machine learning uses the k-nearest neighbor supervised classification method, that is also termed discriminant analysis method. The second chapter on machine learning is classical statistics, using the naive Bayes discriminant analysis method. The third chapter in this general context uses a decision tree approach. There are two related chapters relating to artificial neural networks.

Other Analytical Methods

A few other analytical processes constitute the final four chapters. First, with a major focus on visualization, there is the use of hierarchical clustering. This constitutes data mining, and this also has the title of unsupervised classification or clustering.

The following chapters cover: causal mediation analysis, structural equation modeling, and case-crossover design.

Fionn Murtagh, PhD
Professor of Data Science, University of Huddersfield, Huddersfield, UK.
(*Email: fmurtagh@acm.org*)

The race is on! Which team or company is able to gain sufficient expertise to extract data generated during medical practice and turn it into useful information? Although multiple attempts have been published over the last years, several facts impeding innovation are worth mentioning.

As patient privacy is a key topic in our business, data availability in comparison with other scientific domains and industries, is relatively limited. The analytics enterprise is driven by dozens of scientists located around the globe. With similarities of a hacking community, the top level scientists recognize each others qualities and form temporary teams to solve specific problems. In an environment where patient data is used, progression is limited by data privacy regulations. This requires new attention as efforts to anonymize or de-identify medical data could provide more available medical data open for exploration with similarities of data lakes. A huge effort in this context has been accomplished by the team behind the MIMIC database (https://mimic.physionet.org/) which is available for research.

Companies focusing on precision medicine are faced with the limitations of their own structure in the sense that they often lack sufficient affinity with daily practice to capture the priorities in medical questioning. As such they often limit their interest to a narrow domain where the real power of analytics is to be found in the analysis of multivariable, high dimensional time series data. Additionally, providing descriptive analysis of the logistics or activity of a hospital is not the high end predictive and prescriptive analytics the medical community is waiting for. Also the medical community should focus on the science of ontology as a significant amount of data is continued to be entered entirely unstructured. Although several scientific papers from the New York State Center of Excellence in Bioinformatics and Life Sciences, University at Buffalo, Buffalo, NY, USA and others cover this topic, a sufficient level of data quality generated by the medical community is frequently lacking resulting in GIGO (garbage in-garbage out) bias.

It is fascinating to observe the way tools and platforms such as R, normally used in basic science and various industries, are adapted to the medical domain thanks to the incredible efforts of a relatively small group of dedicated data scientists being fully active as health care professional in a busy medical practice. Combining two jobs seems an essential requirement to fully understand the path between medical data and generating clinical useful information.

Sven Van Poucke, MD, PhD
Anesthesiologist, Emergency Physician,
Data Scientist Ziekenhuis Oost-Limburg, Genk, Belgium

Table of Contents

Perspectives on Big Data Clinical Research

Data Management

Model Building Strategy

Survival Analysis

Machine Learning

Others

Big data and clinical research: perspective from a clinician

Zhongheng Zhang

Submitted Oct 11, 2014. Accepted for publication Nov 13, 2014.
doi: 10.3978/j.issn.2072-1439.2014.12.12
View this article at: http://dx.doi.org/10.3978/j.issn.2072-1439.2014.12.12

Introduction

The 21st century is an era of information technology and people face information explosion with large amount of data. The term big data has been defined in Wikepedia as: big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. In all areas of disciplines, such big data may help to explore underlying mechanisms of various phenomenons, and further facilitate decision-making. For example, in the 2012 USA presidential election, data-driven decision-making played a huge role in helping Obama win (1). In biomedical field, big data also begin to show their important role. Medical decision-making is becoming more and more dependent on data analysis, rather than conventional experiences and intuitions (2).

The present paper will focus on how to do clinical research by using big data. Firstly, I will review some basic concepts of clinical research and some characteristics of big data. Next, I will take some examples to illustrate how to address clinical uncertainty via data mining. The aim of the article is to provide more insights into clinical research based on big data in the hope that more people will take initiatives to conduct investigations using clinical big data. I cannot discuss all detailed technical issues in this article, but I will refer them in references. Interested investigators may take these references as the tutorials to guide them through the challenging journey of data exploration.

Bewilderedness in the current clinical research

It is well known that clinical research can be generally categorized into experimental and observational studies. The former is an experimental study that certain interventions will be given to participants. The most commonly performed randomized controlled trial (RCT) belongs to this kind of study. RCT usually has strict inclusion and exclusion criteria for participants. The procedure of randomization is employed to balance potential confounding factors between treatment and control arms. RCTs and/or systematic reviews involving high quality RCTs are the gold standard for clinical guidelines. However, with more and more RCTs being conducted, its limitations have arisen. In the area of sepsis and critical care medicine, we have compared the results obtained from RCTs and observational studies, and found that interventional effects obtained from these two types of studies were quite different (3,4). Biological efficacy, a measurement of effect size under strict experimental condition, can be obtained by RCT. However, this biological efficacy may be attenuated or even not take place at real world setting. In this circumstance the biological efficacy cannot be translated into clinical effectiveness (5). Clinical effectiveness is the most relevant to clinicians. The next question goes to why biological efficacy cannot be translated to clinical effectiveness. A plausible explanation is that RCTs are usually conducted with strict experimental design. There is a long list of inclusion and exclusion criteria. The participants are highly selected to exclude potential confounders. The protocol of intervention is also strictly defined with respect to timing, dosing and sequence of procedures. However, in reality, the conditions defined in RCTs cannot be fulfilled. Patients are usually complicated by comorbidities such as renal dysfunction, hypertension, and congestive heart failure. The timing of drug administration may be delayed due to admissions at busy hours. As a result, it is probably that we treat our patients based on knowledge derived from a minority of highly selected patients. That is to say, the conclusion from RCTs may not be generalizable to the "real world" patients.

Although there is no definitive solution to the above-mentioned limitations of current clinical research, big data may provide some insights into future direction of clinical trials (6,7). Wang and colleagues have proposed that big-data clinical trial (BCT) may become a mainstay type of

Figure 1 Flow chart of exploring MIMIC-II database. SQL, structural query language; MIMIC-II, multiparameter intelligent monitoring in intensive care II.

Figure 2 Certificate of completion of the training cause "Protecting Human Research Participants". NIH, provided by National Institute of Health.

clinical research and complement RCT in an important way (8,9). Big data in clinical study refer to the information collected using electronic database. These data come from daily routine clinical practice without modification or screening with strict inclusion and exclusion criteria, therefore retaining its real-world features (10). The advantage of BCT is that the result directly reflects clinical effectiveness. Big data in medical or epidemiological research generally comprise electronic medical record system, administrative registry for chronic and infectious diseases and medical insurance system (11). As clinicians, our research may focus on EMR system comprising information on demographics, laboratory findings, microbiology data, medical order, procedures, surgery and clinical outcomes (12). However, big data are not panacea without limitations. BCT is a kind of observational study in nature and has inherent limitations of its kind. For example, the observed and unobserved baseline characteristics cannot be well balanced. The conclusion may not be generalizable to other institutions if data were collected from a single center. Such limitation can be addressed with advanced statistical method such as random effect model and bootstrap estimation of coefficients.

How to do clinical research with big data: an example from multiparameter intelligent monitoring in intensive care II (MIMIC-II)

This section will take MIMIC-II as an example to illustrate how to incorporate big data into clinical research (13). The flow chart of analysis is shown in *Figure 1*.

MIMIC-II is an open access database comprising clinical data of ICU patients from Beth Israel Deaconess Medical Center (http://physionet.org/mimic2/). The database is consistently updating with current version of 2.6 that contains >30,000 ICU patients from 2001 to 2008 (14). MIMIC-II consists of clinical data and high resolution waveforms.

Gaining access to MIMIC-II

The investigator should register a username on the website, and then apply for access to the database. An online training course named "Protecting Human Research Participants" should be completed and a certification number will be assigned to an individual investigator (*Figure 2*). With this certification number, one is qualified to apply for an access to the database. After a couple of days, the whole database is accessible and data analysis can be performed according to one's study protocol

Conceiving clinical research ideas

With such a huge amount of clinical data, the next question is how to conceive clinical research ideas. Firstly, I would like to enumerate several types of clinical research by using big data (*Table 1*). The first one involves the exploration of risk factors, for which high resolution data are usually required for confounding controls. Multivariable models, stratification and propensity score analysis are useful tools for such kind of analysis. The second involves the assessment of an effectiveness of an intervention. Similarly, this type of study requires high-resolution data to control for confounding factors. Bias associated with selective treatment may play an important role and should be considered in study design. The third involves prediction model building, which aims to fit a model for future prediction. The forth type is an assessment of the cost-effectiveness of an intervention

Another key issue is how to conceive research ideas that

Table 1 Types of clinical studies by using big data

Types of studies	Examples (research question)	Requirement for clinical data*	Note
Risk factor evaluation (independency)	Is urine output on ICU entry associated with mortality outcome?	High resolution (other risk factors should be provided)	Multivariable model, stratified analysis and propensity score analysis can be employed
Effectiveness of intervention	Will PiCCO monitoring improve outcome of patients with septic shock?	High resolution (including a large number of confounding factors)	Intervention may be given for patients with different conditions. These conditions should be controlled to avoid "selective treatment"
Prediction model	Prediction model for ICU delirium	Moderate resolution (general description of risk factors)	The predictive value of whole model is stressed, rather than a single risk factor
Epidemiological study	The incidence and prevalence of catheter-related blood stream infection in ICU	Low resolution	A simple description is enough and no risk factor adjustment is required
Implementation and efficacy of healthcare policy	Is the policy of screening and controlling hypertension effective in lowering cardiovascular event rate?	Low resolution	No complex clinical data are required

*, resolution of clinical data refers to the intensity of data recording. For example, the study on urine output requires it be recorded on hourly basis. The resolution is higher for hourly urine output than daily urine output. For risk factor analysis, every covariate should be completely recorded. Otherwise, the resolution is not enough if some covariates are missing in the dataset.

can be addressed with big data. There are two approaches: one is to perform data mining according to your research question, and the other is to adapt your research question to the database. Sometimes these two approaches may be used simultaneously. In a study investigating the association of lactate and clinical outcome (15), we planed to explore lactate measured on ICU entry at the outset. However, after data mining we found that lactate was usually measured repeatedly and decision was made to explore the trend of lactate (lactate clearance). The study protocol was adapted accordingly.

To conceive research idea based on your data is another way. One can perform statistical description for a dataset, by using traditional central tendency (mean, median) and discrete tendency (full range, 95% confidence interval). Graphical demonstration may be particularly helpful. For example, contour plot may help to explore associations among three variables; histogram can be used to explore the distribution of a variable. However, someone may contend that a peek at dataset before drafting a study protocol may introduce bias (e.g., the problem of multiple testing and selective reporting is of this kind). That is to say, twenty times statistical testing for associations will result in one with $P<0.05$ among independently generated random variables. I acknowledge this limitation, but such study can still be used as hypothesis generating and provide the rationale for further hypothesis-driven studies.

Another type of study is to investigate simple and easily obtainable parameters. By using such parameters, your clinical questions can be addressed by using various databases. Our study group has previously performed association study involving urine output and mortality (16). Because urine output is an essential but easily obtainable variable, it should be recorded in all kinds of ICUs. There is no reason to omit this recording, just like all other vital signs. We were confident at the outset that urine output must be carefully recorded in MIMIC-II, and the study was expected to be conducted smoothly. Such simple variables included temperature, electrolytes and heart rate. In another study we investigated the association of ionized calcium and mortality (17). However, studies involving simple variables are usually criticized for the lack of novelty, which may be an important reason for a manuscript to be rejected.

Data extraction

There is a demo version of the MIMIC-II database that can be accessed via the query builder (https://mimic2app.csail.mit.edu/querybuilder/). A limited number of rows can

Figure 3 Dual flow chart of data management and statistical analysis by using STATA.

be exported from this version, and thus it is primarily used for testing structural query language (SQL). I found that it was useful for the preliminary exploration of the data. I had attempted to do research on brain natriuretic peptide (BNP) when I first encountered the MIMIC-II database (18,19), the first idea coming to my mind was to use these big data to establish a linkage between BNP and clinical outcome. However, when I started to explore the database I found that measurements on BNP was scarce. In the end I realized that the payment for medical insurance in USA is closely monitored and not every one had the indication to get BNP measured (e.g., BNP is only covered by insurance when it is used to determine the cause of respiratory distress in the emergency room). Query builder can be accessed via windows operating system which is convenient for most Chinese users, whereas the whole database can only be extracted via virtual machine on Ubuntu operating system.

Accessing to the complete MIMIC-II database requires users to have some basic knowledges on virtual machine and Ubuntu operating system. The downloaded package compressed file with suffix ".tar", taking disc space of 30 G. The files can be directly imported into the virtual box (Oracle). After entering username and password, one gains access to the Ubuntu operating system. The username and password are mimic2 and 2CIMIM_2v6, respectively. PGadmin is the most popular development platform for PostgreSQL. It can be opened to extract data under Ubuntu operating system.

Some investigators may want to export data for further analysis under Windows system. Data transferring between different operating systems can be performed via email. The file extracted has a suffix of .csv, which can be imported into Stata statistical package. Other statistical softwares such as SAS and SPSS also support this format.

Data processing using Stata

The author is familiar with Stata statistical package in data processing, and the following section will introduce key steps in data processing with Stata.

Data processing using Stata consists of data management and statistical analysis. In my experience around 80% of time and energy are spent on data management. This step included several aspects: (I) to generate new variables, for example, one wants to transform the continuous variable age into a binary variable (e.g., old *vs.* young); (II) variable checking, the sum module can check for some senseless values (e.g., age =200); (III) to transform string variables into numerical variables, or vise versa; (IV) combination of datasets. Different types of variables are stored in different relational tables. They should be combined into one dataset for the purpose of statistical analysis. Stata provide useful modules such as merge and append for this purpose. Statistical analysis is based on correctly performed data management.

One advantage of Stata is its ability to record the complete process of data analysis. Data analysis can be performed in three ways: windows pull-down menu, command input in command window and do-file. I suggest that data analysis be performed by using do-file, which is able to record the whole process of how data are managed and analyzed. This will facilitate replication of analysis and made revisions. Furthermore, cooperation among researchers also requires do-file. The other two methods (e.g., windows pull-down menu and command input in command window) are mainly used to test a Stata syntax or to facilitate draw graphs with complex options.

A complete dual flow chart of data analysis using stata is shown in *Figure 3*. I suggest split data analysis into two parts: data management and statistical analysis. The left column shows data management that will make changes to the dataset in memory. Dataset generated by using do-file will be stored in memory with the suffix ".dta". The right column simply performs statistical analysis and will not change the dataset.

Using structural query language (SQL) to extract data

An important step in preparing data for analysis is data extraction by using SQL. Efficient use of SQL will save a lot of time and disc space. Some tasks previously mentioned during data management can be performed at the step of data extraction. For example, one intends to restrict data analysis to adult population. This can be performed by using Stata command "if", or using the SQL "where" clause. I prefer to use Stata for data management because it documents the process of data management. However, it is at the discretion of the investigator.

A simple SQL syntax can be written as: Select *variable name* from *table name* where *conditions*, where variable name refers to variables contained in a relational table. The asterisk "*" can replace the variable name if all variables are expected to be extracted. The table name refers to the name of the relational table. The conditions are a set of expressions used to select observations fulfilling certain criteria. For example, the expression "age >65" can be used if your target population is old people. For complex SQL syntax, readers may consult textbooks on SQL such as the language of SQL: How to Access Data in Relational Databases edited by Larry Rockoff and SQL Cookbook by Anthony Molinaro.

Conclusions

The exploration of big data is a process of trial and error. Someone may feel that it is a task of distress, but I feel it is a hard way filled with success and surprise. The secret of human disease may lurk under the vast ocean of big data, waiting us to decode and understand them. The article is not intended to serve as a step-by-step guidance on using big data but to inspire people who are interested in doing exploration on it. In China, it is possible to establish our own high quality big data because we have the largest population in the world.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Scherer M. Inside the Secret World of the Data Crunchers Who Helped Obama Win. Available online: http://swampland.time.com/2012/11/07/inside-the-secret-world-of-quants-and-data-crunchers-who-helped-obama-win/
2. Margolis R, Derr L, Dunn M, et al. The National Institutes of Health's Big Data to Knowledge (BD2K) initiative: capitalizing on biomedical big data. J Am Med Inform Assoc 2014;21:957-958.
3. Zhang Z, Ni H, Xu X. Do the observational studies using propensity score analysis agree with randomized controlled trials in the area of sepsis? J Crit Care 2014;29:886.e9-15.
4. Zhang Z, Ni H, Xu X. Observational studies using propensity score analysis underestimated the effect sizes in critical care medicine. J Clin Epidemiol 2014;67:932-939.
5. Nallamothu BK, Hayward RA, Bates ER. Beyond the randomized clinical trial: the role of effectiveness studies in evaluating cardiovascular therapies. Circulation 2008;118:1294-1303.
6. Schneeweiss S. Learning from big health care data. N Engl J Med 2014;370:2161-2163.
7. Psaty BM, Breckenridge AM. Mini-Sentinel and regulatory science--big data rendered fit and functional. N Engl J Med 2014;370:2165-2167.
8. Wang SD. Opportunities and challenges of clinical research in the big-data era: from RCT to BCT. J Thorac Dis 2013;5:721-723.
9. Wang SD, Shen Y. Redefining big-data clinical trial (BCT). Ann Transl Med 2014;2:96.
10. Albert RK. "Lies, damned lies .." and observational studies in comparative effectiveness research. Am J Respir Crit Care Med 2013;187:1173-1177.
11. Cooke CR, Iwashyna TJ. Using existing data to address important clinical questions in critical care. Crit Care Med 2013;41:886-896.
12. Peters SG, Buntrock JD. Big data and the electronic health record. J Ambul Care Manage 2014;37:206-210.
13. Saeed M, Villarroel M, Reisner AT, et al. Multiparameter Intelligent Monitoring in Intensive Care II: a public-access intensive care unit database. Crit Care Med 2011;39:952-960.
14. Scott DJ, Lee J, Silva I, et al. Accessing the public MIMIC-II intensive care relational database for clinical research. BMC Med Inform Decis Mak 2013;13:9.
15. Zhang Z, Chen K, Ni H, et al. Predictive value of lactate in unselected critically ill patients: an analysis using fractional polynomials. J Thorac Dis 2014;6:995-1003.

16. Zhang Z, Xu X, Ni H, et al. Urine output on ICU entry is associated with hospital mortality in unselected critically ill patients. J Nephrol 2014;27:65-71.
17. Zhang Z, Xu X, Ni H, et al. Predictive value of ionized calcium in critically ill patients: an analysis of a large clinical database MIMIC II. PLoS One 2014;9:e95204.
18. Zhang Z, Zhang Z, Xue Y, et al. Prognostic value of B-type natriuretic peptide (BNP) and its potential role in guiding fluid therapy in critically ill septic patients. Scand J Trauma Resusc Emerg Med 2012;20:86.
19. Zhang Z, Ni H, Lu B, et al. Changes in brain natriuretic peptide are correlated with changes in global end-diastolic volume index. J Thorac Dis 2013;5:156-160.

Cite this article as: Zhang Z. Big data and clinical research: perspective from a clinician. J Thorac Dis 2014;6(12):1659-1664. doi: 10.3978/j.issn.2072-1439.2014.12.12

Big data and clinical research: focusing on the area of critical care medicine in mainland China

Zhongheng Zhang

Abstract: Big data have long been found its way into clinical practice since the advent of the era of information technology. Medical records and follow-up data can be efficiently stored and extracted with information technology. It continuously produces a large amount of data including laboratory findings, medications, fluid balance, progressing notes and imaging findings after admission of a patient. Clinicians and clinical investigators should make every effort to make full use of the big data that are being generated by the electronic medical record (EMR) system and other healthcare databases. At this stage, more training courses on data management and statistical analysis are required before clinicians and clinical investigators can handle big data and translate them into advances in medical science. China is a large country with a population of 1.3 billion and can contribute greatly to clinical advances by the generation and utilization of reliable and high-quality big data.

Keywords: Big data; critical care medicine; mainland China

Submitted Aug 26, 2014. Accepted for publication Aug 27, 2014.
doi: 10.3978/j.issn.2223-4292.2014.09.03
View this article at: http://dx.doi.org/10.3978/j.issn.2223-4292.2014.09.03

The 21st century is an era of big data involving all aspects of human life, including economics, biology, and medicine. In Wikipedia, the term big data is defined as any collection of data sets so large and complex that it becomes difficult to process by using on-hand data management tools or traditional data processing applications. Big data can be seen in all aspects of our daily life. For instance, most of us hold a VIP card when we go shopping in a supermarket. The card records information on our individual characteristics such as age, gender, career and education. Every time when we use VIP card the supermarket can obtain information on the category of goods that we have bought, as well as the time when these goods are most likely to be sold. If we go shopping twice a week, there will be more than 100,000 observations in a year for a supermarket with daily volume of 1,000 people. Such a large sample size provides sufficient statistical power for a variety of complex model fittings (e.g., fractional polynomials, spline function with more than five knots, and complex interactions) (1). In the field of marketing research, it is of particular interest to explore the association between particular characteristics of customers and their preference of buying. If a certain brand of perfume is more likely to be sold between 7 and 8 o'clock at night, promotion activities of it can be performed at this time period.

Big data have long been found its way into clinical practice since the advent of the era of information technology. Medical records and follow-up data can be more efficiently stored and extracted with information technology. A patient, immediately after admission, produces a large amount of data including laboratory findings, medications, fluid balance, progressing notes and imaging findings. This is particularly true for critically ill patients requiring intensive care unit (ICU) admission, because vital signs and urine outputs are recorded on hourly basis for them. In our institution, every bed is equipped with a monitor and a computer so that data on vital signs (e.g., respiratory rate, heart rate, blood pressure and body temperature) and other physiological signals (e.g., extravascular lung water when transpulmonary thermodilution measurement is performed) can be automatically extracted from the monitor and stored in the electronic medical system in a predefined time interval (*Figure 1*). Furthermore, other clinical observations such as pupil size, Glasgow coma scale (GCS), fluid input and output can be input into the computer on hourly basis. Such "high resolution" data allow extensive studies to explore the effectiveness of certain interventions and independent predictors of clinical outcomes.

Figure 1 In Jinhua Municipal Central Hospital, every bed is equipped with a monitor (upper left) and a computer (upper right) so that data on vital signs (e.g., respiratory rate, heart rate, blood pressure and body temperature) and other physiological signals (e.g., extravascular lung water when transpulmonary thermodilution measurement is performed) can be automatically extracted from the monitor and stored in the electronic medical record (EMR) system in a predefined time interval. The lower screen displays parameters of ventilator settings, and these parameters can also be extracted and stored in the EMR.

Big data have not been fully utilized yet for clinical research in mainland China, despite the wide use of electronic medical record (EMR) system in the majority of tertiary hospitals. We have previously worked with the department of information technology of our hospital to extract data from electronic record system. Using these data, we carried out several interesting clinical researches with these data (2-4). Researches based on EMR system are feasible and efficient because all data are stored electronically and it saves a lot of time as compared to manual data extraction. However, our study was based on a single center and the sample size was relatively small (less than 2,000 in our previous studies), which significantly compromised the generalizability of our conclusions. In the future, electronic clinical data may be incorporated from dozens or hundreds of hospitals to form the big data, and studies by using these data will be more generalizable.

However, clinical study based on big data has been widely used in other Western countries (5-8). I personally have been working with some famous databases for a while and herein I would like to share some of my experience on using these databases. The most distinguished database in my experience is the Multiparameter Intelligent Monitoring in Intensive Care database (MIMIC-2), and now it has been updated to version 2.6 (9). This database comprises more than 30,000 ICU patients enrolled between 2001 and 2008 in Beth Israel Deaconess Medical Center (Boston, MA, USA). The information technology required to construct and maintain the database is provided by Massachusetts Institute of Technology (MIT). It is publically available and requires completion of an online training course before one can obtain the full access to the database (10). Access of our team to the database was approved after completion of the NIH web-based training course named "Protecting Human Research Participants" (certification number: 1132877). At first inspection of the dataset, one may be overwhelmed by the large amount of data as it occupies nearly 80 G of the computer disc after decompression. Everything happened during hospital stay is recorded in the database, including demographics, diagnosis based on ICD-9, medications, report of imaging study, vital signs recorded on hourly basis and laboratory findings. Scores for severity of illness such as sequential organ failure assessment (SOFA) and simplified acute physiology score (SAPS) are calculated after secondary analysis of the original data.

The exploration of the MIMIC-2 database requires some degree of experience of computer science. One needs a virtual machine to mount the data and extract it from Linux

operation system. Data are stored in relational tables that are connected to each other by using the primary key (e.g., icustay_id or hadm_id). Therefore, structure query language (SQL) is mandatory for data extraction which takes the form of "select * from *tablename* where a *condition*". A good online material to learn SQL could be found at http://www.w3schools.com/sql/sql_like.asp.

The first work I have done with the MIMIC-2 was to explore the association of urine output and mortality in critically ill patients (11). The idea comes from the fact that while another indicator of renal function serum creatinine has been extensively studied for its association with mortality risk, the urine output has not received so many attentions despite the widely accepted notion that urine output should be tightly associated with mortality risk based on clinical experience. The MIMIC-2 database provided a good material to quantitatively establish the association between urine output and mortality risk. After successful completion of the first work, I continued to do several other investigations involving the clinical implications of lactate and ionized calcium (12,13). Up to now, these projects have been completed.

The following section will discuss about the difference between clinical researches based on big data and randomized controlled trials (RCT). There is no doubt that RCT is at the top of the evidence pyramid and can provide high quality evidence on the efficacy of a certain intervention. However, RCT is not a panacea that solves all clinical problems (14). As I have previously mentioned, RCT has at least the following limitations which may be potentially addressed by using big data derived from "real world" setting (15,16). Although some investigators argue that a high quality RCT should be as close to real word as possible, RCT may differ from observational studies in that they provide evidence in biological efficacy of an intervention which may not be translated to clinical effectiveness in "real world" setting (17). Firstly, RCT is performed with strict inclusion/exclusion criteria that maybe only 20% of population of interest is included, and the generalizability of the conclusion to the rest 80% patients is questionable. With aging population, more and more patients encountered in daily clinical practice will have multiple comorbidities, which would have been excluded from RCTs. Secondly, interventions performed in RCTs may be complex and not applicable to other institutions or routine practice. In contrast, the big data using EMR system provide patients' information from real world setting and researches based on such design are more applicable to patients encountered in daily practice. Thirdly, some interventions cannot be explored by using RCT due to ethical constraints. Likewise, when an intervention becomes widespread, clinicians are unwilling to "experiment" with alternatives. For instance, the impact of timing of cardiopulmonary resuscitation (CPR) on cerebral function recovery cannot be investigated with controlled trials. However, such studies can be done by using techniques such as propensity score analysis and stratification based on big data. Fourthly, RCT is generally time consuming and more costly than observational studies based on big data (18). For a novel intervention or diagnostic test whose beneficial effect is largely unknown, preliminary data based on electronic record system or other healthcare data for administration purpose may be needed before large amount of funds are granted for a well-designed RCT.

In conclusion, clinicians and clinical investigators should make every effort to make full use of the big data that are being continuously generated by EMR system and other healthcare databases. It is totally a waste of resources by leaving electronic data on the disc without exploiting its potential values on revealing mechanisms underlying disease progression and resolution. Unfortunately, this is a common situation in mainland China. At this stage, more training courses on data management and statistical analysis are required before clinicians and clinical investigators can handle big data and translate them into advances in medical science. I personally feel that it is of much more interest to clinicians to perform clinical research based on big data than to culture cells and feed mice. Of note, China is a large country with a population of 1.3 billion and can contribute greatly to clinical advances by the generation and utilization of reliable and high-quality big data, but now this goal is far from being achieved.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Harrell FE Jr, Lee KL, Pollock BG. Regression models in clinical studies: determining relationships

between predictors and response. J Natl Cancer Inst 1988;80:1198-1202.
2. Zhang Z, Xu X, Fan H, Li D, Deng H. Higher serum chloride concentrations are associated with acute kidney injury in unselected critically ill patients. BMC Nephrol 2013;14:235.
3. Zhang Z, Xu X, Ni H, Deng H. Red cell distribution width is associated with hospital mortality in unselected critically ill patients. J Thorac Dis 2013;5:730-736.
4. Zhang Z, Xu X, Ni H, Deng H. Platelet indices are novel predictors of hospital mortality in intensive care unit patients. J Crit Care 2014;29:885.e1-6.
5. Cooke CR, Iwashyna TJ. Using existing data to address important clinical questions in critical care. Crit Care Med 2013;41:886-896.
6. Margolis R, Derr L, Dunn M, Huerta M, Larkin J, Sheehan J, Guyer M, Green ED. The National Institutes of Health's Big Data to Knowledge (BD2K) initiative: capitalizing on biomedical big data. J Am Med Inform Assoc 2014;21:957-958.
7. Psaty BM, Breckenridge AM. Mini-Sentinel and regulatory science--big data rendered fit and functional. N Engl J Med 2014;370:2165-2167.
8. Schneeweiss S. Learning from big health care data. N Engl J Med 2014;370:2161-2163.
9. Saeed M, Villarroel M, Reisner AT, Clifford G, Lehman LW, Moody G, Heldt T, Kyaw TH, Moody B, Mark RG. Multiparameter Intelligent Monitoring in Intensive Care II: a public-access intensive care unit database. Crit Care Med 2011;39:952-960.
10. Scott DJ, Lee J, Silva I, Park S, Moody GB, Celi LA, Mark RG. Accessing the public MIMIC-II intensive care relational database for clinical research. BMC Med Inform Decis Mak 2013;13:9.
11. Zhang Z, Xu X, Ni H, Deng H. Urine output on ICU entry is associated with hospital mortality in unselected critically ill patients. J Nephrol 2014;27:65-71.
12. Zhang Z, Xu X, Ni H, Deng H. Predictive value of ionized calcium in critically ill patients: an analysis of a large clinical database MIMIC II. PLoS One 2014;9:e95204.
13. Zhang Z, Chen K, Ni H, Fan H. Predictive value of lactate in unselected critically ill patients: an analysis using fractional polynomials. J Thorac Dis 2014;6:995-1003.
14. Wang SD. Opportunities and challenges of clinical research in the big-data era: from RCT to BCT. J Thorac Dis 2013;5:721-723.
15. Zhang Z, Ni H, Xu X. Do the observational studies using propensity score analysis agree with randomized controlled trials in the area of sepsis? J Crit Care 2014;29:886.e9-886.e15.
16. Zhang Z, Ni H, Xu X. Observational studies using propensity score analysis underestimated the effect sizes in critical care medicine. J Clin Epidemiol 2014;67:932-939.
17. Nallamothu BK, Hayward RA, Bates ER. Beyond the randomized clinical trial: the role of effectiveness studies in evaluating cardiovascular therapies. Circulation 2008;118:1294-1303.
18. Sibbald B, Roland M. Understanding controlled trials. Why are randomised controlled trials important? BMJ 1998;316:201.

Cite this article as: Zhang Z. Big data and clinical research: focusing on the area of critical care medicine in mainland China. Quant Imaging Med Surg 2014;4(5):426-429. doi: 10.3978/j.issn.2223-4292.2014.09.03

Accessing critical care big data: a step by step approach

Zhongheng Zhang

Submitted Jan 10, 2015. Accepted for publication Feb 08, 2015.
doi: 10.3978/j.issn.2072-1439.2015.02.14
View this article at: http://dx.doi.org/10.3978/j.issn.2072-1439.2015.02.14

Introduction

Since the publication of an introductory article in the *Journal of Thoracic Disease* and the Chinese version in *Journal of Clinical and Pathological Research* (1,2), I received many letters inquiring about detailed steps in accessing the multiparameter intelligent monitoring in intensive care-2 (MIMIC-2) database. When I told them there were very good instructions on the website that can guide them accessing the whole database in a step-by-step approach, most of them were still confusing on some technical details. Probably, most of the readers are critical care clinicians who are interested in conducting researches by using this database but they are lack of some basic knowledge on computer science. Therefore, I write this step-by-step guidance on how to access the MIMIC-2 clinical database, assuming that the readers have no knowledge on virtual machine and database management. In the manuscript, I used many figures to illustrate the detailed steps in establishing virtual machine. I hope that interested investigators can access this database freely without cumbering by technical difficulties.

Getting access to MIMIC-2 clinical database

(I) Go to the website by clicking on "https://physionet.org/pnw/login". Create an account on this page, or login if you have already registered and had an account.

(II) Scroll down to select "MIMIC II Clinical Database". Then you will see two items. If you are authorized users you may enter directly. Otherwise you have to apply for an access. You will be directed to complete an on-line training program. After completion of the on-line training course, you will receive a certificate and the access to the clinical database. Many clinicians will have no difficulty in this step because there is detailed step-by-step instructions on the website.

(II) Authorized users can enter a web page where one can download the whole database to your own disc. On the top of the webpage, there is a general introduction to the database and two references are provided to cite for authors whose work is based on the database (*Figure 1*).

Obtaining database to your own computer

(I) There are several means to get the clinical database to your computer disc. I would like to use the virtual machine to download it. I have succeeded in obtaining the whole database in both OS X and Windows hosts. I use the software Oracle VM Virtualbox which can be downloaded from https://www.virtualbox.org/wiki/Downloads. Download the appropriate version of Virtualbox (select from OS X, Windows, Linux and Solaris hosts according to your computer operating system).

(II) There is a link to download the file "MIMIC2_VM_v1-disk1.vmdk". This is the MIMIC II Virtual Machine Encrypted Hard Disk file that is compressed to 4 GiB, and can be expanded to 80 GiB. It takes hours to days to download this file, depending on the transmission rate of your internet.

(III) Launch the VM Virtualbox manager and start a new machine. The name of the new machine can be anything you like and here I name it "MIMIC2". Choose the type to be Linux and leave the version to be Ubuntu (64 bit may be mandatory since when I used 32 bit in Windows system, the virtual machine cannot be opened) as default. Then click the "next" button.

(IV) Set the size of the memory and the default is 512 M. Then click the "next" button.

(V) Choose the disc file "MIMIC2_VM_v1-disk1.vmdk" from directory where you have saved it. Then click the "create" button. The virtual machine is then created. The instruction from the website states the use of MIMIC2_VM_v1.ovf file to start the virtual machine. However, I found the MIMIC2_VM_v1.ovf file was not

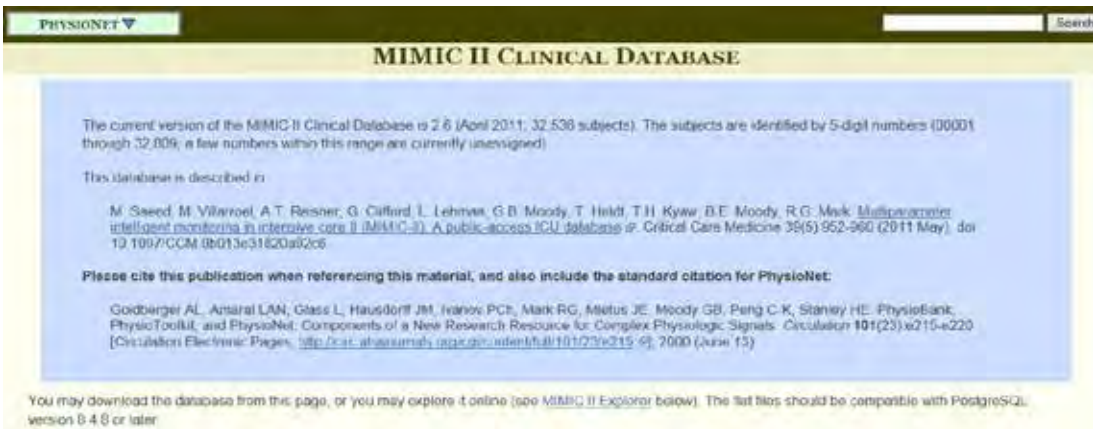


Figure 1 A general introduction to the multiparameter intelligent monitoring in intensive care-2 (MIMIC-2) clinical database.

downloadable and it could not be opened with Oracle VM Virtualbox. By clicking on the downloading link to MIMIC2_VM_v1.ovf, the internet explorer opened a new page with overwhelming computer codes. It doesn't matter. We choose to use the MIMIC2_VM_v1-disk1.vmdk file.

(VI) By returning to the starting interface of Oracle VM Virtualbox, you will see a new icon named "MIMIC2" on the leftmost column. It is the virtual machine that we have just created (*Figure 2*). Enter it by double clicking this icon.

(VII) When prompted for the encryption password, enter the password: 2CIMIM_2v6. The password is case-sensitive (*Figure 3*).

(VIII) Login and change the mimic2 user password (please note that the encryption password will not be changed). Use the following credentials for the mimic2 user login:

(i) Username: mimic2;

(ii) Password: 2CIMIM_2v6.

Downloading and unpacking the data

The whole process can be found in the website at https://physionet.org/works/MIMICIIClinicalDatabase/files/virtual_machines/MIMIC2_VM_README.txt. I repeated them here to keep the whole process complete. Furthermore, I illustrate each key step with figures to help better understanding for clinicians.

Login into the virtual machine with above username and password. You will see the desktop of the Ubuntu operating system. Open a terminal (click black screen icon on top left of the VM desktop, *Figure 4*). Type the following command (*Figure 5*):

./get_mimic_data.sh –c

Note: this command enables you to connect to PhysioNetWorks using your user name (your registered PhysioNetWorks email address) and password. The entire download process should take a few minutes on a 100 MBps network connection. After completion, the MIMIC II User's Guide and additional documentation will be automatically installed and you can find them on the desktop.

Installing the database

After the flat files have been downloaded and unpacked in the previous steps, you can then proceed to import them into the PostgreSQL database. The downloaded file was stored in the temporary folder. You'd better not to shut down VM between the last step and this one. Open a terminal and go to where the flat-file is saved by using cd:

cd /tmp/MIMIC2-Importer-2.6/

Run the importer by entering the following two commands:

sudo./prep.sh

./import.sh

Note: the process takes a long time due to the large file size. Depending on the memory (and disk, and CPU) available in your virtual machine, unpacking and loading each batch of ~1,000 subjects may require 20 minutes to several hours. This process should not be interrupted until it is finished. Take a cup of coffee during the process, but do not close or shut down the virtual machine until the process is completed.

After confirming the completion of installation, delete the /tmp/MIMIC2-Importer-2.6/ directory by typing:

Figure 2 Setup of a virtual machine. MIMIC-2, multiparameter intelligent monitoring in intensive care-2.

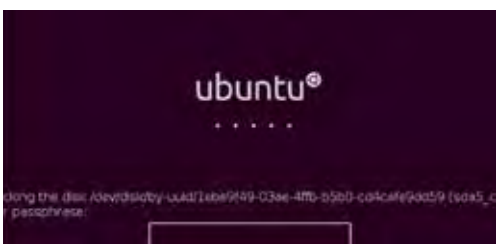


Figure 3 Launching the Ubuntu operating system with password.

```
cd
rm -rf /tmp/MIMIC2-Importer-2.6/
```

Logging into the database using pgAdmin

The PostgreSQL database in the VM doesn't use password authentication, it uses identity authentication. To open

Figure 4 The black screen icon on top left of the VM desktop.

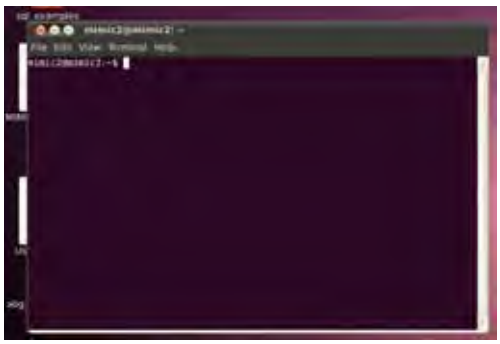


Figure 5 After opening the terminal, this window will pop up where we can type command.

Figure 6 Open the pgAdmin III as illustrated in this figure.

the pgAdmin III, use the following path (*Figure 6*): applications- > programming- > pgAdmin III. When you open pgAdmin III, click the plug icon on the left most of the toolbar and open the window for new server registration. The Host should be blank. The Username should be "mimic2". The "Maintenance DB" should be "MIMIC2" (*Figure 7*).

After registration, you can find the server in the object

Figure 7 New server registration. The Host should be blank. The Username should be "mimic2". The "Maintenance DB" should be "MIMIC2". MIMIC-2, multiparameter intelligent monitoring in intensive care-2.

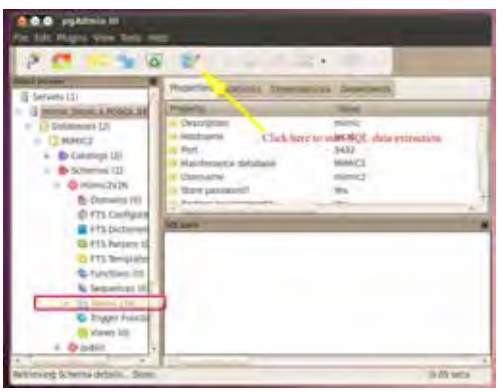


Figure 8 Structure of the MIMIC local server. Expand the schema and you will find there are 38 relational tables in the database, which exactly matches what we can see in the query builder explorer. MIMIC-2, multiparameter intelligent monitoring in intensive care-2.

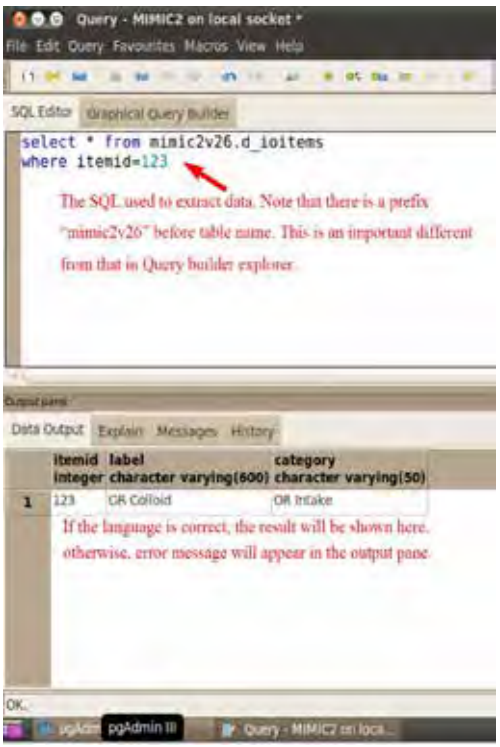


Figure 9 The query language is input at the upper panel and the lower panel displays the result of the query. Make sure that the SQL is exactly correct.

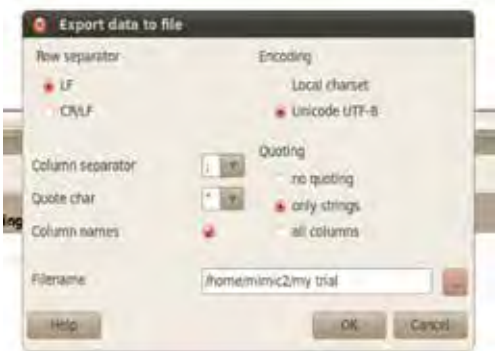


Figure 10 Export data to file. Specify the directory you want to save your data.

browser on the left. Expand the schema and you will find there are 38 relational tables in the database (*Figure 8*). That exactly matches what we can see in the query builder explorer. By clicking the SQL icon in the toolbar at the top of the window, the interface where you can extract data with SQL is opened. The query language is input at the upper panel and the lower panel displays the result of the query (*Figure 9*). When you are satisfied with what you have extracted, you can export the file to the virtual machine. Select a pathway to save your data (*Figure 10*). The exported file can be sent by email and then saved to your computer for further statistical analysis.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

Cite this article as: Zhang Z. Accessing critical care big data: a step by step approach. J Thorac Dis 2015;7(3):238-242. doi: 10.3978/j.issn.2072-1439.2015.02.14

References

1. Zhang Z. Big data and clinical research: perspective from a clinician. J Thorac Dis 2014;6:1659-1664.
2. Zhang Z. Big data and clinical research. J Clin Pathol Res 2014;34:492-497.

When doctors meet with AlphaGo: potential application of machine learning to clinical medicine

Zhongheng Zhang

Submitted Mar 10, 2016. Accepted for publication Mar 11, 2016.
doi: 10.21037/atm.2016.03.25
View this article at: http://dx.doi.org/10.21037/atm.2016.03.25

March is a reluctant beginning of spring. After suddenly getting warmer, the weather has turned cold again. Yesterday, the friend circle of Wechat was full of a breaking news that the Google's artificial intelligence AlphaGo beat human Go champion Lee Sedol. Or at least it was a temporary lead because there will be several rounds in following days. The news was a headline because it was related to the debate on whether artificial intelligence could defeat human brain. Many cultural celebrities posted their comments on the news. Shen Lei, a writer who is famous for his interpretation on Jin Yong's Kung Fu novels, stated that: "*The last warrior falls*." Tang Fenng also posted on his Wechat that: "*All calculations and computations can be left for machines from now on. What we humans can do are: burning incense, sipping tea, Washing inkstone, playing the Chinese zither*…." In other words, all works with rules can be completed by machines, and humans have to retreat to their emotional domain. Why is it the Go, instead of other chesses such as Modern Ludo and animal checker, or other intellectual activities? That is because the Go represent the most complicated intellectual activity. The chessboard of Go is made up of 361 cross points, which in turn can generate numerous chess compositions. There are around 10^{170} possible board configurations in the average 150-move game (1)! After all, this is much more challenging than medical decision making. Two decades ago, the IBM supercomputer Deep Blue defeated the world chess champion Garry Kasparov (2). However, the chess is slightly inferior to the Go in its complexity. Thus the AlphaGo's victory has more far-reaching significance.

Turning to the medical community, how can we view the progress made by artificial intelligence? First of all, let's take a look at the role that doctors can play in human society. Long ago before in the Middle Ages, medical treatment was primarily provided by priests. Patients at that time could only get console from these healthcare providers and there was no evidence of biological efficacy (3,4). A sick person could get the hope of survival from that spiritual treatment, and he or she might recover with the placebo effect. For most illnesses, the spiritual console may have healing power. After Renaissance, the modern science developed rapidly and was introduced into medical treatment. The evidence-based medicine dictated that medical treatment should be based on empirical evidence rather than experience or other super-nature power. The scientific knowledge greatly improved human health and longevity, owing to reduction of infectious diseases and malnutrition (5). However, the disease spectrum changed from infectious diseases to tumors and cardiovascular diseases, which imposed a great challenge to medical community in the 21st century (6). Although millions of research papers have been published declaring that they have found novel mechanisms underlying these diseases, the translation of the basic scientific knowledge into healthcare improvement is far from satisfactory (7).

When it comes into the 21st century, the advances of information technology turn the big data science into reality. Big data dredging have found their way into all areas of scientific field such as politics, marketing and biology (8,9). Machine learning is commonly used in dealing with big data, and artificial neural network (ANN) is an interesting machine learning technique. It is the ANN procedure that defeated Mr. Lee Sedol. The underlying mechanism of ANN is very similar to that of a biological human brain. Initially, the ANN has no experience of playing Go, but it will study millions of positions from expert games and glean abstract information on the state of play from board data (10). After training, the AlphaGo program is able to select the best move by scanning possible simulated future games (1). The ANN machine learning employs a black-box algorithm that researchers may have no idea on the mechanisms of why Lee Sedol moves in such a way. The learning algorithm focuses on input feature variable and the response variable, and a model with hidden layer can be fitted to reflect proper relationship between input variable and outcomes.

The black-box algorithm has its application in medical

field. Since it appears difficult to translate bench work to clinical efficacy in recent decades, it may be possible to develop diagnostic algorithm and treatment strategy by learning from millions of previous samples. In this black-box approach, the biological or pathophysiological mechanisms underlying treatment outcome are not important. After all, the medical science is not deterministic and can not be predicted accurately with formula and several parameters. It is the work by Newton that the appearance of a planet can be accurately predicted by formula. Such deterministic phenomenon is not applicable in the field of clinical research. There are random errors in medicine. They are called "random errors" because the underlying molecular pathway remains largely unknown. It is possible that the outcome of a disease is the results of interactions among millions of molecular pathways, but our state-of-art knowledge only have disentangled dozens of them. That is why statistics is indispensible to the clinical research. In my opinion, intelligent activities involved in medical decision-making are much simpler than that in Go match. It is possible that artificial intelligence can replace human brain to make medical decisions since there are rules governing them.

Does it mean that artificial intelligence will replace doctors to treat patients? The answer is of course no. Medical treatment involves biological human body as well as the soul of patients (11,12). The most important role as a physician is not to cure disease, but being a comforter to the sick (13). As the motto goes: "*To cure sometimes, to relieve often, to comfort always.*" (14). Therefore, human doctors are indispensible to medical treatment, by playing the role of a soul comforter. Artificial intelligence can only take a small part of medical activity, because there is no calculation algorithm to understand human emotions.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Silver D, Huang A, Maddison CJ, et al. Mastering the game of Go with deep neural networks and tree search. Nature 2016;529:484-489.
2. McGrew T. Collaborative intelligence. The internet chess club on game 2 of kasparov vs. deep blue. IEEE Internet Computing 1997;1:38-42.
3. French RK. Medicine before science: The business of medicine from the Middle Ages to the Enlightenment. Cambridge: Cambridge University Press, 2003.
4. Getz F. Medicine in the English middle ages. Princeton, NJ: Princeton University Press, 1998.
5. Kass EH. History of the specialty of infectious diseases in the United States. Ann Intern Med 1987;106:745-756.
6. Jones DS, Podolsky SH, Greene JA. The burden of disease and the changing task of medicine. N Engl J Med 2012;366:2333-2338.
7. Zerhouni EA. Translational and clinical science--time for a new vision. N Engl J Med 2005;353:1621-1623.
8. Zhang Z. Big data and clinical research: focusing on the area of critical care medicine in mainland China. Quant Imaging Med Surg 2014;4:426-429.
9. Monteith S, Glenn T, Geddes J, et al. Big data are coming to psychiatry: a general introduction. Int J Bipolar Disord 2015;3:21.
10. Gibney E. Google AI algorithm masters ancient game of Go. Nature 2016;529:445-446.
11. Marvel MK, Epstein RM, Flowers K, et al. Soliciting the patient's agenda: have we improved? JAMA 1999;281:283-287.
12. Barry CA, Bradley CP, Britten N, et al. Patients' unvoiced agendas in general practice consultations: qualitative study. BMJ 2000;320:1246-1250.
13. Cayley WE Jr. Comfort always. Fam Pract Manag 2006;13:74.
14. Coote B. Medical humanities: to cure sometimes, to relieve often, to comfort always. Med J Aust 2005;182:430, 432; author reply 432.

Cite this article as: Zhang Z. When doctors meet with AlphaGo: potential application of machine learning to clinical medicine. Ann Transl Med 2016;4(6):125. doi: 10.21037/atm.2016.03.25

Release of the national healthcare big data in China: a historic leap in clinical research

Zhongheng Zhang

Received: 13 January 2017; Accepted: 14 January 2017; Published: 09 February 2017.
doi: 10.21037/amj.2017.02.03
View this article at: http://dx.doi.org/10.21037/amj.2017.02.03

At the beginning of 2017, the National scientific data sharing platform for population and health (NSDSPPH) released 49.1 TB data on Chinese population and health. The data comprise 280 million observations involving the fields of biomedicine, basic medicine, clinical medicine, public health, traditional Chinese medicine, pharmacology, population and reproductive health (http://www.ncmi.cn/1). NSDSPPH is a major project of the national science and technology infrastructure. It was launched in 2003. Clinical investigators may be interested in clinical data contained in the NSDSPPH. The clinical data center is jointly established by Peking Union Medical College Hospital and Chinese PLA General Hospital. After more than 10 years of endeavor, the clinical data center has become more and more sophisticated. Clinical data were collected from hospitals nationwide with strict criteria for data entry. Data are presented in several forms such as the patient-level data, summary data and statistical report. The clinical data are accessible to the public and researchers. Additionally, the center also provides services of statistical analysis and data mining. Numerous scientific papers have been published using the database. Some examples include the correlation between waist circumference and respiratory function in teenagers (1), epidemiological study on coexisting prehypertension and prediabetes in northern China (2), and the association between γ-glutamyltransferase and prehypertension (3).

Medical big data are big treasure that can provide valuable information for scientific researches (4,5). Investigators can test their ideas and hypothesis by using the big data. Although China has the largest population, as well as the patient population in the world, there is little voice in the research field of clinical medicine. In major medical journals, especially the clinical journals, most high-impact articles are reported based on data from western countries. As a result, Chinese patients receive medical treatment based on guidelines derived from western countries. It is largely unknown whether results or conclusions based on western populations are generalizable to Chinese patients. The cause of this contradictory situation is partly due to the lack of awareness of the importance of heath care data exploration. Furthermore, clinical studies based on China are seldom accessible to other researchers. In other words, collected data are seldom openly accessible to the public and other investigators. The situation is that while the government invests large amount of funds on research projects and the funded research group collects a large body of healthcare-related data, the researchers may only exploit a minority of these data and selected results are reported. This is a waste of data and funds. If the data are openly accessible, other researchers can perform data mining, individual-patient level meta-analysis from a distinct perspective. In other words, more useful information can be extracted from the dataset for medical decision making. In an analogy to gold mine, the value of big data can never be fully exploited unless it is openly accessible to researchers. Data opening is also in concordance with the international open data campaign, many journals such as *PloS One* and *British Medical Journal* have declared that original articles published in their journals must include a statement on data availability (6,7).

However, datasets released at this initial stage are not without limitations. For example, there is a dataset established based on critically ill patients treated in the intensive care unit (http://124.207.187.26:8080/lab/index.jsp). The number of observation is very small that there are less than 200 patients and/or observations. The recorded variables are also limited with less than 100 pieces of items. Recorded variables include demographics, physical variables, laboratory findings, severity scores, antibiotics and outcomes. However, the

recording is over simplified. For example, only the name of antibiotics is available and there is no information regarding the time of initiation and discontinuation, dosage and administration route. Laboratory variable is simply presented as a numeric value and there is no information on the time of measurement. Usually, investigators are interested in serial changes of laboratory findings, which is indicative of therapeutic efficacy, disease progression and recovery. With lack of these multi-dimensional data, it is difficult to establish a learning algorithm for medical decision making. Missing values are also prevalent in this dataset, which may impose challenges in statistical analysis (8). It appears that the ICU dataset is from a single hospital, which significantly compromised the generalizability of the results. However, we have to appreciate that the opening of the dataset is a historic leap in clinical research in China. We cannot expect a perfect dataset at its initial phase.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

doi: 10.21037/amj.2017.02.03
Cite this article as: Zhang Z. Release of the national healthcare big data in China: a historic leap in clinical research. AME Med J 2017;2:19.

References

1. Feng K, Chen L, Han SM, et al. Ratio of waist circumference to chest circumference is inversely associated with lung function in Chinese children and adolescents. Respirology 2012;17:1114-1118.
2. Wu J, Yan WH, Qiu L, et al. High prevalence of coexisting prehypertension and prediabetes among healthy adults in northern and northeastern China. BMC Public Health 2011;11:794.
3. Qin X, Tang G, Qiu L, et al. Association between γ-glutamyltransferase and prehypertension. Mol Med Rep 2012;5:1092-1098.
4. Zhang Z. Big data and clinical research: focusing on the area of critical care medicine in mainland China. Quant Imaging Med Surg 2014;4:426-429.
5. Zhang Z. Big data and clinical research: perspective from a clinician. J Thorac Dis 2014;6:1659-1664.
6. Henry D, Fitzpatrick T. Liberating the data from clinical trials. BMJ 2015;351:h4601.
7. Alsheikh-Ali AA, Qureshi W, Al-Mallah MH, et al. Public availability of published research data in high-impact journals. PLoS One 2011;6:e24357.
8. Zhang Z. Missing values in big data research: some basic skills. Ann Transl Med 2015;3:323.

Data management by using R: big data clinical research series

Zhongheng Zhang

Abstract: Electronic medical record (EMR) system has been widely used in clinical practice. Instead of traditional case report forms produced by manual input, the EMR makes big data clinical research feasible. The most important feature of big data research is its real-world setting. Furthermore, big data research can provide all aspects of information related to healthcare. However, big data research requires some skills on data management, which however, is always lacking in the curriculum of medical education. This greatly hinders doctors from testing their clinical hypothesis by using EMR. To make ends meet, a series of articles introducing data management techniques are put forward to guide clinicians to big data clinical research. The present tutorial firstly introduces some basic knowledge on R language, followed by some data management skills on creating, recoding and renaming variables. These are very basic skills and may be used in every project of big data research.

Keywords: Big-data clinical trial; electronic medical record (EMR); R language

Submitted Oct 12, 2015. Accepted for publication Nov 11, 2015.
doi: 10.3978/j.issn.2305-5839.2015.11.26
View this article at: http://dx.doi.org/10.3978/j.issn.2305-5839.2015.11.26

Introduction

Electronic medical record (EMR) system has already been widely used in most hospitals in China, and it serves as a potential reservoir to provide resources for clinical research (1-3). EMR can be regarded as a form of big data because the data volume of EMR is ever expanding (4). All information of a patient, from outpatient medication to inpatient management, can be easily extracted from established database. Furthermore, hospital admissions and outpatient visits can be linked together by using unique patient identification number. However, clinicians are always experts in clinical practice but lack necessary skills in the management of big data. They are usually overwhelmed by the complexity of the structure of EMR data. As a result, many research questions cannot be answered by using the readily available big data. To conduct prospective experimental or observational studies is usually time consuming and even impossible for the extremely busy clinicians. To make ends meet, a series of tutorial articles introducing data management skills are put forward to guide clinicians to big data clinical research.

R

R is not a name of software, but it is a language and environment for data management, graphic plotting and statistical analysis (5,6). R is freely available and is an open source environment that is supported by world research community. Thousands of statistical and graphing packages are available for use, and the package pool is ever expanding. One attractive feature of R is its graphing capability, allowing for nearly any customizations of figures (7). R can be downloaded from Comprehensive R Archive Network (CRAN) at http://cran.r-project.org. Users can easily complete the installation following guidance on the website. After installation, R console should be launched where one can input commands for data analysis. In the following sections, I assume that users have already been familiar with the preparations of R. This section will focus on creating, recoding the renaming variables. Although these techniques look simple, they must be used in each project of big data exploration.

For the ease of reading, I denote codes that can be executed in R console with the beginning symbol ">".

Working example

A data frame is created to include original variables such as PaO_2, FiO_2, Glasgow coma scale (GCS), mean arterial pressure (MAP), bilirubin, platelet count, creatinine and

urine output. The simulated dataset is used for illustration purpose, and it has no practical meaning.

Continuous variables are created by assuming that they have normal distributions, whereas categorical variables are created by assuming binomial distributions.

```
>pao2<-round(rnorm(100, mean = 300, sd = 30))
>fio2<-round(rnorm(100, mean = 0.5, sd = 0.1),2)
>gcs<-round(rnorm(100, mean = 80, sd = 20)/10)
>map<-round(rnorm(100, mean = 65, sd = 15))
>dop<- round(rnorm(100, mean = 10, sd = 3),1)
>dob<- rbinom(100, 1, 0.4)
>epi<- round(rnorm(100, mean = 10, sd = 3)/100,2)
>nor<- round(rnorm(100, mean = 10, sd = 3)/100,2)
>bilirubin<-round(rnorm(100, mean = 80, sd = 20),1)
>platelet<-round(rnorm(100, mean = 180, sd = 50))
>cr<-round(rnorm(100, mean = 150, sd = 34))
>uo<-rnorm(100, mean = 1000, sd = 500)
>uo<-round(ifelse(uo>0,uo,-uo))
```

The function rnorm() is used to randomly generate variables with mean and standard deviation (sd) arguments. Because generated observations may have decimal places which is not in line with the real world situation, round() function is used to obtain integers (e.g., this function was applied to variable uo and GCS). Ifelse() function is used to convert negative variables into positive ones. rbinom() function is used to generate categorical variables with binomial distributions. In the calculation of SOFA score, we only need to know whether dobutamine is used or not. However, these steps generated separate vectors that are stored in R workspace and we need to combine them into a data frame.

```
>data<-data.frame(pao2,fio2,gcs,map,dop,
 dob,epi,nor,bilirubin,platelet,cr,uo)
>head(data,8)
```

The results are shown in *Table 1*. There are 100 observations in the dataset but only the first 8 are displayed. The first row is the variable name and the first column is the number of observations. This dataset illustrates a typical example that we can extract from EMR. We will use it for the illustration of several basic R functions in the following sections.

Creating new variable

It is a common practice to create new variables in data analysis. These variables are called secondary variables, to distinguish them from original variables that can be extracted directly from EMR. In clinical practice, the most widely used secondary variables are a variety of scores. Particularly in critical care medicine, there are numerous risk stratification scores that can be calculated from original physiological and laboratory variables. Sequential Organ Failure Assessment (SOFA) score is one of such example and I would like to illustrate how it can be calculated from original variables.

SOFA score is used to determine the extent of organ dysfunctions. It is based on six different scores for respiratory, hepatic, cardiovascular, coagulation, neurological and renal systems (*Table 2*) (8). The SOFA score equals the sum of scores of each organ system, and it simplifies multi-dimension parameters into one dimension.

First, we calculate scores for each organ system. Because there are five categories for each system, we use the ifelse() function.

```
>data$respiratory<-ifelse(data$pao2/data$fio2>=400,0,
 ifelse(data$pao2/ data$fio2>=300,1,
 ifelse(data$pao2/ data$fio2>=200,2,
 ifelse(data$pao2/ data$fio2>=100,3,4))))
>data$neuro<-ifelse(data$gcs >14,0,
 ifelse(data$gcs >=13,1,
 ifelse(data$gcs >=10,2,
 ifelse(data$gcs >=6,3,4))))
>data$liver<-ifelse(data$ bilirubin <20,0,
 ifelse(data$ bilirubin <=32,1,
 ifelse(data$ bilirubin <=101,2,
 ifelse(data$ bilirubin<=204,3,4))))
>data$coagulation<-ifelse(data$ platelet >=150,0,
 ifelse(data$ platelet >=100,1,
 ifelse(data$ platelet >=50,2,
 ifelse(data$ platelet >=20,3,4))))
```

The renal score is a little more complex than others because it encompasses serum creatinine and urine output. The "or" relationship means that for an individual patient we use the maximum scores derived from either urine

Table 1 Simulated dataset for illustration (with the first 8 observations displayed)

No.	PaO_2	FiO_2	GCS	MAP	Dop	Dob	Epi	Nor	Bilirubin	Platelet	Cr	Uo
1	326	0.63	9	66	14.6	0	0.11	0.09	70.6	246	164	1144
2	308	0.51	8	59	10.4	0	0.10	0.10	67.8	138	196	578
3	274	0.66	7	42	12.6	1	0.13	0.10	91.2	128	123	1629
4	291	0.57	11	69	6.2	1	0.12	0.09	80.7	94	158	619
5	269	0.50	6	67	11.6	0	0.08	0.11	39.7	383	86	989
6	328	0.54	11	61	7.5	1	0.07	0.12	60.9	190	129	39
7	322	0.61	8	64	7.9	1	0.07	0.05	60.6	182	117	1129
8	298	0.40	5	68	5.4	1	0.06	0.08	72.9	92	141	450

GCS, Glasgow coma scale; MAP, mean arterial pressure; Dop, dopamine; Dob, dobutamine; Epi, epinephrine; Nor, norepinephrine; Cr, creatinine; Uo, urine output.

output or creatinine. Therefore, we first calculate scores for the urine output and the creatinine, respectively; then the maximum score is used as the final renal score.

```
>cr.score<-ifelse(data$ cr <110,0,
 ifelse(data$ cr <=170,1,
 ifelse(data$ cr <=229,2,
 ifelse(data$ cr <=440,3,4))))
>uo.score<-ifelse(data$uo>=500,0,ifelse(data$uo
>=200,3,4))
>data$renal<-pmax(cr.score, uo.score)
```

Scores for cardiovascular system comprise five variables (e.g., map, dop, dob, epi and nor). We can apply pmax() function with more variables as its arguments.

```
>map.score<-ifelse(data$map>=70,0,1)
>dop.score<-ifelse(data$dop<=5,2,
 ifelse(data$dop<=15,3,4))
>dob.score<-ifelse(data$dob==1,2,0)
>epi.score<-ifelse(data$epi==0,0,ifelse(data$epi
<=0.1,3,4))
>nor.score<-ifelse(data$nor==0,0,ifelse(data$nor
<=0.1,3,4))
>data$cardio<-pmax(map.score, dop.score,
 dob.score, epi.score, nor.score)
```

Then the total SOFA score can be calculated by summing up all individual system scores.

```
>data$sofa.score<- data$cardio+ data$renal+
 data$coagulation+ data$liver+ data$neuro+
data$respiratory
>head(data)
```

We can now take a look at the new dataset by using head() function (*Table 3*). The dataset contains scores for each individual system (from variable cardio to coagulation) and the last column is the SOFA score.

Recoding variables

The most commonly used technique in recoding variable is to change a continuous variable into a set of categories. To recode data, one can use R's logical operators (*Table 4*).

Table 2 Components of Sequential Organ Failure Assessment (SOFA) score

Systems	0	1	2	3	4
Respiratory (PaO_2/FiO_2) (mmHg)	≥400	300-400	200-300	100-200 and mechanical ventilation	<100 and mechanical ventilation
Neurological (GCS)	>14	13-14	10-12	6-9	<6
Cardiovascular (MAP or vasopressor[¶])	MAP ≥70	MAP <70	Dop ≤5 or dob (any dose)	Dop >5 or epi ≤0.1 or nor ≤0.1	Dop >15 or epi >0.1 or nor >0.1
Liver [Bilirubin (μmol/L)]	<20	20-32	33-101	102-204	>204
Coagulation (platelets $\times 10^3/\mu L$)	≥150	100-150	50-100	20-50	<20
Renal [creatinine (μmol/L) or urine output]	<110	110-170	171-229	300-440 (or <500 mL/d)	>440 (or <200 mL/d)

[¶], vasopressor drug doses are in μg/kg/min. GCS, Glasgow coma scale; MAP, mean arterial pressure; Dop, dopamine; dob, dobutamine; epi, epinephrine; nor, norepinephrine.

Table 3 New dataset comprising original variables and newly created variables

No.	PaO_2	FiO_2	GCS	MAP	Dop	Dob	Epi	Nor	Bilirubin	Platelet	Cr	Uo	Cardio	Renal	Respiratory	Neuro	Liver	Coagulation	Sofa.score
1	326	0.63	9	66	14.6	0	0.11	0.09	70.6	246	164	1,144	4	4	0	3	2	0	13
2	308	0.51	8	59	10.4	0	0.10	0.10	67.8	138	196	578	4	4	0	3	2	1	14
3	274	0.66	7	42	12.6	1	0.13	0.10	91.2	128	123	1,629	4	4	0	3	2	1	14
4	291	0.57	11	69	6.2	1	0.12	0.09	80.7	94	158	619	4	4	0	2	2	2	14
5	269	0.50	6	67	11.6	0	0.08	0.11	39.7	383	86	989	4	4	0	3	2	0	13
6	328	0.54	11	61	7.5	1	0.07	0.12	60.9	190	129	39	4	4	0	2	2	0	12

GCS, Glasgow coma scale; MAP, mean arterial pressure; Dop, dopamine; Dob, dobutamine; Epi, epinephrine; Nor, norepinephrine; Cr, creatinine; Uo, urine output.

Table 4 Logical operators in R

Operators	Meaning
<	Less than
<=	Less than or equal to
>	More than
>=	More than or equal to
==	Equal to
!=	Not equal to
!a	Not a
a|b	a or b
a&b	a and b
isTRUE(a)	Test if a is true

These logical operators create logical expression and return the value of TRUE or FALSE.

Suppose that we want to make classifications of acute respiratory distress syndrome (ARDS) based on Berlin definition. This definition proposed 3 mutually exclusive categories of ARDS based on the degree of hypoxemia: severe (PaO_2/FiO_2 ≤100 mmHg), moderate (100 mmHg < PaO_2/FiO_2 ≤200 mmHg), and mild (200 mmHg < PaO_2/FiO_2 ≤300 mmHg) (9). Then we can recode continuous variable PaO_2/FiO_2 to categorical variable berlin (severe, moderate and mild). First, we create a new variable named 'oxyindex'.

```
>data$oxyindex<- data$pao2/data$fio2
```

Because patients with oxygen index greater than 500 are less likely to have ARDS, we need to exclude them.

```
>data$oxyindex[data$oxyindex>=500]<-NA
```

This statement assigned null (NA) values to observations with oxygen index greater than 500. The statement 'variable[conditions]<-expression' takes value from expression when the condition is TRUE.

Next we can use the following code to create the berlin variable:

```
>data$berlin[data$oxyindex<=100] <- "severe"
>data$berlin [data$oxyindex >100 &
 data$oxyindex <= 200] <- "moderate"
>data$berlin [data$oxyindex >200] <- "mild"
```

Data frame names are used in these codes to ensure the new variables are saved back to the data frame. Alternatively, if you do not want to repeat data frame name, you can use within() function to write the code more compactly.

```
>data <- within(data,{
 berlin <- NA
 berlin[oxyindex<=100] <- "severe"
 berlin[oxyindex>100 & oxyindex<=200] <- "moderate"
 berlin[oxyindex>200& oxyindex<500] <- "mild"})
```

Another very useful function shipped with R is cut(), which is able to convert continuous variables into factor variables.

```
>data$berlin<-cut(data$oxyindex,
 breaks=c(500,300,200,100),
 labels=c("mild","moderate","severe"))
```

The first argument of the cut() function is a numeric vector to be converted to a factor variable. The second argument can be a numeric vector containing two or more cutoff points. The last argument labels each level of the factor variable. Note that we do not need to specify NA values to values greater than 500, and the newly created factor variable automatically excludes observations with oxygen index >500. We need to ensure there is no value less than 100.

Renaming variables

When working with big data, you may have hundreds of variables at hand and you need to rename variables to avoid confusion. In our example, if you are not happy with the name oxyindex, you can change it by using names() function.

```
>names(data)[20]<- "oxygen.index"
>names(data)[11]<- "creatinine"
```

As you can see, the names() function extract variable names of a data frame.

```
> names(data)
[1]      "pao2"          "fio2"          "gcs"
[4]      "map"           "dop"           "dob"
```

```
[7]   "epi"          "nor"          "bilirubin"
[10]  "platelet"     "creatinine"   "uo"
[13]  "respiratory"  "neuro"        "liver"
[16]  "coagulation"  "renal"        "cardio"
[19]  "sofa.score"   "oxygen.      "berlin"
                     index"
```

Alternatively, you can use rename() function for the same purpose (10). Rename() function is in the reshape package and you should load it to the working environment first.

```
>install.packages("reshape")
>library(reshape)
>data <- rename(data,
 c(oxyindex="oxygen.index", cr="creatinine"))
```

This code looks more compact and it is particularly useful to change a series of variable names.

Summary

The tutorial introduces some basic R functions for data management. Differently from other R tutorials, this article illustrates the R functions in the context of clinical research. The questions discussed were those that have been encountered by the author, and the R functions were considered as the most fundamental skills. Creating new variable is to create new variable based on other original variables that are directly extracted from EMR. The ifelse() function is very useful. In recoding variables, logical operators are always used. There is also a useful function called cut() which is able to convert continuous variables into factor variables. Renaming variable is applied when there are hundreds of variables and you are not satisfied with its original forms. The principle of naming variable is to make it concise and informative.

Cite this article as: Zhang Z. Data management by using R: big data clinical research series. Ann Transl Med 2015;3(20):303. doi: 10.3978/j.issn.2305-5839.2015.11.26

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Zhang Z. Big data and clinical research: focusing on the area of critical care medicine in mainland China. Quant Imaging Med Surg 2014;4:426-429.
2. Zhang Z. Big data and clinical research: perspective from a clinician. J Thorac Dis 2014;6:1659-1664.
3. Monteith S, Glenn T, Geddes J, et al. Big data are coming to psychiatry: a general introduction. Int J Bipolar Disord 2015;3:21.
4. Potash JB. Electronic medical records: fast track to big data in bipolar disorder. Am J Psychiatry 2015; 172:310-311.
5. Kabacoff R. R in Action. Shelter Island: Manning Publications Co., 2011.
6. Lander JP. R for Everyone: Advanced Analytics and Graphics. Boston: Addison-Wesley Professional, 2014.
7. Horton NJ, Kleinman K. Using R for data management, statistical analysis, and graphics. Clermont: CRC Press, 2010.
8. Vincent JL, Moreno R, Takala J, et al. The SOFA (Sepsis-related Organ Failure Assessment) score to describe organ dysfunction/failure. Intensive Care Med 1996;22:707-710.
9. ARDS Definition Task Force, Ranieri VM, Rubenfeld GD, et al. Acute respiratory distress syndrome: the Berlin Definition. JAMA 2012;307:2526-2533.
10. Wickham H. Reshaping data with the reshape package. Journal of Statistical Software 2007;21:1-20.

Missing values in big data research: some basic skills

Zhongheng Zhang

Submitted Oct 15, 2015. Accepted for publication Nov 23, 2015.
doi: 10.3978/j.issn.2305-5839.2015.12.11
View this article at: http://dx.doi.org/10.3978/j.issn.2305-5839.2015.12.11

Introduction

Missing values occur when there is no data value for a variable in an observation. The phenomenon of missing value is universal in clinical researches involving big data. Nurses may forget to record urine output at a certain time point. Patients may have only one measurement of blood lactate, while the researcher is interested in exploring the impact of lactate trend on mortality outcome. Other reasons of missing values include but not limited to coding errors, faulty equipment and nonresponses (1). In statistical packages, some commands (e.g., logistic regression) may automatically delete observations with missing values. There is no problem if there are a few incomplete observations. However, when there are a large number of observations with missing values, the default listwise deletion may result in significant loss of information. In such a situation, analysts should take a close look at the missing patterns and find appropriate means to cope with it. The present article will introduce how missing values are handled in R, and provide some basic skills in dealing with missing values.

How missing value is handled in R

Missing value is represented by the symbol NA (not available) in R. When you read an Excel spreadsheet containing empty cells into R console, these empty cells will be replaced by NAs. This is different from STATA where empty cells are replaced with ".". The same missing value symbol is used in R for both numeric and character variables. R provides several functions for handling missing values (*Table 1*).

The function is.na() is the most commonly used method to indicate which element is NA. It returns logical values (FALSE or TRUE) and has the same length as its argument. Suppose we have six patients. Five lactate values are recoded and one is missing.

```
>lactate<-c(0.2,3.3,4.5,NA,6.1,2.4)
>is.na(lactate)
[1] FALSE FALSE FALSE TRUE FALSE FALSE
> which(is.na(lactate))
[1] 4
```

The length of the returning vector of is.na() is six. In the fourth place the value is TRUE, indicating lactate value is missing in the forth patient.

Someone may think of using logical test (e.g., lactate==NA) to examine the missing pattern. This can never be TRUE because missing values are considered non-comparable and you have to use missing value functions. The "==" operator returns NA when either argument is NA. By using which() function, you can locate the element of a vector which contains NA. In the example, which() function returns 4, indicating the forth patient has missing lactate.

Next, you may want to describe lactate levels of the six patients. In statistical description, mean, variance and standard deviation are among the most commonly used statistics.

```
>mean(lactate)
[1] NA
>sum(lactate)
[1] NA
>var(lactate)
[1] NA
>sd(lactate)
[1] NA
```

All of these functions return the NA value because the vector lactate contains missing values. Fortunately, there is a function na.rm() to remove NAs when applying statistical functions.

Table 1 R functions to handle missing values

Functions	Description
is.na()	To indicate which element is NA
na.rm()	It is used within other functions, as an optional argument
na.fail()	It can be used to detect missing values in a dataset
na.omit()	It returns the object with incomplete cases removed
complete.cases()	It returns a logical vector indicating which cases have no missing values
na.tree.replace()	Adds a new level called "NA" to any discrete predictor in a data frame that contains NAs
na.gam.replace()	A discrete variable with missing value is replaced by a new level labelled "NA". A missing numeric vector has its missing values replaced by the mean of the non-missing values

```
>mean(lactate,na.rm=TRUE)
[1] 3.3
>sd(lactate,na.rm=TRUE)
[1] 2.219234
```

The results are exactly what you want. Both mean and standard deviation are calculated based on the five patients with lactate values available.

Data frame with missing values

In real world setting, what you encounter is missing values in a data frame. Thus, this section focuses on how to handle data frames with missing values. First we create a data frame of three variables and five observations.

```
>ptid<-c(1,2,3,4,5)
>sex<-c("m","f",NA,"f","m")
>lactate<-c(0.2,3.3,4.5,NA,6.1)
> data<-data.frame(ptid,sex,lactate)
> data
```

	ptid	sex	lactate
1	1	m	0.2
2	2	f	3.3
3	3	<NA>	4.5
4	4	f	NA
5	5	m	6.1

Note that the third patient has missing value on sex, and the forth patient has missing value on lactate.

```
> na.fail(data)
Error in na.fail.default(data) : missing values in object
```

The function na.fail() can be used to detect missing values in a dataset. If there is no missing value, it returns the assigned object (see below). If there is missing value, it returns an error message indicating there is one or more missing values in the dataset.

Although some good default settings in regression model can effectively ignore observations with missing values, it is useful to create a new data frame by omitting observations with missing values.

```
> na.omit(data)
```

	ptid	sex	lactate
1	1	m	0.2
2	2	f	3.3
5	5	m	6.1

The function na.omit() returns an object with incomplete cases removed. As you can see, the third and forth patients with missing values in one variable are removed. Alternatively, the same purpose can be achieved by using the following code.

```
> complete.data<- data[complete.cases(data), ]
> complete.data
```

	ptid	sex	lactate
1	1	m	0.2
2	2	f	3.3
5	5	m	6.1

By applying na.fail() to the complete dataset obtained by na.omit(), the error message is replaced by a new

complete dataset. The test passed when the new dataset contains no NAs.

```
> na.fail(complete.data)
     ptid  sex  lactate
1    1     m    0.2
2    2     f    3.3
5    5     m    6.1
```

Sometimes you may want to locate which variable of a patient contains missing value. Try functions is.na() and which() as described previously.

```
> is.na(data)
      ptid    sex     lactate
[1,]  FALSE   FALSE   FALSE
[2,]  FALSE   FALSE   FALSE
[3,]  FALSE   TRUE    FALSE
[4,]  FALSE   FALSE   TRUE
[5,]  FALSE   FALSE   FALSE
> which(is.na(data))
[1] 8 14
```

It returns 8 and 14 indicating the location of missing values. R counts the place in a column-wise fashion. There is no problem when the position number is not large. However, if the dataset is large (always the case in big data exploration) you are overwhelmed by a long list of location numbers. The following code helps you to select observations with at least one missing value.

```
> unique (unlist (lapply (data,
 function (x) which (is.na (x)))))
[1] 3 4
```

As expected, the returned results indicate the third and forth patients who have at least one missing variable. The function lapply() returns a list of the locations of missing values across variables ptid, sex and lactate. Unlist() simplifies list structure to produce a vector. When there is more than one missing value in an observation, unique() simplified it to list observations with missing values only once.

In occasions, you may be interested in counting missing values for a variable (number of missing values in a column). Then you can restrict your analysis to variables with less than one fifth missing variables.

```
> unlist(lapply(data,
 function(x) sum(is.na(x))))/nrow(data)
ptid    sex     lactate
0.0     0.2     0.2
```

Missing values in regression model

Regression model building is probably the most commonly used method in statistical analysis. However, details on missing values are always omitted in regression model fitting. With small number of missing values, it is safe to fit a model by default argument. Problem may arise with substantial number of missing values and analysts have to understand how missing values are handled in model building. Here the difference between na.omit() and na.exclude() will be shown.

For the illustration purpose, I will regress lactate on patient id. Of course, this has no practical meaning.

```
> model.omit <- lm(ptid ~lactate, data = data, na.action
= na.omit)
> model.exclude <- lm(ptid ~lactate, data = data,
na.action = na.exclude)
> resid(model.omit)
1           2           3           5
0.3895652  -0.60522    -0.37739    0.593044
> resid(model.exclude)
1           2           3           4           5
0.3895652  -0.60522    -0.37739    NA          0.593044
> fitted(model.omit)
1           2           3           5
0.6104348   2.605217    3.377391    4.406957
> fitted(model.exclude)
1           2           3           4           5
0.6104348   2.605217    3.377391    NA          4.406957
```

As you can see, the na.exclude() function pads the residuals and fitted values with NAs where there are missing values. However, the na.omit() function simply exclude observations with missing values. In this regard, na.exclude() is a placeholder for missing values.

Some advanced functions for missing values

There are situations when you don't want to simply delete

observations with missing values. Or missing values may have special clinical relevance. In our example, missing lactate values may indicate that the patient have been fully recovered from shock (e.g., lactate is a biomarker of tissue hypoperfusion and hypoxia and clinicians typically do not order lactate for patients who are hemodynamically stable). Therefore, NAs in lactate indicate stable patients. Because NAs contain important information, it is wise to add a new category called "NA" to replace the missing values. You can try na.tree.replace() in the tree package for this purpose (2). However, this function is limited by the fact that it is only applicable for discrete variables with missing value. I create a new data frame for the illustration.

```
>install.packages("tree")
>library(tree)
>data.discrete<-data.frame(ptid=c(1,2,3,4,5),lactate
=c("low",NA,"moderate",NA,"high"),death=c("y","y",
NA,"n","y"))
> na.fail(data.discrete)
Error in na.fail.default(data.discrete) : missing values
in objects
> newdata.discrete<-na.tree.replace(data.discrete)
> na.fail(newdata.discrete)
  ptid lactate death
1    1     low     y
2    2    <NA>     y
3    3 moderate  <NA>
4    4    <NA>     n
5    5    high     y
```

The data frame data.discrete contains three variables and the last two discrete variables have NAs. The error message returned by na.fail() indicates NAs in the data frame data.discrete. Then, a new data frame called newdata.discrete is created by using na.tree.replace(). The na.fail() function returns the new data frame without error. As you can see missing values are replaced by string value "NA". The limitation of na.tree.replace() is that it stops if any continuous variable contains NAs.

```
> na.tree.replace(data)
Error in na.tree.replace(data):
  continuous variable lactate contained NAs
```

Note that an error message is returned because the data contains continuous variable with NAs. The problem can be solved by using na.gam.replace() function in the gam package. Note that gam package requires simultaneous installation of foreach package (3).

```
> install.packages("gam")
> library(foreach)
> library(gam)
> newdata<-na.gam.replace(data)
> na.fail(newdata)
  ptid sex lactate
1    1   m   0.200
2    2   f   3.300
3    3  NA   4.500
4    4   f   3.525
5    5   m   6.100
```

Note that the data frame newdata pass the na.fail() test and returns a new dataset. For variable sex the missing value is replaced by string value "NA", and the missing numeric value of lactate is replaced by the mean of the available lactate values.

Summary

The article reviews some basic skills in dealing with missing values in R. Missing values in R cannot be compared by using logical operators and thus specific function is.na() is fundamental in judging whether an object contains missing value or not. There is a variety of tips to do with missing values in data frame. Some useful codes are to locate and count missing values by using lapply() function. Lapply() is very useful in that it applies functions across all variables while avoiding the time-consuming loop functions. Regression model performs casewise deletion of missing values by default. However, the performances of na.omit() and na.exclude() with respect to residuals and fitted values of NAs are different. There are also some advanced packages for handling missing values. These include na.tree.replace() in the tree package and na.gam.replace() in the gam package.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Montez-Rath ME, Winkelmayer WC, Desai M. Addressing missing data in clinical studies of kidney diseases. Clin J Am Soc Nephrol 2014;9:1328-1335.
2. Ripley B. Tree: classification and regression trees (2007). (R package version 1.0-26).
3. Hastie T. Gam: generalized additive models (2011). (R package version 1.06.2).

Cite this article as: Zhang Z. Missing values in big data research: some basic skills. Ann Transl Med 2015;3(21):323. doi: 10.3978/j.issn.2305-5839.2015.12.11

Missing data exploration: highlighting graphical presentation of missing patterns

Zhongheng Zhang

Abstract: Functions shipped with R base can fulfill many tasks of missing data handling. However, because the data volume of electronic medical record (EMR) system is always very large, more sophisticated methods may be helpful in data management. The article focuses on missing data handling by using advanced techniques. There are three types of missing data, that is, missing completely at random (MCAR), missing at random (MAR) and not missing at random (NMAR). This classification system depends on how missing values are generated. Two packages, Multivariate Imputation by Chained Equations (MICE) and Visualization and Imputation of Missing Values (VIM), provide sophisticated functions to explore missing data patterns. In particular, the VIM package is especially helpful in visualization of missing data. Finally, correlation analysis provides information on the dependence of missing data on other variables. Such information is useful in subsequent imputations.

Keywords: Big-data Clinical Trial; Multivariate Imputation by Chained Equations (MICE); missing completely at random (MCAR); missing at random (MAR); not missing at random (NMAR)

Submitted Nov 15, 2015. Accepted for publication Dec 05, 2015.
doi: 10.3978/j.issn.2305-5839.2015.12.28
View this article at: http://dx.doi.org/10.3978/j.issn.2305-5839.2015.12.28

Introduction

The previous article of big-data clinical trial series has introduced basic techniques in dealing with missing values. There are several R packages that allow advanced methods for managing missing data. Some useful methods include visual presentation of missing data pattern and correlation analysis (1). This article firstly creates a dataset containing five variables. Three missing data classes are illustrated in creating the dataset by simulation. Then various tools for the exploration of missing data are introduced.

Classification of missing data

Statisticians typically classify missing data into three categories. Missing completely at random (MCAR) refers to the presence of missing values on a variable that is unrelated to any other observed and unobserved variables (2,3). In other words, there is no systematic reason for the missing pattern. Missing at random (MAR) is the presence of missing values on a variable that is related to other observed variables but not related to its own unobserved values. Not missing at random (NMAR) is the presence of missing values on a variable that is neither MCAR nor MAR. For example, a patient with lower lactate value is more likely to have a missing lactate value. A hemodynamically stable patient typically has a lower lactate value. In the situation, a treating physician is less likely to order test for lactate.

Dataset simulation

A dataset of 200 observations is created by simulation. The dataset is used for illustration purpose and there is no clinical relevance. There are five variables which are age, sex, lactate (*lac*), white blood cell (*wbc*) and C-reactive protein (*crp*). In each simulation, I set a seed to allow readers to replicate the results.If all codes are run at once, only one set.seed() function is needed.

```
> set.seed(123456)
> age<-round(abs(rnorm(200, mean = 67, sd = 19)))
> set.seed(12345)
> sex<-rbinom(200, 1, 0.45)
> set.seed(12356)
```

```
> sex.miss.tag<-rbinom(200, 1, 0.3) #MCAR
> sex[sex.miss.tag==1]<-NA
> sex[sex==1]<-"male"
> sex[sex==0]<-"female"
> set.seed(12456)
> lac<-round(abs(rnorm(200, mean = 3, sd = 4)),1)
> set.seed(13456)
> lac.miss.tag<-rbinom(200, 1, 0.3)
> lac[lac<=3&lac.miss.tag==1]<-NA # NMAR
> set.seed(23456)
> wbc<-round(abs(rnorm(200, mean = 12, sd = 4)),1)
> set.seed(123)
> wbc.miss.tag<-rbinom(200, 1, 0.3)
> wbc[wbc.miss.tag==1]<-NA
> set.seed(1234)
> crp<-round(abs(rnorm(200, mean = 50, sd = 100)),1)
> set.seed(3456)
> crp.miss.tag<-rbinom(200, 1, 0.4)
> crp[wbc<=12&crp.miss.tag==1]<-NA # MAR
> data<-data.frame(age,sex,lac,wbc,crp)
```

The variable *age* has complete values for all observations. It is assumed that our population has mean age of 67 with a standard deviation of 19. The abs() function is employed to avoid negative values. The results are rounded to integers by using round() function with default argument for decimal place. The variable sex is a categorical variable and it is assumed to have binominal distribution. Missing values on *sex* is set to MCAR. The variable *lac* has normal distribution with a mean value of 3 and a standard deviation of 4. It is set to NMAR and missing values occur more likely at *lac* values equal to or less than 3. The variable *wbc* has a normal distribution and missing values are MCAR. The variable *crp* assumes a normal distribution and missing values occur more frequently at *wbc* values equal to or less than 12. The rationale behind this missing pattern is that in clinical practice physicians may first order white blood cell count and for those with high WBC values they will further order test for *crp*.

Exploring missing pattern with md.pattern() function

The md.pattern() function shipped with Multivariate Imputation by Chained Equations (MICE) package can be used to produce a table displaying the missing pattern (4).

```
> install.packages("mice")
> library(mice)
> md.pattern(data)
```

	age	lac	crp	wbc	sex	
58	1	1	1	1	1	0
42	1	1	1	1	0	1
7	1	0	1	1	1	1
32	1	1	1	0	1	1
20	1	1	0	1	1	1
5	1	0	1	1	0	2
16	1	1	1	0	0	2
4	1	0	1	0	1	2
9	1	1	0	1	0	2
3	1	0	0	1	1	2
3	1	0	1	0	0	3
1	1	0	0	1	0	3
	0	23	33	55	76	187

In the main body of the output table, "1" indicates nonmissing value and "0" indicates missing value. The first column shows the number of unique missing data patterns. There are 58 observations with nonmissing values, and there are 42 observations with nonmissing values except for the variable sex. The rightmost column shows the number of missing variables in a particular missing pattern. For example, the first row has no missing value and it is "0" in the column. The last row counts the number of missing values for each variable. For example, the variable *age* contains no missing values and the variable *crp* contains 33 missing values. This table can be helpful when you decide to drop some observations with missing variables exceeding a preset threshold.

Visual presentation of missing data patterns

Although the above table displays missing pattern compactly and effectively, you may also want to show it in a figure. As the saying goes "*one look is worth a thousand words.*" The Visualization and Imputation of Missing Values (VIM) package is very powerful in visually displaying missing data patterns (5). This package contains advanced tools for the visualization of missing or imputed values. It is helpful for exploring the structure of the missing or imputed values. The missing data pattern is essential for selecting an appropriate imputation method to estimate missing

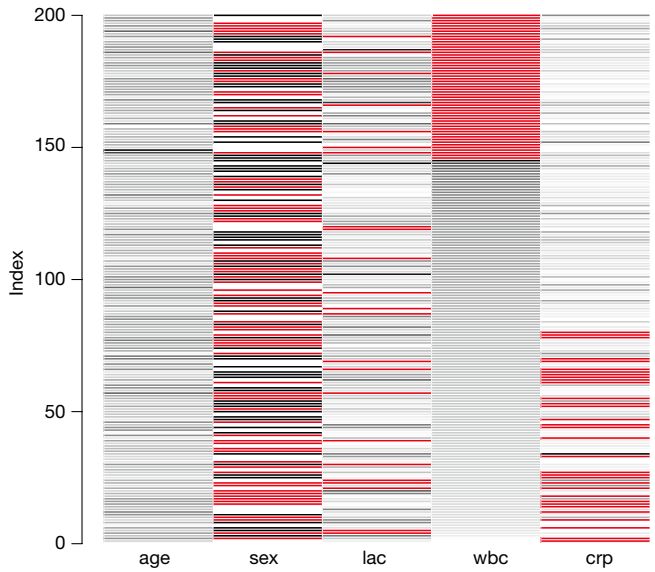


Figure 1 Matrix plot of nonmissing and missing values by observations. The matrix is sorted by the variable *wbc*.

values. Thus the visualization tools should be applied before imputation and the use of diagnostic tools afterwards. There are three functions can be used for this purpose: matrixplot(), scattMiss() and aggr().

```
> install.packages("VIM")
> library(VIM)
> matrixplot(data)
```

Click in a column to sort by the corresponding variable.

To regain use of the VIM GUI and the R console, click outside the plot region.

Matrix plot sorted by variable 'wbc'.

The matrixplot() is interactive and when the message "Click in a column to sort by the corresponding variable" pops up, I click on the *wbc* column. The result is shown in *Figure 1*. In the figure, missing values are represented in red. The continuous variable is rescaled and represented by grayscale colors. Lighter colors indicate lower values and darker colors suggest greater values. Note that missing values on *crp* occur only at lower levels of *wbc*, which is consistent with the rule used in the simulation. Because I choose to sort by the *wbc* variable, it is displayed firstly with missing values and then in descending order.

```
> barMiss(data) # similar function histMiss(data)
```

Click in the left margin to switch to the previous variable or in the right margin to switch to the next variable.

To regain use of the VIM GUI and the R console, click anywhere else in the graphics window.

```
> nrow(data[lac<=1&!is.na(lac),])
[1] 23
> table(complete.cases(data[lac<=1&!is.na(lac),]))
FALSE                          TRUE
18                             5
> table(complete.cases(data[is.na(lac),]
[,c("age","sex","wbc","crp")]))
FALSE                          TRUE
16                             7
```

The barMiss() and histMiss() functions produce similar figures and I would like to illustrate barMiss() only. By default, barMiss() produce interactive plot and one can click to choose on which variable to display. Here, I displayed lac variable. The horizontal axis is *lac* values. *Figure 2* displays barplot with highlighting of missing values in other variables by splitting each bar into two parts. Additionally, information about missing values in the variable *lac* is shown on the right hand side. There are 23 observations with lac ≤1. Of the 23 observations, there are 18 cases with missing values on other variables and 5 cases contain no missing values on other variables. The right hand side bar shows there are 23 missing values on the variable lac. Of them, there are 16 cases with missing values on other variables and 7 cases contain no missing values on other variables.

```
> aggr(data, numbers = TRUE, prop=FALSE)
```

The aggr() function produces missing data pattern as shown in *Figure 3*. The left panel displays the number of missing values on each variable. As expected, *age* has no missing values and *lac* has around 20% missing values. The right panel expresses the same information as the table produced by md.pattern() function. There are 58 complete observations without missing values. Forty-two observations contain missing values on the *sex* variable.

```
> marginplot(data[c("wbc","crp")], pch=c(20),
col=c("green", "red", "blue"))
```

The result of marginplot() is shown in *Figure 4*.

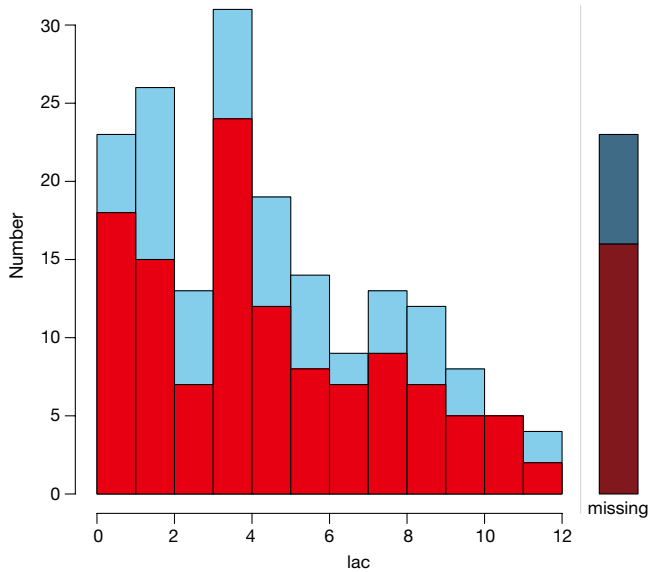


Figure 2 Barplot highlighting missing values in other variables by splitting each bar into two parts. One part represents missing values and the other represents nonmissing values.

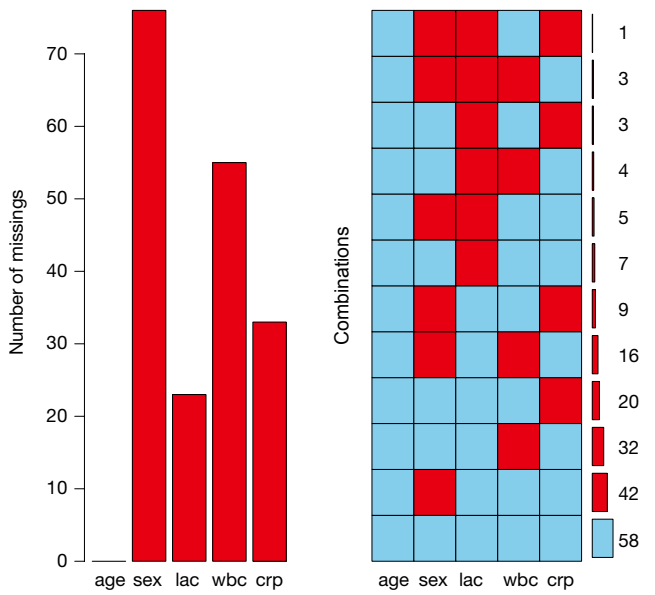


Figure 3 Missing data pattern produced by aggr() function.

Nonmissing values are displayed in green color and missing values are in red color. There are 33 missing values on *crp*, and the mean value of *wbc* with missing values on crp is around 9. *Wbc* with missing values on *crp* is significantly lower than *wbc* with complete values on *crp* (comparing horizontal red and green box plots). However, there is no difference between *crp* values in cases with and without missing values on *wbc* (vertical red and green box plots).

```
> marginmatrix(data)
```

This is an extension of the marginplot() function that creates a scatterplot matrix with information about missing values in the plot margins of each panel (*Figure 5*). Interpretation of each panel is the same as *Figure 4*.

```
> spineMiss(data)
```

Click in the left margin to switch to the previous variable or in the right margin to switch to the next variable.

To regain use of the VIM GUI and the R console, click anywhere else in the graphics window.

The spineMiss() function produces plot similar to that produced by barMiss(). The spineplot highlights missing values in other variables by splitting each cell into two parts (*Figure 6*). Additionally, information about missing values in the variable of interest (lac) is shown on the right hand side. The vertical axis is proportion instead of counts in barplot produced by barMiss() function.

```
> scattmatrixMiss(data)
```

Click in a diagonal panel to add to or remove from the highlight selection.

To regain use of the VIM GUI and the R console, click anywhere else in the graphics window.

Highlighted missing in any of the variables ‘age’, ‘sex’, ‘lac’, ‘wbc’, ‘crp’.

The scattmatrixMiss() produces Scatterplot matrix in which cases with missing values in certain variables (‘age’, ‘sex’, ‘lac’, ‘wbc’, ‘crp’) are highlighted (*Figure 7*). Variables with missing values to be highlighted can be added or removed by clicking in a diagonal panel. The diagonal panels display density plots for non-highlighted and highlighted observations. The red-cross symbols represent observations with missing values on any of the variables *age*, *sex*, *lac*, *wbc* and *crp*.

Exploring missing data patterns by correlation matrix

Correlation matrix can be utilized to explore which two variables tend to have missing values together, or the relationship between the presence of missing values in a

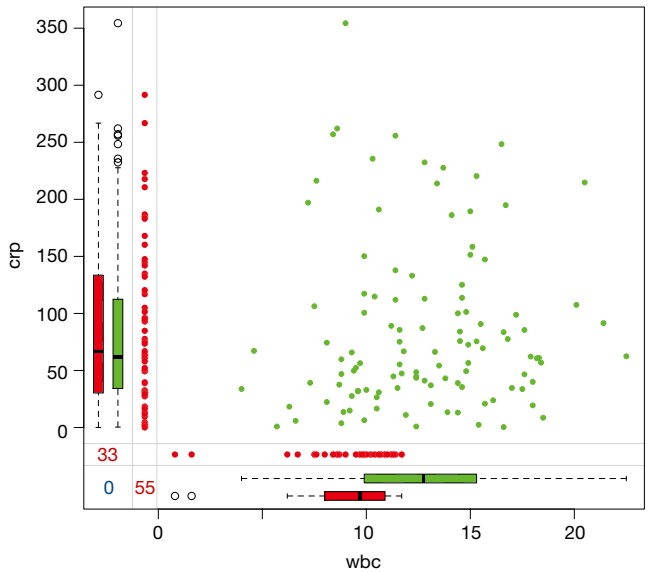


Figure 4 Scatter plot between *wbc* and *crp*, with missing values displayed on the margins.

variable and the observed values on other variables. To complete this task, one may need to create a shadow matrix in which missing values are replaced by "1", and nonmissing values are replaced by "0".

```
> shadow<- as.data.frame(abs(is.na(data)))
```

Next, you can create a new data frame in which only variables with one or more missing values are retained.

```
> miss.shadow<-shadow[
,which(unlist(lapply(shadow,sum))!=0)]
> round(cor(miss.shadow),3)
```

	sex	lac	wbc	crp
sex	1.000	0.008	-0.044	-0.070
lac	0.008	1.000	0.024	0.009
wbc	-0.044	0.024	1.000	-0.274
crp	-0.070	0.009	-0.274	1.000

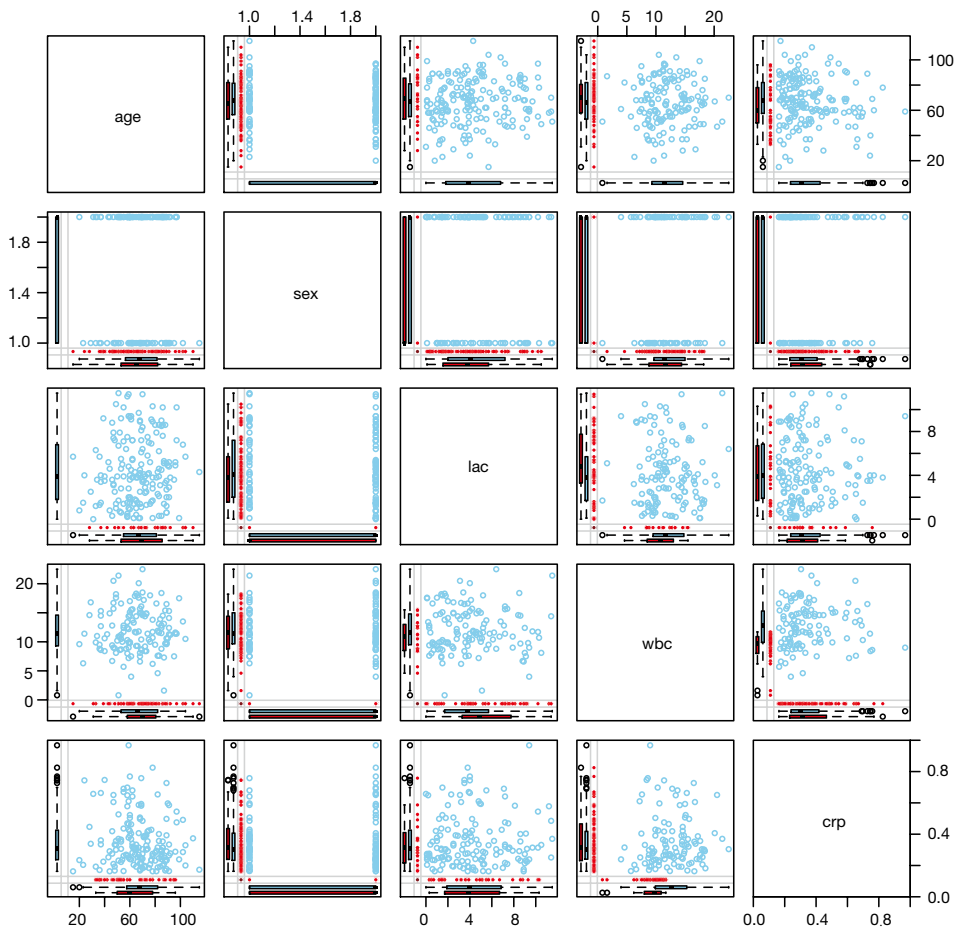


Figure 5 Scatterplot matrix with information about missing values in the plot margins of each panel.

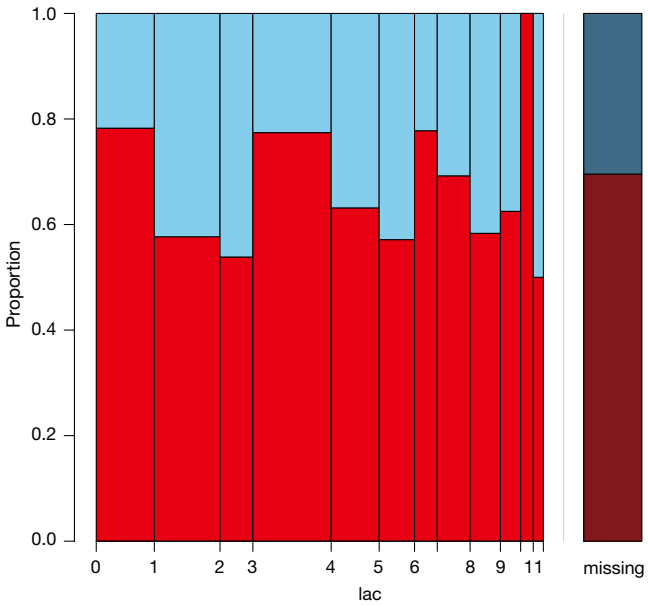


Figure 6 Spineplot highlighting missing values in other variables by splitting each cell into two parts. Additionally, information about missing values in the variable lac is shown on the right hand side.

There is no strong correlation among these variables and one can safely conclude that the presence of missing values in one variable is not related to missing values in other variables. Next, you can examine the relationship between the presence of missing values in a variable and the observed values on other variables. Before running the cor() function, you need to retain only numeric variable in the first argument of cor() function. The round() function is again used to make the output more succinct.

```
> round(cor(data[!names(data)%in%c("sex")],
 miss.shadow, use="pairwise.complete.obs"),3)
```

	sex	lac	wbc	crp
age	-0.031	0.055	0.082	-0.077
lac	-0.101	NA	0.163	0.029
wbc	-0.073	-0.115	NA	-0.413
crp	-0.019	-0.030	0.012	NA

As you can see there is a negative correlation between *crp* and *wbc* (r=–0.413), indicating that missing values on *crp* are

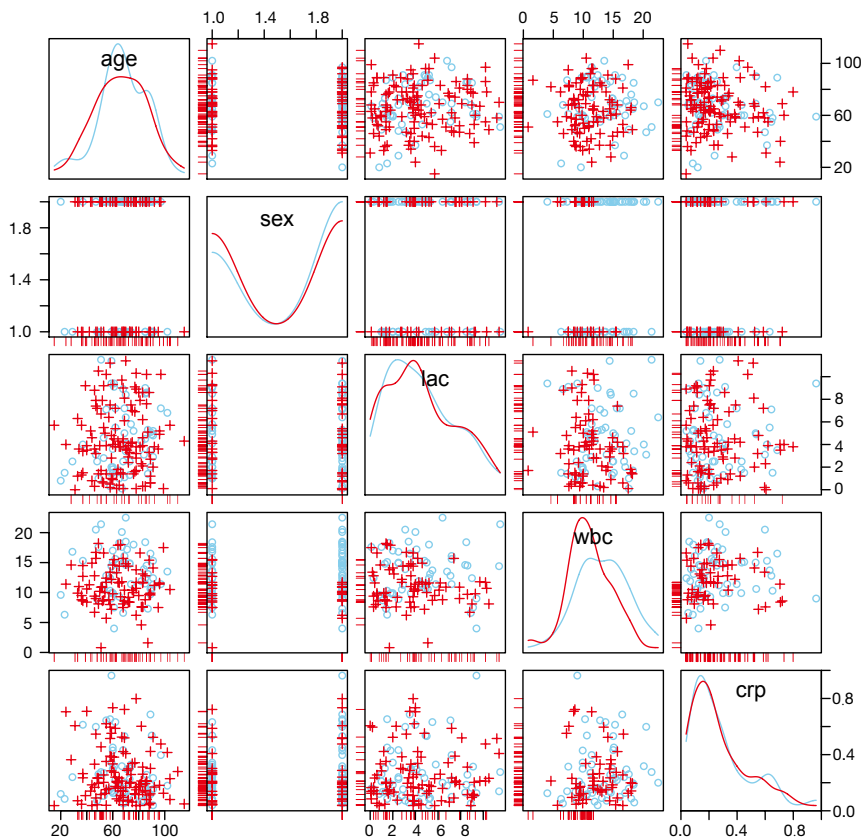


Figure 7 Scatterplot matrix in which observations with missing values in certain variables (*age*, *sex*, *lac*, *wbc*, *crp*) are highlighted.

more likely to occur at lower levels of *wbc*. The command is a little complex. The "names(data)%in%c("sex")" returns a logical vector with TRUE for each element in names(data) that matches "sex" and FALSE otherwise. The "!" symbol reverses the values of the logical vector. However, correlation analysis cannot replace using external information to judge whether missing data are NMAR. In other words, judgment from subject-matter knowledge is of critical importance to rule out NMAR.

Summary

Missing data are ubiquitous in big-data clinical research and sometimes the mechanisms underlying the missing pattern may be complicated. In this situation some advanced techniques in dealing with missing data may be helpful. Classified by the mechanism of missing, there are three types of missing data which are MCAR, MAR and NMAR. While imputations depending on other covariates can be used for the first two types, subject-matter knowledge is required in dealing with the last type. Missing patterns can be illustrated in table manner. Furthermore, the VIM package provides many functions for graphical presentations of missing data. Relationships between missing data and values of other variables provide further insights into mechanisms underlying the missing data. This can be explored by using correlation analysis.

Cite this article as: Zhang Z. Missing data exploration: highlighting graphical presentation of missing pattern. Ann Transl Med 2015;3(22):356. doi: 10.3978/j.issn.2305-5839.2015.12.28

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Kabacoff R. R in Action. Shelter Island: Manning Publications Co., 2011.
2. Montez-Rath ME, Winkelmayer WC, Desai M. Addressing missing data in clinical studies of kidney diseases. Clin J Am Soc Nephrol 2014;9:1328-1335.
3. Dziura JD, Post LA, Zhao Q, et al. Strategies for dealing with missing data in clinical trials: from design to analysis. Yale J Biol Med 2013;86:343-358.
4. Buuren SV, Groothuis-Oudshoorn K. mice: Multivariate Imputation by Chained Equations in R. Journal of Statistical Software 2011;45:1-67.
5. Templ M, Alfons A, Kowarik A, et al. Package VIM: Visualization and Imputation of Missing Values (2013). R package version 3.0. 3.1.

Reshaping and aggregating data: an introduction to reshape package

Zhongheng Zhang

Abstract: It is common that data format extracted from clinical database does not meet the requirement of statistical analysis. In clinical research, variables are frequently measured repeatedly over the follow-up period. Such data can be displayed either in wide or long format. Transformation between these 2 forms can be challenging by hand. Fortunately, there are sophisticated packages in R environment. Data frame should firstly be melted and then casted to a format that you want. Aggregation over unique combination of id variables is also allowable. Additionally, the article also introduces 2 functions colsplit() and funstofun() that can be useful in some circumstances.

Keywords: Aggregating; reshape package; R

Submitted Dec 20, 2015. Accepted for publication Jan 09, 2016.
doi: 10.3978/j.issn.2305-5839.2016.01.33
View this article at: http://dx.doi.org/10.3978/j.issn.2305-5839.2016.01.33

Introduction

In clinical studies involving electronic health records, variables are frequently grouped by one or more other variables (1). For example, when you follow up stroke patients for their quality of life as measured by some certain scales, the scales are recorded at each visit. In other words, these scales are nested within each patient. The display of such data can adopt long or wide format. The former listed each visit scale longitudinally in a column and a variable denoting patient identification is mandatory. One patient can take up several rows in long format. On the other hand, the wide format displays 1 patient per row, and each visit scale is listed consecutively in a single row. Both formats have their advantages and disadvantages depending on the purpose of analysis. These are simple examples illustrating the importance of data shape during analysis, and some are far more complex. This article introduces a powerful R package named "reshape", which is able to handle a variety of data formats (2).

Working example

Suppose we have 3 patients, and each of them has blood partial pressure of oxygen measured on daily basis for 3 days. Blood oxygen can be measured from arterial line (*PaO2*) and central venous line (*PcvO2*).

```
> id<-rep(1:3,each=3)
> time<-rep(1:3,3)
> PaO2<-round(rnorm(9,mean=70,sd=10))
> PcvO2<-round(rnorm(9,mean=40,sd=8))
> data<-data.frame(id,time,PaO2,PcvO2)
> data
```

	id	time	PaO2	PcvO2
1	1	1	78	42
2	1	2	86	36
3	1	3	72	39
4	2	1	89	37
5	2	2	81	47
6	2	3	66	36
7	3	1	65	30
8	3	2	60	46
9	3	3	88	30

Oxygen content is significantly lower in the central vein than that in the artery, which is consistent with the common sense that arterial oxygenation is much greater than venous oxygenation.

Melting a dataset

The melt() function converts a wide format into long format. A number of variables listed in columns can be stacked into a single column. As in our example, both venous and arterial partial pressures can be stacked in a single column after application of the melt() function. This function requires an *id* variable and variables of interests to be stacked. The generic form of melt() function is like this:

```
melt(data, id.vars, measure.vars,
     variable_name = "variable",
    na.rm = !preserve.na, preserve.na = TRUE, ...)
```

If either *id.vars* or *measure.vars* is specified, the function takes the remainder variable in the data frame belong to the other. If neither is specified, the function assumes the character and factor variable as the *id.vars*, and the remainders are *measure.vars*. To avoid confusion, you'd better specify both of them. The "variable_name" argument specifies the name of a new variable that will be created to store stacked variables. Now let's take a close look at how melt() works by using our working example.

```
> data.melt<-melt(data, id=(c("id", "time")),
measure.vars=(c("PaO2","PcvO2")),
variable_name="PO2")
> data.melt
```

	id	time	PO2	value
1	1	1	PaO2	78
2	1	2	PaO2	86
3	1	3	PaO2	72
4	2	1	PaO2	89
5	2	2	PaO2	81
6	2	3	PaO2	66
7	3	1	PaO2	65
8	3	2	PaO2	60
9	3	3	PaO2	88
10	1	1	PcvO2	42
11	1	2	PcvO2	36
12	1	3	PcvO2	39
13	2	1	PcvO2	37
14	2	2	PcvO2	47
15	2	3	PcvO2	36
16	3	1	PcvO2	30
17	3	2	PcvO2	46
18	3	3	PcvO2	30

Both *id* and *time* are *id.vars* and thus they remain unchanged. PaO2 and PcvO2 are stacked into a single column and a new variable called "PO2" is added to distinguish between arterial and venous oxygen. This melted format is not only useful for statistical analysis but also helpful in reshaping and aggregating data.

Casting a data frame

The cast() function contained in reshape package works on melted dataset. It transforms long data into wide format and can aggregate variable within any combinations of id variables. The generic form of cast() function takes the following form:

```
cast(data, formula = ... ~ variable, fun.aggregate=NULL,
...,
     margins=FALSE, subset=TRUE, df=FALSE,
     fill=NULL, add.missing=FALSE,
     value = guess_value(data))
```

the argument data is a melted dataset. The cast formula has the format:

```
x_variable + x_2 ~ y_variable + y_2 ~ z_variable ~ ... |
list_variable + ...
```

The x variables in combination define the rows, and y variables in combination define the columns. If a set of x variables does not uniquely identify a row, the *fun.aggregate* argument should be given. Then the aggregate function can be applied to rows identified by a certain combination of x variables. List variables and z variables are usually not required. We don't have dataset of that complex! Next, I will show how to cast the melted data to different forms.

```
> cast(data.melt,id~PO2,mean)
```

	id	PaO2	PcvO2
1	1	78.66667	39.00000
2	2	78.66667	40.00000

```
3     3       71.00000       35.33333
```

The rows are defined by the *id* variable on the left side of the formula, and columns are defined by *PO2* variable. There are 2 levels contained in the *PO2*, thus we obtain 2 columns. Because the *id* variable does not uniquely identify a row, function mean is applied to vectors identified by *id* variable.

```
> cast(data.melt,time~PO2,mean)
      time      PaO2      PcvO2
1     1     77.33333  36.33333
2     2     75.66667  43.00000
3     3     75.33333  35.00000
```

When *id* is replaced by *time* in the above codes, the row represents mean value across *id* variable.

```
> cast(data.melt,id+time~PO2)
      id    time    PaO2    PcvO2
1     1     1       78      42
2     1     2       86      36
3     1     3       72      39
4     2     1       89      37
5     2     2       81      47
6     2     3       66      36
7     3     1       65      30
8     3     2       60      46
9     3     3       88      30
```

Variables *id+time* on the left side of the formula define the rows. Each unique combination of id and time defines a row. Columns are in line with the levels of *PO2* variable. Because the 3 variables have identified a unique row, aggregating function is no longer applicable.

```
> cast(data.melt, id ~ time+PO2,
subset=time < 3 & id < 3)
      id    1_PaO2    1_PcvO2    2_PaO2    2_PcvO2
1     1     78        42         86        36
2     2     89        37         81        47
```

Data can be subset before reshaping. In the above example, we restrict subset of data with *time*<3 and *id*<3.

```
> cast(data.melt,id~time~PO2)
, , PO2 = PaO2

   time
id      1       2       3
1       78      86      72
2       89      81      66
3       65      60      88

, , PO2 = PcvO2

   time
id      1       2       3
1       42      36      39
2       37      47      36
3       30      46      30
```

When a *z* variable is applied, we can see that the melted data is splitted into 2 datasets by variable *PO2*. Each item can be directly called by using the following code:

```
> cast(data.melt, id~time| PO2)$PaO2
      id    1       2       3
1     1     78      86      72
2     2     89      81      66
3     3     65      60      88
```

Row and column margins can be calculated with following code.

```
> cast(data.melt, time ~ PO2, mean,
margins=c("grand_row", "grand_col"))
      time      PaO2      PcvO2      (all)
1     1     77.33333  36.33333  56.83333
2     2     75.66667  43.00000  59.33333
3     3     75.33333  35.00000  55.16667
4     (all) 76.11111  38.11111  57.11111
```

Split character vector into multiple columns

The function colsplit() in reshape package is to split

character vector into multiple columns on certain expression. This can be helpful in handling a list of variable names. Suppose that we have 3 laboratory items measured on 3 consecutive days. Their variable names can be denoted by: lac_1, lac_2, lac_3, wbc_1, wbc_2, wbc_3, hb_1, hb_2 and hb_3. In analysis, you want to list laboratory values in a column, and the type of value and measurement days are denoted by separate variables.

```
> data.split<-data.frame(lac_1=2.3, lac_2=3.4,
lac_3=4.5, wbc_1=12, wbc_2=11, wbc_3=6, hb_1=60,
hb_2=77, hb_3=89)
> variable.name<-colsplit(names(data.split),
"_",c("lab","days"))
> data.reshape<-cbind(variable.name,t(data.split))
> row.names(data.reshape)<-NULL
> names(data.reshape)[3]<-"value"
> data.reshape
```

	lab	days	value
1	lac	1	2.3
2	lac	2	3.4
3	lac	3	4.5
4	wbc	1	12.0
5	wbc	2	11.0
6	wbc	3	6.0
7	hb	1	60.0
8	hb	2	77.0
9	hb	3	89.0

The first line creates a data frame that is typically encountered in practice. All variable names are composed of type of measurement and day, and the latter two are separated by "_". In such case, colsplit() can be used to separate the variable names into 2 columns, with each representing the type of laboratory measurement and the day, respectively. Next, values of measurements are added by cbind() function. The following lines rename the variable names to make them easy to understand.

Producing baseline characteristics of cohort automatically

In big-data clinical study, there are numerous covariates under consideration. The first step is usually to take a look at these variables one by one. Suppose you want to have a look at the mean, median, range, and standard deviation of a variable. The traditional way is to execute functions one by one. Alternatively, you can combine these functions into a single one.

```
> round(funstofun(mean, median,min,max,sd)
(data$PaO2),1)
```

mean	median	min	max	sd
76.1	78.0	60.0	89.0	10.8

```
> round(funstofun(mean, median,min,max,sd)
(data$PcvO2),1)
```

mean	median	min	max	sd
38.1	37.0	30.0	47.0	6.1

Sometimes, the *summary()* function contained in the base R package can fulfill the task of general description of variables. However, the output parameters are fixed. The *funstofun()* function overcomes this limitation that functions can be flexibly adapted to the needs of a specific purpose.

Summary

The article provides a gentle introduction to data reconstruction and aggregating. It is common that the format of data output from case report form (CRF) does not meet the purpose of statistical analysis. In clinical research, variables are frequently measured repeatedly over the follow-up period. Such data can be displayed either in wide or long format. Transformation between these 2 forms can be challenging by hand. Fortunately, there are sophisticated packages in R environment. Data frame should firstly be melted and then casted to format that you want. Aggregation over unique combination of id variables is also allowable. Additionally, the article also introduces 2 functions colsplit() and funstofun() that may be useful in some situations.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Twisk JW. Applied longitudinal data analysis for epidemiology: a practical guide. Second edition. Cambridge, England: Cambridge University Press, 2013:321.
2. Wickham H. Reshaping data with the reshape package. Journal of Statistical Software 2007;21:1-20.

Cite this article as: Zhang Z. Reshaping and aggregating data: an introduction to reshape package. Ann Transl Med 2016;4(4):78. doi: 10.3978/j.issn.2305-5839.2016.01.33

Missing data imputation: focusing on single imputation

Zhongheng Zhang

Abstract: Complete case analysis is widely used for handling missing data, and it is the default method in many statistical packages. However, this method may introduce bias because some useful information is omitted from the analysis. Therefore, many imputation methods are developed to overcome this problem. The present article focuses on single imputation. Imputations with mean, median and mode are simple but, like complete case analysis, can introduce bias on mean and deviation. Furthermore, they ignore relationships with other variables. Regression imputation can preserve relationship between missing values and other variables. There are many sophisticated methods to handle missing values in longitudinal data. This article focuses primarily on how to perform single imputation, while avoiding complex mathematical calculations.

Keywords: Big-data clinical trial; missing data; single imputation; longitudinal data; R

Submitted Nov 18, 2015. Accepted for publication Dec 08, 2015.
doi: 10.3978/j.issn.2305-5839.2015.12.38
View this article at: http://dx.doi.org/10.3978/j.issn.2305-5839.2015.12.38

Introduction

Missing data are ubiquitous in big-data clinical trials. Although many studies do not explicitly report how they handle missing data (1,2), some implicit methods are used in statistical softwares. As a result, different packages may handle missing data in different ways (or the default methods are different) and results may not be reproducible by using different statistical software packages. Sometimes this may not lead significantly different results, but the scientific soundness of the study is compromised. The best practice is to explicitly state how missing values are handled. For simplicity, many investigators simply delete incomplete case (listwise deletion), which is also the default method in many regression packages (3). This method gets reliable results only when the number of missing values is not large and the missing pattern is missing completely at random (MCAR) or missing at random (MAR). Another disadvantage of complete case analysis is information loss. This can be a big problem when there are a large number of variables (columns). A substantial number of cases can be deleted because deletion is based on missingness on one or more variables. Furthermore, complete case analysis can lead to unpredictable bias (3-5). The solution to this problem is imputation. Missing values are replaced by imputed values. Since imputation is an area of active research, there are numerous methods and packages developed for imputation. This article intends to introduce some basic imputation methods for missing data. Multiple imputations will be discussed in the following articles of the big-data clinical trial series.

Dataset simulation

A dataset of 150 observations is created by simulation. The dataset is used for illustration purpose and there is no clinical relevance. There are three variables which are sex, mean arterial blood pressure (map) and lactate (lac). In each simulation, I set a seed allowing readers to replicate the results.

```
> set.seed(12365)
> sex<-rbinom(150, 1, 0.45)
> sex[sex==1]<-"male"
> sex[sex==0]<-"female"
> set.seed(123567)
> sex.miss.tag<-rbinom(150, 1, 0.3) #MCAR
> sex.miss<-ifelse(sex.miss.tag==1,NA,sex)
> set.seed(124564)
> map<-round(abs(rnorm(150, mean = 70, sd = 30)))
```

```
> map<-ifelse(map<=40,map+30,map)
> set.seed(12456)
> lac<- rnorm(150, mean = 5, sd = 0.7) -map*0.04
> lac<-abs(round(lac,1))
> set.seed(134567)
> lac.miss.tag<-rbinom(150, 1, 0.3)
> lac.miss<-ifelse(lac.miss.tag==1,NA,lac)
> data<-data.frame(sex.miss,map,lac.miss)
```

In the dataset, lac is created to have correlation with map. Serum lactate is a reflection of tissue perfusion, and the latter is dependent on mean arterial pressure. A negative correlation coefficient is assumed for map ~ lac relationship. In order to add noise, the intercept is generated by using random number generator [rnorm() function]. Sex is generated in an assumption of MCAR.

```
> sd(lac.miss,na.rm=TRUE)
[1] 1.105589
> summary(lac.miss)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
0.100	1.200	2.100	2.051	2.800	4.600	47

There are 47 missing values in the lac variable. The standard deviation is 1.11 and the mean is 2.051.

```
>library(car)
>scatterplot(lac ~ map | lac.miss.tag, lwd=2,
            main="Scatter Plot of lac vs. map by #
            missingness",
            xlab="Mean Aterial Pressure (mmHg)",
            ylab="Lactate (mmol/l)",
            legend.plot=TRUE,
            id.method="identify",
            boxplots="xy"
            )
```

Figure 1 is the scatter plot of lac versus map and missing values on lac is denoted by red triangle. Black and red curves are fitted by nonparametric-regression smooth for nonmissing and missing values, respectively. It is noted that missing values on lac distribute evenly across lac range and is independent of the variable map. This is in consistent with the MCAR.

Rough estimation of missing values with mean, mode or median

A quick approach to missing values is to replace them with mean, median or mode. The initialise() function shipped with VIM package can be used for this purpose. However, it is primarily used internally by some imputation algorithms and has no advantage over other basic methods in performing simple imputation. Suppose we want to impute missing values in data by mean for numeric variables and by mode for categorical variables.

```
> lac.mean<-round(ifelse(is.na(lac.miss),
mean(lac.miss,na.rm=TRUE),lac.miss),1)
```

Next, you can take a look at how the imputed values fill the lac ~ map scatter plot.

```
> scatterplot(lac.mean ~ map | lac.miss.tag, lwd=2,
            main="Scatter Plot of lac vs. map by #
            missingness",
            xlab="Mean Aterial Pressure (mmHg)",
            ylab="Lactate (mmol/l)",
            legend.plot=TRUE, smoother=FALSE,
            id.method="identify",
            boxplots="xy"
            )
```

It is noted that all imputed values are at the mean lac value of 2.1 mmol/L (*Figure 2*). The mean and standard deviation are biased. Imputations with mode and median work in the same manner and they are left to readers for practice. Although rough imputation provides fast and simple methods for missing values, it underestimates variance, compromises relationship between variables, and biases summary statistics. Thus rough imputations can only be used when a handful of values are missing, they are not recommended for general use.

Regression imputation

Imputation with regression on other one or more variables

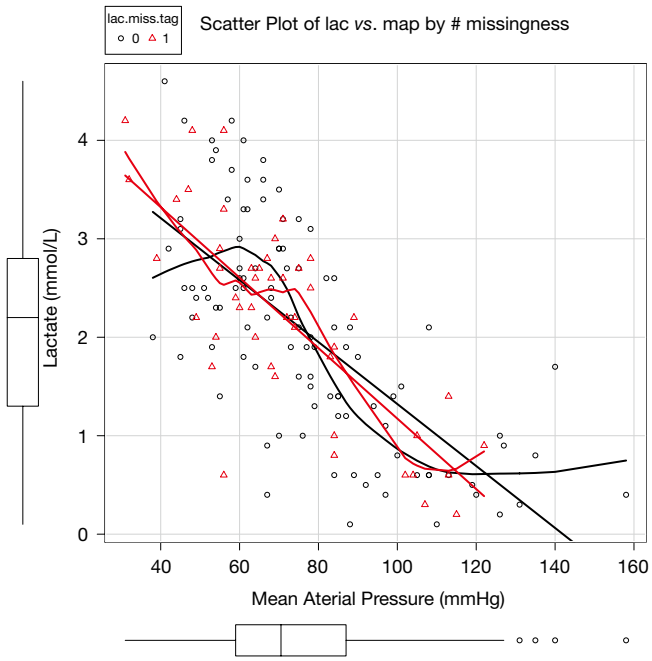


Figure 1 Scatter plot of lac *vs.* map and missing values on lac is denoted by red triangle.

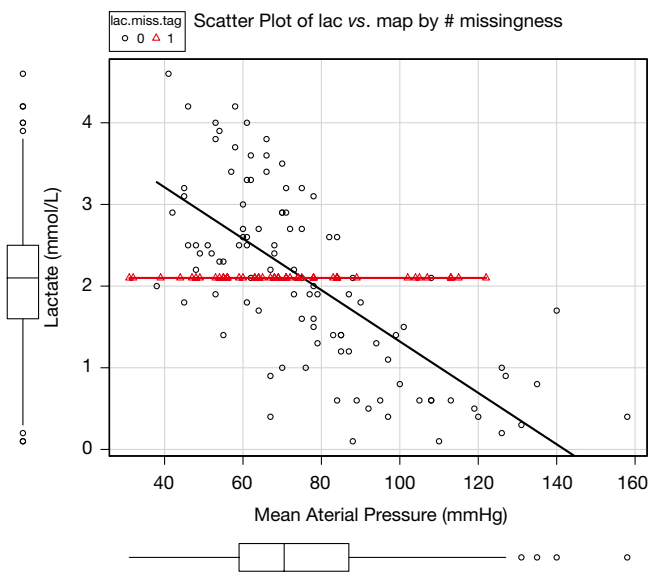


Figure 2 Scatter plot of lac *vs.* map with missing values on lac replaced by the mean value of observed lac.

may produce smarter values. Firstly, investigators need to fit a regression model by setting the variable of interest as response variable and other relevant variables as covariates.

The coefficients are estimated, and then missing values can be predicted by the fitted model. Take the dataset for example, one can build a linear regression model between lac and map. Thereafter, missing values on lac can be predicted by the fitted model equation.

```
> fit <- lm(lac.miss ~ map, data = data)
> lac.pred <- predict(fit,newdata=data)
> lac.regress<-round(ifelse(is.na(lac.miss),
lac.pred,lac.miss),1)
> scatterplot(lac.regress ~ map | lac.miss.tag,
lwd=2,
                main="Scatter Plot of lac vs. map by #
                missingness",
                xlab="Mean Aterial Pressure(mmHg)",
                ylab="Lactate (mmol/l)",
                legend.plot=TRUE, smoother=FALSE,
                id.method="identify",
                boxplots="xy"
                )
```

The estimated values are on the regression line without noise (*Figure 3*). This looks more rational than that estimated with mean. However, this method increases correlation coefficients between map and lac. The variability of imputed data is underestimated. Alternatively, you can add some noises to the regression by using mice() function (6).

```
> library(mice)
> imp <- mice(data[, 2:3], method = "norm.nob",m = 1,
                    maxit = 1, seed = 123456)
> lac.stoc<-complete(imp, action = 1, include =
FALSE)$lac.miss
> scatterplot(lac.stoc ~ map | lac.miss.tag, lwd=2,
                main="Scatter Plot of lac vs. map by #
                missingness",
                xlab="Mean Aterial Pressure (mmHg)",
                ylab="Lactate (mmol/l)",
                legend.plot=TRUE, smoother=FALSE,
                id.method="identify",
                boxplots="xy"
                )
```

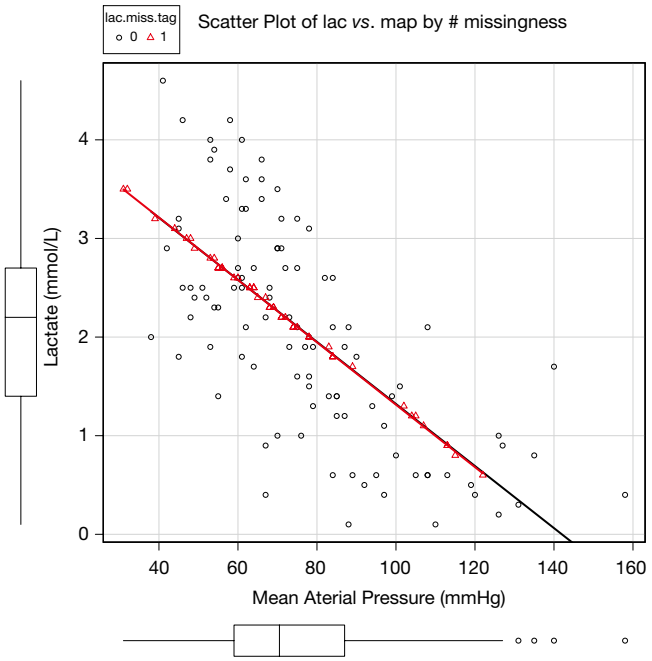



Figure 3 Scatter plot of lac *vs.* map with missing values on lac replaced by values predicted by fitted regression model.

The core of the mice() function is the method="norm.nob" argument which first estimates the slope, intercept and residual variance with linear regression, then predicts missing values with these specifications. The addition of residual variance opens up the distribution of imputed values (e.g., they are not exactly on the regression line) (*Figure 4*). However, the limitation is that one imputed value falls below zero, which is practically impossible.

Indicator method

Indictor method is an alternative to deal with missing values. This method replaces missing data by zero, and can be easily done by modifying the previous R code. I leave it to your practice. Indicator method has once been popular because it is simple and retains the full dataset. On the other hand, it allows for systematic difference between observed and unobserved data. However, indicator method is criticized that it can bring unpredictable bias into regression model, even with small percentage of missing values (4). Some authors have argued against its use in general practice (7).

Imputation of longitudinal data

The function imputation() shipped with longitudinal data package provides powerful algorithm for the imputation of longitudinal data (8). Longitudinal data are characterized by correlation between repeated measurements for a certain variable. Thus, missing values imputed depending on neighboring values are more reliable than methods mentioned above. For example, for a given patients, his or her serum lactate levels are correlated in consecutive measurements.

Suppose we have four patients and serum lactate levels are measured on a daily basis. However, there are many missing values. R code for creating the dataset is shown below.

```
> matMissing <- matrix(
   c(NA,1.8,NA,2.3,2.2,NA,1.4,NA,NA,1.1,
    9.4,8.4,NA,9.6,7.7,NA,8.1,NA,7.9,NA,
    3.1,NA,4,3.3,3.1,3.4,2.4,3,NA,2.1,
     5.1,4,5.6,NA,NA,4.1,4.4,NA,NA,6.2
   ),4,byrow=TRUE
)
```

The first step in analyzing such dataset is to estimate the missing values. Since they are longitudinal data, it is reasonable that missing values are correlated to their immediate observed values. However, there are many methods for the imputation. Longitudinal imputation uses non-missing data of the same subject to estimate missing values. The imputation is independent of other individual subjects or cases. There are also a variety of methods for the longitudinal imputation (*Table 1*) (9-11). In the present article, I want to show several simple methods for the imputation of longitudinal data. Readers interested in more complex methods can refer to the reference (9).

```
> library(longitudinalData)
> par(mfrow=c(2,2))
> matplot(t(imputation(matMissing,"crossMean")),
              type="b",ylim=c(0,10),
              lty=1,col=1,main="crossMean",
              ylab="Lactate values (mmol/L)")
```


Figure 4 Missing values are predicted by linear regression. Note that residual variance is added to reflect uncertainty in estimation.

```
> matlines(t(matMissing),type="o",
col=2,lwd=3,pch=16,lty=1)
> matplot(t(imputation(matMissing,"trajMean")),
                type="b",ylim=c(0,10),
                ylab="",
                lty=1,col=1,main="trajMean")
> matlines(t(matMissing),type="o",
col=2,lwd=3,pch=16,lty=1)
> matplot(t(imputation(matMissing,
"linearInterpol.locf")),
                type="b",ylim=c(0,10),
                lty=1,col=1,main="linearInterpol.locf",
                xlab="Measurement time points",
                ylab="Lactate values (mmol/L)")
> matlines(t(matMissing),type="o",col=2,
lwd=3,pch=16,lty=1)
> matplot(t(imputation(matMissing,
"copyMean.locf")),
                type="b",ylim=c(0,10),
                lty=1,col=1,main="copyMean.locf",
                xlab="Measurement time points",
                ylab="")
> matlines(t(matMissing),type="o",col=2,
lwd=3,pch=16,lty=1)
```

The par() function is powerful in setting R graphical parameters. The mfrow=c(2,2) argument specifies that subsequent figures will be drawn in a two-by-two array on the device by row. In order to illustrate how each imputation method works, I plot observed and imputed lactate measurements on graphics by using matplot() function. Imputation methods are carried out by the imputation() function. The first argument specifies the matrix of trajectory to impute. The second argument specifies the name of the imputation method. In the example I used "crossMean", "trajMean", "linearInterpol.locf" and "copyMean.locf". Different methods resulted in different imputed values (*Figure 5*). To distinguish observed values from those which are imputed, the matlines() function was used to highlight observed values with red points and lines.

Summary

Missing data are ubiquitous in big-data clinical trials. Some investigators use the method of complete case analysis and this can get reliable results when missing values are at random and the proportion is not large. However, it is common that complete case analysis may result in information attrition when there are many variables. Imputation is an alternative that can help to obtain reliable results. This article introduces some simple imputation methods. Mean, median and mode imputations are simple, but they underestimate variance and ignore the relationship with other variables. Regression method can preserve their correlation with other variables but the variability of missing values is underestimated. Variability can be adjusted by adding random errors to the regression model. Indicator method is to replace missing values with zeros, which is not recommended for general use. Longitudinal data are special and there are many methods exist for their imputations. This is an area of active research and it is controversial on which method is the best. Based on simulation study, the copy mean method may be a good choice (9).

Table 1 Imputation methods for longitudinal data

Imputation methods	Brief description
Cross sectional imputation	
Cross mean	Replace missing value with mean of values observed at that time
Cross median	Replace missing value with median of values observed at that time
Cross hot deck	Replace missing value with a randomly chosen value among values observed at that time
Longitudinal imputation	
Traj mean	Replace missing value by average values of that subject (trajectory)
Traj median	Replace missing value by median value of that subject (trajectory)
Traj hot deck	Replace missing value by a value chosen randomly from that subject (trajectory)
LOCF	Replace missing value by previous non-missing value of that subject (trajectory)
Linear interpolation	Values immediately surrounding the missing are join by a line
Spline interpolation	Values immediately surrounding the missing are joined by a cubic spline
Cross and longitudinal imputation	
Copy mean	Combine linear interpolation and imputation using population's mean trajectory
Linear regression	Predict missing value by constructing a model

LOCF, last occurrence carried forward.

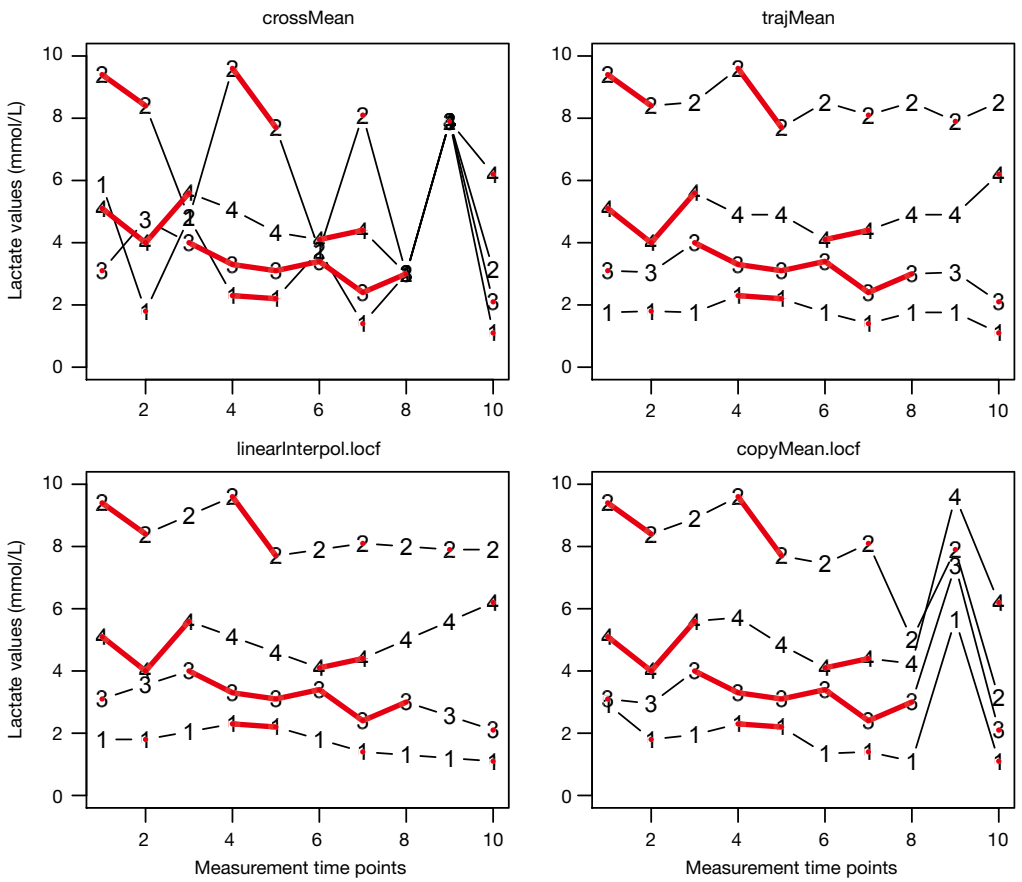


Figure 5 Longitudinal imputations with different methods.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Wood AM, White IR, Thompson SG. Are missing outcome data adequately handled? A review of published randomized controlled trials in major medical journals. Clin Trials 2004;1:368-376.
2. Bell ML, Fiero M, Horton NJ, et al. Handling missing data in RCTs; a review of the top medical journals. BMC Med Res Methodol 2014;14:118.
3. Demissie S, LaValley MP, Horton NJ, et al. Bias due to missing exposure data using complete-case analysis in the proportional hazards regression model. Stat Med 2003;22:545-557.
4. Knol MJ, Janssen KJ, Donders AR, et al. Unpredictable bias when using the missing indicator method or complete case analysis for missing confounder values: an empirical example. J Clin Epidemiol 2010;63:728-736.
5. Masconi KL, Matsha TE, Erasmus RT, et al. Effects of different missing data imputation techniques on the performance of undiagnosed diabetes risk prediction models in a mixed-ancestry population of South Africa. PLoS One 2015;10:e0139210.
6. Buuren SV, Groothuis-Oudshoorn K. mice: Multivariate Imputation by Chained Equations in R. Journal of Statistics Software 2011;45:1-67.
7. van der Heijden GJ, Donders AR, Stijnen T, et al. Imputation of missing values is superior to complete case analysis and the missing-indicator method in multivariable diagnostic research: a clinical example. J Clin Epidemiol 2006;59:1102-1109.
8. Genolini C. longitudinalData: Longitudinal Data. Available online: https://cran.r-project.org/web/packages/longitudinalData/longitudinalData.pdf
9. Genolini C, Écochard R, Jacqmin-Gadda H. Copy Mean: A New Method to Impute Intermittent Missing Values in Longitudinal Studies. Open Journal of Statistics 2013;3:26-40.
10. Twisk J, de Vente W. Attrition in longitudinal studies. How to deal with missing data. J Clin Epidemiol 2002;55:329-337.
11. Engels JM, Diehr P. Imputation of missing longitudinal data: a comparison of methods. J Clin Epidemiol 2003;56:968-976.

Cite this article as: Zhang Z. Missing data imputation: focusing on single imputation. Ann Transl Med 2016;4(1):9. doi: 10.3978/j.issn.2305-5839.2015.12.38

Multiple imputation with multivariate imputation by chained equation (MICE) package

Zhongheng Zhang

Abstract: Multiple imputation (MI) is an advanced technique for handing missing values. It is superior to single imputation in that it takes into account uncertainty in missing value imputation. However, MI is underutilized in medical literature due to computational challenges and the lack of familiarity to clinical investigators. The article provides a step-by-step approach to perform MI by using R multivariate imputation by chained equation (MICE) package. The procedure firstly imputes m sets of complete dataset by calling mice() function. Then statistical analysis, such as univariate analysis and multivariable regression analyses, can be performed within each dataset by calling with() function. This function sets the environment for statistical analysis. Lastly, the results obtained from each analysis are combined by using the pool() function.

Keywords: Big-data clinical trial; multiple imputation (MI); multivariate imputation by chained equation (MICE) package; R; imputed complete dataset

Submitted Nov 05, 2015. Accepted for publication Dec 15, 2015.
doi: 10.3978/j.issn.2305-5839.2015.12.63
View this article at: http://dx.doi.org/10.3978/j.issn.2305-5839.2015.12.63

Introduction

Multiple imputation (MI) is an advanced method in handling missing values. In contrast to single imputation, MI creates a number of datasets (denoted by *m*) by imputing missing values. That is, one missing value in original dataset is replaced by *m* plausible imputed values. These values take imputation uncertainty into consideration. Statistics of interest are estimated from each dataset and then combined into a final one. While single imputation has been criticized for its bias (e.g., overestimation of precision) and ignorance of uncertainty about the estimation of missing values, MI, if performed properly can give an accurate estimate of the real result (1). However, MI is underutilized in medical literature due to computational challenges and the lack of familiarity to clinical investigators. To make clinicians become familiar with MI, the present article provides a step-by-step tutorial to the use of R package to conduct MI for missing values. Before that, a brief description of basic ideas behind MI will be given.

Fundamentals of MI

MI procedure replaces each missing value with multiple possible values. Compared with single imputation, this procedure takes into account uncertainty behind missing value estimation. The procedure produces several dataset from which parameters of interest can be estimated. For example, if you are interested in coefficient for a covariate in a multivariable model, the coefficients will be estimated from each dataset resulting in *m* number of coefficients. Finally, these coefficients are combined to give an estimate of the coefficient, taking into account uncertainty in the estimation of missing values. The variance of coefficient estimated in this way is less likely to be underestimated as compared with single imputation.

The imputation procedure is carried out by firstly creating a prediction model for a target variable with missing values from all other variables (*Figure 1*). In other words, the variable under imputation is the response variable and other relevant variables are independent variables. By default, predictive mean matching is used for continuous variables and logistic regression is used for dichotomous variables (2). Variables included into imputation are advised to be (I) predictive of missingness, (II) associated with the variable being imputed and (III) the outcome variable of the current analysis (3,4).

Working example

To establish a working example for illustration, I borrow the scenario from one of my recent research projects which investigated the relationship between lactate and mortality outcome (5). While the research was conducted by using MIMIC-II database involving >30,000 patients (6), the working example will artificially generate 150 patients with simulation. There are roughly 30% missing values in the *lac* variable.

```
> set.seed(12365)
> sex<-rbinom(150, 1, 0.45)
> sex[sex==1]<-"male"
> sex[sex==0]<-"female"
> set.seed(123567)
> sex.miss.tag<-rbinom(150, 1, 0.3) #MCAR
> sex.miss<-ifelse(sex.miss.tag==1,NA,sex)
> set.seed(124564)
> map<-round(abs(rnorm(150, mean = 70, sd = 30)))
> map<-ifelse(map<=40,map+30,map)
> set.seed(12456)
> lac<- rnorm(150, mean = 5, sd = 0.7) -map*0.04
> lac<-abs(round(lac,1))
> set.seed(134567)
> lac.miss.tag<-rbinom(150, 1, 0.3)
> lac.miss<-ifelse(lac.miss.tag==1,NA,lac)
> set.seed(111)
> mort<-rbinom(150, 1, 0.25)
> mort[mort==1]<-"dead"
> mort[mort==0]<-"alive"
> data<-data.frame(sex.miss,map,lac.miss,mort)
```

MI with multivariate imputation by chained equation (MICE) package

R provides several useful packages for MI. The commonly used packages include Amelia, MICE (7), and MI. In this article I will introduce how MICE works by using the simulated dataset.

```
> library(Rcpp) #Rcpp package is mandatory
> library(mice)
> imp <- mice(data, seed=12345)
```

The first argument in mice() function is a data frame containing missing variables. I set a seed in the second place to make sure that readers can replicate the result. The result produced by mice() is stored in a list object *imp* which contains *m* imputed dataset, along with relevant information on how imputations are performed. You can take a look at the contents in *imp*.

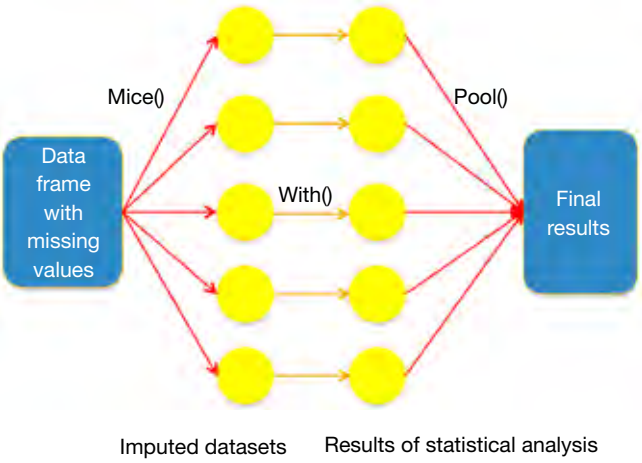


Figure 1 Schematic illustration of how MICE package works with data frame with missing values. Note the sequential use of mice(), with() and pool() functions.

```
> imp
Multiply imputed data set
Call:
mice(data = data, seed = 12345)
Number of multiple imputations: 5
Missing cells per column:
sex.miss      map  lac.miss     mort
      43        0       47        0
Imputation methods:
sex.miss      map  lac.miss     mort
"logreg"       ""    "pmm"       ""
VisitSequence:
sex.miss lac.miss
       1        3
PredictorMatrix:
         sex.miss map lac.miss mort
sex.miss        0   1        1    1
map             0   0        0    0
lac.miss        1   1        0    1
mort            0   0        0    0
Random generator seed value: 12345
```

In this call to function mice(), five datasets are imputed.

There are 43 and 47 missing values in variables *sex.miss* and *lac.miss*, respectively. The imputation method for dichotomous variable *sex.miss* is logistic regression and for continuous variable *lac.miss* is predictive mean matching. Visit sequence tells you the column order to impute data during one pass through the data. Visit sequence can be defined by the argument *visitSequence =(1:ncol(data))* [*apply(is.na(data), 2, any*]. In the working example, *sex.miss* is first imputed, followed by *lac.miss*. Predictor matrix contains 0/1 data specifying which variables are used to predict a target variable. Row indicates target variables (variables to be imputed). A value "1" means that the column variable is used to predict row variable. For instance, the first row indicates that column variables *map*, *lac.miss* and *mort* are used to predict *sex.miss*. No variable is used to predict *map* and *mort* because neither of them contains missing value. You can choose any set of variables used for prediction by specifying the *predictorMatrix =(1-diag(1, ncol(data)))* argument. If you do not want to use *mort* to predict *sex.miss*, change the predictor matrix as follows:

```
> predmatrix<-1-diag(1, ncol(data))
> predmatrix[c(2,4),]<-0
> predmatrix[1,4]<-0
> predmatrix
     [,1] [,2] [,3] [,4]
[1,]    0    1    1    0
[2,]    0    0    0    0
[3,]    1    1    0    1
[4,]    0    0    0    0
```

Imputations for a particular variable can be viewed in the following way. I use the head() function to save space (otherwise there will be 47 rows).

```
> head(imp$imp$lac.miss)
   1   2   3   4   5
2 3.9 2.3 3.0 1.9 2.7
5 3.6 1.8 3.5 4.0 3.3
8 1.0 1.9 2.1 1.0 0.5
11 1.1 1.8 0.1 0.6 0.1
17 2.9 3.7 1.9 2.2 0.4
20 3.8 3.8 2.5 4.0 4.2
```

In this matrix you can have a look at what mice() has imputed in each imputation. Recall that you have not specified the number of imputations and the default is 5. The first row indicates the row number of missing values in the original dataset. If the matrix contains negative values, you may need to review the imputation method. You can also view each of the five complete dataset by following code. Again head() is used to save space. The argument *action=4* specifies that the forth imputation is visited.

```
> head(complete(imp, action=4))
  sex.miss map lac.miss mort
1     male 126      0.2 alive
2     male  55      1.9 alive
3   female  89      0.6 alive
4   female 158      0.4 alive
5   female  63      4.0 alive
6   female  48      2.5 alive
```

Statistical analysis after imputation

Now that you have five complete datasets at hand, conventional statistical analysis can be performed. In observational studies, the first step is usually to perform univariate analysis to find out which variable is associated with the outcome of interest. The following example illustrates how to perform t test and multivariable regression analysis.

```
> ttest<-with(imp,t.test(lac.miss~mort))
> ttest
call :
with.mids(data = imp, expr = t.test(lac.miss ~ mort))
call1 :
mice(data = data, seed = 12345)
nmis :
sex.miss      map lac.miss     mort
      43        0       47        0
analyses :
[[1]]

        Welch Two Sample t-test

data:  lac.miss by mort
t = 0.61547, df = 59.653, p-value = 0.5406
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3059795  0.5779093
sample estimates:
mean in group alive
           2.135965

                [...output deleted to save space...]
```

The generic form of with() function is *with(data, expr)*. It allows an R expression to be executed in an environment

from data. The assignments within expression take place within data environment instead of the user's workspace. In our example, the t.test() function takes place within imputed complete datasets contained in the object *imp*. Otherwise, if with() is not called the t.test() will use original incomplete dataset existing in the workspace. The above output only displays analysis of the first imputed dataset and the remaining four are omitted to save space.

Next I want to find out variables associated with mortality outcome by using logistic regression model. Note that logistic regression model is specified using the glm() function.

```
> fit<-with(imp,glm(mort~sex.miss+map+lac.miss,family = binomial))
> fit
call :
with.mids(data = imp, expr = glm(mort ~ sex.miss + map + lac.miss,
        family = binomial))
call1 :
mice(data = data, seed = 12345)
nmis :
sex.miss        map        lac.miss        mort
43              0          47              0
analyses :
[[1]]
Call:  glm(formula = mort ~ sex.miss + map + lac.miss, family = bino-
mial)
Coefficients:
(Intercept)          sex.miss2          map
0.99982              -0.30750           -0.01772
Degrees of Freedom: 149 Total (i.e. Null);  146 Residual
Null Deviance:       165.3
Residual Deviance: 162.1        AIC: 170.1
[…output deleted to save space…]
```

Now the list object fit contains results from five logistic regression analyses (note that there are five sets of logistic regression, and here I showed analysis from the first imputed dataset) and relevant information. In the first estimation, the coefficient for *lac.miss* is –0.33 for each one unit increase in *lac.miss*. Note there is a "2" behind *sex.miss*, which indicates level 1 is used as reference (recall that R treats *sex.miss* as factor variable). This is not the end of the story and you need to combine the five sets of results into one.

```
> pooled <- pool(fit)
> round(summary(pooled),2)
             est  se   t     df    Pr(>|t|) Lo 95 Hi 95 nmis fmi   lambda
(Intercept) 0.22 1.67 0.13 20.57 0.90     -3.27 3.70  NA   0.44  0.39
sex.miss2   0.01 0.51 0.01 18.45 0.99     -1.07 1.08  NA   0.47  0.41
map        -0.01 0.01 -0.89 35.14 0.38     -0.04 0.02  0    0.31  0.27
lac.miss   -0.22 0.33 -0.69 16.09 0.50     -0.91 0.47  47   0.50  0.45
```

The pool() function is shipped with MICE package. It combines the results of *m* imputed complete data analysis. The variance is computed by taking into account the uncertainty in the missing value imputation.

Summary

This article introduces how to perform MI by using MICE package. The idea of MI is to take into account uncertainty in predicting missing values by creating multiple complete datasets. There is a variety of imputation methods and users can choose the most appropriate one. In practice, the default setting is usually satisfactory. The with() function sets the environment for the expression to be performed, and the argument for the environment is the imputed datasets. A variety of expressions can be executed including univariate analysis and multivariable regression models. Finally, the pool() function is employed to combine results from analysis of each imputed dataset. The variance obtained from pool() function takes uncertainty produced in the missing value imputations into account.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Donders AR, van der Heijden GJ, Stijnen T, et al. Review: a gentle introduction to imputation of missing values. J Clin Epidemiol 2006;59:1087-1091.
2. Morris TP, White IR, Royston P. Tuning multiple imputation by predictive mean matching and local residual draws. BMC Med Res Methodol 2014;14:75.
3. White IR, Royston P, Wood AM. Multiple imputation using chained equations: Issues and guidance for practice. Stat Med 2011;30:377-399.
4. Moons KG, Donders RA, Stijnen T, et al. Using the outcome for imputation of missing predictor values was

preferred. J Clin Epidemiol 2006;59:1092-1101.
5. Zhang Z, Chen K, Ni H, et al. Predictive value of lactate in unselected critically ill patients: an analysis using fractional polynomials. J Thorac Dis 2014;6:995-1003.
6. Zhang Z. Accessing critical care big data: a step by step approach. J Thorac Dis 2015;7:238-242.
7. Buuren SV, Groothuis-Oudshoorn K. mice: Multivariate Imputation by Chained Equations in R. Journal of Statistical Software 2011;45:1-67.

Cite this article as: Zhang Z. Multiple imputation with multivariate imputation by chained equation (MICE) package. Ann Transl Med 2016;4(2):30. doi: 10.3978/j.issn.2305-5839.2015.12.63

Multiple imputation for time series data with Amelia package

Zhongheng Zhang

Abstract: Time series data are common in medical researches. Many laboratory variables or study endpoints are measured repeatedly over time. Multiple imputation (MI) without considering time trend of a variable may cause it to be unreliable. The article illustrates how to perform MI by using Amelia package in a clinical scenario. Amelia package is powerful in that it allows for MI for time series data. External information on the variable of interest can also be incorporated by using prior or bound argument. Such information may be based on previous published observations, academic consensus, and personal experience. Diagnostics of imputation model can be performed by examining the distributions of imputed and observed values, or by using the over-imputation technique.

Keywords: Multiple imputation (MI); Amelia package; R

Submitted Nov 12, 2015. Accepted for publication Dec 23, 2015.
doi: 10.3978/j.issn.2305-5839.2015.12.60
View this article at: http://dx.doi.org/10.3978/j.issn.2305-5839.2015.12.60

Introduction

Time series data are frequently encountered in medical researches (1,2). In such datasets, individuals are measured for a particular variable repeatedly over time. For example, during general anesthesia patients may have their blood pressures and heart rates recorded every 5 min. A study comparing the efficacy of two analgesics may use the blood pressure and heart rate as the study end points. There is a variety of terms for such time series data in the literature. In politics and economics, investigators call them time series cross sectional data (3,4). In psychology, researchers follow participants for their developmental trends across life span and called this longitudinal study (or longitudinal survey). Despite of the disparity in terminology, one important feature of such data is that time series values usually progress smoothly over time, and conventional multiple imputation (MI) algorithm fails to take this into consideration. Recall that we have previously used imputation() function from longitudinalData package for single imputation for longitudinal data. However, single imputation fails to consider imputation uncertainty and usually underestimates the variance. This article is a step-by-step tutorial on how to perform MI for time series data. Some basic ideas behind the algorithms are provided. Interested readers could refer to references for more details on mathematical equations.

Basic ideas behind Amelia package

Bootstrap-based EM algorithm is employed to impute missing values. The algorithm draws m (the number of imputation dataset) samples of size n (the size of original dataset) from original dataset. Point estimations of mean and variance (both are vectors) are performed in each sample by using EM method. Remember there are m sets of estimates. Then each set of estimates is used to impute the missing observations from original dataset. The result is m sets of imputed data that can be used for subsequent analyses (*Figure 1*). Detailed description of bootstrap-based EM algorithm may be too complex for non-statistician readers and readers can refer to chapter (4) if they need more detailed descriptions.

By assuming that time series data vary smoothly over time, observed values close in time to the missing value can greatly aid imputation of that value. However, the trend pattern may vary over time. A patient may experience sustained hypotension and lactate raises rapidly. Then after adequate resuscitation, he or she may recover from shock and the lactate value drops. In different situations, observed values would be used in different ways to impute missing values. One advantage of *Amelia* is its ability to incorporate polynomials of time to fit a model to predict missing values. In the following example, I will show the difference in imputations with and without polynomials of time.

Working example

For hemodynamically unstable patients, mean blood pressure (*map*) and serum lactate (*lac*) are measured on daily basis. The latter is a reflection of tissue perfusion, and high *lac* values are the result of hypotension and tissue hypoperfusion. I created 15 patients and each one is followed up for 10 days.

```
> id<-rep(1:15,each=10)
> time<-rep(1:10,15)
> set.seed(1234)
> map.raw<-abs(round(rnorm(150,mean=50,sd=25)))
> map<-round(ifelse(map.raw>=30,
map.raw,map.raw+50))
> set.seed(1234)
> lac<-round(3-map*0.06+rnorm(150,mean=0,sd=0.4)-
0.4*time*time+4.3*time,1)
> set.seed(1234)
> lac.miss.tag<-rbinom(150, 1, 0.3)
> lac.miss<-ifelse(lac.miss.tag==1,NA,lac)
> set.seed(123456)
> age<-rep(round(abs(rnorm(15, mean = 65, sd =
19))),each=10)
> data<-data.frame(id,time,age,map,lac.miss)
```

The *id* variable repeated for 10 times for every patient, and each measurement time is denoted by *time* variable from 1 to 10. Mean blood pressure (*map*) is simulated by assuming a normal distribution with a mean of 50 and a standard deviation of 25. An arbitrary value of 50 is added to *map* values less than 30. Lactate (*lac*) is the function of *map* and *time*. Higher *map* is associated with lower values of *lac*. Furthermore, *lac* follows a natural disease progression pattern, which goes up at the initial phase (disease progression), reaches a plateau and then returns to normal range thereafter. It is like a quadratic function with negative coefficient for the quadratic term. There is an uncertainty in *lac* values that cannot be accounted for by either *map* or *time*. Roughly 30% of *lac* values are missing, and the missing pattern is missing completely at random (MCAR). The variable age assumes a normal distribution and one patient has a fixed age value (e.g., age does not change within ten days). Lastly, simulated variables are aggregated into a data frame.

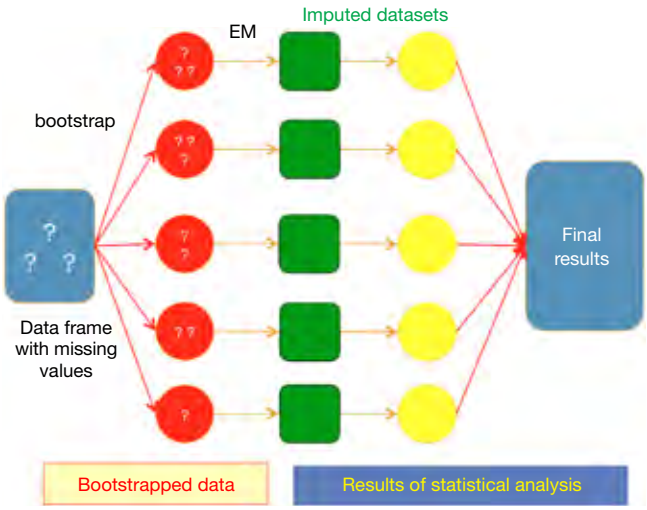


Figure 1 Schematic illustration of multiple imputation based on bootstrap-based EM algorithm.

Multiple imputation (MI)

MI with the amelia() function can be simply performed by the following code (5).

```
> library(Amelia)
> a.out <- amelia(data, m = 5, ts = "time", cs = "id")
```

The first argument assigns a data frame with missing values to the amelia() function. The number of imputed datasets to create is defined by *m*. In this function, the effect of time is not incorporated into the model. The imputed datasets can be visited by the following code. Because there are five imputed datasets, I create five data frame objects and each contains the imputed dataset.

```
> imp1<-a.out$imputations[[1]]
> imp2<-a.out$imputations[[2]]
> imp3<-a.out$imputations[[3]]
> imp4<-a.out$imputations[[4]]
> imp5<-a.out$imputations[[5]]
```

It would be interesting to visualize the imputed data in each individual. For the purpose of visualization of time series data, you may use the ggplot2 package (6). The

graphical grammar of this package is based on the Grammar of Graphics (7). *ggplot2* works in a layered fashion, starting with a layer showing the raw data then adding layers of additional symbols. In the example, longitudinal trends of imputed datasets are drawn layer after layer.

```
> install.packages("ggplot2")
> library(ggplot2)
> p<- ggplot(data =data[31:60,],
aes(x = time, y = lac.miss, group = id))
>p+
geom_point(data=imp1[31:60,],aes(colour="red"))+
geom_line(data=imp1[31:60,])+
geom_point(data=imp2[31:60,],aes(colour="red"))+
geom_line(data=imp2[31:60,])+
geom_point(data=imp3[31:60,],aes(colour="red"))+
geom_line(data=imp3[31:60,])+
geom_point(data=imp4[31:60,],aes(colour="red"))+
geom_line(data=imp4[31:60,])+
geom_point(data=imp5[31:60,],aes(colour="red"))+
geom_line(data=imp5[31:60,])+
geom_point(data=data[31:60,])+
facet_grid(. ~ id)
```

Figure 2 shows the imputed values (red dot) and observed values (black dot). Each panel represents one individual patient. Here I arbitrarily present the patients 4, 5 and 6. Note that the imputed data distributed across a wide range, suggesting a remarkable uncertainty, and the time trend in *lac* variation is not taken into consideration.

Otherwise, the imputed values can be visualized by using tscsPlot() function shipped with Amelia package (*Figure 3*).

```
> tscsPlot(a.out, cs = c(3,4,5,6), main = "without
polynomials of time", var ="lac.miss")
```

Incorporating polynomials of time

Next, I will show how to incorporate the variable time into imputation.

```
> a.out2 <- amelia(data, m = 5, ts = "time", cs =
"id",polytime=2)
```

As you can see, polynomials of time is assigned by the argument "polytime=2". A polynomial power between 0 and 3 is allowed to account for the effects of time on variation of the variable of interest. Other settings are the same to the above. Next, I would like to visualize the result of imputation when polynomials of time are incorporated into the model.

```
> tscsPlot(a.out2, cs = c(3,4,5,6), main = "with
polynomials of time", var ="lac.miss")
```

Figure 4 displays the same individuals as shown in *Figure 3*. Red dot and line represent posterior distribution of imputed values. Note that the distribution of imputed values is much more aggregated, suggesting more certainty in imputed values after incorporating the time polynomials.

Lags and leads

An alternative to take time into account is to use lags and leads of a variable. In Amelia package, lags are to take the value of another variable in the previous time, and leads are to take the value of another variable in the next time point. This task can be done by simply assign arguments to "lags=" and "leads=". Note this assignment will take longer chain lengths.

```
> a.out3 <- amelia(data, m = 5, ts = "time", cs = "id",lags
= "lac.miss",leads = "lac.miss")
> tscsPlot(a.out3, cs = c(3,4,5,6), main = "with lags and
leads", var ="lac.miss")
```

The results are shown in *Figure 5*. As compared with *Figure 4*, the imputed values distributed more widely. However, the imputation is more influenced by lags and leads and the distribution appears better than *Figure 3*.

Prior information

Occasionally, investigators may have prior knowledge on the missing values based on previous published observations, academic consensus, and personal experience. When these are available, incorporation of such external information into imputation would greatly reduce uncertainty on imputations. The prior information can be used within the Bayesian framework. With Amelia package, prior information can be applied using "priors=" argument. This

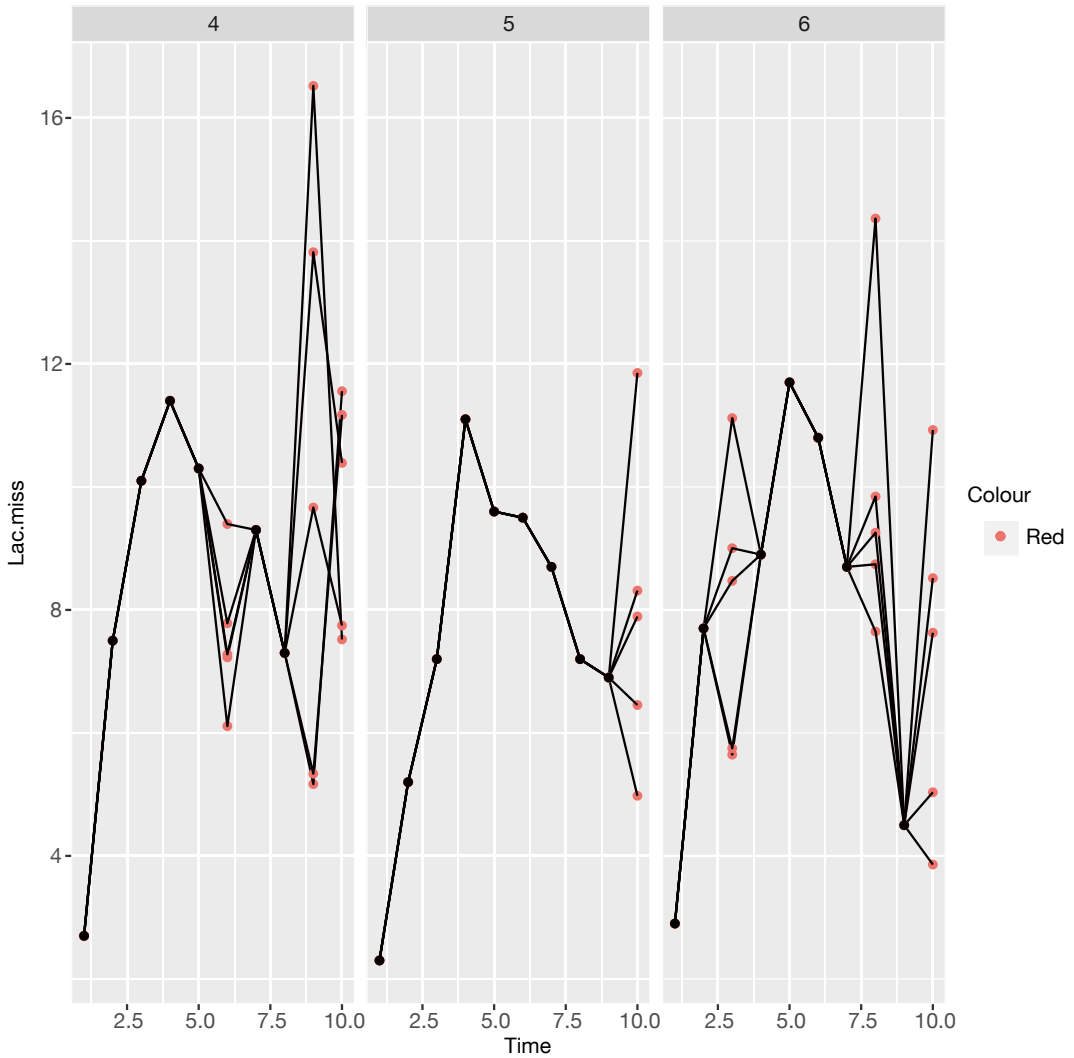


Figure 2 Plots produced by ggplot2 package showing the imputed values (red dot) and observed values (black dot). Each panel represents one individual patient.

argument receives a four or five-column prior matrix.

```
> data[data$id == "6",]
    id time age map lac.miss
51   6    1  81  55      2.9
52   6    2  81  35      7.7
53   6    3  81  72       NA
54   6    4  81  75      8.9
55   6    5  81  46     11.7
56   6    6  81  64     10.8
57   6    7  81  91      8.7
58   6    8  81  31       NA
59   6    9  81  90      4.5
60   6   10  81  71       NA
```

Supposed that I have prior information on lactate levels for the 6th patient. The mean lactate value is 3 because he survives the episode of shock and literature shows that patients with a mean lactate level of 3 are very likely to survive. However, there is uncertainty on this mean *lac* and a standard deviation of 1.2 is added. The prior matrix can

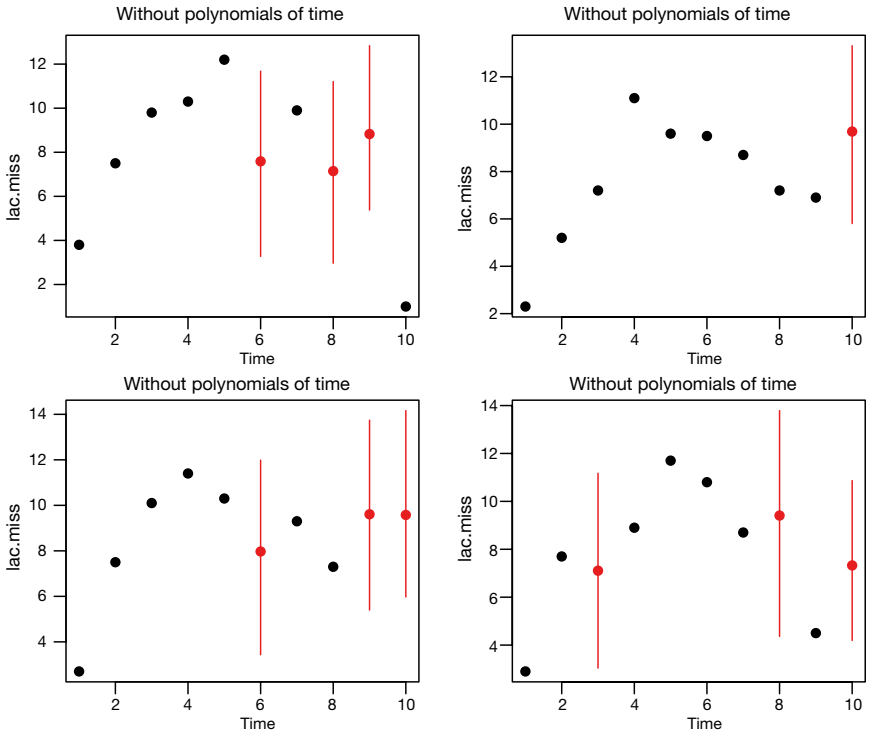


Figure 3 Lactate values of each individual patient are plotted against time. Posterior distributions of missing values are shown. Red dot indicates the mean of imputed values and red line represent 95% credible interval.

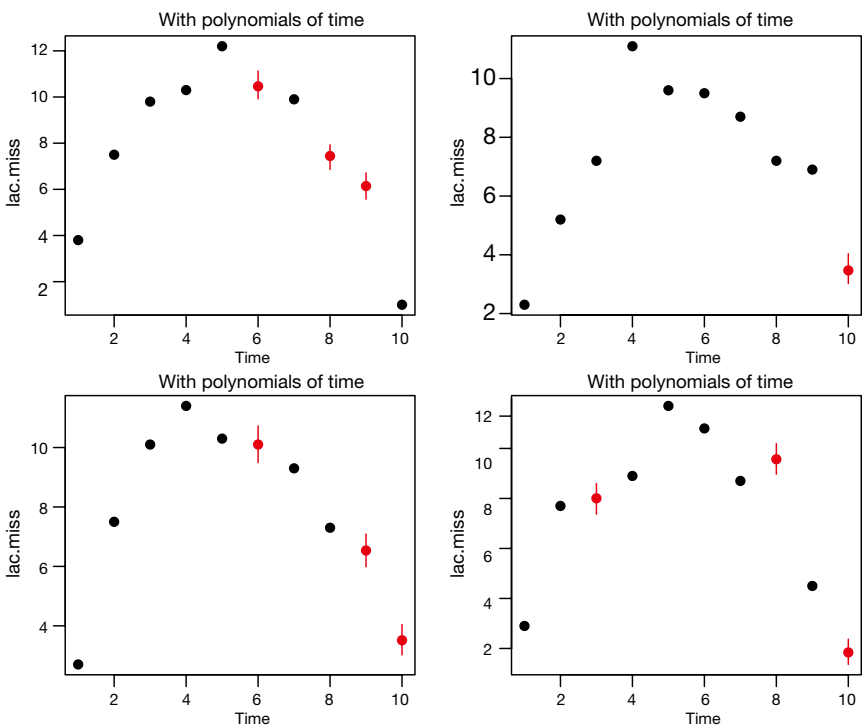


Figure 4 Imputation with polynomials of time. Note that the distribution of imputed values is much more aggregated, suggesting more certainty in imputed values after incorporating of the time polynomials.

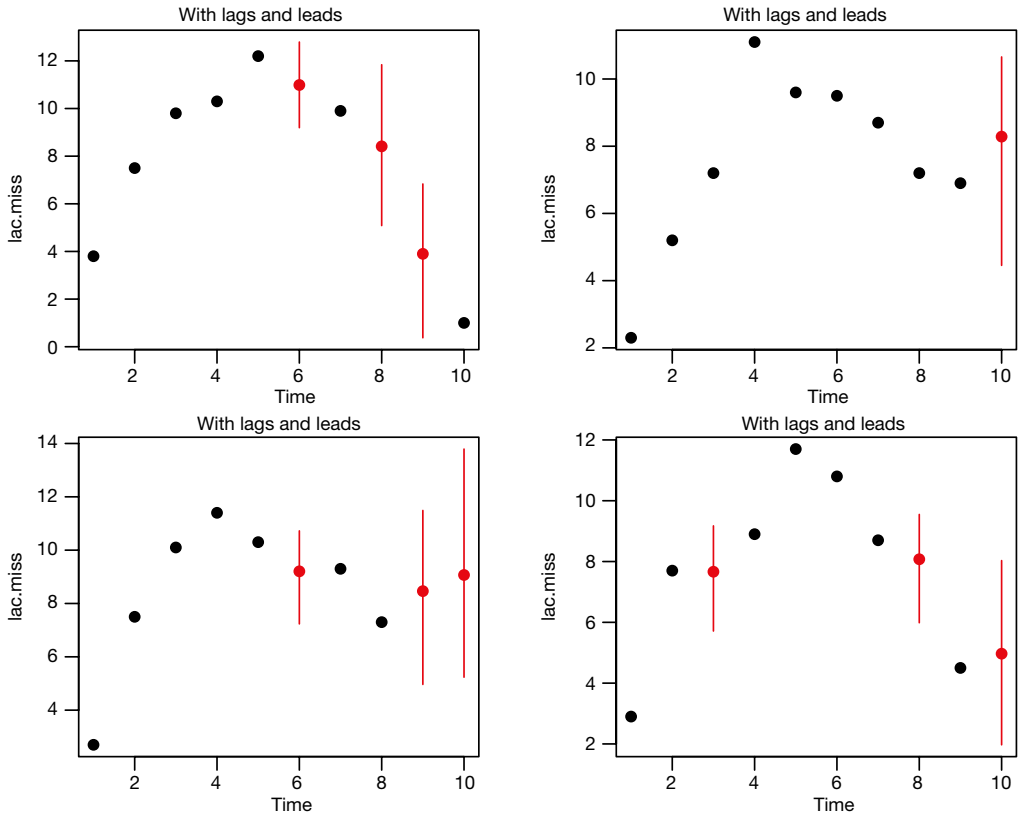


Figure 5 Imputation with lags and leads.

be created by the following codes.

```
> pr <- matrix(c(53,58,60,5,5,5,3,3,3,1.2,1.2,1.2),
nrow=3, ncol=4)
> pr
```

	[,1]	[,2]	[,3]	[,4]
[1,]	53	5	3	1.2
[2,]	58	5	3	1.2
[3,]	60	5	3	1.2

The first column is the row number of the missing values. The second column indicates the number of column of the variable with missing values. The third column contains the presumed mean of each missing values and the forth column is the standard deviation. With the prior matrix, MI can be performed with Amelia.

```
> a.out.pr <- amelia(data, ts = "time", cs = "id", priors =
pr)
```

External information can also be used by logical bounds. Amelia can take draws from a truncated distribution, and the truncation is performed by setting bounds. In the example, suppose that you are certain that the missing values of the 6th patient fall between 3 and 5. Firstly, you need to create a bound matrix. Then the bound matrix can be passed to the amelia() function.

```
> bds <- matrix(c(5, 3, 5), nrow = 1, ncol = 3)
> bds
```

	[,1]	[,2]	[,3]
[1,]	5	3	5

```
> a.out.bds <- amelia(data, ts = "time", cs = "id",
bounds = bds, max.resample = 1000)
> tscsPlot(a.out.bds, cs = c(3,4,5,6), main = "with
bounds", var ="lac.miss")
```

The results are shown in *Figure 6*. Note that the imputed values are restricted to the bound between 3 and 5.

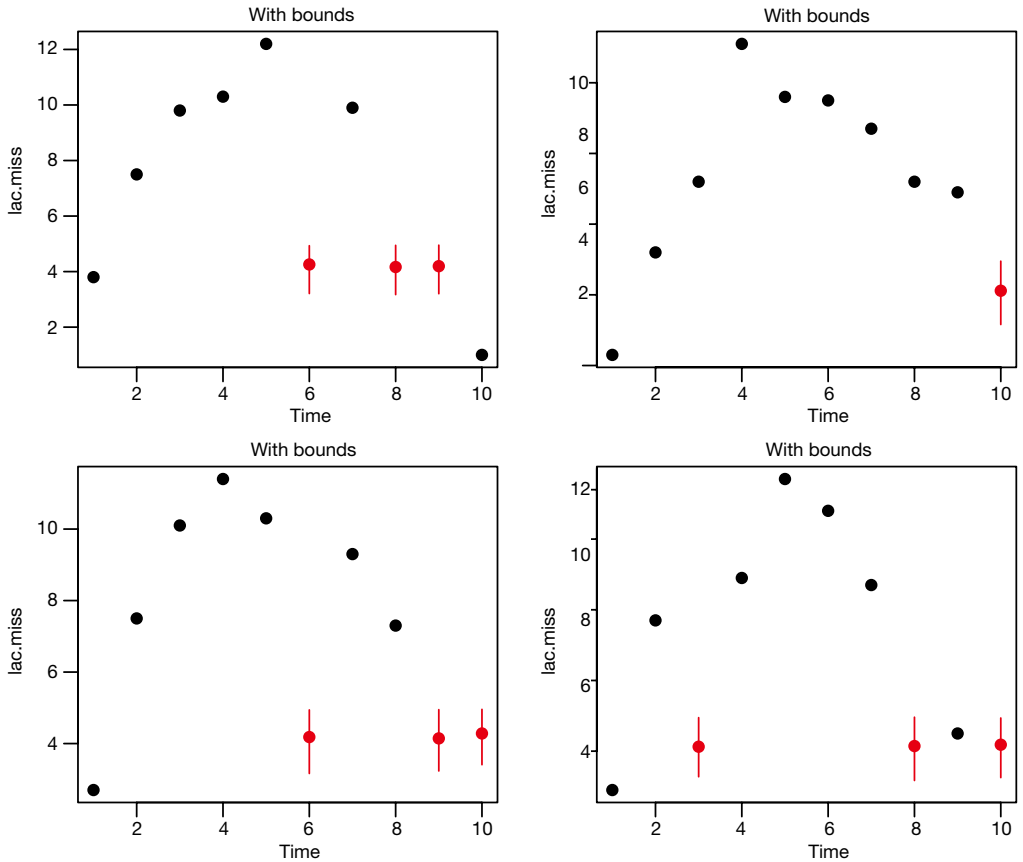


Figure 6 Imputation with arbitrary of bound between 3 and 5. Note that the imputed values are restricted to the bound between 3 and 5.

Imputation diagnostics

Like diagnostics for regression model, imputed values also need to be checked for their plausibility. The amelia package provides several methods for the diagnostics of imputation. You may compare distributions of imputed and observed values with the following codes.

```
> par(mfrow=c(2,2))
> compare.density(a.out, var = "lac.miss",
main="without polynomials of time")
> compare.density(a.out2, var = "lac.miss",main="with
polynomials of time")
> compare.density(a.out3, var = "lac.miss",main="lags
and leads")
> compare.density(a.out.bds, var = "lac.miss",
main="bounds of 3-5")
```

The par() function is to set graphical parameters. The "mfrow= c(2,2)" argument dictates that subsequent figures will be drawn in an 2-by-2 array. The compare.density() function is used to draw distributions of imputed and observed values (*Figure 7*). The results show that the imputation with polynomials of time fits the best. Imputation with arbitrary bounds fits poorly. Another way to examine the imputation model is over-imputation.

```
> par(mfrow=c(2,1))
> overimpute(a.out, var = "lac.miss",
main="without polynomials of time")
> overimpute(a.out2, var = "lac.miss",
main="with polynomials of time")
```

Over-imputation is a technique designed to test the fitness of imputation model. Each observed value is assumed to be missing, and imputed by using the imputation model. The horizontal line displays the actual observed value, and the vertical line denotes the imputed

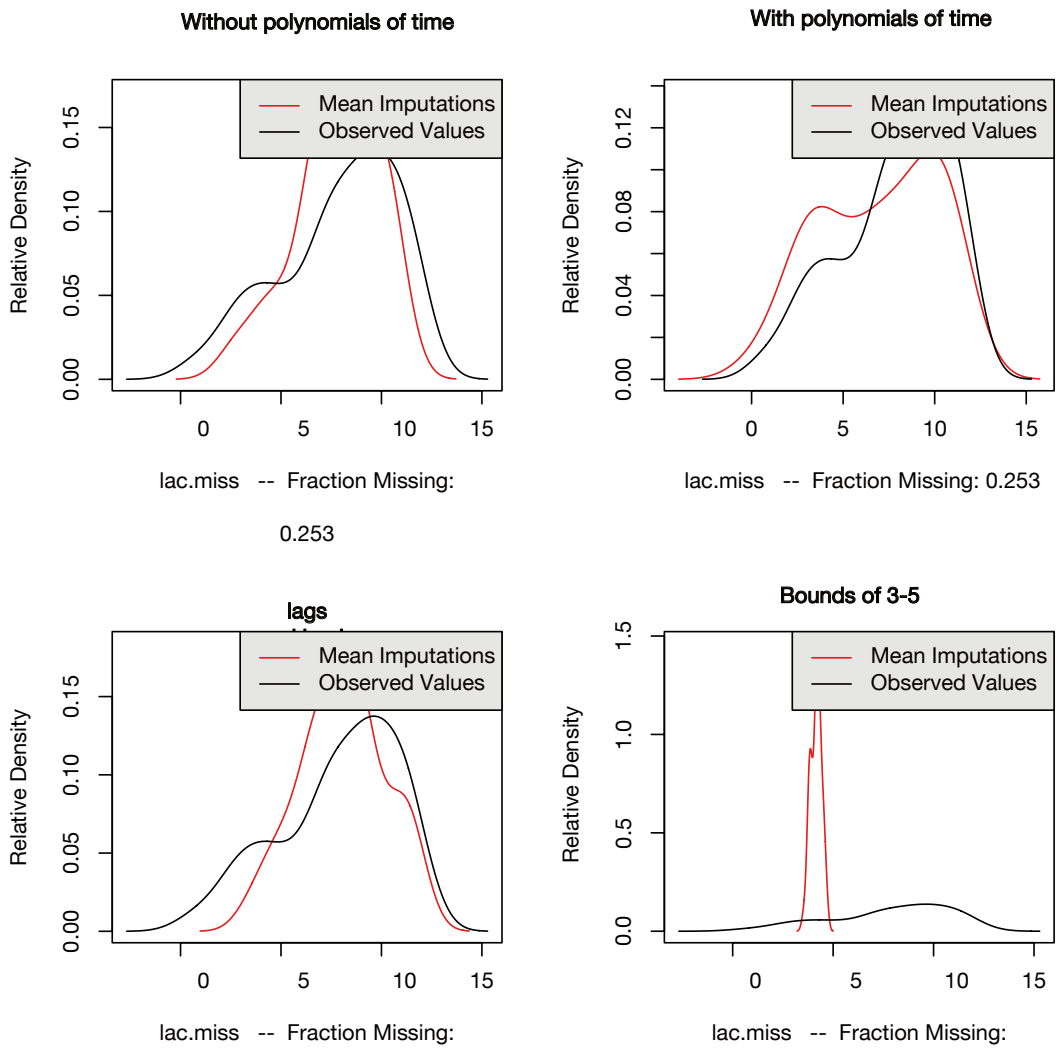


Figure 7 Diagnostics of imputation model performed by comparing distributions of imputed and observed values. The results show that imputation with polynomials of time fits best. Imputation with arbitrary bounds fits poorly.

value pretending that the observed value are missing. The dots are mean value of imputation and the lines represent 90% confidence interval. If mean values are on the diagonal line, the imputation model is exactly accurate. It is obvious from *Figure 8* that imputation model with polynomials of time predicts well for missing values. The color of the line (in the bottom legend) shows the fraction of missing values in the missing pattern for that patient.

Summary

The article illustrates how to perform MI by using the Amelia package in a clinical scenario. Amelia package is powerful in that it allows for MI for time series data. External information based on previous published observations, academic consensus, and personal experience can be incorporated by using prior or bound arguments. Diagnostics of imputation model can be performed by examining the distributions of imputed and observed values,

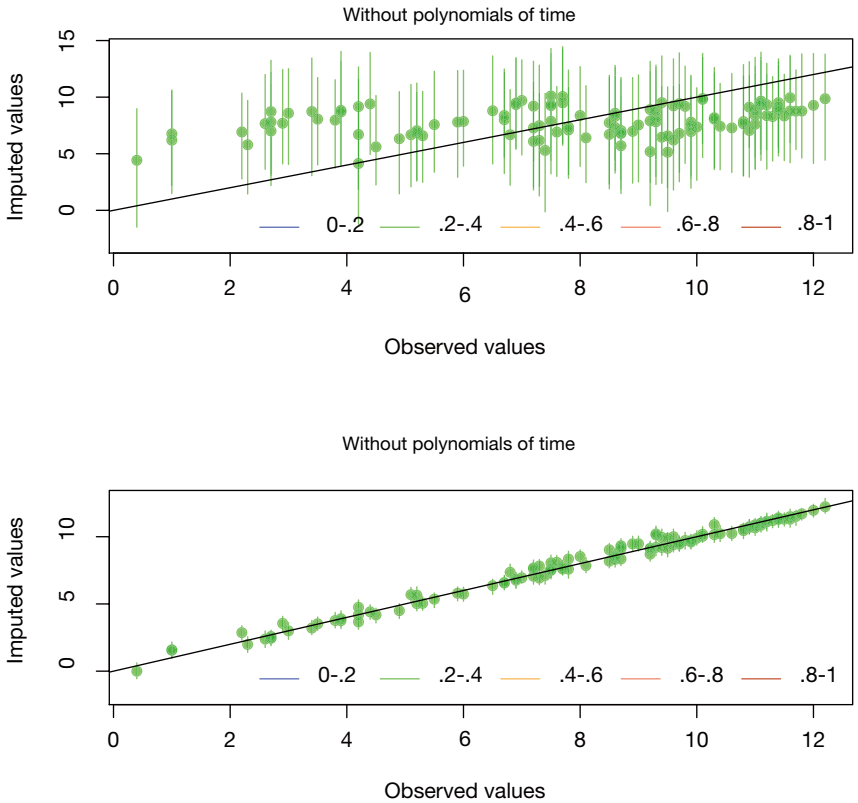


Figure 8 Diagnostics of imputation model performed by over-imputation technique. It is apparent that imputation model with polynomials of time predicts well for missing values.

or by using over-imputation technique.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Chen B, Sumi A, Toyoda S, et al. Time series analysis of reported cases of hand, foot, and mouth disease from 2010 to 2013 in Wuhan, China. BMC Infect Dis 2015;15:495.
2. Kennedy CE, Aoki N, Mariscalco M, et al. Using Time Series Analysis to Predict Cardiac Arrest in a PICU. Pediatr Crit Care Med 2015;16:e332-e339.
3. Beck N, Katz JN, Tucker R. Taking Time Seriously: Time-Series-Cross-Section Analysis with a Binary Dependent Variable. American Journal of Political Science 1998;42:1260-1288.
4. Honaker J, King G. What to Do about Missing Values in Time-Series Cross-Section Data. American Journal of Political Science. American Journal of Political Science 2010;54:561-581.
5. Honaker J, King G, Blackwell M., Amelia II. A program for missing data. Journal of Statistical Software. 2011;45:1-47.
6. Wickham H. ggplot2: elegant graphics for data analysis. New York: Springer-Verlag, 2009.
7. Wilkinson L. The grammar of graphics. New York: Springer-Verlag, 2012.

Cite this article as: Zhang Z. Multiple imputation for time series data with Amelia package. Ann Transl Med 2016;4(3):56. doi: 10.3978/j.issn.2305-5839.2015.12.60

Univariate description and bivariate statistical inference: the first step delving into data

Zhongheng Zhang

Abstract: In observational studies, usually the first step is to explore the data distributions and the baseline differences between groups. Data description includes their central tendency (e.g., mean, median, and mode) and dispersion (e.g., standard deviation, range, interquartile range). There is a variety of bivariate statistical inference methods such as Student's t-test, Mann-Whitney U test and Chi-square test, for normal, skewed and categorical data, respectively. The article shows how to perform these analyses with R. Furthermore, I believe that the automation of the whole workflow is of paramount importance in that (I) it allows for others to replicate your results; (II) you can easily find out how you performed analysis during manuscript revision; (III) it spares manual data input and is less error-prone; and (IV) when you correct your original dataset, the final result can be automatically corrected by executing the codes. Therefore, the process of making a publication-quality table incorporating all above-mentioned statistics and P values is provided, allowing readers to customize these codes to their own needs.

Keywords: Univariate description; bivariate statistical inference; R; table; baseline characteristics; automation

Submitted Dec 25, 2015. Accepted for publication Jan 10, 2016.
doi: 10.21037/atm.2016.02.11
View this article at: http://dx.doi.org/10.21037/atm.2016.02.11

Introduction

When data are well-prepared by using previously described methods such as correcting, recoding, rescaling and missing value imputation, the next step is to perform statistical description and inference (1,2). In observational studies, the first table is usually a display of descriptive statistics of overall population, as well as statistical inference for the difference between groups. This table is important in that it estimates the differences in baseline characteristics, and provides evidence for further multivariable analysis. The article first gives an overview of methods used for bivariate analysis, and then provides a step-by-step tutorial on how to perform these analyses in R. Finally, I will show how to make the table automatically. This can be useful when there are a large number of variables.

Univariate description and bivariate statistical methods

A variety of methods are available for the univariate description and bivariate inference. *Table 1* displays central tendency and dispersion for different types of data. Mean and standard deviation are probably the most widely used statistics to describe normally distributed data. For skewed data, we employ the median and interquartile range. For nominal data, mode can be used for description of central tendency. However, in practice we frequently describe it by the number of each category and relevant percentage. Student's *t*-test is typically applied when variable under test follows a normal distribution (3). Mann-Whitney U test is a non-parametric test that does not require the assumption of normal distribution (4). For studies with multiple groups, analysis of variance (ANOVA) may be the best choice.

Working example

In the working example, I create three types of data that are most commonly encountered in practice. Nominal variables include those with multiple levels and those with two levels. Continuous variables include those with normal and non-normal distributions.

```
> set.seed(12)
> gender<-factor(rbinom(100, 1, 0.4),levels=c(0,1),
labels=c("male","female"))
> set.seed(12)
```

```
> trt<-factor(rbinom(100, 1, 0.5),levels=c(0,1),
labels=c("treat","control"))
> set.seed(88)
> diagnosis<-factor(rbinom(100, 3, 0.5),
levels=c(0,1,2,3),labels=c("heart failure",
"renal dysfunction","ards","trauma"))
> set.seed(88)
> age<-rnorm(100,mean=67,sd=20)
> set.seed(88)
> wbc<-round(exp(rnorm(100,mean=9,sd=0.8)))
> data<-data.frame(gender,age,trt,diagnosis,wbc)
```

In this dataset, gender is a binomial variable with two levels of "male" and "female". The variable *trt* is also binomial, but it is used for grouping purpose. The variable *diagnosis* is a categorical variable with four levels. The variable *age* is normally distributed with a mean of 67 and a standard deviation of 20. The last variable *wbc* is skewed in distribution. The last line combines these variables into a single data frame.

Examination of skewness and kurtosis

Because the choice of statistical methods depends on the distribution, the first step is to examine the normality of the data. The distribution can be visualized using histogram (*Figure 1*).

```
> par(mfrow=c(1,2))
> hist(data$age)
> hist(data$wbc)
```

The first line calls par() function to dictate that subsequent figures will be drawn in 1×2 array. It is for the purpose of better visualization and users can omit it in their own practices. As we can see from *Figure 1*, the distribution of *age* was symmetrical, while the variable *wbc* is skewed. However, graphic visualization only gives a hint on the distribution of data. To make formal judgment, we need statistical tests. The package *moments* provides good functions to do the task (5). Data distribution can be described by skewness and kurtosis. The former is a measure of the asymmetry of the probability distribution, and the latter is a measure of the "tailedness" of the probability distribution.

```
> install.packages("moments")
> library(moments)
> agostino.test(data$age)

        D'Agostino skewness test

data: data$age
skew = 0.2462, z = 1.0571, p-value = 0.2905
alternative hypothesis: data have a skewness
> agostino.test(data$wbc)

        D'Agostino skewness test

data: data$wbc
skew = 2.7039, z = 7.1426, p-value = 9.158e-13
alternative hypothesis: data have a skewness
```

The package employs the D'Agostino skewness test, details of this method can be found in reference (6). The alternative hypothesis is that the data have a skewness. When $P<0.05$ as for the variable *wbc*, the alternative hypothesis is accepted and there is skewness.

```
> anscombe.test(data$age)

        Anscombe-Glynn kurtosis test

data: data$age
kurt = 2.8616000, z = -0.0028691, p-value = 0.9977
alternative hypothesis: kurtosis is not equal to 3

> anscombe.test(data$wbc)

        Anscombe-Glynn kurtosis test

data: data$wbc
kurt = 11.463, z = 5.209, p-value = 1.898e-07
alternative hypothesis: kurtosis is not equal to 3
```

Anscombe-Glynn kurtosis test is employed to test kurtosis (6,7). Data should have kurtosis equal to 3 under the hypothesis of normality. This test incorporates such null hypothesis and is useful to detect a significant difference

Table 1 Central tendency and dispersion for difference types of data

Data type	Central tendency	Description	Dispersion	Description	Statistical inference
Nominal	Mode	Value of highest frequency	NA	NA	Chi-square test
Skewed data	Median	Center value	Interquartile range	Difference between the upper and lower quartiles	Mann-Whitney U test
Normal distribution	Mean	Mathematical average	Standard deviation	How much deviate from the mean	Student's t-test

NA, not applicable.

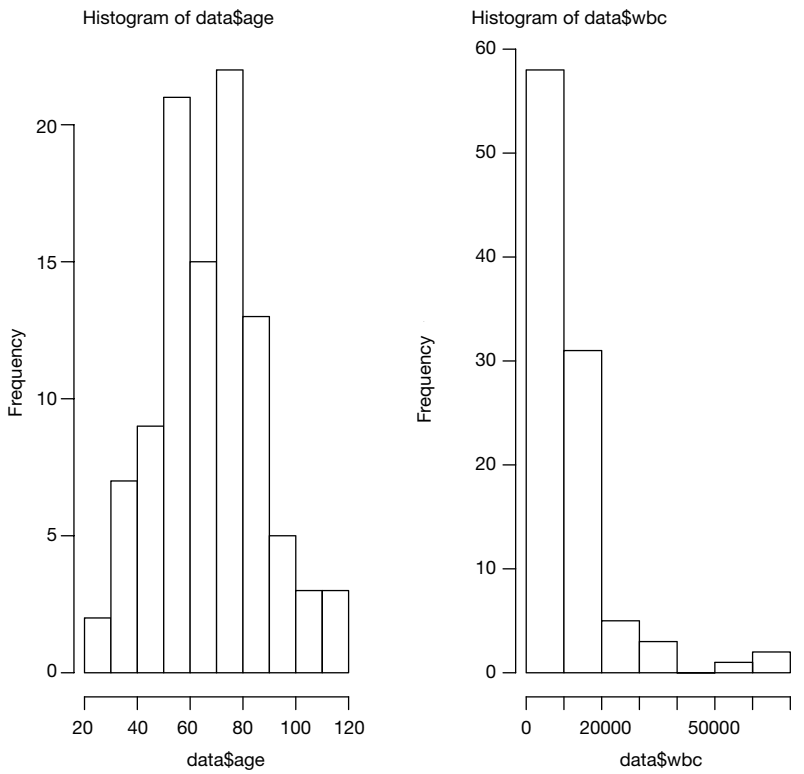


Figure 1 Histograms of the variables of age and *wbc*. It appears that the distribution of *age* was symmetrical, while the variable *wbc* is skewed.

of kurtosis in normally distributed data. As expected, the variable *wbc* has kurtosis that is significantly different from normality.

Univariate description

Since we know the distribution of the data, we need to provide central tendency and dispersion. The variable *wbc* will be expressed as median and interquartile range, and *age* will be expressed as mean and standard deviation. Other categorical variables will be expressed as the number and percentage.

```
> summary(age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  26.18   52.18   68.59   67.21   78.78  118.10
> sd(age)
[1] 19.86663
> summary(wbc)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```



```
1583    4479     8634      11360      12980     62670
> table(diagnosis)
diagnosis
heart failure     renal dysfunction  ards     trauma
11                38                 37       14
> prop.table(table(diagnosis))
diagnosis
heart failure     renal dysfunction  ards       trauma
0.11              0.38               0.37       0.14
```

The above codes are used for description of the overall cohort. We also want to describe variables separately by the treatment group.

```
> tapply(data$wbc, data$trt, summary)
$treat
Min.    1st Qu.  Median   Mean     3rd Qu.   Max.
1639    4599     8414     10170    12630     60830

$control
Min.    1st Qu.  Median   Mean     3rd Qu.   Max.
1583    4508     8634     12750    13380     62670

> table(data$diagnosis,data$trt)

                      treat          control
heart failure         5              6
renal dysfunction     20             18
ards                  18             19
trauma                11             3
> prop.table(table(data$diagnosis,data$trt),margin=2)

                    treat          control
heart failure       0.09259259     0.13043478
renal dysfunc-0.37037037           0.39130435
tion
ards                0.33333333     0.41304348
trauma              0.20370370     0.06521739
```

The function tapply() "applies a function to each cell of a ragged array, that is to each (non-empty) group of values given by a unique combination of the levels of certain factors." (8). In our example, the function summary() is applied to the variable *wbc*, stratified by the treatment (trt). The results give summary values separately for each level of the variable *trt*. The table() function is easy to use for cross tabulation.

Bivariate statistical inference

After the univariate description, investigators can have a general impression on the effectiveness of a treatment. However, we still don't know whether the difference is caused by random error, or there is a real difference. A variety of sophisticated methods are designed to answer this question.

```
> wilcox.test(wbc ~ trt, data=data)

        Wilcoxon rank sum test with continuity correction

data: wbc by trt
W = 1166.5, p-value = 0.604
alternative hypothesis: true location shift is not equal to 0
> chisq.test(table(data$diagnosis,data$trt))

        Pearson's Chi-squared test

data: table(data$diagnosis, data$trt)
X-squared = 4.1814, df = 3, p-value = 0.2425
> t.test(age ~ trt, data=data)

        Welch Two Sample t-test

data: age by trt
t = -0.79721, df = 90.569, p-value = 0.4274
alternative hypothesis: true difference in means is not
equal to 0
95 percent confidence interval:
-11.229622   4.797661
sample estimates:
mean in group treat      mean in group control
65.72955                 68.94553
```

In the above codes, the functions wilcox.test(), chisq.test() and t.test() are employed for skewed, categorical and normal data, respectively. The outputs of these functions

include the name of the test, dataset, statistics and P value. These values can be stored as an object for further use.

Making publication style table automatically

It is easy to copy and paste the above statistical outputs into a table in Microsoft Word when there are a limited number of variables. However, it is a tedious task when variables expand to dozens. Furthermore, it is error-prone to make a blank table and input values manually. We may take the advantage of R that it allows for customized codes to automate the process of table generation. It also allows to round numbers to a desired decimal place.

```
> overall.age<-paste(round(mean(data$age),1),
"±",round(sd(data$age),1))
> trt.age<-paste(round(t.test(age ~ trt,
data=data)$estimate[1],1), "±",round(tapply(data$age,
data$trt, sd)[1],1))
> contrl.age<-paste(round(t.test(age ~ trt,
data=data)$estimate[2],1), "±",round(tapply(data$age,
data$trt, sd)[2],1))
> age.p<-round(t.test(age ~ trt, data=data)$p.value,2)
> row.age<-c(overall.age,trt.age,contrl.age,age.p)
```

Many journals require that the description of a variable should be put in a single cell, thus we need to use character vector to store the output in the form like "mean ± SD". The paste() function can connect mean and standard deviation by using the symbol "±". The round() function is used to save only one decimal place. Skewed data can be stored in a similar way.

```
> overall.wbc<-paste(summary(data$wbc)[3],
"(",summary(data$wbc)[2],"~",
summary(data$wbc)[5],")")
> wbc.trt.all<-tapply(data$wbc, data$trt,
summary)$treat
> wbc.control.all<-tapply(data$wbc, data$trt,
summary)$control
> trt.wbc<-paste(wbc.trt.all[3],"(",
wbc.trt.all[2],"~",wbc.trt.all[5],")")
> contrl.wbc<-paste(wbc.control.all[3],"(",
wbc.control.all[2],"~",wbc.control.all[5],")")
> wbc.p<-round(wilcox.test(wbc ~ trt,
data=data)$p.value,2)
> row.wbc<- c(overall.wbc,trt.wbc,contrl.wbc,wbc.p)
```

This time I break the whole work into several pieces to make it clearer. The first line calculates the overall mean and standard deviation, and merges them in to a single cell. The second and third lines calculate the summary statistics for the treatment and control groups, respectively. Because not all summary statistics are wanted, we extracted the second, third and fifth values, which represent the first quartile, median and third quartile values. The last line combines all statistics into a character vector.

```
> overall.gender<-paste(table(data$gender)[1],"(",
prop.table(table(data$gender))[1]*100,"%",")")
> trt.gender<-paste(table(data$gender,data$trt)[1],"(",
round(prop.table(table(data$gender,data$trt),
margin=2)[1],3)*100,"%",")")
> contrl.gender<-paste(table(data$gender,data$trt)[3],
"(",round(prop.table(table(data$gender,data$trt),
margin=2)[3],3)*100,"%",")")
> p.gender<-round(chisq.test(table(data$gender,
data$trt))$p.value,2)
> row.gender<-c(overall.gender,trt.gender,
contrl.gender,p.gender)
```

Next we need to combine all row variables into a matrix.

```
> table<-rbind(row.age,row.gender,row.wbc)
> table
```

	[,1]	[,2]	[,3]	[,4]
row.age	"67.2 ± 19.9"	"65.7 ± 18.7"	"68.9 ± 21.2"	"0.43"
row.gender	"62 (62 %)"	"34 (63 %)"	"28 (60.9 %)"	"0.99"
row.wbc	"8634 (4479 ~ 12980)"	"8414 (4599 ~ 12630)"	"8634 (4508 ~ 13380)"	"0.6"

The matrix output appears nearly what we want. However, the column and row names are not exactly meet the requirements for submission. We can rename them easily. For the column name I added a new character vector so that each of the names can have a double quote.

```
> row.names(table)<-c("Age (years)",
"Gender (male,%)","WBC/ul")
```


Figure 2 Settings in Microsoft Word to convert text to table. Note that the double quote mark is used to separate text.

```
> table.pub<-rbind(c("Overall","Treatment",
"Control","p value"),table)
> table.pub
```

	[,1] "Overall"	[,2] "Treatment"	[,3] "Control"	[,4] "p value"
Age (years)	"67.2 ± 19.9"	"65.7 ± 18.7"	"68.9 ± 21.2"	"0.43"
Gender (male,%)	"62 (62 %)"	"34 (63 %) "	"28 (60.9 %) "	"0.99"
WBC/ul	"8634 (4479 ~ 12980) "	"8414 (4599 ~ 12630) "	"8634 (4508 ~ 13380) "	"0.6"

There are double quotes for each value, which is not the typical format for a publication-quality table. The next task is done in Microsoft Word (MS). Users can copy the table from R console to MS and use the "convert text to table" function of the Word processor. In the "separate text at" option, you can use double quotes as the symbol to separate text. After click the "OK" button, the double quotes mark will be replaced by lines forming cells of a table (*Figure 2*). Creating table in this way avoids you from the tedious copy-and-paste task and can avoid typos due to manual input. Furthermore, because you have recorded every step for making the table in R scripts, you and others can reproduce the results. This is very important in collaborations and manuscript revisions. The rule of thumb in data management and statistical analysis is to make your results exactly reproducible.

Summary

The article provides a gentle introduction to univariate statistical description and bivariate statistical inference, which is typically the first step in exploring data. Statistical description includes statistics for central tendency such as mode, mean, and median. Dispersion statistics include standard deviation, range, and interquartile range. They are applied to different types of data. Also, there are several statistical inference methods, which are Student's *t*-test, Mann-Whitney U test and Chi-square test. The results of these analyses should be summarized in a table for publication or conference presentation. The process can be automated by using R codes, making the process fully reproducible in collaborations and manuscript revisions.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Zhang Z. Missing values in big data research: some basic skills. Ann Transl Med 2015;3:323.
2. Zhang Z. Data management by using R: big data clinical research series. Ann Transl Med 2015;3:303.
3. Fay MP, Proschan MA. Wilcoxon-Mann-Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules. Stat Surv 2010;4:1-39.
4. Corder GW, Foreman DI. Nonparametric statistics: A step-by-step approach. New York: Wiley, 2014.
5. Komsta L, Novomestky F. moments: moments, cumulants, skewness, kurtosis and related tests. 2012. R package version 0.13; 2014.
6. Ghasemi A, Zahediasl S. Normality tests for statistical analysis: a guide for non-statisticians. Int J Endocrinol

Metab 2012;10:486-489.
7. DeCarlo LT. On the meaning and use of kurtosis. Psychological Methods 1997;2:292-307.
8. Breiman L. R. A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, 2012. Machine Learning 2001;45:5-32.

Cite this article as: Zhang Z. Univariate description and bivariate statistical inference: the first step delving into data. Ann Transl Med 2016;4(5):91. doi: 10.21037/atm.2016.02.11

Model building strategy for logistic regression: purposeful selection

Zhongheng Zhang

Abstract: Logistic regression is one of the most commonly used models to account for confounders in medical literature. The article introduces how to perform purposeful selection with R. I stress on the use of likelihood ratio test to see whether deleting a variable will have significant impact on the model fit. A deleted variable should also be checked for whether it is an important adjustment for remaining covariates. Interactions should be checked to disentangle complex relationships between covariates and their synergistic effect on the response variable. Models should be checked for the goodness-of-fit (GOF). In other words, how the fitted model reflects the real data. Hosmer-Lemeshow GOF test is the most widely used checking for logistic regression models.

Keywords: Logistic regression; interaction; R; purposeful selection; linearity; Hosmer-Lemeshow

Submitted Dec 30, 2015. Accepted for publication Jan 19, 2016.
doi: 10.21037/atm.2016.02.15
View this article at: http://dx.doi.org/10.21037/atm.2016.02.15

Introduction

Logistic regression model is one of the most widely used method to investigate the independent effect of a variable on binomial outcomes in medical literature. However, the model building strategy is not explicitly stated in many studies, compromising the reliability and reproducibility of the results. There is a variety of model building strategies reported in the literature, such as the purposeful selection of variables, the stepwise procedure selection and the best subsets method (1,2). However, there is no one that has been proven to be superior to others and the model building strategy is "part science, part statistical methods, and part experience and common sense" (3). However, the principal of model building is to select as less variables as possible, but the model (parsimonious model) should still reflect the true outcomes of the data. In this article, I will introduce how to perform the purposeful selection procedure in R. Variable selection is the first step of model building. Other steps will be introduced in following section.

Working example

In the example, I create five variables *age*, *gender*, *lac*, *hb* and *wbc* for the prediction of mortality outcome. The outcome variable is binomial which takes the values of "die" and "alive". To illustrate the selection process, variables *age*, *hb* and *lac* are associated with the outcome, while *gender* and *wbc* are not (4-6).

```
> set.seed(888)
> age<-abs(round(rnorm(n=1000,mean=67,sd=14)))
> lac<-abs(round(rnorm(n=1000,mean=5,sd=3),1))
> gender<-factor(rbinom(n=1000,size=1,prob=0.6),
labels=c("male","female"))
> wbc<-abs(round(rnorm(n=1000,mean=10,sd=3),1))
> hb<-abs(round(rnorm(n=1000,mean=120,sd=40)))
> z<-0.1*age-0.02*hb+lac-10
> pr = 1/(1+exp(-z))
> y = rbinom(1000,1,pr)
> mort<-factor(rbinom(1000,1,pr),
labels=c("alive","die"))
> data<-data.frame(age,gender,lac,wbc,hb,mort)
```

Step one: univariable analysis

The first step is to use univariable analysis to explore the unadjusted association between variables and the outcome. In our example, each of the five variables will be included in a logistic regression model, one for each time.

```
> univariable.age<-glm(mort~age, family = binomial)
```

```
> summary(univariable.age)
```

Note that logistic regression model is built by using generalized linear model in R (7). The family argument is a description of the error distribution and link function to be used in the model. For the logistic regression model, the family is binomial with the link function of logit. For the linear regression model, Gaussian distribution with identity link function is assigned to the family argument. The summary() function is able to show you the results of the univariable regression. A P value of smaller than 0.25 and other variables of known clinical relevance can be included for further multivariable analysis. A cutoff value of 0.25 is supported by the literature (8,9). The results of univariable regression for each variable are shown in *Table 1*. According to the rule, the variables *age*, *hb* and *lac* will be included for further analysis. The allowance to include clinically relevant variables, even if they are statistically insignificant, reflects the "part experience and common sense" nature of the model building strategy.

Step two: multivariable model comparisons

This step fits the multivariable model comprising all variables identified in step one. Variables that do not contribute to the model (e.g., with a P value greater than traditional significance level) should be eliminated and a new smaller mode fits. These two models are then compared by using partial likelihood ratio test to make sure that the parsimonious model fits as well as the original model. In the parsimonious model the coefficients of variables should be compared to coefficients in the original one. If a change of coefficients ($\Delta\beta$) is greater than 20%, the deleted variables have provided important adjustment for the effect of the remaining variables. Such variables should be added back to the model. This process of deleting, adding variables and model fitting and refitting continues until all variables excluded are clinically and statistically unimportant, while variables remained in the model are important. In our example, suppose that the variable *wbc* is also added because it is clinically relevant.

```
> model1<-glm(mort~lac+hb+wbc+age, family =
binomial)
> summary(model1)
```

Table 1 Univariable analysis for each variable

Variable	Coefficient	Standard error	P value
Age	0.049	0.005	<0.001
Gender	−0.044	0.131	0.736
wbc	−0.004	0.021	0.845
hb	−0.009	0.002	<0.001
lac	0.740	0.047	<0.001

The result shows that P value for the variable *wbc* is 0.845, which is statistically insignificant. Therefore, we exclude it.

```
> model2<-glm(mort~lac+hb+age, family = binomial)
```

All variables in model2 are statistically significant. Then we will compare the changes in coefficients for each variable remained in model2.

```
> delta.coef<-abs((coef(model2)-coef(model1)[-4])/
coef(model1)[-4])
> round(delta.coef,3)
(Intercept)        lac         hb        age
      0.029      0.004      0.000      0.004
```

The function coef() extracts estimated coefficients from the fitted model. The fitted model2 is passed to the function. Because there is a coefficient for wbc in model1, which has nothing to compare with in model2, we drop it by using "[-4]". The result shows that all variables change at a negligible level and the variable *wbc* is not an important adjustment for the effects of other variables. Furthermore, we will compare the fit of model1 and model2 by using partial likelihood ratio test.

```
> library(lmtest)
> lrtest(model1,model2)
Likelihood ratio test
Model 1: mort ~ lac + hb + wbc + age
Model 2: mort ~ lac + hb + age
#  Df  LogLik      Df  Chisq  Pr(>Chisq)
1   5  -322.73
2   4  -323.08     -1 0.6867      0.4073
```


The result shows that the two models are not significantly different in their fits for the data. In other words, model2 is as good as model1 in fitness. We choose model2 for the principal of parsimony. Alternatively, users can employ analysis of variance (ANOVA) to explore the difference between models.

```
> anova(model1,model2,test="Chisq")
Analysis of Deviance Table
Model 1: mort ~ lac + hb + wbc + age
Model 2: mort ~ lac + hb + age
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       995     645.47
2       996     646.15 -1  -0.6867   0.4073
```

The results are exactly the same. Because of our simple example, we do not need to cycle the process and we can be confident that the variables *hb*, *age* and *lac* are important for the mortality outcome. At the conclusion of this step, we obtain a preliminary main effects model.

Step three: linearity assumption

In the step, continuous variables are checked for their linearity in relation to the logit of the outcome. In this article, I want to examine the smoothed scatter plot for the linearity.

```
> par(mfrow=c(2,2))
> scatter.smooth(age,log(pr/(1-pr)),cex=0.5)
> scatter.smooth(lac,log(pr/(1-pr)),cex=0.5)
> scatter.smooth(hb,log(pr/(1-pr)),cex=0.5)
> scatter.smooth(wbc,log(pr/(1-pr)),cex=0.5)
```

The smoothed scatter plots show that the variables *age*, *lac* and *hb* are all linearly associated with mortality outcome in logit scale (*Figure 1*). The variable *wbc* is not related to the mortality in logit scale. If the scatterplot shows non-linearity, we shall apply other methods to build the model such as the incorporations of 2 or 3-power terms, fractional polynomials and spline functions (10,11).

Step four: interactions among covariates

In this step we check for potential interactions between covariates. An interaction between two variables implies that the effect of one variable on response variable is dependent on another variable. The choice of interaction pairs can be started from clinical perspective. In our example, I assume that there is interaction between *age* and *hb*. In other words, the effect of *hb* on mortality outcome is dependent on the age.

```
> model.interaction<-glm(mort~lac+hb+age+hb:age,
data=data,family = binomial)
 > summary(model.interaction)
                (output omitted to save space)
> lrtest(model2,model.interaction)
Likelihood ratio test

Model 1: mort ~ lac + hb + age
Model 2: mort ~ lac + hb + age + hb:age
#  Df  LogLik      Df  Chisq      Pr(>Chisq)
1  4   -323.08
2  5   -322.91     1   0.3373     0.5614
```

Note that I use the ":" symbol to create the interaction term. There are several ways to produce interaction terms in R (*Table 2*). The results show that the P value for the interaction term is 0.56, which is far away from the significance level. When the model with interaction term is compared to the preliminary main effects model, no statistically significant difference is identified. Thus, the interaction term is dropped. However, if there are interaction effects, users may be interested in visualizing how the effect of one variable changes depending on different levels of another covariate. Suppose we want to visualize how the probability of death (y-axis) changes across an entire range of *hb*, stratified by *age* groups. We will plot at age values of 20, 40, 60 and 80.

```
> newdata<-data.frame(hb=rep(seq(from=40,
to=150),length.out=100,4),lac=mean(lac),
age=rep(c(20,40,60,80),100))
> newdata1 <- cbind(newdata, predict(model.interaction,
newdata = newdata, type = "link",se = TRUE))
> newdata1 <- within(newdata1,{
    age<-factor(age)
    PredictedProb <- plogis(fit)
    LL <- plogis(fit - (1.96 * se.fit))
```

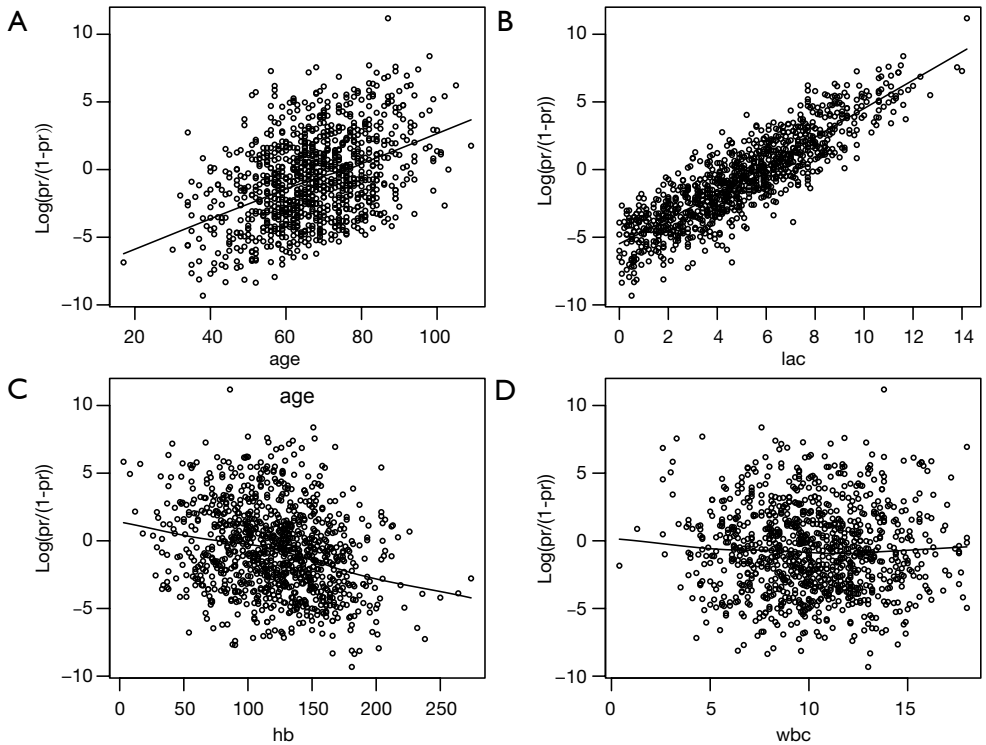



Figure 1 Smoothed scatter plots showing the relationship between variable of interest with mortality outcome in logit scale.

Table 2 Methods to create interaction terms in R

Symbols	Remarks
:	A simple and direct way to denote an interaction between predictor variables. The formula y ~ a + b + a:b is to predict y from a, b, and the interaction between a and b
*	The code y ~ a * b * c can be expanded to y ~ a +b +c + a:b + a:c + b:c + a:b:c. This is a shortcut for all possible interaction terms
^	The code y ~ (a +b + c)^2 can be expanded to y ~ a + b + c + a:b + a:c + b:c. The interaction is in specified degree. In this case, the interaction is defined to be at degree 2

```
    UL <- plogis(fit + (1.96 * se.fit))
})
```

The first command creates a new data frame that contains new patients. Variables of each patient are artificially assigned. Variable *hb* is defined between 4 and 15, with a total of 100 patients at each age group. *lac* is held at its mean value. The next line applies the fitted model to the new data frame, aiming to calculate the fitted values in logit scale and relevant standard error. The plogis() function transforms fitted values into probability scale, which is much easier to understand for subject-matter audience. Lower and upper limits of the confidence intervals are transformed in similar way. The continuous variable age is transformed into a factor that will be helpful for subsequent plotting.

```
> library(ggplot2)
> ggplot(newdata1,
        aes(x = hb, y = PredictedProb)) +
geom_ribbon(aes(ymin = LL,
        ymax = UL, fill = age), alpha = 0.2) +
geom_line(aes(colour = age),
        size = 1)
```

The result is shown in *Figure 2*. Because there is no significant interaction, the lines are generally parallel.

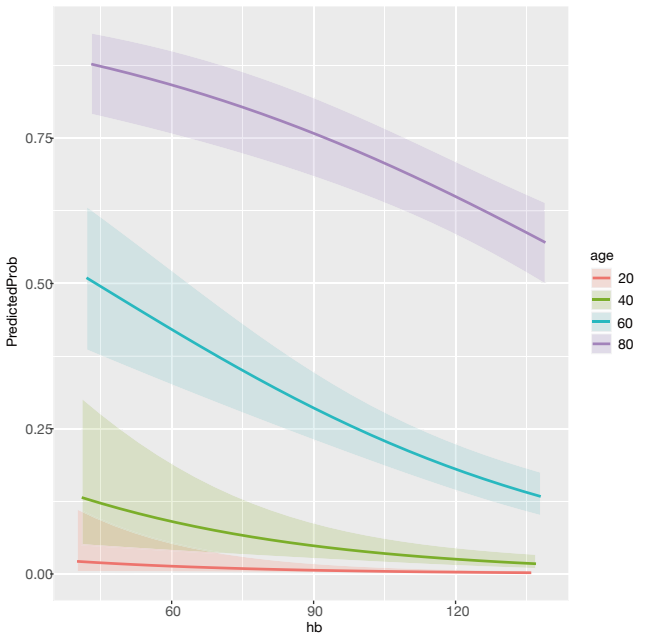


Figure 2 Effect of *hb* on the probability of mortality, stratified by different age groups.

While the probability of death increases with higher *age*, increasing *hb* is associated with greater mortality rate.

Step five: assessing fit of the model

The final step is to check the fit of the model. There are two components in checking for model fit: (I) summary measures of goodness of fit (GOF) and; (II) regression diagnostics. The former uses one summary statistics for the assessment of model fit, including the Pearson Chi-square statistic, deviance, sum-of-square, and the Hosmer-Lemeshow tests (12). These statistics measure the difference between observed and fitted values. Because Hosmer-Lemeshow test is the most commonly used measure for model fit assessment, it is illustrated in the following section.

```
> library(ResourceSelection)
> hoslem.test(model2$y, fitted(model2))

        Hosmer and Lemeshow goodness of fit (GOF) test

data:  model2$y, fitted(model2)
X-squared = 4.589, df = 8, p-value = 0.8005
```

The P value is 0.8, indicating that there is no significant difference between observed and predicted values. Model fit can also be examined in plots.

```
> Predprob<-predict(model2,type="response")
> plot(Predprob,jitter(as.numeric(mort),0.5),cex=0.5,
ylab="Jittered mortality outcome")
> library(Deducer)
> rocplot(model2)
> library(lattice)
> histogram(Predprob|mort)
```

Figure 3 is the plot of jittered outcome (alive=1; die=2) versus estimated probability of death from the fitted model. The classification of the model appears good that most survivors have an estimated probability of death less than 0.2. Conversely, most non-survivors have an estimated probability of death greater than 0.8. *Figure 4* is the histogram of estimated probability of death, stratified by observed outcomes. It also reflects the classification of the model. Survivors mostly have low estimated probability of death. *Figure 5* is the receiver operating characteristic curve (ROC) reflecting the discrimination power of the model. We consider it an outstanding discrimination when the area under ROC reaches above 0.9.

Summary

The article introduces how to perform model building by using purposeful selection method. The process of variable selection, deleting, model fitting and refitting can be repeated for several cycles, depending on the complexity of included variables. Interaction helps to disentangle complex relationships between covariates and their synergistic effects on the response variable. Model should be checked for the GOF. In other words, how the fitted model reflects the real data. Hosmer-Lemeshow GOF test is the most widely used method for the assessment of the logistic regression model. However, it is a summary statistic for checking model fit. Investigators may be interested in whether the model fits across entire range of covariate pattern, which is the task of regression diagnostics. This will be introduced in the next chapter.

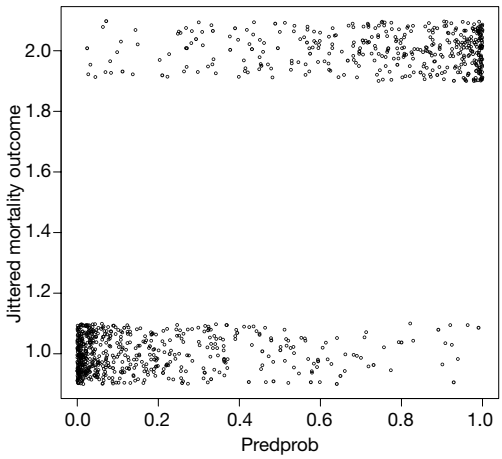


Figure 3 The plot of jittered outcome (alive=1; die=2) versus estimated probability of death from fitted model.

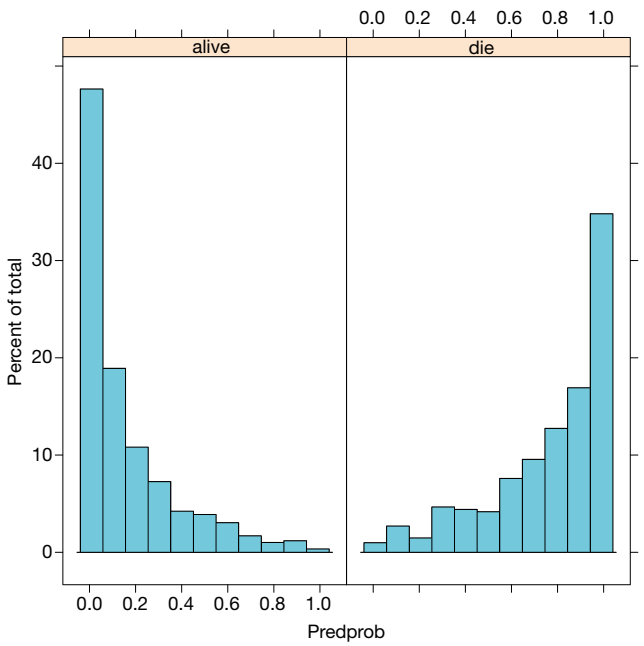


Figure 4 Histogram of estimated probability of death, stratified by observed outcome.

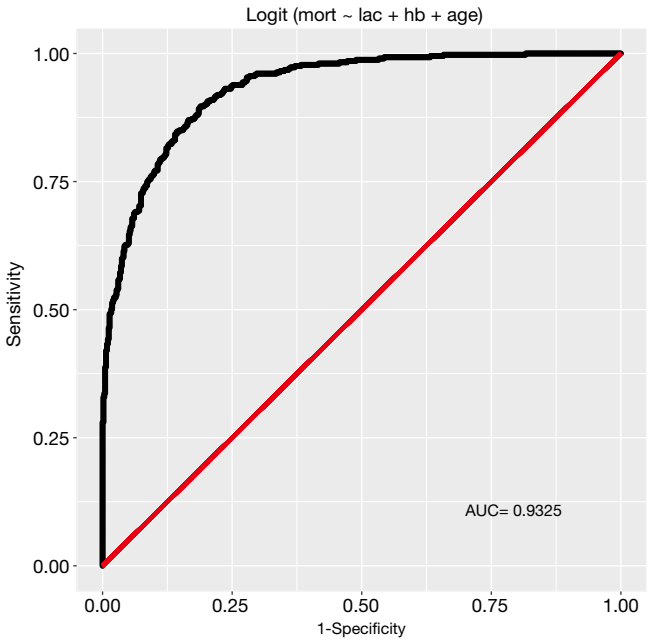


Figure 5 The receiver operating characteristic curve (ROC) reflecting the discrimination power of the model.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Bursac Z, Gauss CH, Williams DK, et al. Purposeful selection of variables in logistic regression. Source Code Biol Med 2008;3:17.
2. Greenland S. Modeling and variable selection in epidemiologic analysis. Am J Public Health 1989;79:340-349.
3. Model-building strategies and methods for logistic regression. In: Hosmer DW Jr, Lemeshow S, Sturdivant RX. Applied logistic regression. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2000;63.
4. Zhang Z, Chen K, Ni H, et al. Predictive value of lactate in unselected critically ill patients: an analysis using fractional polynomials. J Thorac Dis 2014;6:995-1003.
5. Zhang Z, Ni H. Normalized lactate load is associated with development of acute kidney injury in patients who underwent cardiopulmonary bypass surgery. PLoS One 2015;10:e0120466.
6. Zhang Z, Xu X. Lactate clearance is a useful biomarker for the prediction of all-cause mortality in critically ill patients: a systematic review and meta-analysis*. Crit Care Med 2014;42:2118-2125.
7. Kabacoff R. R in action. Cherry Hill: Manning Publications Co; 2011.
8. Bendal RB, Afifi AA. Comparison of stopping rules in

forward regression. Journal of the American Statistical Association 1977;72:46-53.
9. Mickey RM, Greenland S. The impact of confounder selection criteria on effect estimation. Am J Epidemiol 1989;129:125-137.
10. Royston P, Ambler G, Sauerbrei W. The use of fractional polynomials to model continuous risk variables in epidemiology. Int J Epidemiol 1999;28:964-974.
11. Royston P, Altman DG. Regression using fractional polynomials of continuous covariates: parsimonious parametric modelling. Applied Statistics 1994;43:429-467.
12. Hosmer DW, Hjort NL. Goodness-of-fit processes for logistic regression: simulation results. Stat Med 2002;21:2723-2738.

Cite this article as: Zhang Z. Model building strategy for logistic regression: purposeful selection. Ann Transl Med 2016;4(6):111. doi: 10.21037/atm.2016.02.15

Variable selection with stepwise and best subset approaches

Zhongheng Zhang

Abstract: While purposeful selection is performed partly by software and partly by human thoughts, the stepwise and best subset approaches are automatically performed by software. Two R functions stepAIC() and bestglm() are well designed for stepwise and best subset regression, respectively. The stepAIC() function begins with a full or null model, and methods for stepwise regression can be specified in the direction argument with character values "forward", "backward" and "both". The bestglm() function begins with a data frame containing explanatory variables and a response variable. The response variable should be in the last column. A variety of goodness-of-fit criteria can be specified in the IC argument. The Bayesian information criterion (BIC) usually results in more parsimonious model than the Akaike information criterion.

Keywords: Logistic regression; interaction; R; best subset; stepwise; Bayesian information criterion (BIC)

Submitted Dec 25, 2015. Accepted for publication Jan 24, 2016.
doi: 10.21037/atm.2016.03.35
View this article at: http://dx.doi.org/10.21037/atm.2016.03.35

Introduction

The previous article introduced purposeful selection for regression model, which allowed incorporation of clinical experience and/or subject matter knowledge into statistical science. There are several other methods for variable selection, namely, the stepwise and best subsets regression. In stepwise regression, the selection procedure is automatically performed by statistical packages. The criteria for variable selection include adjusted R-square, Akaike information criterion (AIC), Bayesian information criterion (BIC), Mallows's Cp, PRESS, and false discovery rate (1,2). Main approaches of stepwise selection are the forward selection, backward elimination and a combination of the two (3). The procedure has advantages if there are numerous potential explanatory variables, but it is also criticized for being a paradigmatic example of data dredging that significant variables may be obtained from "noise" variables (4,5). Clinical experience and expertise are not allowed in the model building process.

While the stepwise regression select variables sequentially, the best subsets approach aims to find out the best fit model from all possible subset models (2). If there are p covariates, the number of all subsets is 2^p. There are a variety of statistical methods to compare the fit of subset models. In this article, I will introduce how to perform stepwise and best subset procedures by using R.

Working example

The working example used in the tutorial is from the package MASS. You can take a look at the representation of each variable.

```
> library(MASS)
> head(bwt)
  low age lwt  race  smoke ptd   ht    ui    ftv
1 0   19  182  black FALSE FALSE FALSE TRUE  0
2 0   33  155  other FALSE FALSE FALSE FALSE 2+
3 0   20  105  white TRUE  FALSE FALSE FALSE 1
4 0   21  108  white TRUE  FALSE FALSE TRUE  2+
5 0   18  107  white TRUE  FALSE FALSE TRUE  0
6 0   21  124  other FALSE FALSE FALSE FALSE 0
```

The *bwt* data frame contains 9 columns and 189 rows. The variable *low* is an indicator variable with "0" indicates birth weight >2.5 kg and "1" indicates the presence of low birth weight. *Age* is mother's age in years. The variable *lwt* is mothers' weight in pounds. *Race* is mother's race and *smoke* is smoking status during pregnancy. The number of previous premature labor is *plt*. Other information includes history of hypertension (*bt*), presence of uterine irritability (*ui*), and the number of physician visits during the first trimester (*ftv*).

Stepwise selection

We can begin with a full model. Full model can be denoted by using the symbol "." on the right side of the formula.

```
> full <- glm(low ~ ., family = binomial, data = bwt)
> summary(full)
Call:
glm(formula = low ~ ., family = binomial, data = bwt)
Deviance Residuals:
Min        1Q      Median    3Q       Max
-1.7038  -0.8068  -0.5008   0.8835   2.2152
Coefficients:
Estimate      Std.Error   z        value   Pr(>|z|)
(Intercept)   0.82302     1.24471  0.661   0.50848
age           -0.03723    0.03870  -0.962  0.33602
lwt           -0.01565    0.00708  -2.211  0.02705*
raceblack     1.19241     0.53597  2.225   0.02609*
raceother     0.74069     0.46174  1.604   0.10869
smokeTRUE     0.75553     0.42502  1.778   0.07546.
ptdTRUE       1.34376     0.48062  2.796   0.00518**
htTRUE        1.91317     0.72074  2.654   0.00794**
uiTRUE        0.68019     0.46434  1.465   0.14296
ftv1          -0.43638    0.47939  -0.910  0.36268
ftv2+         0.17901     0.45638  0.392   0.69488
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Dispersion parameter for binomial family taken to be 1)
   Null deviance: 234.67  on 188  degrees of freedom
Residual deviance: 195.48  on 178  degrees of freedom
AIC: 217.48
Number of Fisher Scoring iterations: 4
```

As you can see in the output, all variables except *low* are included in the logistic regression model. Variables *lwt*, *race*, *ptd* and *ht* are found to be statistically significant at the conventional level. With the full model at hand, we can begin our stepwise selection procedure.

```
> step <- stepAIC(full, trace = FALSE)
> step$anova
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
low ~ age + lwt + race + smoke + ptd + ht + ui + ftv

Final Model:
low ~ lwt + race + smoke + ptd + ht + ui

  Step   Df   Deviance   Resid. Df Resid. Dev  AIC
1                        178       195.4755    217.4755
2 - ftv  2    1.358185   180       196.8337    214.8337
3 - age  1    1.017866   181       197.8516    213.8516
```

All arguments in the stepAIC() function are in their default settings. If you want to set direction of stepwise regression (e.g., backward, forward, both), the direction argument should be assigned. The default is both.

```
> forward<-stepAIC(full, direction="forward",trace = FALSE)
> forward$anova
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
low ~ age + lwt + race + smoke + ptd + ht + ui + ftv

Final Model:
low ~ age + lwt + race + smoke + ptd + ht + ui + ftv

  Step   Df   Deviance   Resid.Df   Resid.Dev   AIC
1                        178        195.4755    217.4755
```

Because the forward stepwise regression begins with full model, there are no additional variables that can be added. The final model is the full model. Forward selection can begin with the null model (intercept only model).

```
> backward<-stepAIC(full, direction="backward",trace =
FALSE)
> backward$anova
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
low ~ age + lwt + race + smoke + ptd + ht + ui + ftv
```

```
Final Model:
low ~ lwt + race + smoke + ptd + ht + ui
```

Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
1			178	195.4755	217.4755
2 –ftv	2	1.358185	180	196.8337	214.8337
3 –age	1	1.017866	181	197.8516	213.8516

The backward elimination procedure eliminates variables *ftv* and *age*, which exactly matches the "both" procedure.

Different criteria can be assigned to the stepAIC() function for stepwise selection. The default is AIC, which is performed by assigning the argument k to 2 (the default option).

```
> BIC<-stepAIC(full,k=log(nrow(bwt))) #BIC method
```

The stepAIC() function also allows specification of the range of variables to be included in the model by using the *scope* argument. The lower model is the model with smallest number of variables and the upper model is the largest possible model. Both upper and lower components of the scope can be explicitly specified. If scope is a single formula, it specifies the upper component, and the lower model is empty. If scope is missing, the initial model is the upper model.

```
> scope<-stepAIC(full,scope=list(lower=~smoke+age,
upper=full),trace=FALSE)
> scope$anova
Stepwise Model Path
Analysis of Deviance Table
Initial Model:
low ~ age + lwt + race + smoke + ptd + ht + ui + ftv

Final Model:
low ~ age + lwt + race + smoke + ptd + ht + ui
```

Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
1			178	195.4755	217.4755
2 - ftv	2	1.358185	180	196.8337	214.8337

When I specify the smallest model to include *age* variable, it will not be excluded by stepwise regression (e.g., otherwise, the age will be excluded as shown above). This function can help investigators to keep variables that are considered to be relevant by subject-matter knowledge. Next, we can have more complicated models for stepwise selection.

```
> step2 <- stepAIC(full, ~ .^2 + I(scale(age)^2)+ I(scale(lwt)^2),
trace = FALSE)
> step2$anova
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
low ~ age + lwt + race + smoke + ptd + ht + ui + ftv

Final Model:
low ~ age + lwt + smoke + ptd + ht + ui + ftv + age:ftv + smoke:ui
```

Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
1			178	195.4755	217.4755
2 +age:ftv	2	12.474896	176	183.0006	209.0006
3 +smoke:ui	1	3.056805	175	179.9438	207.9438
4 -race	2	3.129586	177	183.0734	207.0734

Recall that the "^" symbol denotes interactions up to a specified degree. In our case, we specified two-degree interactions among all possible combinations of variables. Elements within I() are interpreted arithmetically. The function scale(age) centers variable age at its mean and scales it by standard deviation. "~ .^2 + I(scale(age)^2)+ I(scale(lwt)^2)" is the scope argument and a single formula implies the upper component. The results show that the interactions between *age* and *ftv*, *smoke* and *ui* are remained in the final model. Other interactions and quadratic terms are removed.

Best subset regression

Best subset regression selects the best model from all possible subsets according to some goodness-of-fit criteria. This approach has been in use for linear regression for several decades with the branch and bound algorithm (6). Later on, lawless and Singhal proposed an extension that could be used for non-normal error models (7). The application of best subsets for logistic regression model was described by Hosmer and coworkers (8). An R package called "bestglm" contains functions for performing best subsets selection.

The bestglm() employs simple exhaustive searching algorithm as described by Morgan (9).

```
> library(leaps) # bestglm requires installation of leaps pack-
age
> library(bestglm)
> args(bestglm)
function (Xy, family = gaussian, IC = "BIC", t = "default", CVArgs
= "default",
    qLevel = 0.99, TopModels = 5, method = "exhaustive", inter-
cept = TRUE,
   weights = NULL, nvmax = "default", RequireFullEnumerationQ
= FALSE,
   ...)
```

Xy is a data frame containing independent variables and the dependent variable. For logistic regression model when family is set to be binomial, the last column is the response variable. The sequence of Xy is important because a formula to specify the dependent and independent variables are not allowed with bestglm() function. We can move the response variable low to the last column and give a new name to the new data frame. The IC argument specifies the information criteria to use. Its values can be "AIC", "BIC", "BICg", "BICq", "LOOCV" and "CV" (10).

```
> bwt.move<-bwt[,-1]
> bwt.move$low<-bwt$low
```

Furthermore, factors with more than two levels should be converted to dummy variables. Otherwise, it returns an error message.

```
> library(dummies)
> race<-data.frame(dummy(bwt$race)[,c(1,2)])
> ftv<-data.frame(dummy(bwt$ftv)[,c(2,3)])
> bwt.dummy<-bwt[,-c(1,4,9)]
> low<-bwt$low
> bwt.dummy<-cbind(bwt.dummy,race,ftv, low)
```

To create dummy variables for factors with more than two levels, we use the *dummies* package. The dummy() function passes a single variable and returns a matrix with the number of rows equal to that of the given variable, and the number of columns equal to the number of levels of that variable. Because only n-1 dummy variables are needed to define a factor with n levels, I remove the base level by simple manipulation of vectors. Finally, a new data frame containing dummy variables is created, with the response variable in the last column.

```
> library(bestglm)
> bestglm(bwt.dummy,IC="BIC",family=binomial)
Morgan-Tatar search since family is non-gaussian.
BIC
BICq equivalent for q in (0.420012802643247,
0.585022045845753)
Best Model:
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	1.01736672	0.85333672	1.192222	0.233174251
lwt	-0.01728027	0.00678715	-2.546028	0.010895659
ptdTRUE	1.40676981	0.42850088	3.283003	0.001027075
htTRUE	1.89397147	0.72108967	2.626541	0.008625764

The model selection by AIC always keeps more variables in the model as follows.

```
> bestglm(bwt.dummy,IC="AIC",family=binomial)
Morgan-Tatar search since family is non-gaussian.
AIC
BICq equivalent for q in (0.797717473187408,
0.882261427360355)
Best Model:
```

	Estimate	Std.Error	zvalue	Pr(>|z|)
(Intercept)	0.64059802	0.859552154	0.7452695	0.456108807
lwt	-0.01424899	0.006583754	-2.1642657	0.030443965
smokeTRUE	0.92821842	0.398653203	2.3283857	0.019891632
ptdTRUE	1.12007778	0.450882008	2.4841927	0.012984553
htTRUE	1.85222596	0.705829936	2.6241816	0.008685745
uiTRUE	0.73543662	0.461731565	1.5927796	0.111209644
race.white	-1.01303877	0.396054355	-2.5578276	0.010532828

Readers can try other options available in the bestglm() function. Different options may result in different models.

Summary

The article introduces variable selection with stepwise and best subset approaches. Two R functions stepAIC() and bestglm() are well designed for these purposes. The stepAIC() function begins with a full or null model, and methods for stepwise regression can be specified in the

direction argument with the character values "forward", "backward" and "both". The bestglm() function begins with a data frame containing explanatory variables and a response variable. The response variable should be in the last column. Factor variables with more than two levels should be converted before running the bestglm(). The dummies package contains good function to convert factor variable to dummy variables. There is a variety of information criteria for the selection of the best subset model.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Hocking RR. A biometrics invited paper. The analysis and selection of variables in linear regression. Biometrics 1976;32:1-49.
2. Hosmer DW Jr, Lemeshow S, Sturdivant RX. Applied Logistic Regression. Hoboken: John Wiley & Sons, Inc, 2013.
3. Harrell FE. Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis. New York: Springer-Verlag New York, 2013.
4. Freedman DA. A note on screening regression equations. The American Statistician 1983;37:152-155.
5. Flack VF, Chang PC. Frequency of selecting noise variables in subset regression analysis: a simulation study. The American Statistician 1987;41:84-86.
6. Furnival GM, Wilson RW. Regressions by leaps and bounds. Technometrics 1974;16:499-511.
7. Lawless JF, Singhal K. ISMOD: an all-subsets regression program for generalized linear models. II. Program guide and examples. Comput Methods Programs Biomed 1987;24:125-134.
8. Hosmer DW, Jovanovic B, Lemeshow S. Best subsets logistic regression. Biometrics 1989;45:1265-1270.
9. Morgan JA, Tatar JF. Calculation of the residual sum of squares for all possible regressions. Technometrics 1972;14:317-325.
10. McLeod AI, Changjiang X. bestglm: best subset GLM.–R package ver. 0.34. 2014. Available online: https://cran.r-project.org/web/packages/bestglm/bestglm.pdf

Cite this article as: Zhang Z. Variable selection with stepwise and best subset approaches. Ann Transl Med 2016;4(7):136. doi: 10.21037/atm.2016.03.35

Multivariable fractional polynomial method for regression model

Zhongheng Zhang

Submitted Dec 25, 2015. Accepted for publication Jan 24, 2016.
doi: 10.21037/atm.2016.05.01
View this article at: http://dx.doi.org/10.21037/atm.2016.05.01

Introduction

One assumption in creating generalized linear model (GLM) is the linearity in its link function. For example, in logistic regression model, covariates are assumed to be linearly associated with response variable in logit scale. However, it is not always the case and the assumption may be wrong. For example, lactate is associated with mortality outcome, but the relationship is not linear (1). Quadratic or cubic terms can be added to an explanatory variable to account for the non-linearity relationship. However, this requires subject-matter knowledge to determine the form of a variable. In exploratory study, such knowledge is always lacking and investigators have to rely on data to determine the functional form. Multivariable fractional polynomial (MFP) method is such a method that it allows software to determine whether an explanatory variable is important for the model and its functional form (2,3). MFP can be used when investigators want to preserve continuous nature of covariates and suspect that the relationship is non-linear. The article aims to describe how to perform MFP methods. Fundamentals on MFP are also provided to make the tutorial article more coherent.

Fundamentals on multivariable fractional polynomial (MFP)

There are two components in the procedure: (I) backward elimination of covariates that are statistically insignificant; and (II) iterative examination of the scale of all continuous covariates. Therefore, we need two significant levels, α1 for the exclusion and inclusion of a covariate, and α2 for the determination of significance of fractional transformation of the continuous covariates (4,5).

The first cycle is to build a multivariable model with all potential explanatory covariates (*Figure 1*). Alternatively, variables with P<0.25 or 0.2 in univariable analysis can be incorporated into the initial model. This is also the starting model for purposeful selection of covariates. All dichotomous and design variables are not subject to fractional polynomial (FP) transformation and are modeled with one degree of freedom. They are tested for their contribution to the model by using α1 (e.g., by Wald test). Continuous variables are modeled by using closed test to examine whether they should be kept or removed using α1, and whether transformation should be performed using α2 (*Figure 2*). The closed test begins by comparing the best-fitting second-degree fractional polynomial (FP2) with null model (*Table 1*). The term is dropped if the test is non-significant. Otherwise the best-fitting FP2 is compared with the linear term. Linear term is adopted if the test is non-significant. Otherwise we continue to compare the best-fitting FP2 to the best-fitting FP1. If the test is significant, the best fitting FP2 is adopted. Otherwise the best-fitting FP1 is adopted (6). The second cycle begins with a fit of model containing significant covariates, either in original or polynomial transformed form. All covariates are then examined in descending order of significance for their inclusion, exclusion and possible transformation. The procedure stops when two consecutive steps contain the same covariates with the same FP transformations.

Working example

In this article, the German Breast Cancer Study Group (GBSG) database is used for the illustration of the MFP method. GBSG dataset in the *mfp* package contains 686 rows and 11 columns.

```
> library(survival)
> library(mfp)
> data(GBSG)
> str(GBSG)
'data.frame':          686 obs. of  11 variables:
 $ id    : int  1 2 3 4 5 6 7 8 9 10 ...
```

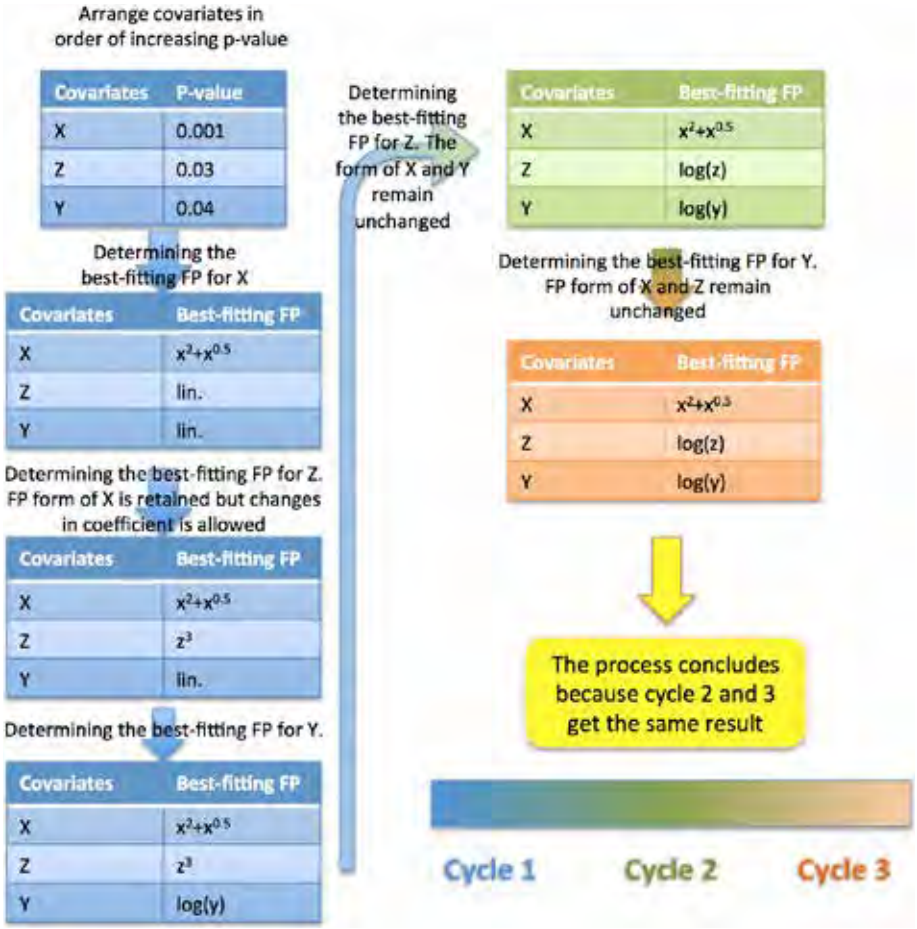


Figure 1 A simple example illustrating the procedure of the multivariable fractional polynomial method. Firstly, predictors are arranged in descending order of significance. The most important predictor is assessed by closed test for its inclusion and fractional polynomial (FP) function. The procedure proceeds sequentially for all predictors. The second cycle begins by examining the second predictor for its inclusion and FP function. Other predictors are kept in their FP form obtained from cycle 1. The procedure concludes when two cycles converge.

```
$ htreat  : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 2 1 1 1 ...
$ age     : int  70 56 58 59 73 32 59 65 80 66 ...
$ menostat: Factor w/ 2 levels "1","2": 2 2 2 2 2 1 2 2 2 2 ...
$ tumsize : int  21 12 35 17 35 57 8 16 39 18 ...
$ tumgrad : Factor w/ 3 levels "1","2","3": 2 2 2 2 2 3 2 2 2 2 ...
$ posnodal: int  3 7 9 4 1 24 2 1 30 7 ...
$ prm     : int  48 61 52 60 26 0 181 192 0 0 ...
$ esm     : int  66 77 271 29 65 13 0 25 59 3 ...
$ rfst    : int  1814 2018 712 1807 772 448 2172 2161 471 2014 ...
$ cens    : int  1 1 1 1 1 1 0 0 1 0 ...
```

The variable *id* is to identify unique patient. Hormonal therapy (*htreat*) is a factor with two levels of no [0] and yes [1]. Menopausal status (*menostat*) is also a factor at two levels premenopausal [1] and postmenopausal [2]. Tumor size (*tumsize*) is a continuous variable measured in millimeter. Tumor grade (*tumgrad*) is an ordered factor at levels 1<2<3. Number of positive nodes (*posnodal*) is a continuous variable with integer values. Progesterone receptor (*prm*) is an integer variable measured in fmol. Estrogen receptor (*esm*) is an integer variable measured in fmol. Recurrence free survival time (*rfst*) is measured in days. Censoring indicator (*cens*) is an integer with 0 indicates censored and 1 for event.

Illustration of multivariable fractional polynomial (MFP) method

The GBSG dataset is a survival data and the model is

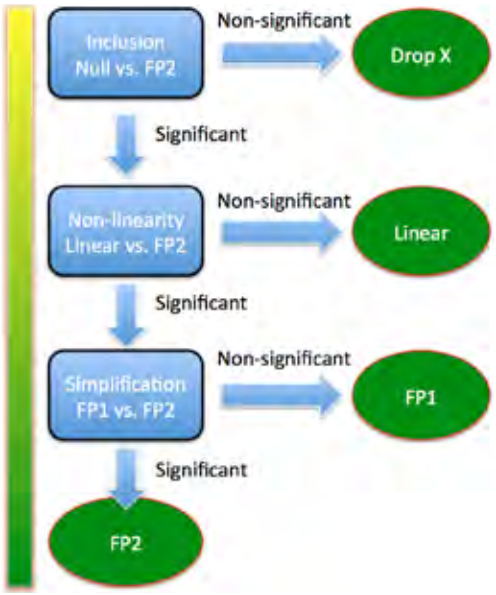


Figure 2 Closed test algorithm for choosing a fractional polynomial model with maximum permitted degree of 2 for a single continuous predictor. The first step is to determine whether a predictor should be included in a model. That is to compare models with and without FP2. If FP2 model is not better than null model, the predictor is dropped. Otherwise, we continue to compare FP2 with linear model. If FP2 is not better than linear one, we choose linear model. Otherwise, we continue to compare FP2 with FP1. If FP2 is not better than FP1, the FP1 model is chosen. Otherwise the FP2 model is chosen. FP, fractional polynomial.

Table 1 Illustration of fractional polynomial terms

Groups	Notations	m	Examples
Linear	lin.		x
First-degree FP	FP1	1	x^2; x^3; $x^{0.5}$; x^{-2}
Second-degree FP	FP2	2	$x+x^2$; $x^{0.5}+x^2$; $x^{-0.5}+x^2$; $x^{0.5}+x^{-2}$
Third-degree FP	FP3	3	$x^{0.5}+x^2+x^{-1}$; $x^{-0.5}+x^2+x$

FP of a certain degree contains numerous terms, depending on the number of powers allowed. By convention, powers are selected from the collection (–2, –1, –0.5, 0, 0.5, 1, 2, 3), where 0 denotes the log transformation. FP3 is usually not needed, and I present it here for better understanding of the fractional polynomial term. FP2 is the most complex and it is compared to the null model. If FP2 is not better than null by statistical test, linear and FP1 of the variable are unlikely to be important to the model. Therefore, the variable is excluded from the model. FP, fractional polynomial.

constructed with survival function. The Surv() function creates a survival object with the time and event as arguments. To make the model simple, only *age* and *prm* are selected for FP transformation. FP2 terms are allowed for *prm* and only FP1 is allowed for *age*. By default argument, FP2 is allowed for *tumsize*. The remaining variables *htreat* and *tumgrad* are linear because they are categorical. The model is built with Cox proportional hazard model by assigning "family = cox".The "verbose=TRUE" argument is to show variable selection details in output.

```
> model<-mfp(Surv(rfst, cens) ~ fp(age, df = 2, select = 0.05)+fp(prm,
df = 4, select = 0.05)+htreat+fp(tumsize)+tumgrad, family = cox,
data = GBSG,verbose=TRUE)

Variable                    Deviance          Power(s)
-----------------------------------------------------
Cycle  1
          prm
                            3530.308
                            3505.751          1
                            3497.597          0.5
                            3495.593          0   0
          tumsize
                            3511.629
                            3497.597          1
                            3495.599          -0.5
                            3493.696          -1   3
          tumgrad2
                            3505.159
                            3497.597          1
          tumgrad3
                            3504.295
                            3497.597          1
          htreat1
                            3504.472
                            3497.597          1
          age
                            3497.698
                            3497.597          1
                            3494.907          -2

Cycle  2
          prm
                            3530.344
                            3505.865          1
```



```
                    3497.698        0.5
                    3495.679        0  0
          tumsize
                    3511.69
                    3497.698        1
                    3495.647        -0.5
                    3493.746        -1  3
          tumgrad2
                    3505.254
                    3497.698        1
          tumgrad3
                    3504.36
                    3497.698        1
          htreat1
                    3504.525
                    3497.698        1


Tansformation
                    shift           scale
prm                 1               100
tumsize             0               10
tumgrad2            0               1
tumgrad3            0               1
htreat1             0               1
age                 0               100

Fractional polynomials
          df.initial select  alpha     df.final  power1  power2
prm       4          0.05    0.05      2         0.5     .
tumsize   4          1.00    0.05      1         1       .
tumgrad2  1          1.00    0.05      1         1       .
tumgrad3  1          1.00    0.05      1         1       .
htreat1   1          1.00    0.05      1         1       .
age       2          0.05    0.05      0         .       .

Transformations of covariates:
                    formula
age                 <NA>
prm                 I(((prm+1)/100)^0.5)
htreat              htreat
tumsize             I((tumsize/10)^1)
tumgrad             tumgrad

Deviance table:
                    Resid. Dev
Null model          3576.209
Linear model        3505.751
Final model         3497.698
```

The first cycle begins by including all covariates into the model and their FP functions are examined. The best-ftting FP functions are shown in the output. For example, the power of best-fitting FP1 is 0.5 for *prm*, and the powers of best-fitting FP2 are -1 and 3 for *tumsize*. The statistical test is performed internally and not shown in the output. In cycle 2, *age* is dropped because the FP1 function is not significantly different from the null model (deviance: 3,494.907 *vs.* 3,497.698). Cycle 2 is the last cycle where the models converge. Transformation of each variable is shown. The variable *prm* is shifted by 1 and divided by 100 before FP transformation. FP functions of each variable remained in the model are shown in the output. *Age* is dropped. All variables are entered in linear form except that *prm* is transformed by FP1 with the power of 0.5. One can see P values of closed test procedure by the following code.

```
> model$pvalues
          p.null          p.lin       p.FP        power2 power4.1 power4.2
prm       5.442815e-07    0.0170453   0.3643537   0.5    0        0
tumsize   1.265929e-03    0.2667106   0.3865315   -0.5   -1       3
tumgrad2  5.980843e-03    NA          NA          NA     NA       NA
tumgrad3  9.851749e-03    NA          NA          NA     NA       NA
htreat1   8.978825e-03    NA          NA          NA     NA       NA
age       2.477346e-01    0.1009907   NA          -2.0   NA       NA
```

In the output, p.null corresponds to the test of inclusion (e.g., comparing best-fitting FP2 against null model); p.lin is the P value for the test of nonlinearity (comparing best-fitting FP2 against linear model) and p.FP is the test of simplification by comparing first degree (FP1) and second degree (FP2) transformations. The best-fitting FP1 power (power2) and best-fitting FP2 powers (power4.1 and power4.2) are also shown. The numbers 2 and 4 describe the corresponding degrees of freedom.

Next, users are interested in the estimated coefficient for each transformed variable.

```
> model$fit
```

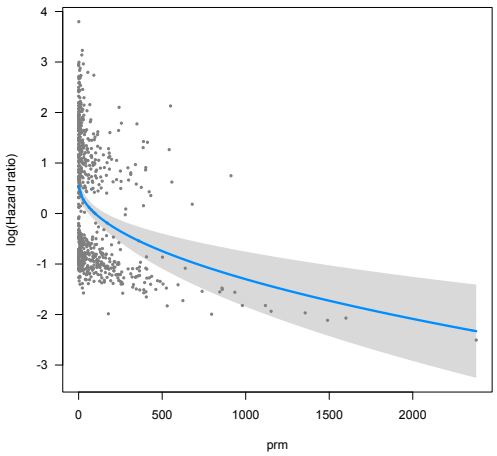



Figure 3 Visualization of non-linear model. Note that the variable prm is not linearly associated with log(hzazrd ratio).

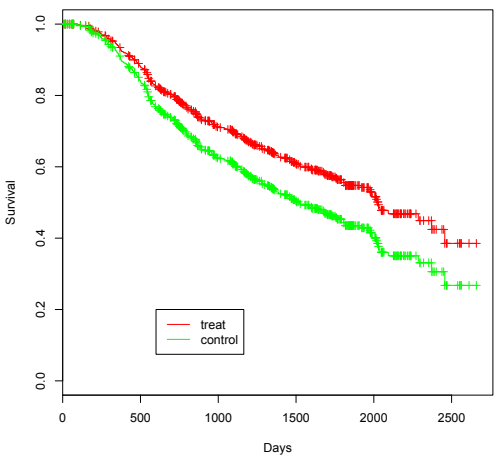


Figure 4 Survival curves for patients with and without hormone therapy, given that the tumor size is 20 mm, the progesterone receptor is 30 fmol, and the tumor grade is 2.

```
Call:
coxph(formula = Surv(rfst, cens) ~ I(((prm + 1)/100)^0.5) +
I((tumsize/10)^1) + tumgrad + htreat, data = GBSG)
```

	coef	exp(coef)	se(coef)	z	p
I(((prm+1)/100)^0.5)	-0.6003	0.5487	0.1145	-5.24	1.6e-07
I((tumsize/10)^1)	0.1442	1.1552	0.0364	3.97	7.3e-05
tumgrad2	0.6342	1.8856	0.2503	2.53	0.011
tumgrad3	0.6700	1.9543	0.2737	2.45	0.014
htreat1	-0.3237	0.7235	0.1261	-2.57	0.010

```
Likelihood ratio test=78.5  on 5 df, p=1.67e-15
n= 686, number of events= 299
```

In the output of "model$fit", one can see the final model is called by the coxph() function. The coefficient for each transformed variable is shown in the table. Exponentiation of coefficient is the hazard ratio. The transformed variable is sometimes obscure to subject-matter audience. Thus, the Visualization of how hazard ratio changes with variable of FP function is interesting.

```
> library(visreg)
> visual<-coxph(formula = Surv(rfst, cens) ~ I(((prm + 1)/100)^0.5)
+ I((tumsize/10)^1) + tumgrad + htreat, data = GBSG)
> visreg(visual,"prm", ylab="log(Hazard ratio)" )
```

For the purpose of visualization of fitted regression model, the *visreg* package is employed. The visreg() function cannot directly receive returned object from mfp() and I refit the Cox model in the form exactly the same to that obtained from mfp(). Then the new model can be visualized using visreg() function (*Figure 3*). Sometimes, users are interested in visualization of survival curves at fixed covariates. For example, we can plot survival curves for patients with and without hormone therapy, given that the tumor size is 20 mm, progesterone receptor is 30 fmol, and tumor grade is 2 (*Figure 4*).

```
> plot(survfit(model$fit, newdata=data.frame(prm=30,tumsize=20,
tumgrad="2",htreat=c("1","0"))), col=c("red","green"),
xlab = "Days", ylab="Survival")
> legend(600, .2, c("treat", "control"), lty = 1,col=c("red","green"))
```

The survfit() function creates survival curves from the previously fitted Cox model, here it is *model$fit*. The argument *newdata* specifies values of covariates to be plotted. The function legend() is used to add legend to make the plot readable.

Summary

The article introduces how to perform MFP procedure by using the mfp() function. Users can define which variable is subject to FP transformation and in what degree. The procedure firstly arranges potential predictors in order of decreasing significance (increasing P value). The purpose is to consider the relatively important variables before unimportant ones. Secondly, predictors are considered consecutively for their best-fitting FP function. The

closed test algorithm is employed to choose a best-fitting FP function. Predictors of interest can be visualized by using the visreg() function. Visualization the of continuous predictors is important because coefficients of high-order terms are difficult to understand for subject-matter audience.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Zhang Z, Chen K, Ni H, et al. Predictive value of lactate in unselected critically ill patients: an analysis using fractional polynomials. J Thorac Dis 2014;6:995-1003.
2. Royston P, Ambler G, Sauerbrei W. The use of fractional polynomials to model continuous risk variables in epidemiology. Int J Epidemiol 1999;28:964-974.
3. Royston P, Altman DG. Regression Using Fractional Polynomials of Continuous Covariates: Parsimonious Parametric Modelling. Applied Statistics 1994;43:429-467.
4. Royston P, Sauerbrei W. Building multivariable regression models with continuous covariates in clinical epidemiology--with an emphasis on fractional polynomials. Methods Inf Med 2005;44:561-571.
5. Sauerbrei W, Royston P, Binder H. Selection of important variables and determination of functional form for continuous predictors in multivariable model building. Stat Med 2007;26:5512-5528.
6. Marcus R, Eric P, Gabriel KR. On closed testing procedures with special reference to ordered analysis of variance. Biometrika 1976;63:655-660.

Cite this article as: Zhang Z. Multivariable fractional polynomial method for regression model. Ann Transl Med 2016;4(9):174. doi: 10.21037/atm.2016.05.01

Residuals and regression diagnostics: focusing on logistic regression

Zhongheng Zhang

Abstract: Up to now, most steps in regression model building and validation has been introduced. The last step is to check whether there are observations that have significant impact on model coefficient and specification. The article firstly describes plotting Pearson residual against predictors. Such plots are helpful in identifying non-linearity and provide hints on how to transform predictors. Next, I focus on observations of outlier, leverage and influence that may have significant impact on model building. Outlier is such an observation that its response value is unusual conditional on covariate pattern. Leverage is an observation with covariate pattern that is far away from the regressor space. Influence is the product of the outlier and leverage. That is, when influential observation is dropped from the model, there will be a significant shift of the coefficient. Summary statistics for outlier, leverage and influence are studentized residuals, hat values and Cook's distance, respectively. They can be easily visualized with graphs and formally tested using functions in the car package.

Keywords: Regression diagnostics; R; leverage; influence; outlier; Pearson residual; studentized residual

Submitted Jan 15, 2016. Accepted for publication Feb 02, 2016.
doi: 10.21037/atm.2016.03.36
View this article at: http://dx.doi.org/10.21037/atm.2016.03.36

Introduction

In previous sections, I introduced several means to select variables (best subset, purposeful selection and stepwise regression) to check for their linearity (multivariable fractional polynomials) and assess for overall fit (Homser-Lemeshow goodness of fit) of the model. That is not the end of the story. The last step is to check individual observations that are unusual. Summary statistics based on Pearson chi-square residuals describe the overall agreement between observed and fitted values. Plotting of residuals against individual predictors or linear predictor is helpful in identifying non-linearity. They are also indicative of which variable needs transformation, and what kind of transformation (e.g., quadratic or cubic?) should be assigned. Regression diagnostics aim to identify observations of outlier, leverage and influence. These observations may have significant impact on model fitting and should be examined for whether they should be included. Sometimes, these observations may be the result of typing error and should be corrected. The article provides a detailed discussion on how to perform these analyses using R. I primarily focus on R functions to implement analysis and the interpretation of the output results. Detailed mathematical equations and statistical theories are omitted.

Working example

I use the Mroz dataset (U.S. Women's Labor-Force Participation) from car package as a working example (1).

```
> library(car)
> str(Mroz)
'data.frame':	753 obs. of  8 variables:
 $ lfp : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ k5  : int  1 0 1 0 1 0 0 0 0 0 ...
 $ k618: int  0 2 3 3 2 0 2 0 2 2 ...
 $ age : int  32 30 35 34 31 54 37 54 48 39 ...
 $ wc  : Factor w/ 2 levels "no","yes": 1 1 1 1 2 1 2 1 1 1 ...
 $ hc  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ lwg : num  1.2102 0.3285 1.5141 0.0921 1.5243 ...
 $ inc : num  10.9 19.5 12 6.8 20.1 ...
```

The data frame contains 753 observations and 8 variables. It is a study exploring labor-force participation of married women. Labor-force participation (lfp) is a factor with two levels of yes and no. The number of children younger than 5 years old (k5), and the number of children older than 6 years old (k618) are recorded. Other information includes

women's age (age), wife's college attendance (wc), husband's college attendance (hc), log expected wage rate (lwg), and family income exclusive of wife's income (inc).

Residual plotting

There are many types of residuals such as the ordinary residual, Pearson residual, and Studentized residual. They all reflect the differences between fitted and observed values, and are the basis of a variety of diagnostic methods. Technical details of these residuals will not be discussed in this article, and interested readers can refer to other references and books (2-4). This article primarily aims to describe how to perform model diagnostics using R. A basic type of graph is to plot residuals against predictors or fitted values. If a model is properly fitted, there should be no correlation between residuals and predictors and fitted values. At best, the trend is a horizontal straight line without curvature. Let's take a look at the following example.

```
> mroz.mod<-glm(lfp~k5+k618+age+wc+hc+lwg+inc,
family=binomial,Mroz)
> residualPlots(mroz.mod)
```

	Test stat	Pr(>|t|)
k5	0.116	0.734
k618	0.157	0.692
age	1.189	0.275
wc	NA	NA
hc	NA	NA
lwg	153.504	0.000
inc	3.546	0.060

The default residual for generalized linear model is the Pearson residual. *Figure 1* plots the Pearson's residual against predictors one by one and the last plot is against the predicted values (linear predictor). Note that the relationship between Pearson residuals and the variable *lwg* is not linear and there is a trend. Visual inspection is only a rough estimation and cannot be used as a rule to modify the model. Fortunately, the residualPlots() function performs formal statistical testing (lack-of-fit test) to see if a variable has relationship with the residuals. The test is performed by adding a squared variable to the model, and to examine whether the term is statistically significant. This is much like the *linktest* in Stata. The idea is that if the model is properly specified, no additional predictors that are statistically significant can be found. The test shows that *lwg* is significant and therefore adding a quadratic term for *lwg* is reasonable.

```
> mroz.mod2<-glm(lfp~k5+k618+age+wc+hc+lwg+
I(lwg^2)+inc,family=binomial,Mroz)
> residualPlots(mroz.mod2)
```

	Test stat	Pr(>|t|)
k5	0.258	0.611
k618	0.129	0.719
age	1.960	0.162
wc	NA	NA
hc	NA	NA
lwg	0.000	1.000
I(lwg^2)	0.266	0.606
Inc	3.042	0.081

In the new model, a quadratic term is added and this term is statistically significant. Lack-of-fit test of the I(lwg^2) is non-significant, suggesting a properly specified model (*Figure 2*). Another way to investigate the difference between observed and fitted value is the marginal model plot (*Figure 3*). Response variable (lfp) is plotted against explanatory variable. Observed data and model prediction are shown in blue and red lines. Consistent with the previous finding, the variable *lwg* fits poorly and requires modification.

```
> marginalModelPlots(mroz.mod)
```

The above-mentioned methods only reflect the overall model fit. It is still unknown whether the fit is supported over the entire set of covariate patterns. This can be accomplished by using regression diagnostics. In other words, regression diagnostics is to detect unusual observations that have significant impact on the model. The following sections will focus on a single or subgroup of observations and introduce how to perform analysis on outliers, leverages and influences.

Outliers

Outlier is defined as an observation with a response value that is unusual conditional on covariate patterns (5). For

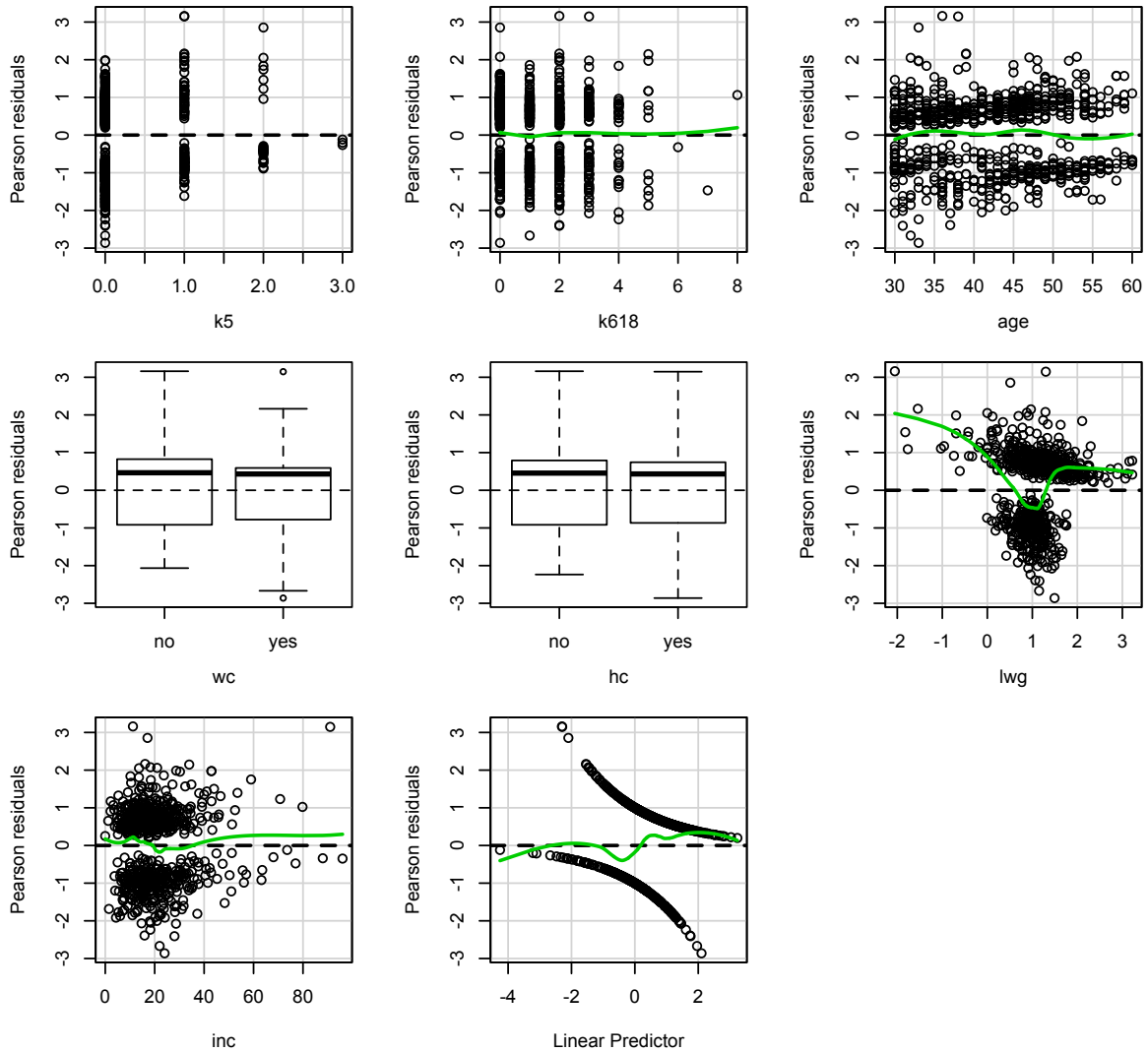


Figure 1 Pearson residuals are plotted against predictors one by one. Note that the relationship between Pearson residuals and the variable lwg is not linear and there is a trend.

example, patients over 80 years old with circulatory shock (hypotension) and renal failure are very likely to die. If the one with these characteristics survives, it is an outlier. Such outlier may have significant impact on model fitting. Outlier can be formally examined using Studentized residuals.

```
> outlierTest(mroz.mod)

No Studentized residuals with Bonferonni p < 0.05
Largest |rstudent|:
     rstudent    Unadjusted p-value    Bonferonni p
119   2.25002      0.024448                NA
```

The results show that there is no outlier as judged by Bonferonni p.

Leverage

Observations that are far from the average covariate pattern (or regressor space) are considered to have high leverage. For example, examinees taking part in the Chinese college entrance examination age between 18 to 22 years old. In this sample, a 76-year-old examinee is considered to have high leverage. Leverage is expressed as the hat value. The

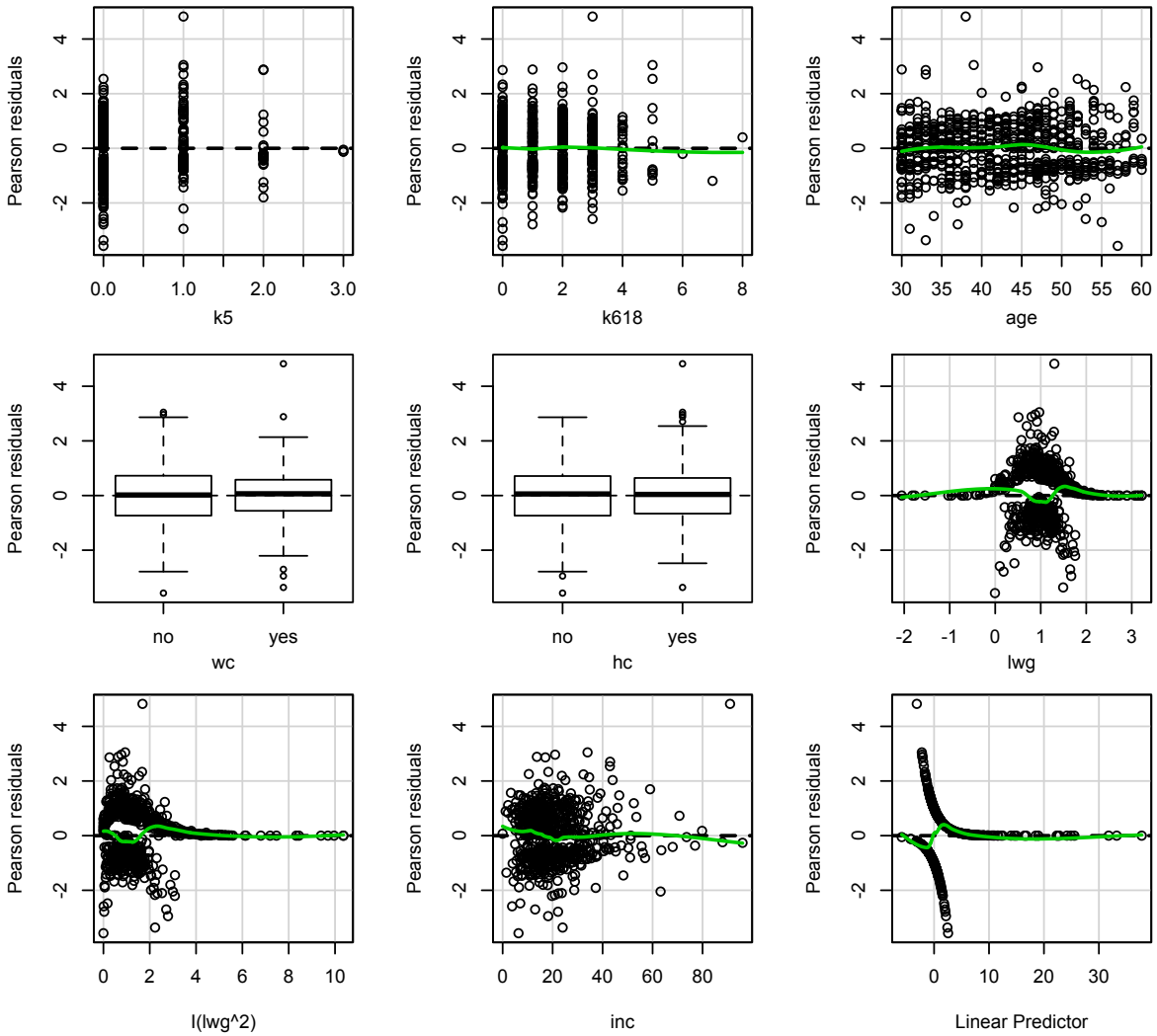


Figure 2 Residual plots showing that after adding a quadratic term to variable lwg, both 1- and 2-degree terms show a flat trend.

hat value of each observation can be obtained using the hatvalues() function from *car* package. There is also another useful plot function to produce graphs of hat values and other relevant values.

```
> influenceIndexPlot(mroz.mod, id.n=3)
```

The id.n=3 argument tells the function to denote three observations that are the farthest away from the average. The results show that the observations 348, 386 and 53 have the largest hat values (*Figure 4*). Observations 119, 220 and 502 are most likely to be outliers (e.g., they have the largest studentized residuals), but all Bonferonni P values are close to 1. Cook's distance will be discussed later.

Influence

If removal of an observation causes substantial change in the estimates of coefficients, the observation is called influential observation. Influence can be regarded as the product of leverage and outlier (e.g., it has high hat value and response value is unusual conditional on the covariate pattern). Cook's distance is a summary measure of the influence (6). A large value of Cook's distance indicates an influential observation. Cook's distance can be examined

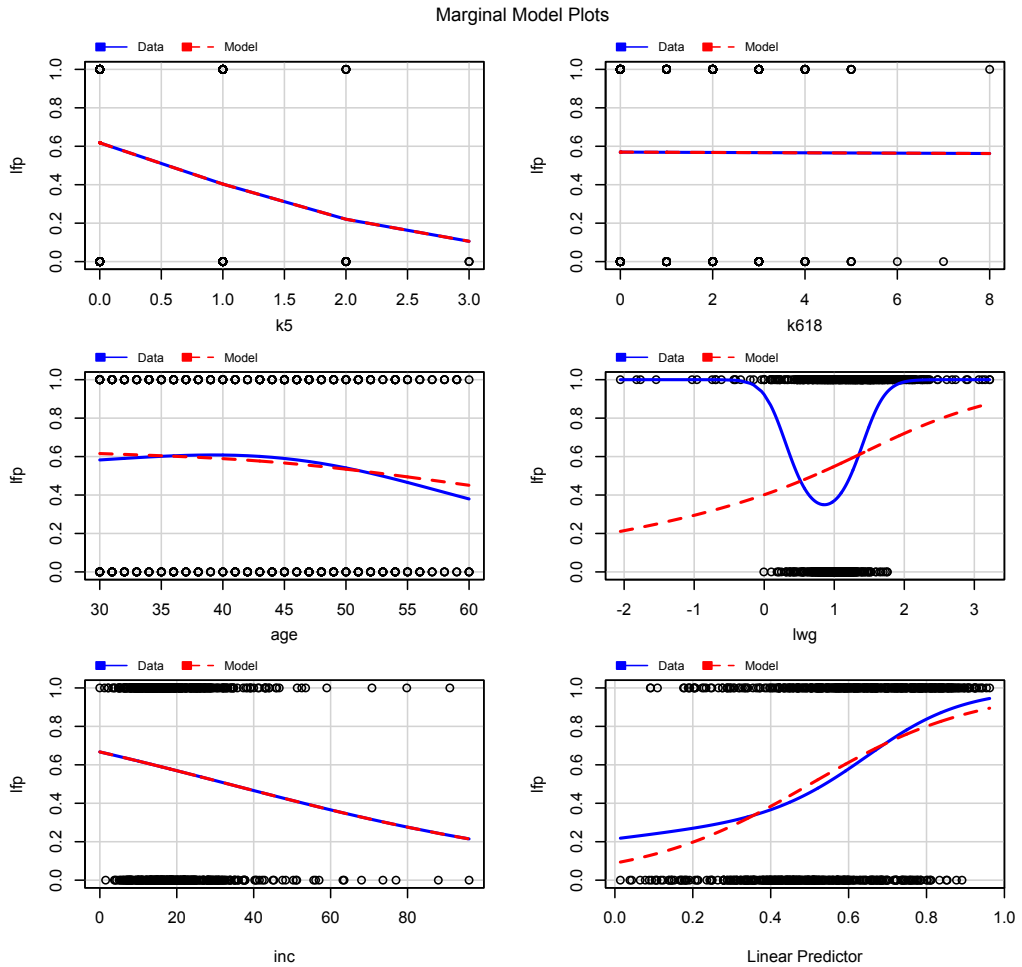


Figure 3 Marginal model plots depicting response variable against each predictor and linear predictor. Fitted values are compared with the observed data.

in *Figure 4*, where observations 119, 220 and 416 are the most influential. Alternatively, influence can be examined by using the influencePlot() function in the *car* package.

```
> influencePlot(mroz.mod,col="red",id.n=3)
```

	StudRes	Hat	CookD
53	1.253157	0.04879224	0.08728345
92	2.128986	0.01268509	0.11515073
119	2.250020	0.02779190	0.19084710
220	2.234546	0.01963334	0.15968200
348	1.283820	0.06455634	0.10467871
386	1.224214	0.05792918	0.09249251
416	1.911026	0.03673921	0.15217866

Figure 5 is very useful in identifying unusual observations because it plots studentized residuals against hat-values, and the size of circle is proportional to Cook's distance. I set the id.n=3, but there are 6 circles being labeled. This is not surprising because I leave the argument id.method to its default setting which means points with large studentized residuals, hat-values or Cook's distances are labeled. Nine points may be identified at maximum, three for each of the three statistics. However, some points may have largest values for two or more statistics. For example, observation 119 has the largest values in both studentized residual and Cook's distance.

We can examine the change of coefficient by removing these influential observations.

```
> mroz.mod119<-update(mroz.mod,subset=c(-119))
> compareCoefs(mroz.mod,mroz.mod119)
```

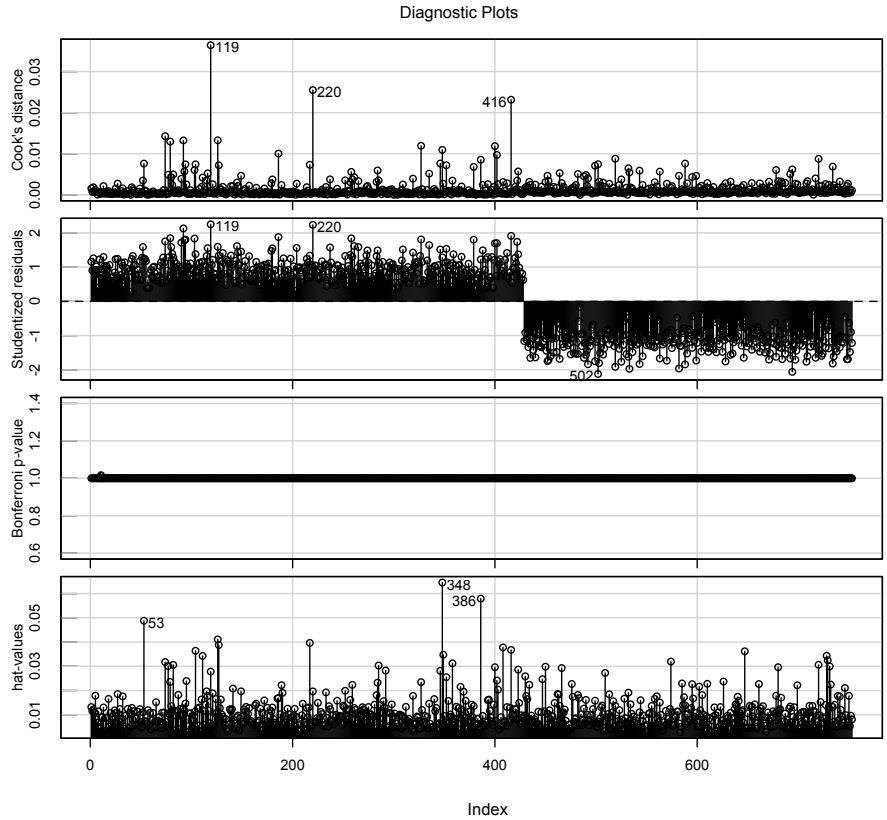



Figure 4 Diagnostic plots combining Cook's distance, studentized residuals, Bonferonni P and hat-values.

Call:

1: glm(formula = lfp ~ k5 + k618 + age + wc + hc + lwg + inc, family

= binomial, data = Mroz)

2: glm(formula = lfp ~ k5 + k618 + age + wc + hc + lwg + inc, family

= binomial, data = Mroz, subset = c(-119))

	Est. 1	SE 1	Est. 2	SE 2
(Intercept)	3.18214	0.64438	3.26635	0.64874
k5	-1.46291	0.19700	-1.49241	0.19899
k618	-0.06457	0.06800	-0.07005	0.06828
age	-0.06287	0.01278	-0.06308	0.01284
wcyes	0.80727	0.22998	0.80212	0.23087
hcyes	0.11173	0.20604	0.13825	0.20721
lwg	0.60469	0.15082	0.61098	0.15128
inc	-0.03445	0.00821	-0.03860	0.00848

You can see from the output table that coefficients are changed minimally, suggesting that the observation 119 is not influential. The cutoff values for these statistics are controversial. Some authors suggest that 2p/n can be the critical value for the leverage, and $\overline{hh}\times\chi^2_{0.95}(1)$ for Cook's distance, where P is the number of predictors, n is the number of observations, $\overline{hh}$ is average of J values of $h_j/(1-h_j)$ and $\chi^2_{0.95}(1)$ is the 0.95 percentile of chi-square distribution with 1 degree of freedom (7).

Summary

The article firstly describes the assessment of model fit by plotting Pearson residual against predictors. Such plots are helpful in identifying non-linearity and provide hints on how to transform predictors. Next, I focus on observations of outlier, leverage and influence that may have significant impact on model building. Outlier is such an observation that its response value is unusual conditional on the covariate pattern. Leverage is an observation with

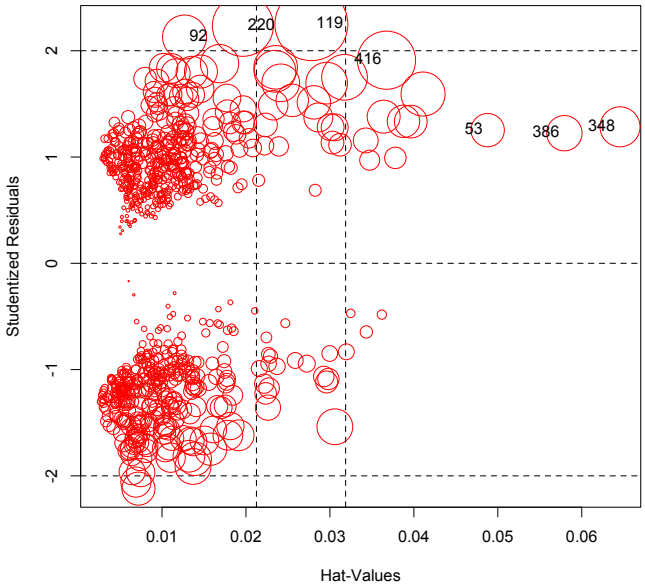


Figure 5 Studentized residuals are plotted against hat-values, and the size of circle is proportional to the Cook's distance. Observation 119 shows the largest Cook's distance, but it is moderate in hat-values.

covariate pattern that is far away from the average regressor space. Influence is the product of outlier and leverage. When influential observation is dropped from the model, there will be a significant shift of the coefficient. Summary statistics for outlier, leverage and influence are studentized residuals, hat values and Cook's distance, respectively. They can be easily visualized with graphs and formally tested using the *car* package.

Cite this article as: Zhang Z. Residuals and regression diagnostics: focusing on logistic regression. Ann Transl Med 2016;4(10):195. doi: 10.21037/atm.2016.03.36

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Fox J, Bates D, Firth D, et al. CAR: Companion to applied regression, R Package version 1.2-16. 2009. Available online: (accessed on August 2012).http://cRAN.R-project.org/web/packagescar/index.html
2. Cordeiro GM, Simas AB. The distribution of Pearson residuals in generalized linear models. Computational Statistics & Data Analysis 2009;53:3397-3411.
3. Hosmer DW Jr, Lemeshow S. Model-Building Strategies and Methods for Logistic Regression. In: Applied Logistic Regression. Hoboken: John Wiley & Sons, Inc; 2000:63.
4. Menard S. Applied logistic regression analysis. 2nd ed. New York: SAGE Publications; 2001:1.
5. Sarkar SK, Midi H, Rana MS. Detection of outliers and influential observations in binary logistic regression: An empirical study. Journal of Applied Statistics 2011;11:26-35.
6. Cook RD. Detection of influential observation in linear regression. Technometrics 1977;19:15-18.
7. Martín N, Pardo L. On the asymptotic distribution of Cook's distance in logistic regression models. Journal of Applied Statistics 2009;36:1119-1146.

Propensity score method: a non-parametric technique to reduce model dependence

Zhongheng Zhang

Abstract: Propensity score analysis (PSA) is a powerful technique that it balances pretreatment covariates, making the causal effect inference from observational data as reliable as possible. The use of PSA in medical literature has increased exponentially in recent years, and the trend continues to rise. The article introduces rationales behind PSA, followed by illustrating how to perform PSA in R with *MatchIt* package. There are a variety of methods available for PS matching, such as nearest neighbors, full matching, exact matching and genetic matching. The task can be easily done by simply assigning a string value to the method argument in the matchit() function. The generic summary() and plot() functions can be applied to an object of class *matchit* to check covariate balance after matching. Furthermore, there is a useful package *PSAgraphics* that contains several graphical functions to check covariate balance between treatment groups across strata. If covariate balance is not achieved, one can modify model specifications or use other techniques such as random forest and recursive partitioning to better represent the underlying structure between pretreatment covariates and treatment assignment. The process can be repeated until the desirable covariate balance is achieved.

Keywords: Propensity score; observational study; logistic regression

Submitted Jun 05, 2016. Accepted for publication Jul 02, 2016.
doi: 10.21037/atm.2016.08.57
View this article at: http://dx.doi.org/10.21037/atm.2016.08.57

Introduction

In clinical research, an important task is to estimate the causal effect of an intervention on patient-important outcomes. Causal effect can be estimated using randomized controlled trial (RCT), in which both measured and unmeasured confounding factors are balanced in both treatment and control arms. While RCT is the gold standard to assess biological efficacy of an intervention, this design is sometimes not feasible due to ethical problem and/or lack of funding (1,2). In contrast, observational studies utilizing electronic medical records are cheap and easy to access (3). Furthermore, such observational studies are conducted in real world setting and can evaluate the clinical effectiveness of an intervention. There are situations in which an intervention shows biological efficacy in RCTs, but loses its clinical effectiveness in the real world setting (4). Therefore, observational studies are widely used in clinical researches in spite of their numerous inherent shortcomings. The major limitation in using observational data to estimate causal effect is the confounding factors. Traditionally, these confounding factors can be adjusted with multivariate models (5-7). However, the distribution of confounding factors may be different between intervention and control groups, and the model extrapolation can be wrong (8). Furthermore, regression models have specific assumptions and specifications such as linearity, normal distribution of error term and interaction (9). Frequently, these assumptions are arbitrarily made without empirical evidence. As a result, the causal effect estimated with regression models can vary substantially depending on different specifications and assumptions of the model. This is termed the model dependence in the literature (8).

Propensity score method is employed to solve the problem of imbalance in baseline characteristics between the intervention and control groups. Initially, the treatment status is used as dependent variable, which is regressed on pretreatment covariates with logistic regression model. Propensity score, or the probability of assigning to the treatment, can be calculated with the fitted model. Then propensity score is used for subsequent causal effect inference. Propensity score is considered as nonparametric although parametric regression model is used to estimate

propensity score. Other advanced models such as random forest, naïve Bayes and repeated partitioning can be used to estimate propensity score. Propensity matching or stratification is nonparametric. The two-step procedure in causal effect estimation is considered doubly robust by Ho and coworkers in that if either propensity score matching or parametric model is correct, the causal estimates can be consistent (8). The article will show how to perform propensity score analysis (PSA) with R packages (10). Readers may consult other references for detailed mathematical descriptions of PSA (11-13).

Working example

To illustrate PSA by using R, I create a dataset including continuous (x.cont) and categorical (x.cat) pretreatment covariates. The functional form between treatment (treat) and covariates includes high order terms and an interaction, reflecting the complexity in real world setting. Furthermore, the outcome (y) is regressed on treatment and pretreatment covariates. A total of 1,000 subjects are created.

```
> set.seed(888)
> x.cat <- rep(0:1,c(200,800))
> x.cont <- rnorm(1000)
> lp <- -3 + 2*x.cat*x.cont+5*x.cont^2+3*x.cont-4*x.cat
> link_lp = exp(lp)/(1 + exp(lp))
> treat <- (runif (1000) < link_lp)
> lp.y<-2 + 3*x.cont+2*x.cat+4*treat
> link_y<-exp(lp.y)/(1 + exp(lp.y))
> y<- (runif (1000) < link_y)
> data<-data.frame(treat,x.cat,x.cont,y)
```

PSA with MatchIt package

The *MatchIt* package contains useful functions to perform PSA. The first step is to install the package and load it to the workspace. The package *rgenoud* should also be installed if you want to perform genetic matching.

```
> install.packages("rgenoud")
> install.packages("MatchIt")
> library(rgenoud)
> library(MatchIt)
> m.out <- matchit(treat ~ x.cat+x.cont, method =
"nearest",discard="both",data = data)
```

The main function matchit() performs PSA. The first argument passes a formula to the function, which defines how the pretreatment covariates influence the treatment assignment. In research practice, investigators usually specify a main effect model without interactions and high-order terms. The *method* argument passes string values including “nearest” (nearest neighbor matching), “exact” (exact matching), “full” (full matching), “genetic” (genetic matching), “optimal” (optimal matching), and “subclass” (subclassification). The default option is the nearest neighbor matching. The discard argument specifies whether to discard observations fall outside of the common support, not allowing them to be used in the matching process. The “both”

```
> summary(m.out)

Call:
matchit(formula = treat ~ x.cat + x.cont, data = data, method = "nearest",
  discard = "both")

Summary of balance for all data:
          Means Treated Means Control SD Control Mean Diff eQQ Med eQQ Mean eQQ Max
distance         0.5137        0.2175     0.1418    0.2963  0.3284   0.3019  0.4488
x.cat            0.6440        0.8698     0.3368   -0.2257  0.0000   0.2233  1.0000
x.cont           0.6084       -0.2543     0.5965    0.8627  1.2080   1.1241  1.9759

Summary of balance for matched data:
          Means Treated Means Control SD Control Mean Diff eQQ Med eQQ Mean eQQ Max
distance         0.4727        0.3643     0.1459    0.1084  0.1276   0.1084  0.1879
x.cat            0.7342        0.6261     0.4849    0.1081  0.0000   0.1081  1.0000
x.cont           0.6391        0.1160     0.6099    0.5231  0.7374   0.7336  1.1489

Percent Balance Improvement:
```


	Mean Diff.	eQQ Med	eQQ Mean	eQQ Max
distance	63.4146	61.1436	64.0798	58.1412
x.cat	52.1097	0.0000	51.5864	0.0000
x.cont	39.3634	38.9554	34.7366	41.8539

Sample sizes:

	Control	Treated
All	691	309
Matched	222	222
Unmatched	469	0
Discarded	0	87

value dictates to discard observations that are outside of the common support in both treatment and control groups.

Balance can be checked with summary() function.

The output of generic function summary() is important in assessing the balance after PSA. The "Means Treated" and "Means Control" columns show the weighted means for the treated and control groups. "SD Control" is the standard deviation for the control group. "Mean Diff" is the mean difference between the control and treated groups. The last three columns show the median, mean and maximum distance between empirical quantile functions of the treated and control groups. A value greater than 0 indicates deviations between the two groups in some part of quantile distributions. The last table shows the number of observations that have been matched or discarded. These statistics can be visualized with generic plot() function.

```
> plot(m.out,type="jitter")
> plot(m.out,type="hist")
> plot(m.out)
```

Figure 1 is a jittered plot showing matched and unmatched observations, as well as their distributions on propensity score values. It appears that many observations with high PS in the treated groups and many with low PS in the control group are excluded. *Figure 2* shows histograms showing the density of PS distribution in the treated and control groups before and after matching. The PS of the treated groups are significantly higher than the control group before matching while the density distributions of the two groups become somewhat similar after matching. Quantile-quantile (QQ) plot compares the probability distributions of a given covariate for the treated and control groups by plotting their quantiles against each other. The points on the QQ plot will lie on the y=x line if two distributions are similar. The results show that although the points are not located on the y=x line exactly after matching, it is much improved as compared to law data (*Figure 3*).

Checking balance with PSAgraphics package

The *PSAgraphics* package provides several functions for evaluating balance of covariates in each stratum (14). So in this section, I first create strata containing treated and control groups, then try to balance covariates between both groups.

```
> m.out.strata<-matchit(treat ~ x.cat+x.cont, method =
"subclass",discard="both",data = data)
```

In the above example, subclassification is used to form strata in which the distribution of covariates in treated and control groups are as similar as possible.

```
> m.data<-match.data(m.out.strata)
> str(m.data)
'data.frame':        913 obs. of 7 variables:
 $ treat : logi TRUE TRUE TRUE FALSE TRUE FALSE ...
 $ x.cat : int 0 0 0 0 0 0 0 0 0 0 ...
 $ x.cont : num -1.951 -1.544 0.73 -0.278 -1.656 ...
 $ y   : logi TRUE TRUE TRUE TRUE TRUE FALSE ...
 $ distance: num 0.105 0.16 0.742 0.464 0.143 ...
 $ weights : num 1 1 1 8.82 1 ...
 $ subclass: num 1 1 5 3 1 3 1 5 5 4 ...
```

Matched dataset can be stored in an object using the match.data() function. In addition to the original variables, three variables *distance*, *weights* and *subclass* are added. The subclass variable denotes the stratum to which an observation belongs. Next I will install the *PSAgraphics* package and proceed to examine covariate balance within each stratum.

```
> install.packages("PSAgraphics")
```

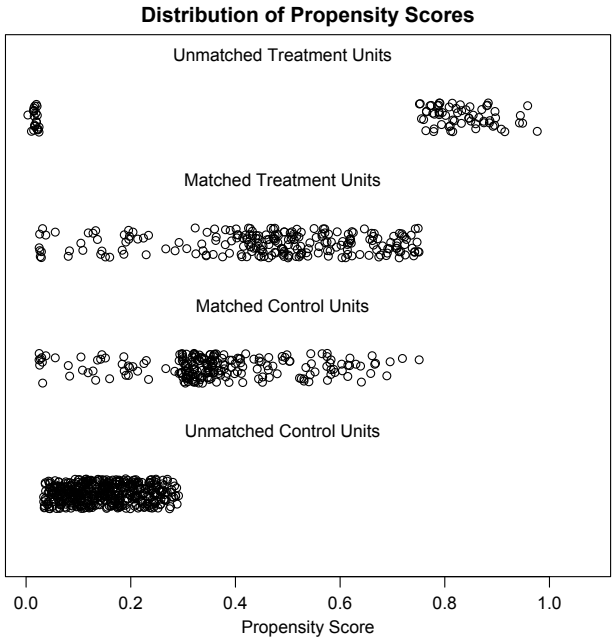


Figure 1 Jittered plot showing matched and unmatched observations, as well as their distributions on propensity score values. It appears that many observations with high propensity scores in the treated group and many with low propensity scores in the control group are excluded.

```
> library(rpart)
> library(PSAgraphics)
> box.psa(m.data$x.cont, m.data$treat,
m.data$subclass, xlab = "Strata",
ylab = "Covariate: x.cont", balance = TRUE)
Press <enter> for bar chart...
```

The box.psa() function depicts a pair of side-by-side boxplots for each stratum to compare the difference between treated and control groups on a given covariate. If balance=TRUE, it calls bal.ms.psa() function to draw a histogram of a permutation distribution and reference statistic to assess balance across strata (*Figure 4*). Balance statistic is defined as:

$$\hat{\delta}_\alpha = \sum_{k=1}^{k} |\hat{\mu}_{0k} - \mu_{1k}|, \quad [1]$$

where $\hat{\delta}_\alpha$ is the balance statistic, the subscript α is to denote a particular subclassification scheme, K is the total number of stratum, and $\hat{\mu}_{0k}$ and $\hat{\mu}_{1k}$ are mean values of the control and treated group within stratum k. Note that smaller value of the balance statistic indicates a better balance on that given covariate. The histogram is drawn by randomly assigning observations to strata and in our example it generated 1,000 balance statistics. By comparing our original balance ($\hat{\delta}_\alpha$) to the mass of permutation distribution, we conclude that the subclassification method has balanced the covariate x.cont as much as possible (e.g., it is the smallest among all randomly generated balance statistics).

By pressing enter as indicated by the output message, it pops up a series of boxplots comparing the difference on *x.cont* between treated and control groups within each stratum. Means of the two groups are connected by a heavy solid line, with the slope of the line indicate size of the difference. *Figure 5* shows that the balance of *x.cont* within each stratum is unsatisfactory and the distribution changes moderately across strata. The sizes (number of observations) of groups are printed below corresponding boxplots. The "balance=TRUE" argument adds Kologmorov-Smirnov p-values to the graph for the test of equivalence of control/treatment distributions within each stratum.

PSAgraphics package has a special function cat.psa() for balance check of categorical variables. The argument of the function is similar to box.psa() function as described above.

The function produces side-by-side barplots comparing proportion of cases in each category (*Figure 6*). The sizes of treatment groups within strata are printed at the top of the bars. It is noted that the subclassification method generates poorly matched strata. Along with the barplot, the cat.psa() function also produces a cross-tabulation between treatment and categorical covariate across strata.

It may be interesting to compare outcomes between treatment groups across strata. The following function circ.psa() is designed for this purpose. While the R code to draw circles is simple, the key lessons are how to interpret the output plot.

```
> circ.psa(m.data$y,m.data$treat,m.data$subclass,
revc = TRUE,xlab = "Treatment", ylab = "Control")
$summary.strata
```

	n.FALSE	n.TRUE	means.FALSE	means.TRUE
1	391	29	0.8286445	0.7931034
2	244	51	0.9180328	1.0000000
3	18	51	0.8333333	1.0000000
4	31	52	0.9032258	1.0000000
5	7	39	1.0000000	1.0000000

```
$wtd.Mn.TRUE
[1] 0.9048231
$wtd.Mn.FALSE
[1] 0.8732947
$ATE
[1] -0.03152831
```

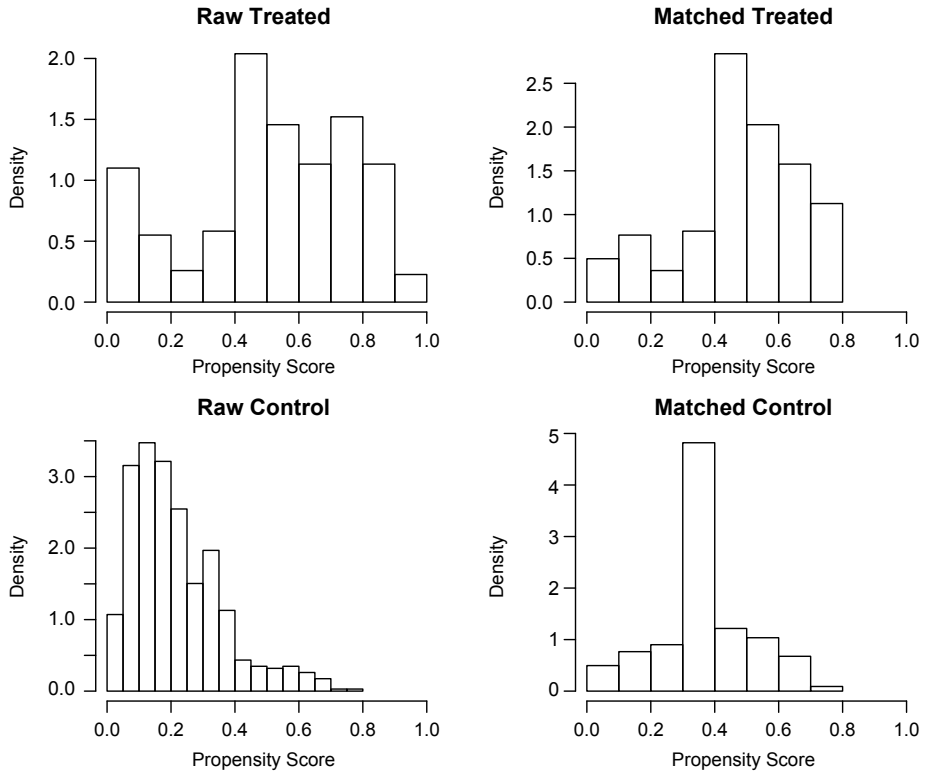



Figure 2 Histograms showing the density of propensity score distribution in the treated and control groups before and after matching.

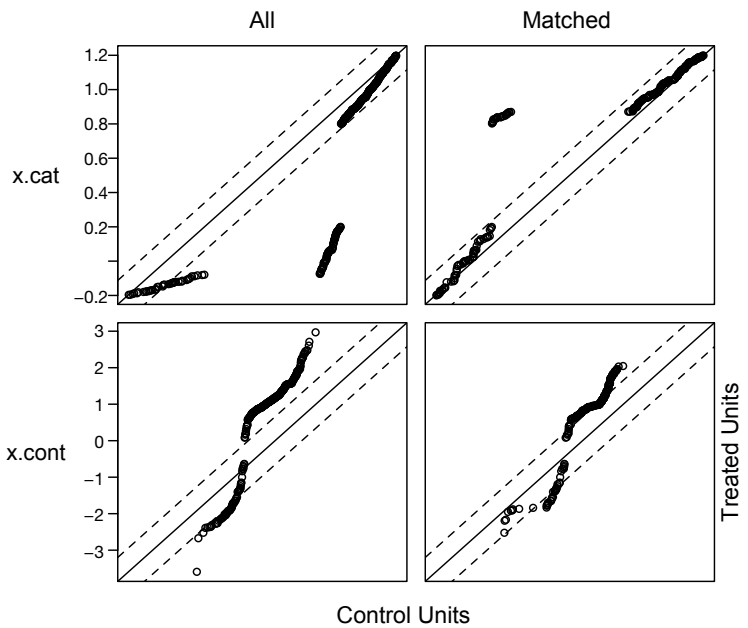


Figure 3 Quantile-quantile (QQ) plot compares the probability distributions of the treated and control groups on a given covariate by plotting their quantiles against each other. The results show that although the points are not located on the y=x line exactly after matching, it is much improved as compared to law data.

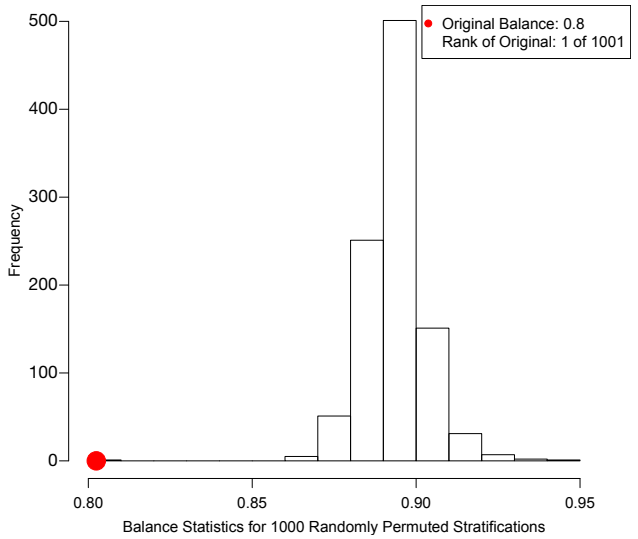


Figure 4 Histogram of a permutation distribution and reference statistic to assess balance across strata. The balance statistic locates at the left end of the mass of permutation distribution, indicating a good balance.

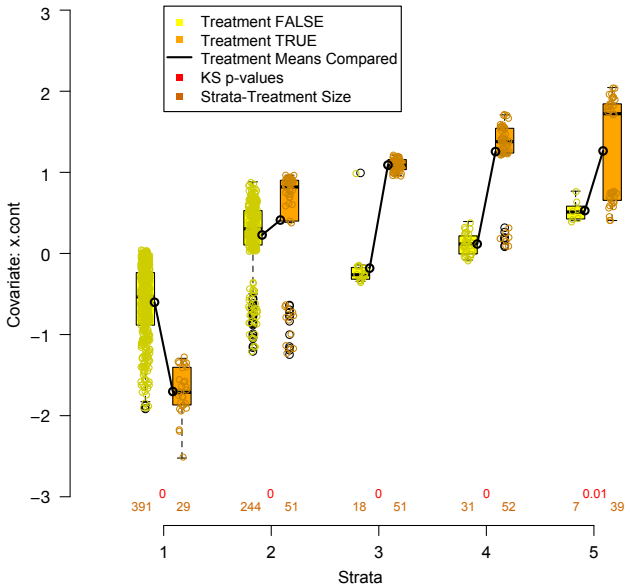


Figure 5 Side-by-side boxplots, 5 strata, for covariate x.cont produced by box.pdf.

```
$se.wtd
[1] 0.02654633
$approx.t
[1] -1.187671
$df
[1] 903
$CI.95
[1] -0.08362798 0.02057137
```

In *Figure* 7, stratum is represented by a circle with the circle size proportional to the number of observations in each stratum. The number within each circle is the stratum number. Because the outcome variable y is binary denoted by 0 and 1, the mean of y in each treatment group is equal to the proportion of outcome events. The center of circle projecting to the x-axis corresponds the outcome means for the treated group, and that projecting to the y-axis is the outcome means for the control group. Circles below the solid identity line (y=x) are those with treatment effect larger than the control group, and vise versa. The dashed blue line parallels to the identity line is the mean difference of the outcome. The cross symbols represent the distribution of strata difference. The horizontal and vertical dashed lines represent the (weighted) means for the treated and control groups respectively. Rug plots on vertical and horizontal sides of the graph show marginal distributions of control and treatment outcome measures. Along with the graph, circ.psa() returns summary statistics for the means on outcome variables in treatment groups across strata. The weighted means for each treatment group are given under objects "wtd.Mn.TRUE" and "wtd.Mn.FALSE". The following objects "ATE", "se.wtd", "approx.t", "df" and "CI.95" respectively represent average treatment effect, weighted standard error (15), approximate t statistics, degree of freedom and 95% confidence interval for the direct adjustment estimator (shown as the heavy green line in *Figure* 7).

Since the above methods indicate that the balance was not achieved by the matching method, one needs to modify model specifications in calculating logistic-regression based propensity scores. Higher order terms and interactions can

```
> cat.psa(m.data$x.cat,m.data$treat, m.data$subclass, xlab = "Strata",ylab = "Proportion for 'x.cat'",barnames =
c("Control", "Treatment"), rtmar = 2)
$`treatment:stratum.proportions`
  FALSE:1 TRUE:1 FALSE:2 TRUE:2 FALSE:3 TRUE:3 FALSE:4 TRUE:4 FALSE:5 TRUE:5
0 0       0.724  0.143   0.235  0.944   0      1       0.135  1       0.487
1 1       0.276  0.857   0.765  0.056   1      0       0.865  0       0.513
```

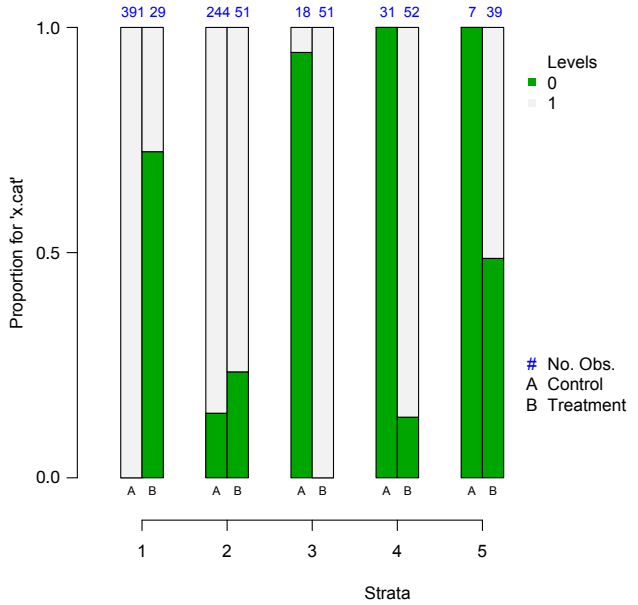



Figure 6 Side-by-side barplots comparing proportion of cases in each category for variable x.cat.

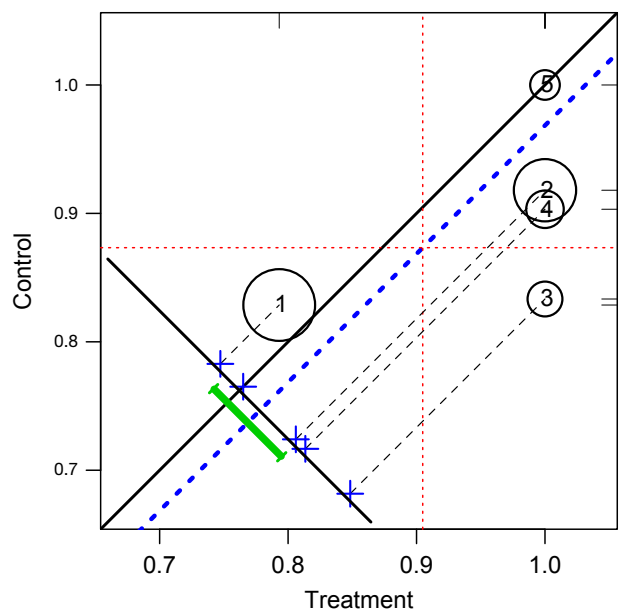


Figure 7 Propensity score analysis assessment plot of outcome variable y, 5 strata, constructed using circ.psa.

be added as follows.

```
> m.out.right <- matchit(treat ~ x.cat*x.cont+x.cont^2,
method = "nearest",discard="both",data = data)
```

The summary output of the object *m.out.right* is omitted to save space. The percent balance improvement of distance is approximately 80%, which is significantly greater than 60% in the original matching. In practice, the matching process can be repeated until the model with the best balance statistics is obtained.

Effect estimation after PSA

Any parametric analyses can be performed on the matched dataset obtained with PSA. The procedures are the same to the ones that have been used without PSA. This is left to readers for practices. Also they can consult the excellent tutorial written by Ho and colleagues on how to perform analysis after matching using *Zelig* package (16).

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Albert RK. "Lies, damned lies .." and observational studies in comparative effectiveness research. Am J Respir Crit Care Med 2013;187:1173-1177.
2. Zhang Z. Big data and clinical research: perspective from a clinician. J Thorac Dis 2014;6:1659-1664.
3. Zhang Z. Big data and clinical research: focusing on the area of critical care medicine in mainland China. Quant Imaging Med Surg 2014;4:426-429.
4. Singal AG, Higgins PD, Waljee AK. A primer on effectiveness and efficacy trials. Clin Transl Gastroenterol 2014;5:e45.
5. Yeh RW, Mauri L. Choosing methods to minimize confounding in observational studies: do the ends justify the means? Circ Cardiovasc Qual Outcomes 2011;4:581-583.
6. Quartey G, Feudjo-Tepie M, Wang J, et al. Opportunities for minimization of confounding in observational research. Pharm Stat 2011;10:539-547.
7. Pfeiffer RM, Riedl R. On the use and misuse of scalar scores of confounders in design and analysis of observational studies. Stat Med 2015;34:2618-2635.
8. Ho DE, Imai K, King G, et al. Matching as nonparametric

preprocessing for reducing model dependence in parametric causal inference. Political Analysis 2007;15:199-236.
9. Zhang Z. Multivariable fractional polynomial method for regression model. Ann Transl Med 2016;4:174.
10. Keller B, Tipton E. Propensity score analysis in R: a software review. Journal of Educational and Behavioral Statistics 2016;41:326-348.
11. Patorno E, Grotta A, Bellocco R, et al. Propensity score methodology for confounding control in health care utilization databases. Epidemiology Biostatistics and Public Health 2013;10:1-16.
12. Li M. Using the propensity score method to estimate causal effects: a review and practical guide. Organizational Research Methods 2013;16:188-226.
13. Little RJ, Rubin DB. Causal effects in clinical and epidemiological studies via potential outcomes: concepts and analytical approaches. Annu Rev Public Health 2000;21:121-145.
14. Helmreich JE, Pruzek RM. PSAgraphics: an RPackage to support propensity score analysis. Journal of Statistical Software 2009;29:1-23.
15. Conniffe D. Evaluating state programmes: "Natural experiments" and propensity scores. Economic and Social Review 2000;31:283-308.
16. Ho D, Imai K, King G, et al. MatchIt: nonparametric preprocessing for parametric causal inference. Journal of Statistical Software 2011;42:1-28.

Cite this article as: Zhang Z. Propensity score method: a non-parametric technique to reduce model dependence. Ann Transl Med 2017;5(1):7. doi: 10.21037/atm.2016.08.57

Drawing Nomograms with R: applications to categorical outcome and survival data

Zhongheng Zhang, Michael W. Kattan

Abstract: Outcome prediction is a major task in clinical medicine. The standard approach to this work is to collect a variety of predictors and build a model of appropriate type. The model is a mathematical equation that connects the outcome of interest with the predictors. A new patient with given clinical characteristics can be predicted for outcome with this model. However, the equation describing the relationship between predictors and outcome is often complex and the computation requires software for practical use. There is another method called nomogram which is a graphical calculating device allowing an approximate graphical computation of a mathematical function. In this article, we describe how to draw nomograms for various outcomes with the nomogram() function. Binary outcome is fit by logistic regression model and the outcome of interest is the probability of the event of interest. Ordinal outcome variable is also discussed. Survival analysis can be fit with parametric models to fully describe the distributions of survival time. Statistics such as the median survival time, survival probability up to a specific time point are taken as the outcome of interest.

Keywords: Nomogram; prediction; regression model; visualization

Submitted Sep 22, 2016. Accepted for publication Mar 23, 2017.
doi: 10.21037/atm.2017.04.01
View this article at: http://dx.doi.org/10.21037/atm.2017.04.01

Introduction

The definition of nomogram is well described in Wikipedia that "A nomogram (from Greek νόμος nomos, "law" and γραμμή grammē, "line"), also called a nomograph, alignment chart or abaque, is a graphical calculating device, a two-dimensional diagram designed to allow the approximate graphical computation of a mathematical function." A nomogram consists of a set of scales that each scale represents a characteristic of the study population. If there are interaction terms in the original regression model, there will be several scales representing certain combinations of interacting variables. Nomogram is actually a visualization of a complex model equation that the behavior of a predictor is represented in scales (1). In clinical medicine, the tool is widely used for prediction of patients' outcomes, considering his or her clinical characteristics. Nomogram is most widely used in clinical oncology to help patients and doctors to make important treatment decisions (2).

Because nomogram is based on regression models such as logistic regression model, parametric survival model and ordinal logistic regression model, the performance of the tool is dependent on the regression models. The discrimination, calibration, and external validation also apply to the nomogram. Therefore, model diagnostics and model fit that are applicable to regression models should also be performed before drawing a nomogram. In this article, these tasks are not described, but readers can consult other tutorials of the big-data clinical trial column (3,4). This article focuses on how to draw a nomogram with R software and how to adjust some interesting and important parameters.

Worked example

The worked example and model strategies for drawing nomograms are adapted from the R documentation. In the following code, a data frame containing 1,000 observations and 6 variables are created. The variable age follows a normal distribution with a mean of 65 and a standard deviation of 11. Lactate (lac) also follows a normal distribution, but abs() function is employed to ensure positive values. Sex and shock are generated as factor variables. Arbitrary coefficients are given to each variable

to produce a linear combination. plogis() function is called the 'inverse logit' that it gives probability according to the linear predictor.

```
> n <- 1000   # sample size
> set.seed(88) # set seed for replication
> age<- rnorm(n, 65, 11)
> lac<- round(abs(rnorm(n, 3, 1)),1)
> sex<- factor(sample(1:2,n,prob=c(0.6,0.4),TRUE),
              labels=c('male','female'))
> shock<-factor(sample(1:4,n,prob=c(0.3,0.3,0.25,0.15),
TRUE),
          labels=c('no','mild','moderate','severe'))
> z<- 0.2*age + 3*lac* as.numeric(sex)+
5*as.numeric(shock) -
rnorm(n,36,15) # linear combination with a bias
> y <- ifelse(runif(n) <= plogis(z), 1, 0)
```

Next, an ordinal categorical variable Y is created. The ifelse() function is useful to create categorical variable. All vectors are combined to create a data frame. Variable names and labels can be used to annotate nomogram, depending on the preference of the investigator. Often, the variable name is usually simple with abbreviations, and the label gives more details on the variable. Therefore, annotation with names makes the nomogram simple, whereas variable labels make the plot more informative. Both methods are illustrated in our example. Here, we assign labels to each variable.

```
> Y <- ifelse(y==0, 0, sample(1:3, length(y), TRUE))
> data<-data.frame(age=age,lac=lac,sex=sex,
shock=shock,y=y,Y=Y)
> var.labels = c(age="Age in Years",
                 lac="lactate",
                 sex="Sex of the participant",
                 shock="shock",
                 y="outcome",
       Y="ordinal")
> library(rms)
> label(data) = lapply(names(var.labels),
                 function(x) label(data[,x]) = var.labels[x])
```

Nomogram for binary outcome

Binary outcome is the most commonly encountered data type in clinical medicine, and can be modeled with logistic regression model. The outcome of interest for prediction is the probability of the event of interest, because this quantity is more intuitive for both clinicians and patients. In the example below, we label the outcome as "Risk of death", which is a patient-important outcome variable.

```
> library(rms)
> ddist <- datadist(data)
> options(datadist='ddist')
> mod.bi<-lrm(y~shock+lac*sex+age,data)
> nom.bi <- nomogram(mod.bi,
    lp.at=seq(-3,4,by=0.5),
    fun=function(x)1/(1+exp(-x)),
    fun.at=c(.001,.01,.05,seq(.1,.9,by=.1),.95,.99,.999),
    funlabel="Risk of Death",
    conf.int=c(0.1,0.7),
    abbrev=TRUE,
    minlength=1,lp=F)
```

In the article, the rms package is employed to fit regression model and depict nomogram. The rms package contains a collection of functions assisting model building and visualization. In essence, nomogram is a kind of visualization of regression models. Firstly, we need to define the distribution summaries for predictor variable with the datadist() function. Specifically, datadist() defines summary statistics for continuous and categorical predictors. These summary statistics include effect and plotting ranges, values to adjust to, and overall ranges. Nomogram only uses the list of categories from datadist() for categorical variables, and the outer limits for continuous ones. The object returned by datadist() is then assigned to option() function so that later predictions and summaries of the fit will not need to access the original data used in the fit. Model fit is performed by lrm() function which is designed to fit logistic regression model. The nomogram() function first takes an object returned by the regression model fit. In the example, it is the object name mod. The argument lp.at takes a vector of numeric values ranging from −3 to 4 by a step of 0.5. These values will be displayed in the

linear predictor axis. A linear predictor function is a linear function (linear combination) of a set of coefficients and explanatory variables (independent variables), whose value is used to predict the outcome of interest. The argument fun is optional that it defines the transformation of linear predictor and plots it on another axis. In the example, the inverse-logit transformation is employed to transform the linear predictor to the probability. Labels of function axis are defined by a vector of numeric values with fun. at argument. In the example, the linear predictor −3 is the minimum value and it corresponds to a probability of 0.0474, thus only function values greater than 0.0474 can be displayed on the function axis. In other words, the initial two values 0.001 and 0.01 will be omitted in the nomogram. The funlabel argument is to change the name of function axis to "Risk of Death". The conf.int argument gives confidence interval to display for each score. The default is to display no confidence limits. In the example, we want to display 0.1 and 0.7 confidence intervals for each score. The narrow interval of 0.1 is useful to mark which score a confidence interval line corresponds to. The abbrev is set to true to abbreviate levels of categorical factors. However, it only abbreviates the factor variables of interaction terms in the current version, which is not consistent with the R documentation. When we set minlength=1, it actually uses minlength=4 for interaction terms. Tick marks for categorical predictors (the variable shock) are supposed to be abbreviated to letters of the alphabet, but actually it doesn't.

```
> plot(nom.bi,lplabel="Linear Predictor",
      fun.side=c(3,3,1,1,3,1,3,1,1,1,1,1,3),
      col.conf=c('red','green'),
      conf.space=c(0.1,0.5),
      label.every=3,
      col.grid = gray(c(0.8, 0.95)),
      which="shock")
> legend.nomabbrev(nom.bi, which='shock', x=4.5, y=.5)
```

The nomogram can be plotted with the generic plot() function (*Figure 1*). It takes an object of class "nomogram" that contains information used in plotting the axes. The lplabel argument is used to rename the linear predictor axis. If tick marks of function axis are too crowded and overlap with each other, one may specify the fun.side argument with a vector of numeric values 1 and 3. The numeral 1 is to position tick marks below axis and the numeral 3 for above the axis. Note that the first element of fun.side corresponds to the first value in the vector defined by fun.at, rather than the first value displayed in the axis. Because the first two function values are omitted, the fun.side argument takes effect from the third element. The argument col. conf=c('red','green') specifies that the 0.1 confidence interval is draw with red color and the 0.7 confidence interval is drawn with green color. Confidence bars defined by the conf.int argument are drawn in the space between predictor axis, and the vertical range within which to draw confidence bars can be modified using the conf.space argument. The conf.space argument takes a two-element vector defining the beginning and ending of vertical range to draw confidence bars. In the example, we draw the first bar at 0.1 and the last at 0.5 (e.g., The entire range between main axis is defined to be 1). The label.every=3 specifies to label every 3 tick marks. Vertical reference lines are drawn with col.grid = gray(c(0.8, 0.95)) argument. Any colors can be designated to the color grid.

If we use abbreviations to label tick marks, a legend should be added with gend.nomabbrev() argument. The x and y arguments tell the function where the legend should be placed. The predictor to which the legend relates is specified in the which argument.

Nomogram for ordinal outcome variable

Ordinal logistic regression is a type of logistic regression that deals with dependent variables that are ordinal—that is, there are multiple response levels and they have a specific order, but no exact spacing between the levels. In the example, the variable Y contains four levels. Similarly to the binary logistic regression model, ordinal model can be fit with lrm() function. In the model, an interaction between sex and lac is included. Furthermore, the variable lac takes a linear tail-restricted cubic spline function by using rcs() function.

```
> mod.ord <- lrm(Y ~ age+rcs(lac,4)*sex)
```

Here we will spend some time to discuss the estimated parameters from ordinal regression model. In the example, the outcome variable Y takes four levels (0, 1, 2 and 3), and

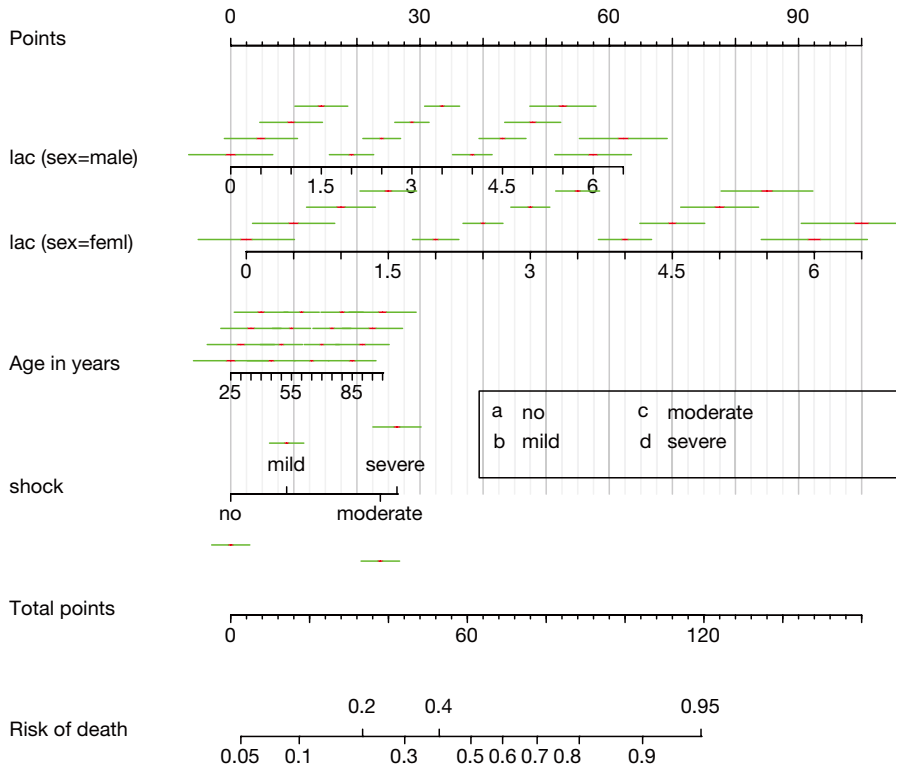


Figure 1 Nomogram for ordinal outcome variable y. Each predictor with a given value can be mapped to the Points axis. The sum of these points can be referred to in the Total Points axis. Then the linear predictor and the probability of death can be obtained from corresponding axis. The green bar indicates the 0.7 confidence limits for each score, and the short red bar corresponds to 0.1 confidence limits. The legend is for abbreviations of categorical variable shock. However, this version of nomogram() does not abbreviate levels of factor variables, but it is supposed to abbreviate levels of categorical factors to letters of the alphabet for tick marks when we set minlength=1.

the ordinal logistic regression model has the form:

$$logit(p_0) \equiv \log \frac{p_0}{1\text{-}p_0} = \alpha_0 + \beta\chi \quad [1]$$

$$logit(p_0 + p_1) \equiv \log \frac{p_0 + p_1}{1 - p_0 - p_1} = \alpha_1 + \beta\chi \quad [2]$$

$$logit(p_0 + p_1 + p_2) \equiv \log \frac{p_0 + p_1 + p_2}{1 - p_0 - p_1 - p_2} = \alpha_2 + \beta\chi \quad [3]$$

$$logit(p_0 + p_1 + p_2 + p_3) \equiv \log \frac{p_0 + p_1 + p_2 + p_3}{1 - p_0 - p_1 - p_2 + p_3} = \alpha_3 + \beta\chi \quad [4]$$

Because p_0+p_1+p_2+p_3=1, the last Eq. [4] is not fit. This model is known as the proportional-odds model because the odds ratio (exponentiation of β) of the event is independent of the outcome category. The odds ratio is assumed to be constant for all categories. The model simultaneously estimated 3 equations and the comparisons of outcome categories are shown in *Table 1*. Eq. [1] models the odds of being in the set of Y=0 on the left versus the set of categories Y={1,2,3} on the right. Because ordinal regression assumes parallel regression, there is only one set of coefficients for each independent variable. Eqs. [1] to [3] share the same coefficient β for covariate x. However, the intercept would be different, as denoted by different annotations from α_0 to α_2. The intercepts can be used to calculate predicted probability for patients with a given set of characteristics of being in a particular category (5).

```
> fun2 <- function(x) plogis(x-mod.ord$coef[1]+
mod.ord$coef[2])
> fun3 <- function(x) plogis(x-mod.ord$coef[1]+
mod.ord$coef[3])
```

The Newlabels() function is used to override the variable

Table 1 Comparisons of outcome categories for each equation in ordinal logistic regression model

Equations	Pooled categories	Versus	Pooled categories	R code for computing probability
1	0		1, 2, 3	plogis(x)
2	0, 1		2, 3	plogis(x-mod.ord$coef[1]+mod.ord$coef[2])
3	0, 1, 2		3	plogis(x-mod.ord$coef[1]+mod.ord$coef[3])

Each equation models the odds of being in the set of categories on the left versus the set of categories on the right. x is the linear predictor of the ordinal regression model.

labels in a fit object. In the example, variable age is assigned a label named "Age in Years". Parameters in the nomogram() function are similar to that described in the above example.

```
> f <- Newlabels(mod.ord, c(age='Age in Years'))
> nom.ord <- nomogram(f, fun=list('Prob Y>=1'=plogis,
                'Prob Y>=2'=fun2,
                'Prob Y=3'=fun3),
                lp=F,
      fun.at=c(.01,.05,seq(.1,.9,by=.1),.95,.99))
> plot(nom.ord, lmgp=.2, cex.axis=.6)
```

The generic function plot() gives a nomogram for ordinal outcome variable (*Figure 2*). Since there is an interaction between sex and lac, continuous variable lac is depicted at two levels of sex. Note that there are three function axes. For each patient with given characteristics (e.g., the linear predictor is fixed), the probabilities of each category are different.

Nomogram for survival data

In follow up study, the time-to-event or survival data are commonly encountered. Multivariable analysis of such data includes semi-parametric and parametric regression modeling. Semi-parametric modeling provides the impact of covariates on survival time, but leaves baseline hazard function unspecified (6). On the other hand, parametric modeling assumes functional forms of the baseline hazard, by which a complete description of survival time can be performed (7). Interesting statistics in survival analysis include among others the survival probability at a given time point, and median survival time. If investigators are interested in several statistics for survival data, more than one transformation of the linear predictor would be plotted in nomogram.

```
> library(survival)
> lung$sex<-factor(lung$sex,labels=c('male','female'))
```

To perform survival analysis, the survival package is employed. In this package, there is a data frame named lung containing variables of patients with advanced lung cancer from the North Central Cancer Treatment Group. Performance score (ph.ecog) rate how well the patient can perform usual daily activities (8). The variable sex is transformed to a factor, with labels of male and female.

```
> mod.sur <- psm(Surv(time,status) ~ ph.ecog+sex+age,
                lung, dist='weibull')
```

Weibull survival model is fit with psm() function, which first takes an object of class Surv returned by Surv(). In the example, only three variables ph.ecog, sex and age are used. Weibull distribution of survival time is specified with the "dist='weibull'" argument. Other options include "exponential", "gaussian", "logistic", "lognormal" and "loglogistic".

```
> med <- Quantile(mod.sur)
> surv <- Survival(mod.sur)
```

The above two lines define two functions med and surv. The med() will take values of linear predictor and return median survival time. By default, the function returns median survival time, but survival time of other quantiles can be specified with q argument. The surv() function will take a time and linear predictor as arguments, and return the probability of survival to that time. R code to produce nomograms for survival data

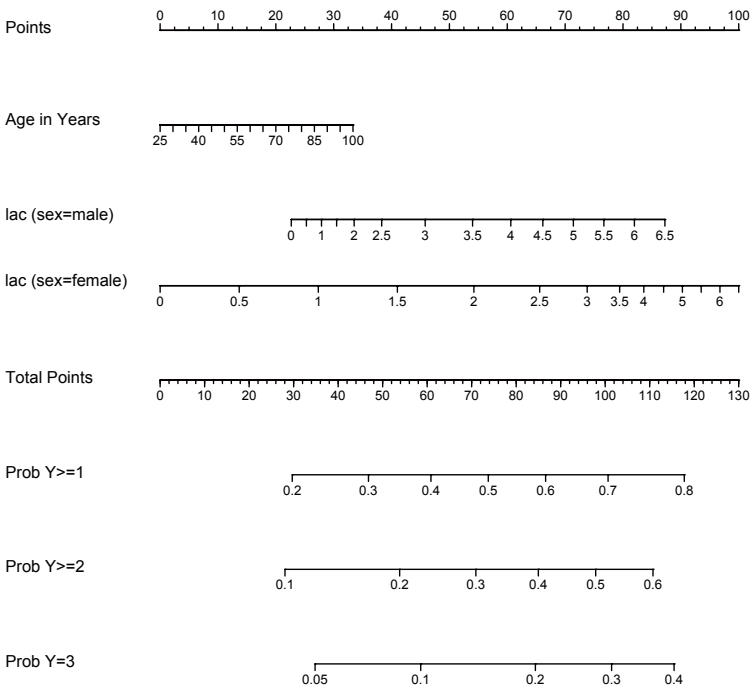


Figure 2 Nomogram for ordinal outcome variable Y. There are three function axes denoting the probability of different sets of outcome categories.

are as follows.

```
> ddist <- datadist(lung)
> options(datadist='ddist')
> nom.sur1<-nomogram(mod.sur,
      fun=list(function(x) med(lp=x, q=0.5),
           function(x) med(lp=x,q=0.25)),
      funlabel=c("Median Survival Time",
           "1Q Survival Time"),
           lp=F)
> plot(nom.sur1,
     fun.side=list(c(rep(1,7),3,1,3,1,3),rep(1,7)),
     col.grid = c("red","green"))
> nom.sur2 <- nomogram(mod.sur, fun=list(function(x)
surv(200, x),
                  function(x) surv(400, x)),
      funlabel=c("200-Day Survival Probability",
           "400-Day Survival Probability"),
            lp=F)
> plot(nom.sur2,
     fun.side=list(c(rep(c(1,3),5),1,1,1,1,1),
                c(1,1,1,rep(c(3,1),6))),
     xfrac=.7,
     col.grid = c("red","green"))
```

Figure 3 shows the nomogram for median and 1-Q survival time. For practical users, it is easy to refer a patient with given characteristics to the corresponding median and 1-Q survival time. In *Figure 4*, there are two function axes for 200- and 400-day survival probabilities.

Nomogram for semiparametric survival models

Semiparametric survival model, also known as the Cox proportional hazard model, is far more popular in medical literature. Thus, in this section we will show how to fit a Cox proportional hazard model with R and draw a nomogram based on the Cox proportional hazard model. Again we use the lung dataset as the above example. The Cox proportional hazard model can be fit with cph() function in the rms package.

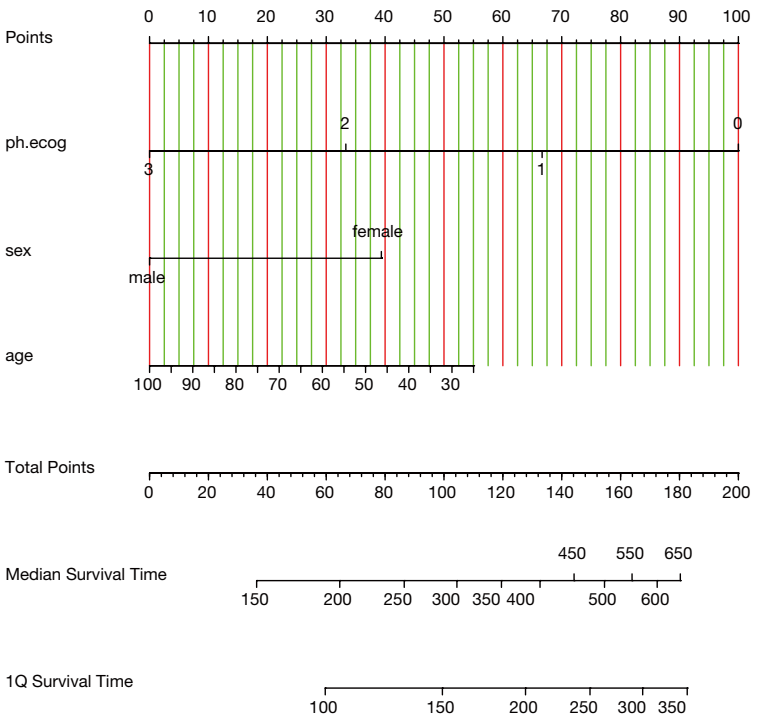


Figure 3 Nomogram for median and 1-Q survival time.

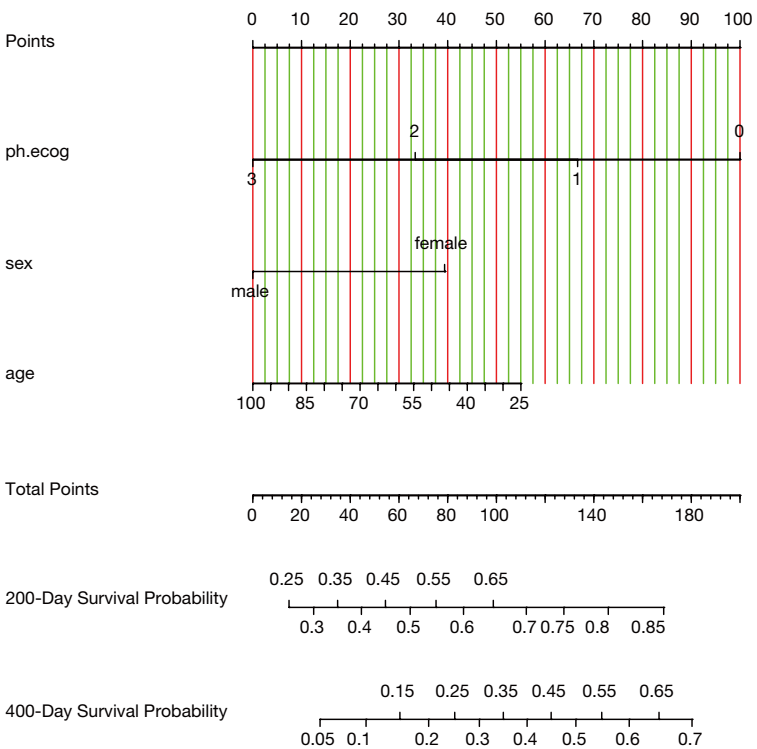


Figure 4 Nomogram for 200- and 400-day survival probabilities.

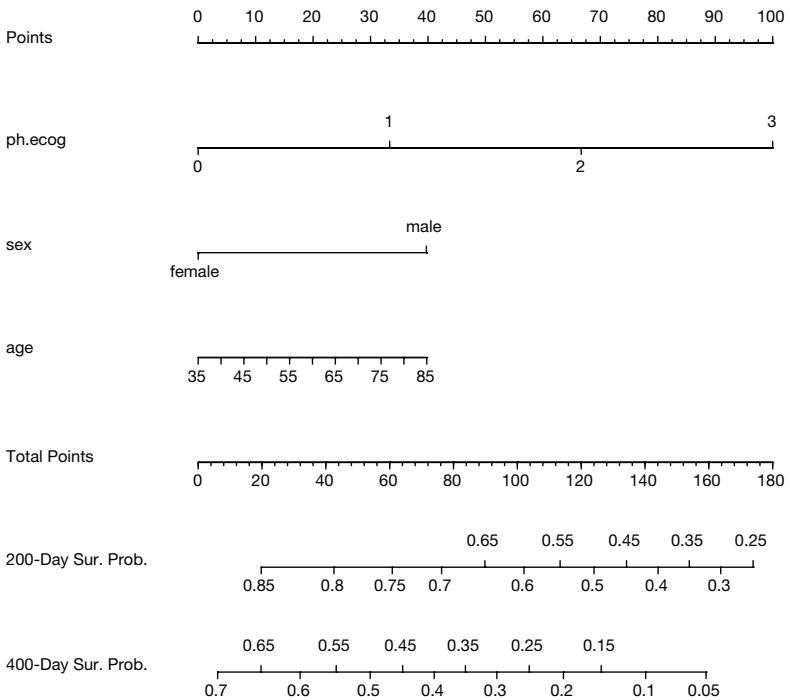


Figure 5 Nomogram for 200- and 400-day survival probabilities. Semiparametric survival model is used to construct the nomogram.

```
> mod.cox <- cph(Surv(time,status) ~ ph.ecog+sex+age,
                 lung, surv=TRUE)
> ddist <- datadist(lung)
> options(datadist='ddist')
> surv.cox <- Survival(mod.cox)
> nom.cox <- nomogram(mod.cox, fun=list(function(x)
surv.cox(200, x),
                      function(x) surv.cox(400, x)),
                    funlabel=c("200-Day Sur. Prob.",
                      "400-Day Sur. Prob."),
                    lp=F)
> plot(nom.cox,
       fun.side=list(c(rep(c(1,3),5),1,1,1,1),
                     c(1,1,1,rep(c(3,1),6))))
```

The result is shown in *Figure 5*. Of note, the linear predictor axis is omitted because most users don't care about it. Other arguments of the nomogram() function and interpretation of results are similar to the previous examples.

Acknowledgements

None.

Footnote

Conflicts of Interest: The authors have no conflicts of interest to declare.

References

1. Kattan MW, Marasco J. What is a real nomogram? Semin Oncol 2010;37:23-26.
2. Su D, Zhou X, Chen Q, et al. Prognostic Nomogram for Thoracic Esophageal Squamous Cell Carcinoma after Radical Esophagectomy. PLoS One 2015;10:e0124437.
3. Zhang Z. Model building strategy for logistic regression: purposeful selection. Ann Transl Med 2016;4:111.
4. Zhang Z. Residuals and regression diagnostics: focusing on logistic regression. Ann Transl Med 2016;4:195.
5. Ordinal Logistic Regression. In: Harrell FE. Regression Modeling Strategies. New York, NY: Springer; 2001:331-43.

6. Introduction to Survival Analysis. In: Harrell FE. Regression Modeling Strategies. New York, NY: Springer; 2001:389-412.
7. Harrell FE. Parametric Survival Models. In: Regression Modeling Strategies. New York, NY: Springer; 2001:413-42.
8. Loprinzi CL, Laurie JA, Wieand HS, et al. Prospective evaluation of prognostic variables from patient-completed questionnaires. North Central Cancer Treatment Group. J Clin Oncol 1994;12:601-607.

Cite this article as: Zhang Z, Kattan MW. Drawing Nomograms with R: applications to categorical outcome and survival data. Ann Transl Med 2017;5(10):211. doi: 10.21037/atm.2017.04.01

Statistical description for survival data

Zhongheng Zhang

Abstract: Statistical description is always the first step in data analysis. It gives investigator a general impression of the data at hand. Traditionally, data are described as central tendency and deviation. However, this framework does not fit for the survival data (also termed time-to-event data). Such data type contains two components. One is the survival time and the other is the status. Researchers are usually interested in the probability of event at a given survival time point. Hazard function, cumulative hazard function and survival function are commonly used to describe survival data. Survival function can be estimated using Kaplan-Meier estimator, which is also the default method in most statistical packages. Alternatively, Nelson-Aalen estimator is available to estimate survival function. Survival functions of subgroups can be compared using log-rank test. Furthermore, the article also introduces how to describe time-to-event data with parametric modeling.

Keywords: Survival analysis; parametric model; Kaplan-Meier; log-rank

Submitted May 03, 2016. Accepted for publication Jun 12, 2016.
doi: 10.21037/atm.2016.07.17
View this article at: http://dx.doi.org/10.21037/atm.2016.07.17

Introduction

Survival analysis encompasses a wide variety of methods for analyzing time-to-event data. In biomedicine, the event of interest may include death, visit to emergency room, myocardial infarction, stroke and intensive care unit (ICU) readmission. The response variable is time. If there is no censoring, traditional regression model can be used to deal with survival data. However, the presence of censoring introduces bias in estimation of survival time distribution. Furthermore, survival data are typically non-negative and positively skewed. Therefore, survival data should be managed with specially designed methods. Instead of focusing on the time (how long) a subject can survive, survival analysis examines the probability of an event at survival time "t" given subjects who are under observation at that survival time. By considering only subjects who are under observation, survival times and survival probability can be estimated without bias given that subjects under observation are representative of the whole study population. A prerequisite assumption is that the censoring mechanism is unrelated to survival time. One scenario that violates this assumption is that when clinical condition deteriorates (e.g., indicating shortened survival time), subjects are more likely to quit a study and they are lost to follow-up. Such censoring is related to survival time. As a result, survival time of subjects who withdraw the study is shorter than those who are still under observation. This is called informative censoring in statistical term.

Here I will not go further into discussion of details and mathematical equations on survival analysis. Instead, I would like to show how survival analysis is performed in R and principles will be introduced with a working example. The first article of this theme focuses on statistical description of survival data.

Working example

The lung dataset (NCCTG Lung Cancer Data) contained in survival package is employed as the working example. This is a dataset containing right-censored survival data.

```
> library(survival)
> str(lung)
'data.frame':        228 obs. of  10 variables:
 $ inst     : num  3 3 3 5 1 12 7 11 1 7 ...
 $ time     : num  306 455 1010 210 883 ...
 $ status   : num  2 2 1 2 2 1 2 2 2 2 ...
 $ age      : num  74 68 56 57 60 74 68 71 53 61 ...
 $ sex      : num  1 1 1 1 1 1 2 2 1 1 ...
 $ ph.ecog  : num  1 0 0 1 0 1 2 2 1 2 ...
```

```
$ ph.karno : num  90 90 90 90 100 50 70 60 70 70 ...
$ pat.karno: num  100 90 90 60 90 80 60 80 80 70 ...
$ meal.cal : num  1175 1225 NA 1150 NA ...
$ wt.loss  : num  NA 15 15 11 0 0 10 1 16 34 ...
```

The dataset contains 228 observations of 10 variables. Institution code (inst) is used to mark different institutions from which patients come. Survival time (time) is measured in days. Censoring status (status) is coded 1 for censored and 2 for dead. Age (age) is measured in years. Male and female sexes are coded as 1 and 2, respectively. ECOG performance score (ph.ecog), Karnofsky performance score rated by physicians (ph.karno) and patients (pat.karno) are also recorded. The last two variables are calories consumed at meals (meal.cal) and weight loss in the last six months (wt.loss).

In real world setting, interval censoring can occur when periodic assessments are used to assess if the event of interest has occurred. In this situation, the survival time is not known precisely until an event of interest occurs. Instead, we only have knowledge that the event of interest falls into a particular interval (1,2). The heart dataset (Stanford Heart Transplant data) is a prototype of interval data.

```
> head(heart)
  start stop event age       year surgery transplant id
1     0   50     1 -17.155373 0.1232033 0         0         1
2     0    6     1   3.835729 0.2546201 0         0         2
3     0    1     0   6.297057 0.2655715 0         0         3
4     1   16     1   6.297057 0.2655715 0         1         3
5     0   36     0  -7.737166 0.4900753 0         0         4
6    36   39     1  -7.737166 0.4900753 0         1         4
```

The start and stop variables represent the entry and exit time for an observation period. Note that one subject can take two rows. Age can take negative values because it is centered at 48.

Declaring a survival data

Survival analysis requires to create a survival object using Surv() function. That is equal to declaring a survival data. The survival object is frequently used as response variable in a model formula.

```
> lung.sur<-Surv(lung$time, lung$status)
```

```
> heart.sur<-Surv(heart$start, heart$stop, heart$event)
```

The lung.sur and heart.sur are objects of class Surv. The Surv() takes the general form:

```
Surv(time, time2, event,
  type=c('right', 'left', 'interval', 'counting', 'interval2', 'mstate'),
  origin=0)
```

If there are two unnamed arguments as shown in the first line, they will match time and event in that order. If there are three unnamed arguments as that in the second line they match time, time 2 and event. The type argument can usually be omitted.

Nonparametric modeling

Distribution of time-to-event data can be estimated with nonparametric methods such as the Kaplan-Meier estimates. The survfit() function can perform this task.

```
> lung.fit<-survfit(lung.sur~1)
> plot(lung.fit,xlab="Days",ylab="Proportion of subjects")
```

The first argument of survfit() is a formula with the response variable (Surv class object) on the left of the "~" operator. The number "1" on the right indicates a single survival curve. The default type of survival curve is estimated using Kaplan-Meier method. If an object of class "survfit" is passed to the generic function plot(), a survival curve is plotted together with estimated confidence interval (CI) (*Figure 1*). The default CI is 95%, and it can be customized using conf.int argument in survfit() function. The cross symbol in the figure represents censored observations.

The summary statistics including cumulative survival probability, standard error and 95% CI can be displayed with the following code.

```
> summary(lung.fit)
Call: survfit(formula = lung.sur ~ 1)

time    n.risk  n.event  survival  std.err  lower    upper
                                            95% CI   95% CI
5       228     1        0.9956    0.00438  0.9871   1.000
11      227     3        0.9825    0.00869  0.9656   1.000
```

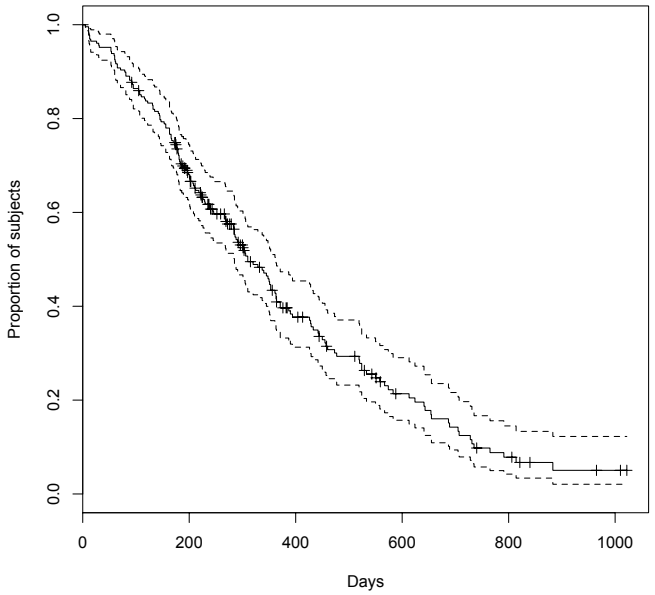


Figure 1 Survival curve plots probability of survival against survival time. The dashed lines are lower and upper limits of 95% confidence interval (CI).

```
12      224     1       0.9781  0.00970 0.9592  0.997
                (omitted to save space)
791     9       1       0.0783  0.02462 0.0423  0.145
814     7       1       0.0671  0.02351 0.0338  0.133
883     4       1       0.0503  0.02285 0.0207  0.123
```

A key feature of the survfit.formula is its ability to fit survival model by strata. The strata are specified by variables on the right of the "~" operator. Factor variables are connected using "+" symbol. For instance, we can compare survival curves by different ECOG performance scores.

```
> lung.fit.strata<-survfit(lung.sur~ph.ecog,lung)
> plot(lung.fit.strata, lty = 2:4,col=2:4,xlab="Days",
ylab="Proportion of subjects")
> legend(700, .9, c("ph.karno=0", "ph.karno=1",
"ph.karno=2","ph.karno=3"), lty = 2:4,col=2:4)
```

In the above example, a variable ph.ecog is added to the right side of the "~" operator to make separate survival curves for different ECOG performance score levels. You can try to add sex variable, which will make 8 (2×4) combinations by different values of sex and ECOG performance scores. The argument lty and col assign different line types and colors to distinguish survival curves. Legend() function is used to add annotations (*Figure 2*).

CIs for the Kaplan-Meier estimator

Although the survfit() function allows adjustment of CI for Kaplan-Meier estimator, the underlying method is limited. Here I introduce the km.ci package for computing pointwise and simultaneous CIs for the Kaplan-Meier estimator. Many options exist for computing CI. The method argument allows assigning a string character including "peto", "linear", "log", "loglog", "rothman", "grunkemeier", "hall-wellner", "loghall", "epband" and "logep". In simulation study, Afifi and colleagues found that the Rothman-Wilson ("rothman"), log and Arcsin transformation methods perform better than other methods (3). Confidence band estimation for Kaplan-Meier estimator is an area of active research and comprehensive enumeration of these methods are out of scope of the current article. References are provided for interested readers to explore more on this topic (4-8). The following example illustrates how to compute CI.

```
> install.packages("km.ci")
> library(km.ci)
> a<-km.ci(lung.fit, conf.level=0.95, tl=NA, tu=NA,
method="loghall")
> plot(a, lty=2, lwd=2)
> lines(lung.fit, lwd=2, lty=1)
> lines(lung.fit, lwd=1, lty=4, conf.int=T)
> linetype<-c(1, 2, 4)
> legend(600, .9, c("Kaplan-Meier", "Hall-Wellner", "Pointwise"),
lty=(linetype))
```

The first argument of km.ci() function is a survival object. The default level of two-sided CI on survival curve is 0.95. Lower (tl) and upper (tu) time boundaries for the simultaneous confidence limits can be specified. If they are missing as in our case, the smallest and largest event times are employed. The Hall-Wellner method is used to compute the confidence bands (9). *Figure 3* compares CIs obtained by different methods.

Nelson-Aalen estimator of the survivorship function

Kaplan-Meier estimator of survival function is the most

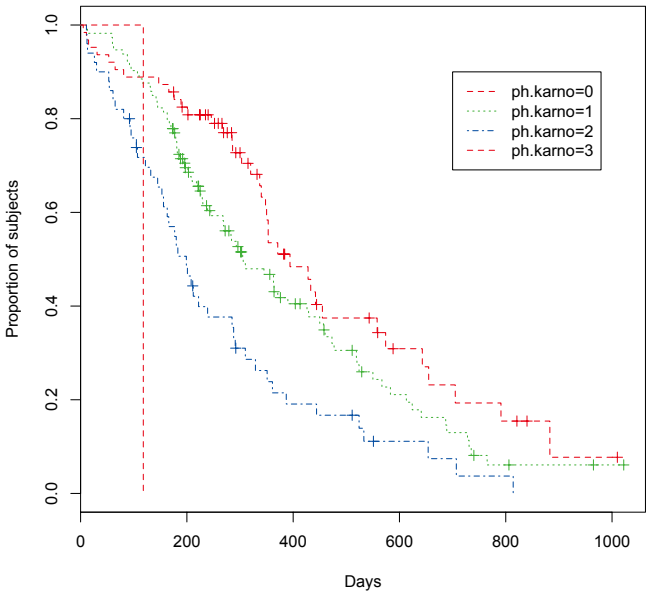


Figure 2 Survival curves stratified by ph.ecog variable. Different ECOG performance score levels are represented by different line types and colors.

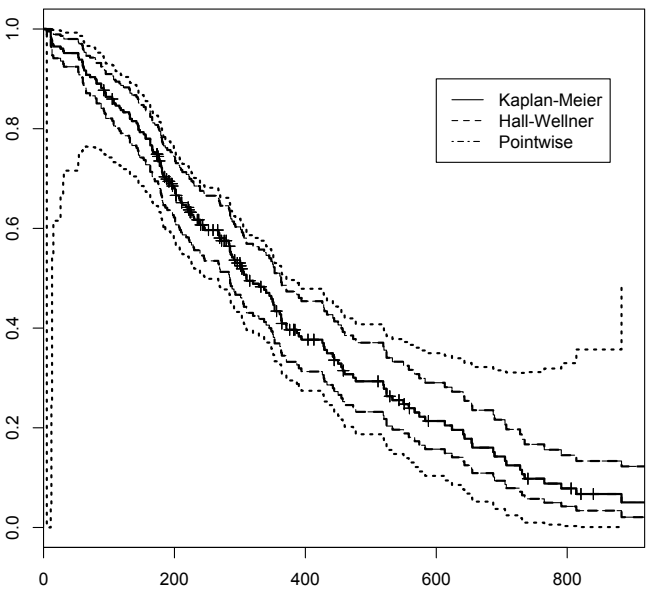


Figure 3 Confidence intervals (CIs) estimated by different methods. The dashed line represents 95% CI estimated by Hall-Wellner method. The dash-dot line is estimated by the default method in survfit() function.

frequently used estimator, partly because it is the default method in many software packages. Alternatively, survival function can be derived from cumulative hazard function. Nelson and Aalen have proposed an easily computed estimator of cumulative hazard, which is now referred to as Nelson-Aalen estimator (10,11). Nelson-Aalen estimator can be computed via cox regression using coxph() function in R.

```
> aalen.fit<- survfit(coxph(lung.sur~1), type="aalen")
> summary(aalen.fit)
Call: survfit(formula = coxph(lung.sur ~ 1), type = "aalen")
```

time	n.risk	n.event	survival	std.err	lower 95% CI	upper 95% CI
5	228	1	0.9956	0.00437	0.9871	1.000
11	227	3	0.9826	0.00865	0.9657	1.000
12	224	1	0.9782	0.00965	0.9594	0.997
(omitted to save space)						
791	9	1	0.0824	0.02504	0.0454	0.149
814	7	1	0.0714	0.02398	0.0370	0.138
883	4	1	0.0556	0.02329	0.0245	0.126

```
> plot(aalen.fit,col="red",lwd=1,lty=1)
> lines(lung.fit, lwd=1, lty=1)
> legend(600, .9, c("Nelson-Aalen","Kaplan-Meier"), lty=c(1,1),
col=c("red","black"))
```

The Nelson-Aalen estimator is designated by type="aalen" argument. Nelson-Aalen estimator of the survival function is always greater than or equal to the Kaplan-Meier estimator (*Figure 4*). If the size of the risk sets is large relative to the number of events, there will be little practical difference between the Nelson-Aalen and the Kaplan-Meier estimators of the survival function.

Comparison between survival curves

In research practice, an important work is to test whether two survival curves are different based on observed data. In our example, we want to explore whether there is enough reason to reject the null hypothesis that survival curves for lung cancer patients with different ECOG performance scores are similar. Function survdiff() calls a family of tests defined by parameter rho. With 'rho =0' it is equivalent to the log-rank or Mantel-Haenszel test, and with 'rho =1' it is the Peto &

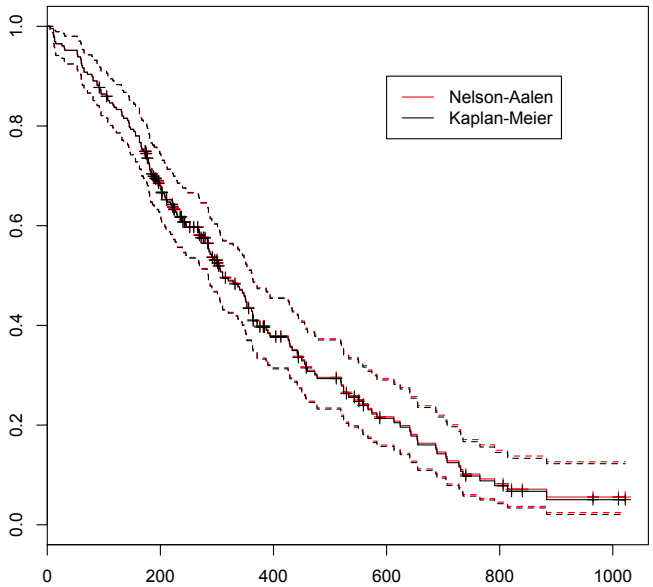


Figure 4 Comparison of survival curves estimated by Kaplan-Meier and Nelson-Aalen methods. Nelson-Aalen estimator of the survival function is always greater than or equal to the Kaplan-Meier estimator.

Peto modification of the Gehan-Wilcoxon test (12).

```
> survdiff(lung.sur~ph.ecog,lung)
Call:
survdiff(formula = lung.sur ~ ph.ecog, data = lung)

n=227, 1 observation deleted due to missingness.

             N  Observed  Expected  (O-E)^2/E  (O-E)^2/V
ph.ecog=0   63        37    54.153     5.4331     8.2119
ph.ecog=1  113        82    83.528     0.0279     0.0573
ph.ecog=2   50        44    26.147    12.1893    14.6491
ph.ecog=3    1         1     0.172     3.9733     4.0040

Chisq= 22 on 3 degrees of freedom, p=6.64e-05
```

Similar to other survival functions, the first argument of survdiff() function is a formula defining the subgroups to be compared. The left side of the "~" symbol is a Surv class object. If subgroups are defined by combinations of factor variables, they can be connected with "+" operators. The output table shows the observed and expected number of events. The Chi-square statistic for a test of equality shows that the probability of observing current distribution of survival curves is extremely small. Thus, there is enough reason that survival curves are different among subgroups with different ECOG performance scores. If there are two subgroups to be compared, survdiff() performs log-rank test. Otherwise, the function implements statistical test according to the method proposed by Harrington and Fleming (13).

Parametric model

Another way to describe survival data is to assume a mathematical model and then estimate coefficients with maximum likelihood method. Parametric modeling is more appealing in multivariable regression model. For the purpose of statistical description, the intercept only model is employed. Full description of parametric modeling will be introduced in future articles. Here we only take a glimpse of how it works.

```
> par.wei<-survreg(lung.sur~1,dist="w")
> par.wei
Call:
survreg(formula = lung.sur ~ 1, dist = "w")

Coefficients:
(Intercept)
  6.034904

Scale= 0.7593936

Loglik(model)= -1153.9   Loglik(intercept only)= -1153.9
n= 228
```

The parametric survival model is fit with survreg() function. The first argument is a formula describing the structure of the model. The above code builds an intercept-only model and thus "1" is assigned on the right of the "~" symbol. I arbitrarily assumed that T (survival time) follows a Weibull distribution and assigned "w" for the dist argument. Before taking a close look at the output of par.wei, we must review parameters of Weibull distribution.

Let T denotes a continuous non-negative random variable representing survival time. T follows Weibull distribution with parameters lambda (λ) and kappa (κ). The hazard function can be written as Eq. [1]:

$$h(t)=\lambda^{\kappa}\kappa t^{\kappa-1} \quad [1]$$

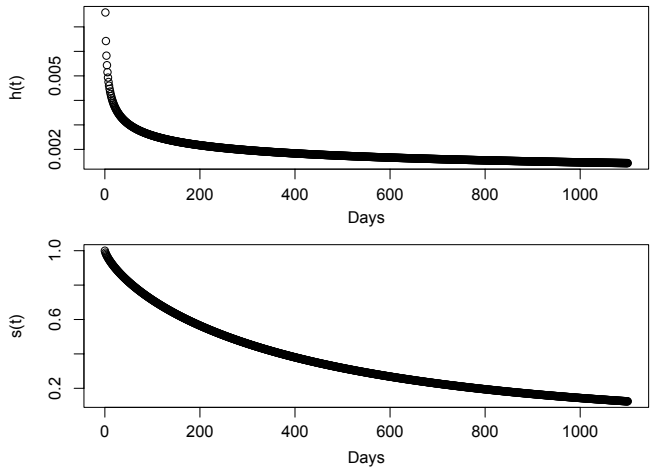


Figure 5 Hazard and survival functions fitted by Weibull parametric model.

and the survival function can be written as Eq. [2]:

$$s(t)=e^{-(\lambda t)k} \quad [2]$$

Then the hazard function and survival function can be plotted with a few lines of commands.

```
> kappa<-par.wei$scale
> lambda<-exp(-par.wei$coeff[1])
> zeit<-seq(from=0,to=1100,length.out=1000)
> s<-exp(-(lambda*zeit)^kappa)
> h<-lambda^kappa *kappa*zeit^(kappa-1)
> par(mfrow=c(2,1))
> plot(zeit,h,xlab="Days",ylab="h(t)")
> plot(zeit,s,xlab="Days",ylab="s(t)")
```

The upper panel shows that the hazard h(t) decreases over time (*Figure 5*). The lower panel shows the survival function, which is comparable to *Figure 1*. The only distinction is that *Figure 1* is depicted with nonparametric method.

Summary

Statistical description is always the first step in data analysis. It gives investigator a general impression of the data at hand. Traditionally, data are described as central tendency and deviation. However, this framework does not fit to the survival data (also termed time-to-event data). Such data type contains two components. One is the survival time and the other is the status. Researchers are interested in the probability of the event at a given survival time point. Hazard function, cumulative hazard function and survival function are commonly used to describe survival data. Survival function can be estimated using Kaplan-Meier estimator, which is also the default method in most statistical packages. Alternatively, Nelson-Aalen estimator is available to estimate survival function. Survivor functions of subgroups can be compared using log-rank test. Furthermore, the article also introduces how to describe time-to-event data with parametric modeling.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Radke BR. A demonstration of interval-censored survival analysis. Prev Vet Med 2003;59:241-256.
2. Lindsey JC, Ryan LM. Tutorial in biostatistics methods for interval-censored data. Stat Med 1998;17:219-238.
3. Afifi AA, Elashoff RM, Lee JJ. Simultaneous non-parametric confidence intervals for survival probabilities from censored data. Stat Med 1986;5:653-662.
4. Fay MP, Brittain EH, Proschan MA. Pointwise confidence intervals for a survival distribution with small samples or heavy censoring. Biostatistics 2013;14:723-736.
5. Fay MP, Brittain EH. Finite sample pointwise confidence intervals for a survival distribution with right-censored data. Stat Med 2016;35:2726-2740.
6. Ahmed N, Subramanian S. Semiparametric simultaneous confidence bands for the difference of survival functions. Lifetime Data Anal 2015. [Epub ahead of print].
7. Parzen MI, Wei LJ, Ying Z. Simultaneous Confidence Intervals for the Difference of Two Survival Functions. Scandinavian Journal of Statistics 1997;24:309-314.
8. Strobl R, Dirschedl P. S41.5: Comparison of simultaneous and pointwise confidence bands for Kaplan-Meier estimators. Biometrical Journal 2004;46:90.
9. Hall WJ, Wellner JA. Confidence bands for a survival curve from censored data. Biometrika 1980;67:133-143.

10. Aalen O. Nonparametric Inference for a Family of Counting Processes. The Annals of Statistics 1978;6:701-726.
11. Nelson W. Theory and Applications of Hazard Plotting for Censored Failure Data. Technometrics 1972;14:945-966.
12. David KG, Mitchel K. Survival Analysis: A Self-Learning Text. Biometrics 2006;62:312.
13. Harrington DP, Fleming TR. A class of rank test procedures for censored survival data. Biometrika 1982;69:553-566.

Cite this article as: Zhang Z. Statistical description for survival data. Ann Transl Med 2016;4(20):401. doi: 10.21037/atm.2016.07.17

Parametric regression model for survival data: Weibull regression model as an example

Zhongheng Zhang

Abstract: Weibull regression model is one of the most popular forms of parametric regression model that it provides estimate of baseline hazard function, as well as coefficients for covariates. Because of technical difficulties, Weibull regression model is seldom used in medical literature as compared to the semi-parametric proportional hazard model. To make clinical investigators familiar with Weibull regression model, this article introduces some basic knowledge on Weibull regression model and then illustrates how to fit the model with R software. The *SurvRegCensCov* package is useful in converting estimated coefficients to clinical relevant statistics such as hazard ratio (HR) and event time ratio (ETR). Model adequacy can be assessed by inspecting Kaplan-Meier curves stratified by categorical variable. The *eha* package provides an alternative method to model Weibull regression model. The check.dist() function helps to assess goodness-of-fit of the model. Variable selection is based on the importance of a covariate, which can be tested using anova() function. Alternatively, backward elimination starting from a full model is an efficient way for model development. Visualization of Weibull regression model after model development is interesting that it provides another way to report scientific findings.

Keywords: Survival analysis; parametric model; Weibull regression model

Submitted May 20, 2016. Accepted for publication Jun 23, 2016.
doi: 10.21037/atm.2016.08.45
View this article at: http://dx.doi.org/10.21037/atm.2016.08.45

Introduction

While semi-parametric model focuses on the influence of covariates on hazard, fully parametric model can also calculate the distribution form of survival time. Advantages of parametric model in survival analysis include: (I) the distribution of survival time can be estimated; (II) full maximum likelihood can be used to estimate parameters; (III) residuals can represent the difference between observed and estimated values of time; (IV) estimated parameters provide clinically meaningful estimates of the effect (1). There are a variety of models to be specified for accelerated failure time model including exponential, Weibull and log-logistic regression models. In this article, Weibull regression model is employed as an example to illustrate parametric model development and visualization.

Weibull regression model

Before exploring R for Weibull model fit, we first need to review the basic structure of the Weibull regression model. The distribution of time to event, T, as a function of single covariate is written as (1):

$$In(T) = \beta_0 + \beta_1 x + \sigma\varepsilon \quad [1]$$

where β_1 is the coefficient for corresponding covariate, ε follows extreme minimum value distribution G(0, σ)and σ is the shape parameter. This is also called the accelerated failure-time model because the effect of the covariate is multiplicative on time scale and it is said to "accelerate" survival time. In contrast, the effect of covariate is multiplicative on hazard scale in the proportional hazard model. The hazard function of Weibull regression model in the proportional hazards form is:

$$\begin{aligned} & h(t,x,\beta,\lambda) \\ & = \lambda t^{\lambda-1} e^{-1(\beta_0+\beta_1 x)} \\ & = \lambda t^{\lambda-1} e^{-\lambda\beta_0} e^{-\lambda\beta_1 x} \\ & = \lambda\gamma t^{\lambda-1} e^{-\lambda\beta_1 x} \\ & = h_0(t) e^{\theta_1 x} \end{aligned} \quad [2]$$

where $\gamma = e^{\frac{-\beta_0}{\sigma}} = e^{\theta_0}$, $\theta_1 = -\beta_1/\sigma$, and the baseline hazard function is $h_0(t) = \lambda\gamma t^{\lambda-1}$. σ is a variance-like parameter on log-time scale. $\gamma = 1/\sigma$ is usually called a scale parameter. Parameter λ is a shape parameter. Parameter θ_1 has a hazard ratio (HR) interpretation for subject-matter audience.

The accelerated failure-time form of the hazard function can be written as:

$$\begin{aligned} & h(t,x,\beta,\lambda) \\ & = \lambda t^{\lambda-1} e^{-\lambda(\beta_0+\beta_1 x)} \\ & = \lambda\gamma\left(te^{-\beta_1 x}\right)^{\lambda-1} e^{-\beta_1 x} \end{aligned}$$

Weibull regression model can be written in both accelerated and proportional forms, allowing for simultaneous description of treatment effect in terms of HR and relative change in survival time [event time ratio (ETR)] (2).

Fitting Weibull regression model with R

The survreg() function contained in *survival* package is able to fit parametric regression model. Let's first load the package into the workspace. To build a Weibull regression model, the *dist* argument should be set to a string value "weibull", indicating that the distribution of response variable follows Weibull distribution. The summary() function is to print content of the returned object of class *survreg*.

```
> library(survival)
> wei.lung<-survreg(Surv(time, status)~ph.ecog+sex+age,lung,
dist='weibull')
> summary(wei.lung)

Call:
survreg(formula = Surv(time, status) ~ ph.ecog + sex + age, data
= lung,
 dist = "weibull")
              Value Std. Error     z       p
(Intercept)  6.27344    0.45358 13.83 1.66e-43
ph.ecog     -0.33964    0.08348 -4.07 4.73e-05
sex          0.40109    0.12373  3.24 1.19e-03
age         -0.00748    0.00676 -1.11 2.69e-01
Log(scale)  -0.31319    0.06135 -5.11 3.30e-07

Scale= 0.731

Weibull distribution
Loglik(model)= -1132.4 Loglik(intercept only)= -1147.4
        Chisq= 29.98 on 3 degrees of freedom, p= 1.4e-06
Number of Newton-Raphson Iterations: 5
n=227 (1 observation deleted due to missingness)
```

The output first recalls the structure of the Weibull regression model, including the covariates. Next, the coefficients of each covariate are shown, together with standard error and P values. Scale is an important parameter in Weibull regression model and is shown in the following line. A log likelihood test shows that the model is significantly better than null model (P=1.4e–06). However, the estimated coefficients are not clinically meaningful. That is why Weibull regression model is not widely used in medical literature. Since Weibull regression model allows for simultaneous description of treatment effect in terms of HR and relative change in survival time, ConvertWeibull() function is used to convert output from survreg() to more clinically relevant parameterization. The function is contained in *SurvRegCensCov* package and we need to install it first.

```
> install.packages("SurvRegCensCov")
> library(SurvRegCensCov)
> ConvertWeibull(wei.lung,conf.level = 0.95)
$vars
             Estimate          SE
lambda  0.0001876914 0.0001506884
gamma   1.3677851193 0.0839087686
ph.ecog 0.4645519368 0.1136759822
sex     -0.5486056737 0.1673299432
age     0.0102247948 0.0092298732

$HR
             HR        LB        UB
ph.ecog 1.5913010 1.2734772 1.9884447
sex     0.5777548 0.4162096 0.8020013
age     1.0102772 0.9921654 1.0287197

$ETR
             ETR        LB        UB
ph.ecog 0.7120280 0.6045610 0.8385984
sex     1.4934525 1.1718447 1.9033242
age     0.9925524 0.9794818 1.0057975
```

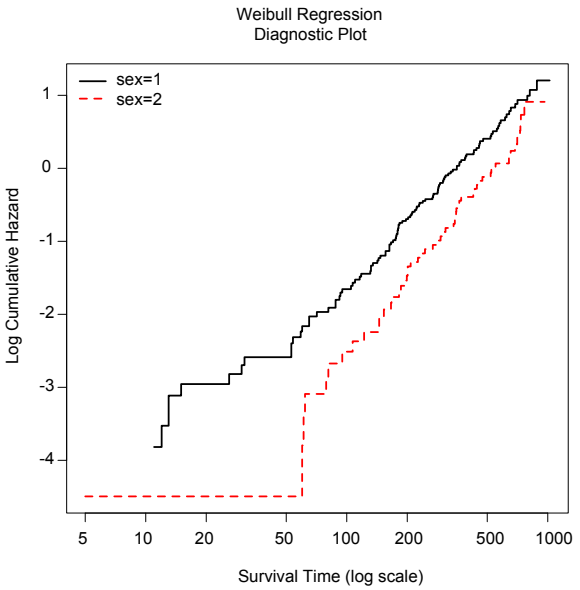



Figure 1 Weibull regression diagnostic plot showing that the lines for male and female are generally parallel and linear in its scale.

The first table of the output displays parameters of the Weibull regression model. Lambda and gamma are scale and shape parameters of Weibull distribution. The estimate for each covariate is different from that displayed in the value column of the summary() output. The relationship can be described by an equation $\beta = -\alpha/\sigma$, where α is parameter for each of the covariate and σ is the scale (2). In our example, β is the estimate in the first table of the ConvertWeibull() output and α is displayed in the output of summary(wei.lung). The second table shows the HR and corresponding 95% confidence interval. The last table displays the ETR and its 95% confidence interval. Female reduces the risk of death compared to male by 42% (HR =0.58), and female significantly increases the survival time by approximately 50% (ETR =1.49). Although HR is more widely reported in the medical literature and is familiar to clinicians, ETR may be easier to understand.

Alternatively, the Weibull regression model can be fit with WeibullReg() function. In essence, it is the combination of survreg() and ConvertWeibull().

```
> wei.lung.alt<-WeibullReg(Surv(time,
status)~ph.ecog+sex+age,data=lung,conf.level=0.95)
```

Adequacy of the Weibull model

Weibull model with categorical variables can be checked for its adequacy by stratified Kaplan-Meier curves. A plot of log survival time versus log[–log(KM)] will show linear and parallel lines if the model is adequate (3).

```
> WeibullDiag(Surv(time,status)~sex,data=lung)
```

Figure 1 is the Weibull regression diagnostic plot showing that the lines for male and female are generally parallel and linear in its scale.

Weibull regression model with eha package

An alternative way to model Weibull regression model is via *eha* package. This package provides a variety of functions for Weibull regression model. Let's now first install the package and load it into the workspace.

```
> install.packages("eha")
> library(eha)
> lung.alt<-weibreg( Surv(time, status)~age+sex+ph.ecog,
data = lung)
> lung.alt
Call:
weibreg(formula = Surv(time, status) ~ age + sex + ph.ecog,
data = lung)

Covariate       Mean      Coef    Exp(Coef)   se(Coef)   Wald p
age            61.963     0.010     1.010      0.009     0.268
sex             1.439    -0.549     0.578      0.167     0.001
ph.ecog         0.853     0.465     1.591      0.114     0.000

log(scale)                6.273   530.296      0.454     0.000
log(shape)                0.313     1.368      0.061     0.000


Events                    164
Total time at risk        69522
Max. log. likelihood      -1132.4
LR test statistic         30
Degrees of freedom        3
Overall p-value           1.3944e-06
```

The argument of weibreg() function is similar to that of the survreg(). The coefficient of covariates in the above output is the HR in log scale. Thus, the exponentiation of coefficient gives the HR.

Hazard, cumulative hazard, density and survivor

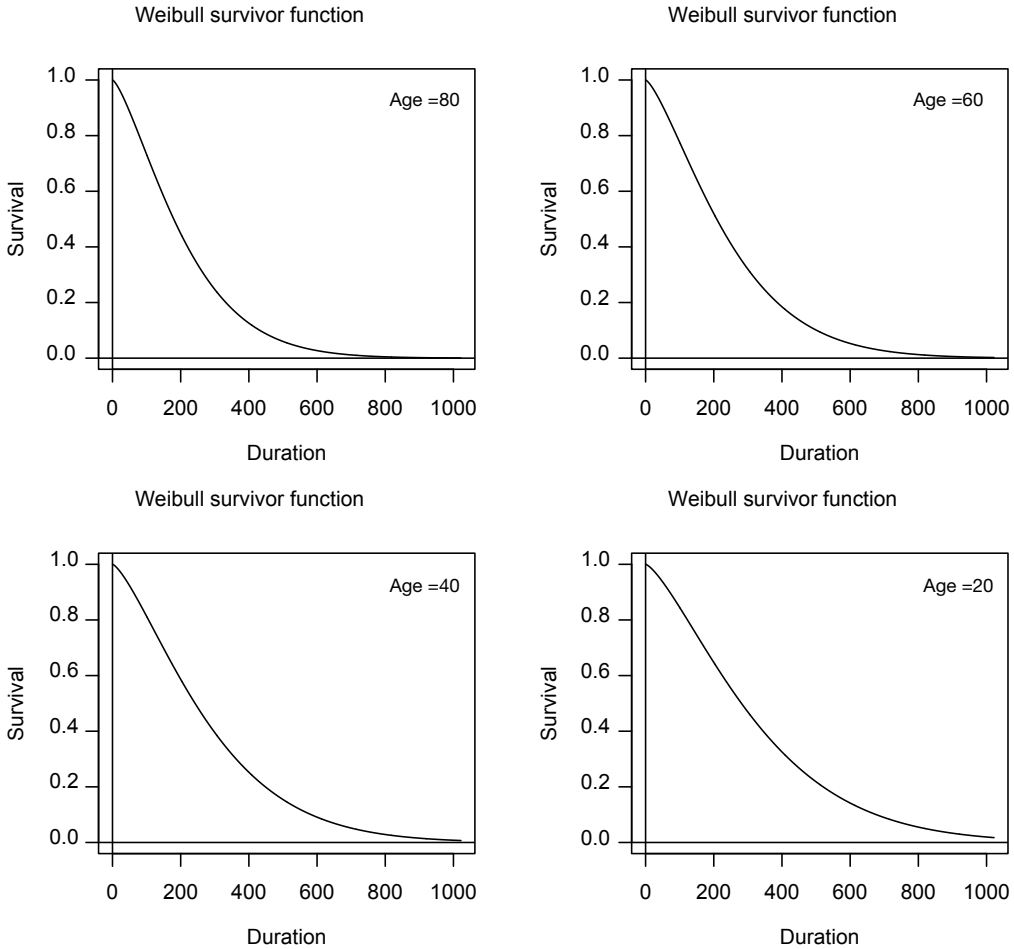


Figure 2 Graphical display of the output of Weibull regression model. The four survivor function plots correspond to ages of 80, 60, 40 and 20. Variables sex and ph.ecog are set to values of 2 and 3, respectively.

functions can be plotted from the output of a Weibull regression model.

```
> par(mfrow=c(2,2))
> plot(lung.alt, fn=c("sur"),new.data=c(80,2,3))
> plot(lung.alt, fn=c("sur"),new.data=c(60,2,3))
> plot(lung.alt, fn=c("sur"),new.data=c(40,2,3))
> plot(lung.alt, fn=c("sur"),new.data=c(20,2,3))
```

Figure 2 is the graphical display of the output of Weibull regression model. The *fn* argument specifies the functions to be plotted. It receives a vector of string values, choosing from "haz", "cum", "den" and "sur". The *newdata* argument specifies covariate values at which to plot the function. If covariates are left unspecified, the default value is the mean of the covariate in the training dataset. In the example, four plots were drawn at age of 80, 60, 40 and 20 years old (in the order from left to right and from top to bottom). The *sex* and *ph.ecog* variables were set at values of 2 and 3, respectively.

Graphical goodness-of-fit test

The *eha* package has a function check.dist() to test the goodness-of-fit by graphical visualization. It compares the cumulative hazards functions for non-parametric and parametric model, requiring objects of "coxreg" and "phreg" as the first and second argument.

```
> phreg.lung<-phreg(Surv(time, status)~ph.ecog+sex+age,
lung, dist='weibull')
> coxreg.lung<-coxreg(Surv(time, status)~ph.ecog+sex+age,
lung)
```

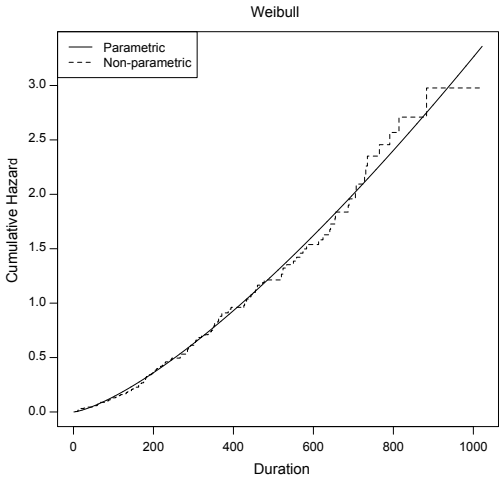



Figure 3 Goodness-of-fit test by graphical comparison between parametric and non-parametric regression models. It appears that the parametric function fits well to the non-parametric function.

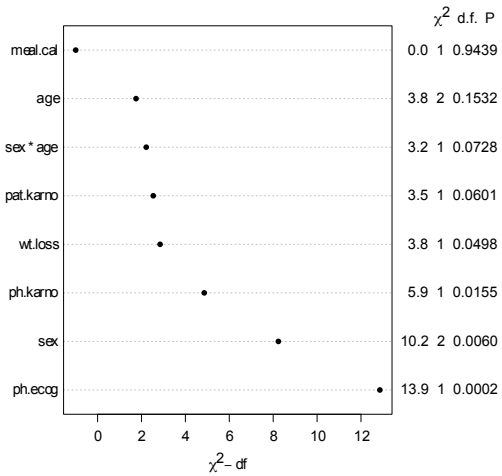


Figure 4 Dot chart showing relative importance of covariates.

```
> check.dist(coxreg.lung,phreg.lung)
```

The solid line is the parametric Weibull cumulative hazard function and the dashed line is non-parametric function. It appears that the parametric function fits well to the semi-parametric function (*Figure 3*). Note the that non-parametric model is closer to the observed data because no function is assumed for the baseline hazard function.

Variable selection and model development

Like generalized linear model development (4), it is essential to include statistically important and clinically relevant covariates into the model in fitting parametric regression model. While clinical relevance is judged by clinical expertise, the statistical importance is determined by software. The anova() function tests the statistical importance of a covariate, an interaction and non-linear terms. The function reports Chi-square statistics and associated P value. Also, it provides dot charts depicting the importance of variables in the model.

```
> psm.lung<-psm(Surv(time, status)~ph.ecog+sex*age+
ph.karno+pat.karno+meal.cal+wt.loss,lung, dist='weibull')
> anova(psm.lung)
     Wald Statistics     Response: Surv(time, status)
```

Factor	Chi-Square	d.f.	P
ph.ecog	13.86	1	0.0002
sex (Factor+Higher Order Factors)	10.24	2	0.0060
All Interactions	3.22	1	0.0728
age (Factor+Higher Order Factors)	3.75	2	0.1532
All Interactions	3.22	1	0.0728
ph.karno	5.86	1	0.0155
pat.karno	3.54	1	0.0601
meal.cal	0.00	1	0.9439
wt.loss	3.85	1	0.0498
sex * age(Factor+HigherOrderFactors)	3.22	1	0.0728
TOTAL	33.18	8	0.0001

```
> plot(anova(psm.lung),margin=c("chisq", "d.f.", "P"))
```

In the example, we included all available covariates into the model to rank their statistical importance. This is often the case in real research setting that researchers have no prior knowledge on which variable should be included. The first argument of psm() function is a formula describing the response variable and covariates, as well as interactions between predictors. The output of anova() includes variable names, Chi-square statistics, degree of freedom and p-values. Dot chart is drawn with generic function plot(). It appears that meal.cal is the least important variable and ph.ecog is the most important one (*Figure 4*). Some pre-specified rules can be applied to inclusion/exclusion of variables (4).

Alternatively, model development can be done with backward elimination on covariates. This method starts with a full model that included all available covariates and

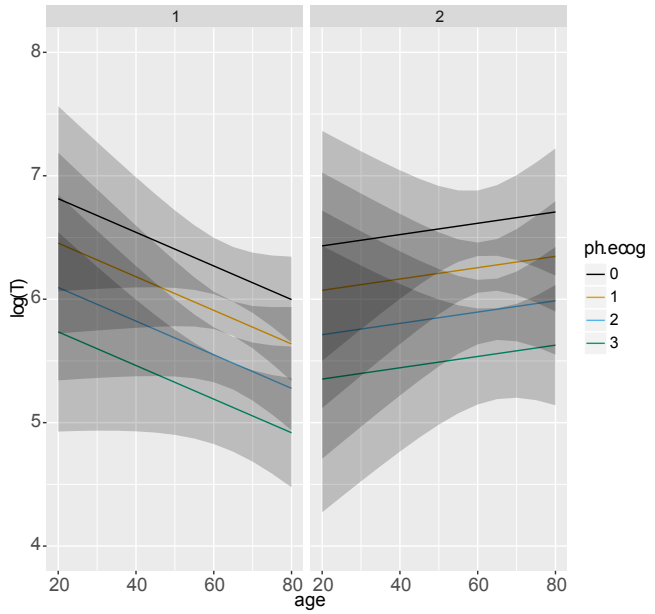


Figure 5 Graphical presentation of the relationship between covariates and survival time on log scale. The effect of age on survival time is dependent on sex. While older patients have shorter survival time in the male, older patients show longer survival time in the female.

then applies Wald test to examine the relative importance of each one. Statistical significance level for a covariate to stay in a model can be specified. R provides a function fastbw() to perform fast backward variable selection.

```
> fastbw(psm.lung,rule="aic")
```

Deleted	Chi-Sq	d.f.	P	Residual	d.f.	P	AIC
meal.cal	0.00	1	0.9439	0.00	1	0.9439	-2.00
sex	1.94	1	0.1634	1.95	2	0.3777	-2.05
pat.karno	2.75	1	0.0970	4.70	3	0.1950	-1.30
wt.loss	2.36	1	0.1248	7.06	4	0.1328	-0.94

Approximate Estimates after Deleting Factors

	Coef	S.E.	Wald Z	P
(Intercept)	8.276947	0.936299	8.840	0.000e+00
ph.ecog	-0.546884	0.137424	-3.980	6.905e-05
age	-0.015444	0.007976	-1.936	5.283e-02
ph.karno	-0.015375	0.007394	-2.080	3.757e-02
sex*age	0.005949	0.002182	2.727	6.395e-03

Factors in Final Model

```
[1] ph.ecog age   ph.karno sex * age
```

The first argument of fastbw() receives an object fit by psm(). The *rule* argument defines stopping rule for backward elimination. The default is Akaike's information criterion (AIC). If P value is used as the stopping rule (rule="p"), the significance level for staying in a model can be modified using *sls* argument (sls =0.1 for example). The output shows that variables meal.cal, sex, pat.karno and wt.loss are eliminated from the model based on AIC. Sometimes if you want to retain a covariate in the model based on clinical judgment, the *force* argument can be employed. It passes a vector of integers specifying covariates to be retained in the model. Intercept is not counted.

Visualization of Weibull regression model

Weibull model can be used to predict outcomes of new subjects, allowing predictors to vary. In Weibull regression model, the outcome is median survival time for a given combination of covariates. We first use Predict() to calculate median survival time in log scale, then use ggplot() function to draw plots.

```
> psm.lung1<-psm(Surv(time, status)~ph.ecog+sex*age,lung,
dist='weibull')
> ggplot(Predict(psm.lung1, age=seq(20,80,by=5),
ph.ecog=c(0,1,2,3),sex=c("male","female")))
```

In the example, an interaction term sex*age is specified. We let variable age to vary between 20 to 80 years old. Both male and female, and all four levels of ph.ecog are considered. *Figure 5* shows the output of ggplot() function. The effect of age on survival time is dependent on sex. While older age is associated with shorter survival time in the male, it is associated with longer survival time in the female.

Figure 5 visualizes relationship between covariates. Occasionally, investigators may be interested in survivor and/or hazard functions of individuals with given covariate patterns. The *smoothSurv* package provides functions for this purpose. Similarly to the previous model building strategy, we first fit a model including interaction terms between sex and age.

```
> install.packages("smoothSurv")
```

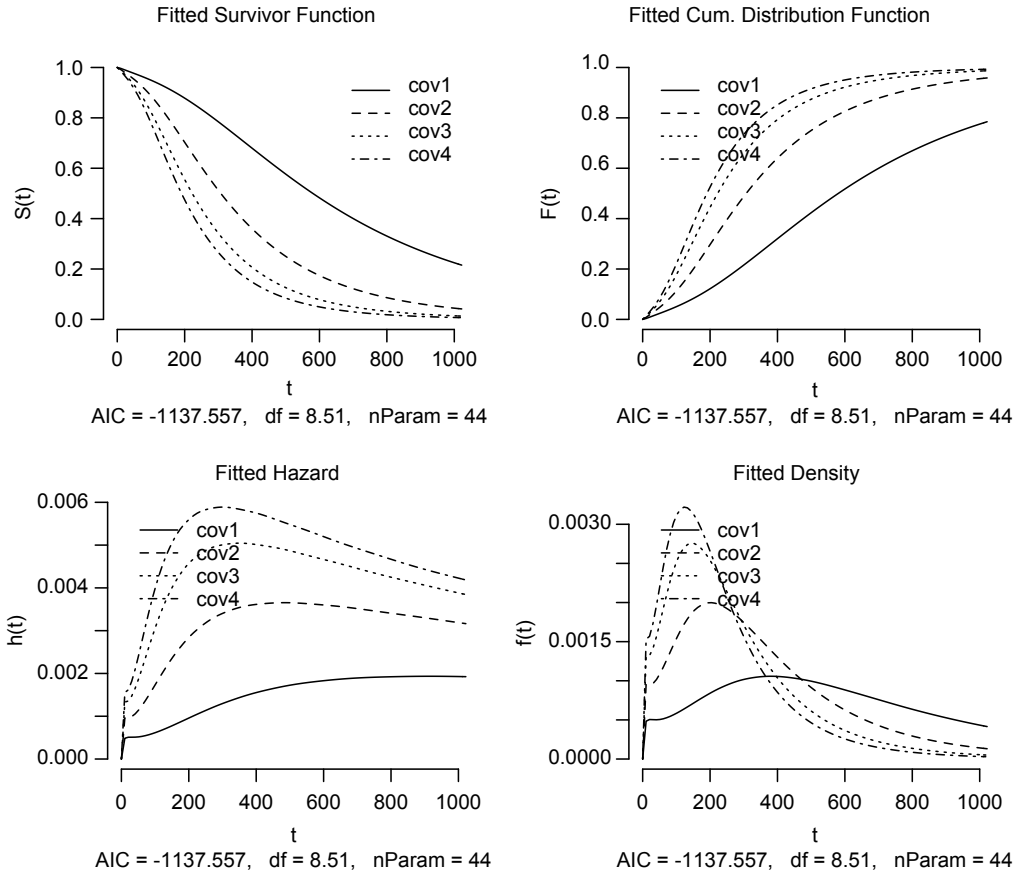


Figure 6 Survivor, cumulative distribution, hazard and density functions of four subjects. Cov1 to cov4 are indicators of four patients with given covariate patterns.

```
> library(smoothSurv)
> smooth.lung <- smoothSurvReg(Surv(time, status)~
ph.ecog+sex*age,data=lung, init.dist='weibull')
> cov<-matrix(c(0,1,2,3,1,2,2,2,20,30,40,70,20,60,80,140),ncol=
4,byrow=FALSE)
> cov
```

	[,1]	[,2]	[,3]	[,4]
[1,]	0	1	20	20
[2,]	1	2	30	60
[3,]	2	2	40	80
[4,]	3	2	70	140

A matrix object of *cov* is created representing 4 patients whose survival time is unknown and the treating physician wants to make a prediction based on the Weibull regression model. The number of columns of the matrix should be equal to the number of covariates in the model, including interaction terms.

```
> par(mfrow=c(2,2))
> survfit(smooth.lung,cov=cov)
> survfit(smooth.lung, cdf = TRUE,cov=cov)
> hazard(smooth.lung,cov=cov)
> fdensity(smooth.lung,cov=cov)
```

The output is a series of plots showing survivor, cumulative distribution, hazard and density functions (*Figure 6*). Cov1 to cov4 are indicators of four patients with given covariate patterns. Important parameters of the model are displayed at the bottom of each plot.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Hosmer DW Jr, Lemeshow S, May S. editors. Applied Survival Analysis, 2nd ed. New York: John Wiley & Sons, Inc., 2008:1.
2. Carroll KJ. On the use and utility of the Weibull model in the analysis of survival data. Control Clin Trials 2003;24:682-701.
3. Klein JP, Moeschberger ML. editors. Survival Analysis: Techniques for Censored and Truncated Data, 2nd ed. New York: Springer, 2005:1.
4. Zhang Z. Model building strategy for logistic regression: purposeful selection. Ann Transl Med 2016;4:111.

Cite this article as: Zhang Z. Parametric regression model for survival data: Weibull regression model as an example. Ann Transl Med 2016;4(24):484. doi: 10.21037/atm.2016.08.45

Semi-parametric regression model for survival data: graphical visualization with R

Zhongheng Zhang

Abstract: Cox proportional hazards model is a semi-parametric model that leaves its baseline hazard function unspecified. The rationale to use Cox proportional hazards model is that (I) the underlying form of hazard function is stringent and unrealistic, and (II) researchers are only interested in estimation of how the hazard changes with covariate (relative hazard). Cox regression model can be easily fit with coxph() function in survival package. Stratified Cox model may be used for covariate that violates the proportional hazards assumption. The relative importance of covariates in population can be examined with the rankhazard package in R. Hazard ratio curves for continuous covariates can be visualized using smoothHR package. This curve helps to better understand the effects that each continuous covariate has on the outcome. Population attributable fraction is a classic quantity in epidemiology to evaluate the impact of risk factor on the occurrence of event in the population. In survival analysis, the adjusted/unadjusted attributable fraction can be plotted against survival time to obtain attributable fraction function.

Keywords: Survival analysis; nonparametric model; Cox proportional hazards; population attributable fraction; relative hazard

Submitted May 06, 2016. Accepted for publication Jun 20, 2016.
doi: 10.21037/atm.2016.08.61
View this article at: http://dx.doi.org/10.21037/atm.2016.08.61

Introduction

A fully parametric hazard function describes the basic underlying distribution of survival time and how that distribution changes as a function of covariates. If we want to describe the circuit life span in continuous renal replacement therapy as a function of serum ionized calcium and pH value, a fully parametric hazard function is required. However, if we want to see whether circuit life span is longer under acidosis (pH<7.35) when compared with that under normal condition, a complete description of survival time is of secondary importance to a description of how serum pH value modifies the survival time (1). A full description of survival time requires assumption of underlying mathematical model, which may be unnecessarily stringent and unrealistic. Survival time model that leaves its dependence on time unspecified but has a fully parametric regression structure is called semi-parametric regression.

The general form of hazard function is written as:

$$h(t, x, \beta)=h_0 \cdot r(x, \beta) \quad [1]$$

where h_0 reflects how hazard function changes with survival time, and $r(x,\beta)$ characterizes how hazard function changes with covariates. Cox has proposed exponential function for r(), and the hazard function is written as:

$$h(t, x, \beta)=h_0 \cdot e^{x\beta} \quad [2]$$

when x changed from x_0 to x_1, the hazard ratio is

$$HR(t, x_1, x_0)=\frac{h(t, x_1, \beta)}{h(t, x_0, \beta)}=\frac{h_0(t)\cdot e^{x_1\beta}}{h_0(t)\cdot e^{x_0\beta}}=e^{\beta(x_1-x_0)} \quad [3]$$

In the literature, the model is termed Cox proportional hazard model (2). Researchers are interested in the parameter β, which is interpreted as changing rate of hazard when the covariate changed by (x_1-x_0) unit. Note that the baseline hazard function $h_0(t)$ remains unknown, that is why the model is called semi-parametric model (3).

Cox proportional hazard model

Ovarian cancer survival data (ovarian) is used to illustrate a fitting Cox proportional hazard model. The study investigated survival in a randomized trial comparing two treatments for ovarian cancer (4).

```
> library(survival)
> cph.ovarian<-coxph(Surv(futime, fustat)~rx+age , ovarian)
> summary(cph.ovarian)
Call:
coxph(formula = Surv(futime, fustat) ~ rx + age, data = ovarian)

 n= 26, number of events= 12

       coef  exp(coef)  se(coef)  z       Pr(>|z|)
rx  -0.80397  0.44755   0.63205  -1.272  0.20337
age  0.14733  1.15873   0.04615  3.193   0.00141**
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

           exp(coef)  exp(-coef)  lower .95  upper .95
rx         0.4475     2.234       0.1297     1.545
age        1.1587     0.863       1.0585     1.268

Concordance= 0.798 (se = 0.091 )
Rsquare= 0.457 (max possible= 0.932 )
Likelihood ratio test= 15.89 on 2 df, p=0.0003551
Wald test     = 13.47 on 2 df, p=0.00119
Score (logrank) test = 18.56 on 2 df, p=9.341e-05
```

The fitting of a Cox regression model appears pretty easy with only one line of R code. The function Surv() creates a survival object. Right side of "~" symbol lists covariates and they are connected with "+" operators. The last argument specifies the dataset containing covariates. Coxph() returns an object of class *coxph*, containing parameters and statistics we are interested in. the summary() function gives a general glimpse of the content of *coxph* object. There are 26 observations and 12 of them have the event of interest.

The next table shows the coefficients of the corresponding covariates. Exponentiation of the coefficient gives the hazard ratio. Age is significantly associated with hazard and the hazard increases by 1.16 times with each year increase in age. Variable age is denoted by double "*" symbols, suggesting a statistical significance with P<0.01.

The index of concordance (Concordance =0.798 in our example) is a "global" index for validating the predictive ability of a survival model. It is the fraction of pairs in the data, where the observation with the higher survival time has the higher probability of survival predicted by the model. Concordance is equivalent to the area under the ROC curve in logistic regression model. A value of 1 indicates perfect agreement, and a value of 0.5 is an agreement that the model is no better than chance. In other generalized linear models, R-square represents the proportion of variation that can be explained by the model (5). However, the R-square in Cox model also depends on the proportion of censored values. In other words, a perfectly adequate model may have a low R-square due to high proportion of censored data. Mathematical details of R-square in Cox regression model have been well described (6-10). The likelihood ratio test explores the difference between models with and without covariates. In the example, this statistic follows a Chi-square distribution with 2 degrees of freedom and thus can be used to obtain P values. Score test can be interpreted in a similar way that the model containing variables *rx* and *age* is significantly better than the null model.

Stratification

The stratified Cox proportional hazard model allows the forms of underlying hazard function to vary across levels of stratification variable. The general form of stratified Cox model is written as:

$$h(t|X, Z = j) = h_j(t) \cdot e^{x\beta} \quad [4]$$

where j is the number of levels in Z. The covariate Z is adjusted for without estimating its effect. Someone may ask the question: why not incorporate Z as a covariate instead of using it as a stratification factor? The reason is that the predictor may not satisfy proportional hazards assumption and it can be very complex to model hazard ratio for that predictor as a function of time. Furthermore, stratification allows for graphical checks of the proportional hazards assumption. A drawback of stratification is that stratifying

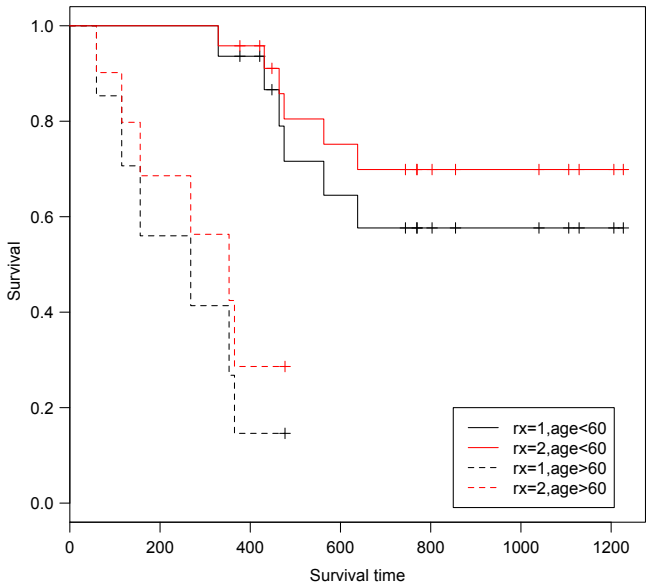


Figure 1 Comparison of survival probability of treated and untreated patients, stratified by age group.

unnecessarily (proportional hazard assumption is met) reduces estimation efficacy.

```
> cph.ovarian.str<-coxph(Surv(futime,
fustat)~rx+strata(age>60), ovarian)
> summary(cph.ovarian.str)
Call:
coxph(formula = Surv(futime, fustat) ~ rx + strata(age >
60),
 data = ovarian)

 n= 26, number of events= 12

       coef     exp(coef)  se(coef)  z        Pr(>|z|)
rx     -0.4300  0.6505     0.6003    -0.716   0.474

       exp(coef)  exp(-coef)  lower .95  upper .95
rx     0.6505     1.537       0.2006     2.11

Concordance= 0.585 (se = 0.115 )
Rsquare= 0.02 (max possible= 0.846 )
Likelihood ratio test= 0.52 on 1 df, p=0.4707
Wald test     = 0.51 on 1 df, p=0.4738
Score (logrank) test = 0.52 on 1 df, p=0.4709
```

The output of stratified Cox model is similar to the previous one. However, coefficient for *age* is not estimated because it is now used as the stratification factor. Instead, two coefficients are estimated for the variable *rx*, showing that the treatment effects are beneficial for young women and it is harmful for those aged over 60 years. One may wish to graphically examine the fitted model. Survival curves of patients with and without treatment stratified by age are depicted. In the survfit() function, a data frame with the same variable names as those that appear in the coxph formula is specified. The curve(s) produced corresponds to a cohort whose covariates equal to the values in *newdata*. If a covariate is not specified, the mean of the covariate used in the *coxph* fit is used. The following example specifies patients with and without treatment (rx=1 and 2).

```
> strata.fit<-survfit(cph.ovarian.str,
newdata=data.frame(rx=c(1,2)))
> summary(strata.fit)
Call: survfit(formula = cph.ovarian.str, newdata = data.
frame(rx = c(1,
 2)))

      age > 60=FALSE
time        n.risk      n.event     survival1   survival2
329         19          1           0.936       0.958
431         16          1           0.866       0.911
464         14          1           0.790       0.858
475         13          1           0.716       0.805
563         12          1           0.645       0.752
638         11          1           0.576       0.699

      age > 60=TRUE
time        n.risk      n.event     survival1   survival2
59          7           1           0.853       0.902
115         6           1           0.707       0.798
156         5           1           0.560       0.686
268         4           1           0.414       0.563
353         3           1           0.268       0.424
```

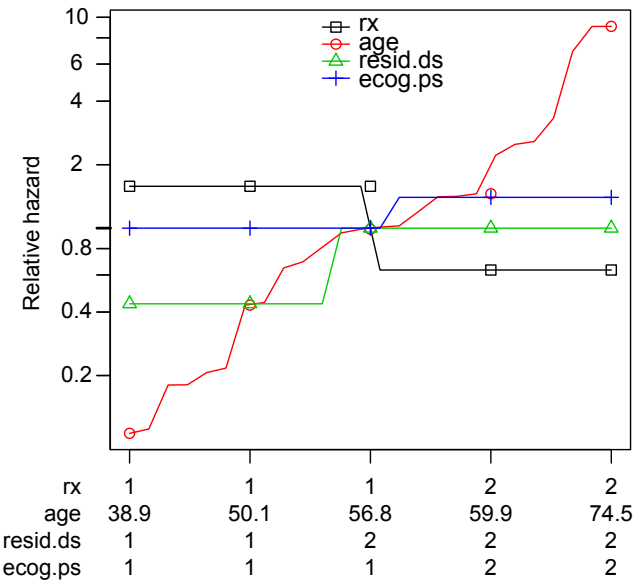



Figure 2 Rank-hazard plot of proportional hazards model where the hazard of ovarian cancer is explained by age (age), residual disease (resid.ds), treatment (rx) and ECOG performance status (ecog.ps). At the horizontal axis, the minimum, first quartile, median, third quartile and maximum values of each covariate are reported.

365	2	1	0.146	0.286

The output shows survival curves of treated and untreated patients, stratified by age group. The first column is survival time. The last two columns display estimated survival probabilities stratified by treatment groups. The result can also be visualized with the following syntax (*Figure 1*).

```
> plot(strata.fit,xlab="Survival time",ylab="Survival",lty=
c(1,1,2,2),col=c(1,2,1,2))
> legend(850,0.2,legend=c("rx=1,age<60","rx=2,age
<60","rx=1,age>60","rx=2,age>60"),col=c(1,2,1,2),lty
=c(1,1,2,2))
```

Visualization of relative importance of covariates

The interpretation of fitted Cox proportional hazards model depends on regression coefficients, significance level and prevalence of covariate patterns. Also, subject-matter audience may be interested in the importance of covariates in a study population. In other words, the importance of a covariate is determined not only by coefficient but also by its distribution in a population. Karvanen and Harrell proposed the relative-hazard plot to visualize relative importance of covariates in proportional hazards model (11). The basic idea is to rank the covariate values and plot relative hazard against ranks. The relative hazard is scaled to the reference hazard. Reference hazard is related to the median of covariates. From this definition, it is obvious that the relative hazard can vary depending on the distribution of covariate values in a population. Rank-hazard plot can be created using rankhazardplot() function. Let's first install the package and load it onto the workspace. In order to make comparisons of relative importance for all covariates, a full model including all available covariates is built.

```
>install.packages("rankhazard")
> library(rankhazard)
> cph.full<-coxph(Surv(futime, fustat)~rx+age+
resid.ds+ecog.ps , ovarian,x=TRUE)
> rankhazardplot(cph.full,data=ovarian)
Y-axis range: 0.106 9.06

Relative hazards for each covariate:
          Min.  1st Qu.  Median  3rd Qu.  Max.
rx       0.633   0.633    1.11    1.58    1.58
age      0.106   0.434    1.00    2.03    9.06
resid.ds 0.438   0.438    1.00    1.00    1.00
ecog.ps  1.000   1.000    1.00    1.40    1.40
```

The output of function rankhazardplot() shows the Y-axis range. Covariates are scaled to an interval of [0,1]. By default, the minimum, first quartile, median, third quartile and maximum values of each covariate are reported. *Figure 2* shows that treatment is the most important determinant of survival at young age, but it becomes less important for old age. Similar to the concept of population attributable fraction that the importance of a risk factor is influenced by its prevalence (12), the X-axis of relative-hazard plot displays scaled covariates for the

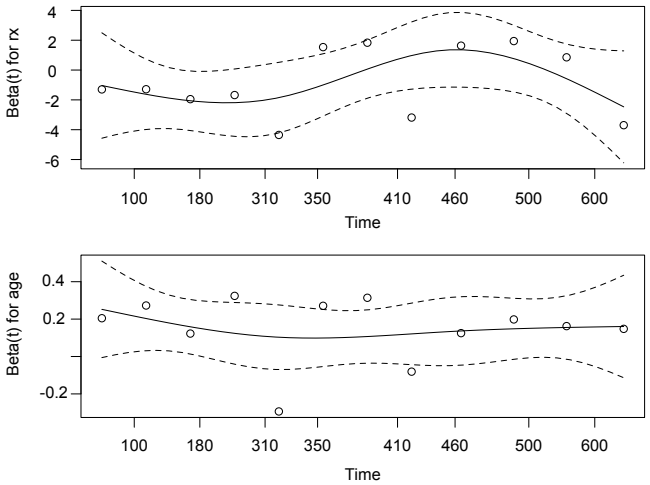


Figure 3 Plotting scaled Schoenfeld residuals against survival time to examine the proportional hazards assumption. The smoothed curve is a flat one for variable age, but there are small curvatures for the treatment (rx).

ranking of their relative importance.

Test the proportional hazards assumption

Interpretation and use of Cox proportional hazards model depends on the proportional hazards assumption. Log-hazard function of proportional hazards model takes the form

$$In[h(t, x, \beta)] = In[h_0(t)] + \boldsymbol{x \cdot \beta} \quad [5]$$

This function contains log of the baseline hazard function $In[h_0(t)]$ and linear predictor $x \cdot \beta$. x and β are highlighted in bold to represent vectors. The proportional hazard assumption dictates the baseline model as a function of time, not of the covariates. Suppose the covariate x has two levels. A plot of log-hazard over time will produce two continuous curve, one for x=0, $In[h_0(t)]$, and the other for x=1, $In[h_0(t)]+\beta$. The difference between the two curves at any time points is β. If log-hazard functions produced by different levels of a given covariate are not equidistant over time, the proportional hazards assumption is violated. Grambsch and Therneau proposed one easily performed test and an associated graph to examine the critical assumption (13).

```
> cox.zph(cph.ovarian)
          rho    chisq      P
rx     0.2072    0.518  0.472
age   -0.0918    0.113  0.736
GLOBAL     NA    0.729  0.695
```

The cox.zph() function directly performed test for proportional hazards assumption. The output is a table containing rho, Chi-square and P value. Rho is the correlation coefficient between transformed survival time and the scaled Schoenfeld residuals. The correlation coefficient follows a chi-square distribution and the statistic is present in the second column. A P value is given for each covariate. A significant P value indicates that the proportional hazards assumption is violated for that covariate. For the global test there is no correlation and NA is entered into the cell.

```
> par(mfrow=c(2,1))
> plot(cox.zph(cph.ovarian))
```

Before drawing plots, rows and columns of graphical layout should be specified. The number of panels is determined by the number of covariates. In the example, there are two covariates and thus two panels are defined. If there is only one panel in the graphical device, only one covariate can be displayed. The Y-axis of the plot is scaled Schoenfeld residuals and X-axis is survival time (*Figure 3*). The proportional hazards assumption dictates that the residual should not change with survival time. Thus, the smoothed curve should be a flat one. Due to limited number of observations in the example, the sensitivity of the test can be quite low.

Hazard ratio curves for continuous predictors

The Cox proportional hazards model assumes that the continuous predictors are in linear association with log-hazard. However, this assumption may not true in reality. Visualization of the relationship between a continuous predictor and the hazard ratio helps investigators to check for the linearity assumption. Flexible hazard ratio curve can be plotted using *smoothHR* package.

```
> install.packages("smoothHR")
```

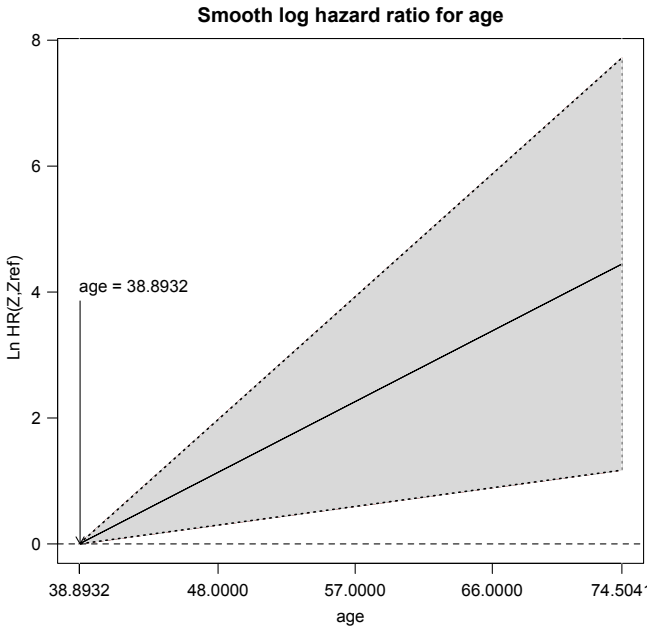



Figure 4 The hazard ratio curve shows that the age is in linear relation to the log-hazard, but the confidence interval is wide due to small number of observations.

```
> library(splines)
> library(smoothHR)
> hr.plot<- smoothHR(data=ovarian, coxfit=cph.full)
> plot(hr.plot, predictor="age", prob=0, conf.level=0.95)
```

The smoothHR() function requires data frame containing covariates and the fitted Cox regression model. The prob argument specifies the reference value of the covariate for hazard ratio. The reference value is the minimum of the hazard ratio curve for prob=0. The hazard ratio curve shows that the age is in linear relation to the log-hazard, but the confidence interval is wide due to small number of observations (*Figure 4*).

Attributable fraction function

Population attributable fraction (PAF) is defined as

$$A = \frac{P(D=1) - P(D=1|Z=0)}{P(D=1)} \quad [6]$$

where D is the binary disease status and Z is the binary exposure indicator (14). PAF is defined as "the reduction in incidence that would be achieved if the population had been entirely unexposed, compared with its current (actual) exposure pattern", and it aims to evaluate the impact of risk factor on the occurrence of event in the population (15). Unlike relative risk, PAF considers the prevalence of risk factors in the population and thus quantifies the population impact of risk factors. PAF can be extended to adjusted attributable fraction (AF) which is defined as the reduction in incidence if some risk factors are eliminated from the population while other risk factors retain their actual levels. When AF is expressed as a function of survival time, it is called attributable fraction function (16). The package *paf* in R is composed to do the task.

```
> install.packages("paf")
> library(paf)
> par(mfrow=c(1,2))
> paf.adj<-paf(Surv(time, status)~sex+age+
ph.ecog+ph.karno+pat.karno+
meal.cal+wt.loss, data=lung, cov=c('age'))
> paf.pop<-paf(Surv(time, status)~age, data=lung,
cov=c('age'))
> plot(paf.pop,ylab=
"Population attributable fraction function")
> plot(paf.adj,ylab="
Adjusted attributable fraction function")
```

The lung dataset (NCCTG Lung Cancer Data) is employed as the worked example (17). Adjusted/unadjusted AF functions of a set of covariates are computed based on the Cox model. The first argument of paf() function is a formula object for the Cox model. Covariates of interest are specified in the *cov* argument and their AF functions are to be plotted against survival time. In the example, the variable age is investigated for its adjusted and unadjusted (population) attributable fraction function. The solid lines pertain to the point estimates of attributable fraction function and the dashed lines show the 95% confidence limits (*Figure 5*). It is seen from the figure that after adjustment for covariates, the attributable fraction of age is decreased.

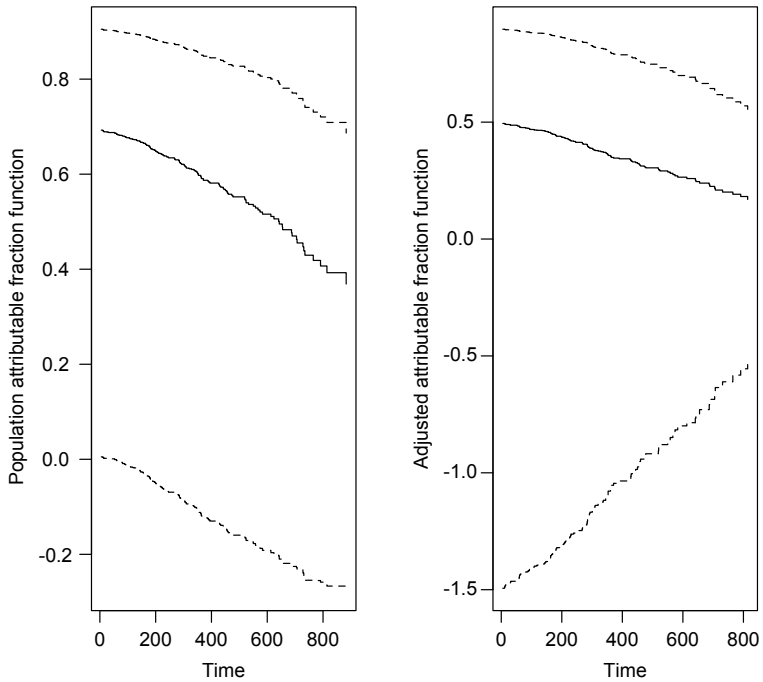


Figure 5 Estimation of attributable fraction functions for NCCTG Lung Cancer Data. The left panel shows the estimates of the population attributable fraction function: the solid curves pertain to the point estimates; and the dashed curves show the corresponding 95% confidence limits; the right panel shows the point estimate of the adjusted attributable fraction function and the corresponding 95% confidence limits.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Zhang Z, Ni H, Lu B. Variables associated with circuit life span in critically ill patients undergoing continuous renal replacement therapy: a prospective observational study. ASAIO J 2012;58:46-50.
2. Cox DR. Regression Models and Life-Tables. Journal of the Royal Statistical Society 1972;34:187-220.
3. Hosmer DW Jr, Lemeshow S, May S. Applied survival analysis: regression modeling of time to event data, second edition. Hoboken: Wiley-Interscience; 2008.
4. Edmonson JH, Fleming TR, Decker DG, et al. Different chemotherapeutic sensitivities and host factors affecting prognosis in advanced ovarian carcinoma versus minimal residual disease. Cancer Treat Rep 1979;63:241-247.
5. Mittlböck M, Schemper M. Explained variation for logistic regression. Stat Med 1996;15:1987-1997.
6. Kejžar N, Maucort-Boulch D, Stare J. A note on bias of measures of explained variation for survival data. Stat Med 2016;35:877-882.
7. Schemper M, Stare J. Explained variation in survival analysis. Stat Med 1996;15:1999-2012.
8. O'Quigley J, Xu R, Stare J. Explained randomness in proportional hazards models. Stat Med 2005;24:479-489.
9. Xu R, O'Quigley J. A measure of dependence for proportional hazards models. Journal of Nonparametric Statistics 1999;12:83-107.
10. Royston P. Explained variation for survival models. Stata Journal 2006;6:83-96.
11. Karvanen J, Harrell FE Jr. Visualizing covariates in proportional hazards model. Stat Med 2009;28:1957-1966.
12. Deubner DC, Tyroler HA, Cassel JC, et al. Attributable risk, population attributable risk, and population

attributable fraction of death associated with hypertension in a biracial population. Circulation 1975;52:901-908.
13. Grambsch PM, Therneau TM. Proportional hazards tests and diagnostics based on weighted residuals. Biometrika 1994;81:515-526.
14. Levin ML. The occurrence of lung cancer in man. Acta Unio Int Contra Cancrum 1953;9:531-541.
15. Rothman KJ, Greenland S, Lash TL. Modern Epidemiology. 3rd ed. Philadelphia: LWW; 2012.
16. Chen L, Lin DY, Zeng D. Attributable fraction functions for censored event times. Biometrika 2010;97:713-726.
17. Loprinzi CL, Laurie JA, Wieand HS, et al. Prospective evaluation of prognostic variables from patient-completed questionnaires. North Central Cancer Treatment Group. J Clin Oncol 1994;12:601-607.

Cite this article as: Zhang Z. Semi-parametric regression model for survival data: graphical visualization with R. Ann Transl Med 2016;4(23):461. doi: 10.21037/atm.2016.08.61

Survival analysis in the presence of competing risks

Zhongheng Zhang

Abstract: Survival analysis in the presence of competing risks imposes additional challenges for clinical investigators in that hazard function (the rate) has no one-to-one link to the cumulative incidence function (CIF, the risk). CIF is of particular interest and can be estimated non-parametrically with the use of cuminc() function. This function also allows for group comparison and visualization of estimated CIF. The effect of covariates on cause-specific hazard can be explored using conventional Cox proportional hazard model by treating competing events as censoring. However, the effect on hazard cannot be directly linked to the effect on CIF because there is no one-to-one correspondence between hazard and cumulative incidence. Fine-Gray model directly models the covariate effect on CIF and it reports subdistribution hazard ratio (SHR). However, SHR only provide information on the ordering of CIF curves at different levels of covariates, it has no practical interpretation as HR in the absence of competing risks. Fine-Gray model can be fit with crr() function shipped with the *cmprsk* package. Time-varying covariates are allowed in the crr() function, which is specified by *cov2* and *tf* arguments. Predictions and visualization of CIF for subjects with given covariate values are allowed for *crr* object. Alternatively, competing risk models can be fit with *riskRegression* package by employing different link functions between covariates and outcomes. The assumption of proportionality can be checked by testing statistical significance of interaction terms involving failure time. Schoenfeld residuals provide another way to check model assumption.

Keywords: Competing risk; Fine-Gary model; hazard function; cumulative incidence

Submitted Jun 20, 2016. Accepted for publication Jul 09, 2016.
doi: 10.21037/atm.2016.08.62
View this article at: http://dx.doi.org/10.21037/atm.2016.08.62

Introduction

Competing risks arise in clinical research when there are more than one possible outcome during follow up for survival data, and the occurrence of an outcome of interest can be precluded by another. The latter is called the competing risk (1-4). In clinical oncology for example, cancer-related mortality may be of primary interest, but other causes of death can prevent its occurrence and deaths caused by reasons other than cancer are typical examples of competing risks. In critical care researches, investigators may examine different strategies to maintain central venous patency in intensive care unit (ICU) (5,6). Patients in different groups were followed up for the occurrence of lumen non-patency. However, patients may die before the occurrence of lumen non-patency. Death can be considered as censoring and Cox proportional hazard model may be applied to estimate the effect of covariate on hazard. However, the effect on hazard cannot be directly liked to the cumulative incidence function (CIF), thus other modeling methods are required. This article will describe different approaches for survival analysis in the presence of competing risks. The basic ideas of each method will be introduced and detailed R code will be provided in the main text. I also highlight the interpretation of statistical output produced by R code.

Key concepts in survival analysis with and without competing risks

Survival data can be characterized by hazard function [$h(t)$] which provides a dynamic description of the instantaneous risk of failing given survival until time t. Cumulative hazard function [$H(t)$] is the $h(t)$ added over time from 0 to t. In contrast to $h(t)$, $H(t)$ has no simple probabilistic

interpretation. However, a plot of $\hat{H}(t)$ against t can provide useful information in that its local slope approximates the $h(t)$ (7). Survival function $[\hat{S}(t)]$ can be estimated non-parametrically using Kaplan-Meier estimator, and the $\hat{H}(t)$ can be estimated using *Nelson-Aalen* estimator. In the absence of competing risks, there is a one-to-one correspondence between S(t) and H(t):

$$S(t) = e^{-H(t)} \quad [1]$$

and the CIF is one minus the survival function:

$$F(t) = 1 - e^{-H(t)} \quad [2]$$

However, in the presence of competing risks, the CIF cannot be directly linked to the hazard function. Let 1 denote the event of interest and 2 denote the competing risk event. Then cumulative incidence for event of interest can be written as (8):

$$F_1(t) = \int_0^t S(s) \cdot h_1(s) \cdot ds \quad [3]$$

where $S(s) = e^{-H_1(s)-H_2(s)}$ is the survival function at time s and is determined by the both event of interest and competing event. Therefore, there is no one-to-one correspondence between cumulative incidence $[F_1(t)]$ and cause-specific hazard $[h_1(s)]$. Cumulative incidence derived from Kaplan-Meier estimator is always larger than that obtained by counting for competing risks. In Kaplan-Meier estimation, an individual is removed from the risk set when the individual experiences competing event. Within competing risk framework, the individual is an event in the calculation of overall survival probability. Therefore, the overall survival $[S(s)]$ of any event is lower when competing risks are considered. When event 2 is considered as non-informative censoring in Kaplan-Meier estimator the overall survival will be larger ($e^{-H_1(S)} \geq e^{-H_1(S)-H_2(S)}$). If hazard is considered as the rate, cumulative incidence is the risk in epidemiology terms. In competing risk analysis, individuals experiencing the competing risk event have zero probability of experiencing the event of interest. In contrast, the naïve Kaplan-Meier approach assumes that these individuals would experience the same probability of event of interest in pure theory (non-informative censoring). Thus, the latter overestimates the cumulative incidence of the event of interest.

The effect of covariates on hazard rate can be estimated using COX proportional hazard regression model. The exponentiation of the coefficient gives the hazard ratio (HR) which is the ratio of rate in epidemiology. Because hazard function has a one-to-one correspondence to the cumulative incidence, the HR also reflects the risk (cumulative incidence) of the study population. However, this relationship does not exist in the presence of competing risks. Although the cause-specific hazard regression model represents the impact of covariates on the cause-specific hazard, it does not necessarily reflect the impact on the cumulative incidence. Because clinicians or investigators may be interested in both rate and risk, the impact of covariates on both quantities can be reported side-by-side in an article (2). The Fine-Gray model was developed to link covariates to cumulative incidence. The statistics derived from Fine-Gray model are subdistribution hazard ratio (SHR) which is actually not equivalent to the HR in conventional framework (9). I will illustrate it in the following examples.

Worked example

The worked example is contained in the *riskRegression* package. A total of 205 patients with melanoma had undergone surgical operation and were followed up until the end of 1977. The dataset can be loaded and viewed in the following way.

```
> install.packages("riskRegression")
> library(riskRegression)
> data(Melanoma)
> str(Melanoma)
'data.frame':        205 obs. of 11 variables:
 $ time  : int 10 30 35 99 185 204 210 232 232 279 ...
 $ status : num 2 2 0 2 1 1 1 1 2 1 ...
 $ event : Factor w/ 3 levels "censored","death.malignant.
melanoma",..: 3 3 1 3 2 2 2 2 3 2 ...
 $ invasion: Factor w/ 3 levels "level.0","level.1",..: 2 1 2 1
3 3 3 3 2 1 ...
 $ ici   : int 2 0 2 2 2 2 2 2 3 2 ...
 $ epicel : Factor w/ 2 levels "not present",..: 2 1 1 1 2 1 2
1 1 1 ...
 $ ulcer : Factor w/ 2 levels "not present",..: 2 1 1 1 2 2 2 2
2 2 ...
 $ thick : num 6.76 0.65 1.34 2.9 12.08 ...
 $ sex   : Factor w/ 2 levels "Female","Male": 2 2 2 1 2 2 2 2
1 1 ...
 $ age   : int 76 56 41 71 52 28 77 49 60 68 ...
 $ logthick: num 1.911 -0.431 0.293 1.065 2.492 ...
```

There are 11 variables. The time (time) was measured in

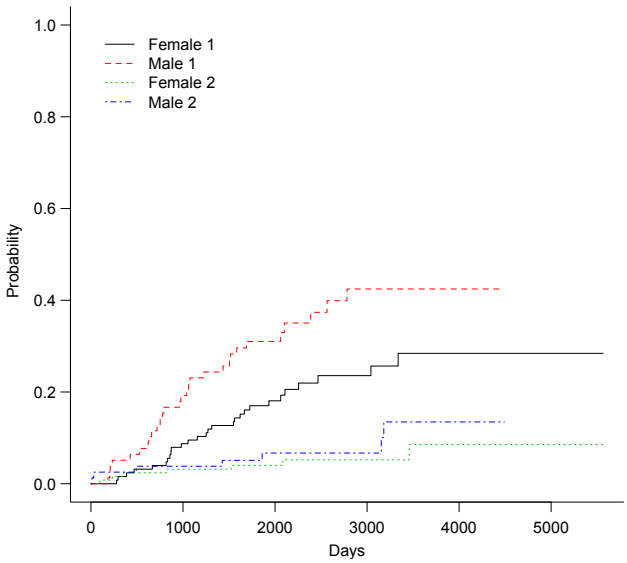


Figure 1 Non-parametric estimates of cumulative incidence functions for deaths from melanoma (status==1) and other causes (status==2). Each outcome is compared between male and female patients.

days from operation. Status assumes numeric values with 0= censored; 1= died from melanoma and 2= died from other causes. Event is in accordance with status but is converted to a factor variable with three levels. Other variables are risk factors under investigation including invasion, inflammatory cell infiltration (ici), ulcer, thick, sex, age and tumor thickness on log scale (logthick).

Non-parametric comparison of CIFs

CIFs for different causes of failure can be employed for statistical description of survival data with competing risks. This task can be done with Kaplan-Meier (KM) estimator in situations without competing risks. However, the KM method may give biased estimates because it takes the competing risk events as censored. Thus, CIFs for different causes of failure provide additional insights into the survival data at hand. The cuminc() function shipped with the *cmprsk* package can estimate the CIFs for different causes of failure and allows comparisons between groups.

```
> library(cmprsk)
> cif<-cuminc(ftime = Melanoma$time, fstatus =
Melanoma$status, group=Melanoma$sex)
> plot(cif,col=1:4,xlab="Days")
```

The arguments specifying an CIF in cuminc() is similar to that in crr() function. The first argument is a failure time variable, and the second one takes a variable with distinct code for different causes of failure. The group argument takes a variable specifying distinct groups. In the example, patients were divided into groups by sex. The estimated CIFs can be visualized with generic function plot(). *Figure 1* shows that male patients have higher risk of death from melanoma and other causes than the female. The difference appears larger for failure from melanoma versus failure from other causes. To perform formal statistical test for the difference between groups, modified χ^2 statistic can be used (10).

```
> cif$Tests
      stat      pv df
1 5.8140209 0.0158989  1
2 0.8543656 0.3553203  1
```

The first column of the output shows χ^2 statistic for between-group test, and the second column shows the respective P values. Male patients are more likely to die from melanoma than the female (P=0.016), but there is no significant difference in mortality risk from other causes for male versus female patients (P=0.36).

Cause-specific hazard regression

Cause-specific hazard regression model can be fit with Cox regression by treating failures from the cause of interest as events and failure from other causes as censored observation. The effect of covariates on cause-specific hazard can be estimated with COX proportional hazard regression. The model is fit with coxph() function in the survival package.

```
> csh<-coxph(Surv(time,status==1)~sex+age+invasion,data=
Melanoma)
> summary(csh)
Call:
coxph(formula = Surv(time, status == 1) ~ sex + age +
invasion,
  data = Melanoma)

 n= 205, number of events= 57

                coef exp(coef) se(coef)     z Pr(>|z|)
sexMale     0.663383  1.941349 0.266320 2.491 0.012741 *
age         0.009823  1.009871 0.008339 1.178 0.238840
invasionlevel.1 1.037168  2.821217 0.328241 3.160 0.001579 **
```

```
invasionlevel.21.403225  4.068300 0.3807443.685 0.000228***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
                exp(coef) exp(-coef)  lower .95  lower .95
sexMale         1.941     0.5151      1.1519     3.272
age             1.010     0.9902      0.9935     1.027
invasionlevel.12.821      0.3545      1.4826     5.368
invasionlevel.24.068      0.2458      1.9290     8.580
Concordance= 0.7 (se = 0.04 )
Rsquare= 0.122 (max possible= 0.937 )
Likelihood ratio test= 26.7 on 4 df, p=2.288e-05
Wald test     = 24.39 on 4 df, p=6.68e-05
Score (logrank) test = 26.85 on 4 df, p=2.13e-05
```

The first argument of the coxph() function takes an object of class *Surv*, where the "status==1" indicates that only status value of 1 is considered as event and other values are considered as censored. The summary output shows the coefficients and corresponding HR. The last five lines display statistics for the fitness of the model. Detailed interpretation of these statistics can be found in previous sections.

Alternatively, the task can be performed using by CSC() function contained in the *riskRegression* package.

```
> install.packages("riskRegression")
> library(prodlim)
> library(riskRegression)
> CSH<-CSC(Hist(time,status)~sex+age+invasion,
data=Melanoma)
> summary(CSH)
```

The summary output is quite similar to that produced by the coxph() function except that CSC() function automatically produces cause-specific hazard models for both types of events (cause 1 and 2). With the fitted regression model, one can predict individual risk with given covariates. For example, I want to predict the risk of a male patient aged 50 years old and has invasion level 2.

```
> library(pec)
> pec:::predictEventProb(CSH,cause=1,
newdata=data.frame(age=50, invasion=factor("level.2",
levels=levels(Melanoma$invasion)), sex=factor("Male",
levels=levels(Melanoma$sex))),time
=c(1000,2000,3000))
          [,1]      [,2]      [,3]
[1,] 0.3146673 0.5288262 0.6722392
```

The output shows that the cumulative incidences of death from melanoma at time points of 1,000, 2,000, 3,000 days are 0.31, 0.53 and 0.67, respectively.

Subdistribution hazards (SHs) model

SHs model is also known as Fine-Gray model. It is a Cox proportional regression model but the cumulative incidence is associated with SHs. The motivation for Fine-Gray model is that the effect of a covariate on cause-specific hazard function may be quite different from that on CIF. In other words, a covariate may have strong influence on cause-specific hazard function, but have no effect on CIF (9). The difference between cause-specific hazard and subdistribution is that the competing risk events are treated differently. The former considers competing risk events as non-informative censoring, whereas the latter takes into account the informative censoring nature of the competing risk events (1).

The Fine-Gray model can be fit using FGR() function shipped with *riskRegression* package. This function calls another function crr() from the *cmprsk* package.

```
> SH <- FGR(Hist(time,status)~sex+age+invasion,
data=Melanoma)
Argument cause missing. Analyse cause: 1
> SH

Right-censored response of a competing.risks model

No.Observations: 205

Pattern:

Cause          event     right.censored
1              57        0
2              14        0
unknown        0         134

Fine-Gray model: analysis of cause 1

Competing Risks Regression

Call:
FGR(formula = Hist(time, status) ~ sex + age + invasion, data
= Melanoma,
  cause = 1)

                coef      exp(coef) se(coef) z      p-value
```

```
                  sexMale 0.62762  1.87      0.27170 2.310  0.0210
                  age     0.00565  1.01      0.00913 0.619  0.5400
                  invasionlevel.1 1.04909  2.86      0.34040 3.082  0.0021
                  invasionlevel.2 1.37802  3.97      0.40137 3.433  0.0006

                                   exp(coef) exp(-coef) 2.5%  97.5%
                  sexMale          1.87      0.534     1.100 3.19
                  age              1.01      0.994     0.988 1.02
                  invasionlevel.1  2.86      0.350     1.465 5.56
                  invasionlevel.2  3.97      0.252     1.806 8.71

                  Num. cases = 205
                  Pseudo Log-likelihood = -274
                  Pseudo likelihood ratio test = 24.6 on 4 df,

                  Convergence: TRUE
```

As you can see, the estimated coefficient for cause 1 deviates a little from that obtained from cause-specific hazard model (HR: 1.87 *vs.* 1.94), reflecting different assumptions for the competing risks. The numerical values derived from Fine-Gray model have no direct interpretation, but it reflects the ordering of cumulative incidence curves (7,9). The cause-specific hazard is the rate of cause 1 failure per time unit for patients who are still alive. However, the cause 1 SH is the rate of cause 1 failure per time unit for patients who are either alive or have already failed from cause 2. In other words, patients who fail from other causes are still in the risk set (8).

The Fine-Gray model can be fit with the crr() function in the *cmprsk* package. The function arguments are different from that in FGR() function. Although the use of model formula is not supported, the model.matrix function can be used to generate suitable matrices of covariates from factors.

```
> cov<-model.matrix(~sex+age+invasion,
data=Melanoma)[,-1]
> crr.model<-crr(Melanoma$time,Melanoma$status,
cov1=cov)
```

Model prediction

The fitted Fine-Gray model can be used to predict new observations with given combinations of covariates. In the following example, a new dataset containing three patients are given. Covariates of age, sex and invasion levels are defined for them.

```
> newdata<-data.frame(sex=factor(c("Male","Male",
"Female"),levels=levels(Melanoma$sex)),age=c(50,31,29),
invasion=factor(c("level.2","level.1","level.2"),
levels=levels(Melanoma$invasion)))
> newdata
  sex    age invasion
1 Male   50  level.2
2 Male   31  level.1
3 Female 29  level.2
```

Characteristics of these three patients are displayed in the above output. The data frame need to be transformed to matrix and factor variables be transformed to dummy variables. The predict() function applied to *crr* object requires the columns of *cov* be in line with that in the original call to crr() function. Because both sex and invasion are factor variables, they need to be transformed to dummy variables with the model.matrix() function. Alternatively, a custom-made function named factor2ind() written by Scrucca and colleagues can be useful (11).

```
> dummy.new<-model.matrix(~sex+age+invasion,
data=newdata)[,-1]
> dummy.new
  sexMale age invasionlevel.1 invasionlevel.2
1       1  50               0               1
2       1  31               1               0
3       0  29               0               1
```

The above output is exactly what we want. Variable sex was coded 1 for male and 0 for female. Invasion was transformed to two 0/1 variables. Age is a continuous variable and remains unchanged.

```
> pred<-predict(crr.model,dummy.new)
> plot(pred,lty=1:3,col=1:3,xlab="Failure time (days)",
ylab="Cumulative incidence function")
> legend("topleft",c("Male,age=50,invasion2",
"Male,age=31,invasion1","Female,age=29,invasion2"),
lty=1:3,col=1:3)
```

An object of *crr* class is passed to the predict() function, followed by a matrix containing covariate combinations. The predict() function returns a matrix (not shown) with the unique cause =1 failure times in the first column, and the other columns giving the estimated subdistribution function corresponding to the covariate combinations at

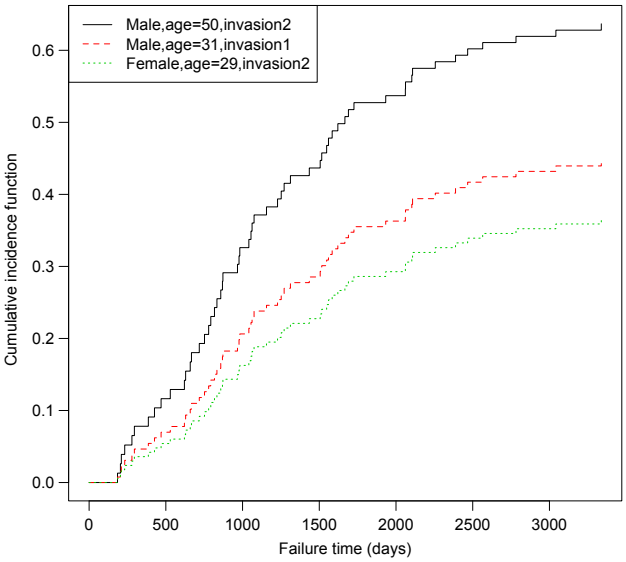


Figure 2 Parametric estimates of cumulative incidence functions for three patients with given covariate values.

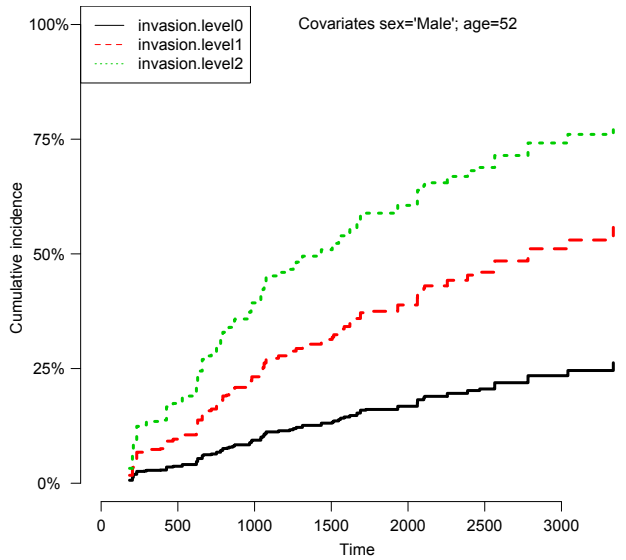


Figure 3 Cumulative incidence functions at different invasion levels, setting age to 52 years and sex to male. There is no evidence of violation to the proportionality assumption for the variable invasion.

each failure time. The generic function plot() can be applied to draw CIF for each observation (*Figure 2*).

The prediction and plotting are more convenient using functions in the *riskRegression* package. The function riskRegression() provides a variety of link functions for survival regression model in the presence of competing risks (12).

```
> reg<-riskRegression(Hist(time, status) ~ sex + age +
invasion, data = Melanoma, cause = 1,link="prop")
> plot(reg,newdata=newdata)
```

The above graphical output gives CIF for patients with characteristics specified in the *newdata*. The link argument controls the link function to be used: "prop" for the Fine-Gray regression model, "relative" for the absolute risk regression model, and "logistic" for the logistic risk regression model.

Model diagnostic

An important assumption of Cox regression model is the proportionality, which assumes the subdistribution with covariates z is a constant shift on the complementary log-log scale from a baseline subdistribution function. The curves will not cross with each other. Model checking may initially be performed by graphical examination of CIFs.

```
> checkdata<-data.frame(sex=factor(c("Male","Male",
"Male"),levels=levels(Melanoma$sex)),age=c(52,52,52),
invasion=factor(c("level.0","level.1",
"level.2"), levels=levels(Melanoma$invasion)))
> plot(reg,newdata=checkdata,lty=1:3,col=1:3)
> text(2000,1,"Covariates sex='Male'; age=52")
> legend("topleft",c("invasion.level0",
"invasion.level1","invasion.level2"),lty=1:3,col=1:3)
```

Figure 3 shows the CIFs at different invasion levels, setting *age* to 52 years and *sex* to male. There is no evidence of violation to the proportionality assumption for the variable invasion. Another method to check proportional assumption is to include time dependent covariate in the regression model.

```
> crr.time<-crr(Melanoma$time,Melanoma$status,cov1=cov,
cov2=cov[,1],tf=function(t) t)
> summary(crr.time)
Competing Risks Regression

Call:
crr(ftime = Melanoma$time, fstatus = Melanoma$status, cov1
= cov,
```

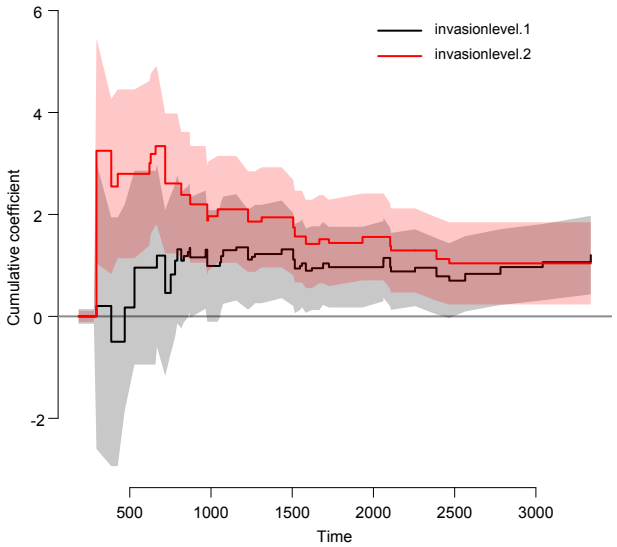



Figure 4 Time-dependent effects in the Fine-Gray regression model for invasion. The nonparametric estimates are shown with 95% pointwise confidence intervals.

```
cov2 = cov[, 1], tf = function(t) t)

                    coef exp(coef) se(coef) z     p-value
sexMale         1.144721 3.14      0.514436 2.225 0.02600
age             0.005667 1.01      0.009013 0.629 0.53000
invasionlevel.1 1.049050 2.85      0.338726 3.097 0.00200
invasionlevel.2 1.375556 3.96      0.397515 3.460 0.00054
cov[, 1]1*tf1  -0.000413 1.00      0.000349 -1.182 0.24000

                exp(coef) exp(-coef) 2.5%  97.5%
sexMale         3.14      0.318      1.146 8.61
age             1.01      0.994      0.988 1.02
invasionlevel.1 2.85      0.350      1.470 5.55
invasionlevel.2 3.96      0.253      1.816 8.63
cov[, 1]1*tf1   1.00      1.000      0.999 1.00

Num. cases = 205
Pseudo Log-likelihood = -273
Pseudo likelihood ratio test = 25.9 on 5 df,
```

The argument cov2 takes a matrix of covariates that will be multiplied by time. The functions of time are specified in the *tf* argument. The function takes a vector of times as an argument and returns a matrix. The jth column of the time matrix will be multiplied by the jth column of cov2. For example, a model of the form $\beta_0 + \beta_1 x_1 + \beta_2 x_1 t + \beta_3 x_1 t^2$ can be specified in crr() function by (cov1 = x_1, cov2=cbind(x_1,x_1), tf=function(t) cbind(t,t^2)). In the summary output of the Fine-Gray model with time-varying covariate, the last term shows no statistical significance (P=0.24), indicating that the effect of sex is time-constant. The *riskRegression* provides a simple resolution to model time-varying covariate.

```
> reg.time<-riskRegression(Hist(time, status) ~ sex + age +
strata(invasion), data = Melanoma, cause = 1,link="prop")
> plotEffects(reg.time,formula=~invasion)
```

Figure 4 shows the time-dependent effects in Fine-Gray regression model for mortality. The curve and corresponding 95% confidence interval are drawn with non-parametric method. It appears that coefficients for level 2 *vs.* 1 are larger during time period from 0 to 1,000 than that in other times, indicating some slight time interactions (12). However, formal statistical test is not allowed in this setting.

The third method for model checking employs Schoenfeld residuals.

```
> par(mfrow=c(2,2))
> for(j in 1:ncol(crr.model$res)) {
    scatter.smooth(crr.model$uft, crr.model$res[,j],
    main =names(crr.model$coef)[j],
    xlab = "Failure time",
    ylab ="Schoenfeld residuals")
    }
```

I plot the Schoenfeld residuals against failure time for each covariate. If the proportional assumption is true, the residual should have a constant mean across time. A scatterplot smoother is added for each covariate to check the assumption (*Figure 5*). It appears that the invasion level 1 has non-constant residuals across time, indicating a potential violation to the proportional assumption. To formally check the assumption of invasion level 1, we can add an interaction term with time.

```
> crr.time2<-crr(Melanoma$time,Melanoma$status,
cov1=cov,cov2=cov[,3],tf=function(t) t)
> crr.time3<-crr(Melanoma$time,Melanoma$status,
cov1=cov,cov2=cbind(cov[,3], cov[,3]),
tf=function(t) cbind(t,t^2),)
```

The model includes an interaction term with linear time function, which shows that the interaction term (time*invasion level1) is statistically significant (P=0.033).

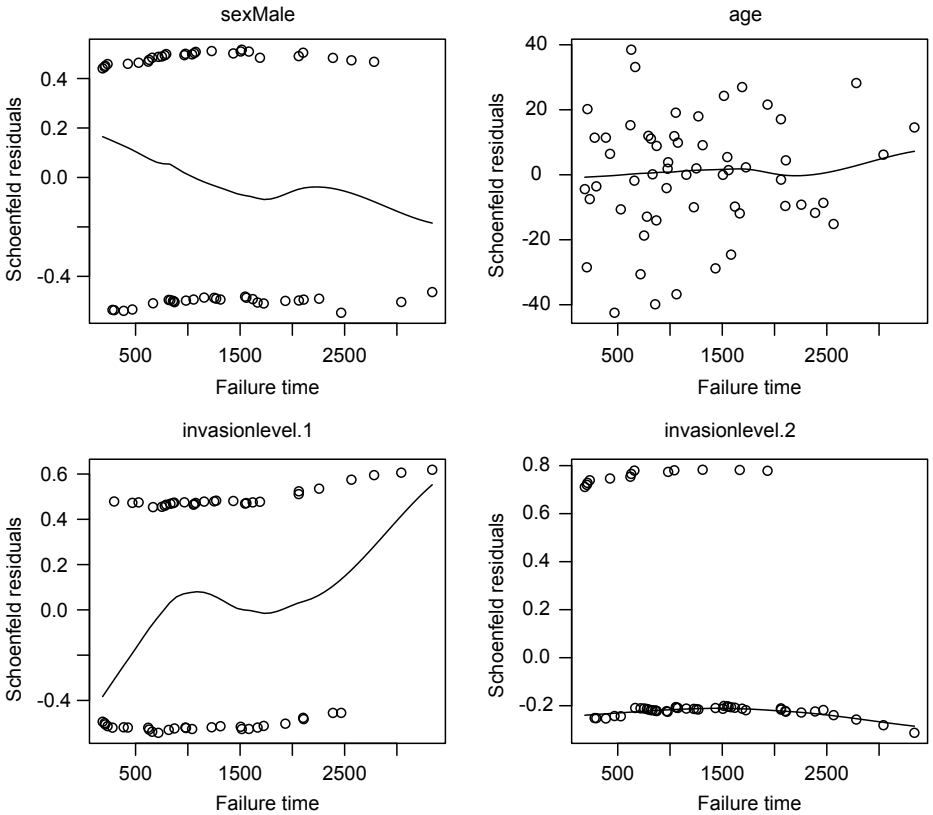


Figure 5 Schoenfeld residuals against failure time for each covariate. It is noted that the residuals follows a non-constant distribution across failure times, indicating a potential violation to the proportional assumption.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Satagopan JM, Ben-Porat L, Berwick M, et al. A note on competing risks in survival data analysis. Br J Cancer 2004;91:1229-1235.
2. Latouche A, Allignol A, Beyersmann J, et al. A competing risks analysis should report results on all cause-specific hazards and cumulative incidence functions. J Clin Epidemiol 2013;66:648-653.
3. Bakoyannis G, Touloumi G. Practical methods for competing risks data: a review. Stat Methods Med Res 2012;21:257-272.
4. Haller B, Schmidt G, Ulm K. Applying competing risks regression models: an overview. Lifetime Data Anal 2013;19:33-58.
5. Schallom ME, Prentice D, Sona C, et al. Heparin or 0.9% sodium chloride to maintain central venous catheter patency: a randomized trial. Crit Care Med 2012;40:1820-1826.
6. Zhang Z, Pan L, Ni H. Lumen nonpatency in the presence of competing risks. Crit Care Med 2012;40:3108-author reply 3108-3109.
7. Andersen PK, Keiding N. Interpretability and importance of functionals in competing risks and multistate models. Stat Med 2012;31:1074-1088.
8. Andersen PK, Geskus RB, de Witte T, et al. Competing risks in epidemiology: possibilities and pitfalls. Int J Epidemiol 2012;41:861-870.
9. Fine JP, Gray RJ. A proportional hazards model for the subdistribution of a competing risk. J Am Stat Assoc 1999;94:496-509.

10. Gray RJ. A class of K-sample tests for comparing the cumulative incidence of a competing risk. Ann Stat 1988;16:1141-1154.
11. Scrucca L, Santucci A, Aversa F. Regression modeling of competing risk using R: an in depth guide for clinicians. Bone Marrow Transplant 2010;45:1388-1395.
12. Gerds TA, Scheike TH, Andersen PK. Absolute risk regression for competing risks: interpretation, link functions, and prediction. Stat Med 2012;31:3921-3930.

Cite this article as: Zhang Z. Survival analysis in the presence of competing risks. Ann Transl Med 2017;5(3):47. doi: 10.21037/atm.2016.08.62

Introduction to machine learning: k-nearest neighbors

Zhongheng Zhang

Abstract: Machine learning techniques have been widely used in many scientific fields, but its use in medical literature is limited partly because of technical difficulties. k-nearest neighbors (kNN) is a simple method of machine learning. The article introduces some basic ideas underlying the kNN algorithm, and then focuses on how to perform kNN modeling with R. The dataset should be prepared before running the knn() function in R. After prediction of the outcome with kNN algorithm, the diagnostic performance of the model should be checked. Average accuracy is the most widely used statistic to reflect the kNN algorithm. Factors such as the k value, distance calculation and the choice of appropriate predictors all have significant impact on the model performance.

Keywords: Machine learning; R; k-nearest neighbors (kNN); class; average accuracy; kappa

Submitted Jan 25, 2016. Accepted for publication Feb 18, 2016.
doi: 10.21037/atm.2016.03.37
View this article at: http://dx.doi.org/10.21037/atm.2016.03.37

Introduction to k-nearest neighbor (kNN)

kNN classifier is to classify unlabeled observations by assigning them to the class of the most similar labeled examples. Characteristics of observations are collected for both training and test dataset. For example, fruit, vegetable and grain can be distinguished by their crunchiness and sweetness (*Figure 1*). For the purpose of displaying them on a two-dimension plot, only two characteristics are employed. In reality, there can be any number of predictors, and the example can be extended to incorporate any number of characteristics. In general, fruits are sweeter than vegetables. Grains are neither crunchy nor sweet. Our work is to determine which category does the sweet potato belong to. In this example we choose four nearest kinds of food, they are apple, green bean, lettuce, and corn. Because the vegetable wins the most votes, sweet potato is assigned to the class of vegetable. You can see that the key concept of kNN is easy to understand.

There are two important concepts in the above example. One is the method to calculate the distance between sweet potato and other kinds of food. By default, the knn() function employs Euclidean distance which can be calculated with the following equation (1,2).

$$D(p,q)=\sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\cdots+(p_n-q_n)^2} \quad [1]$$

where p and q are subjects to be compared with n characteristics. There are also other methods to calculate distance such as Manhattan distance (3,4).

Another concept is the parameter k which decides how many neighbors will be chosen for the kNN algorithm. The appropriate choice of k has significant impact on the diagnostic performance of kNN algorithm. A large k reduces the impact of variance caused by random error, but runs the risk of ignoring small but important patterns. The key to choose an appropriate k value is to strike a balance between overfitting and underfitting (5). Some authors suggest to set k equal to the square root of the number of observations in the training dataset (6).

Working example

For illustration of how kNN works, I create a dataset that has no practical meaning.

```
> set.seed(seed=888)
> df1 <- data.frame(x1=runif(200,0,100),
x2=runif(200,0,100))
> df1 <- transform(df1, y=1+ifelse(100 - x1 - x2
+ rnorm(200,sd=10) < 0, 0, ifelse(100 - 2*x2 +
rnorm(200,sd=10) < 0, 1, 2)))
> df1$y<-as.factor(df1$y)
```

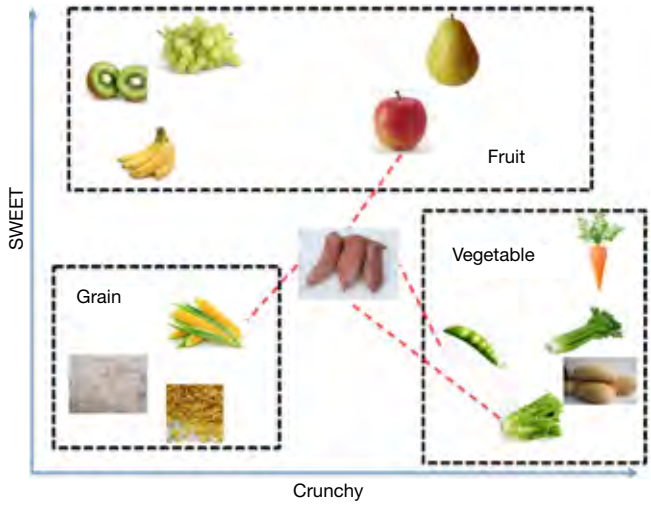


Figure 1 Illustration of how k-nearest neighbors' algorithm works.

```
> df1$tag<-c(rep("train",150),rep("test",50))
```

The first line sets a seed to make the output reproducible. The second line creates a data frame named df1, and it contains two variables x1 and x2. Then I add another categorical variable y, and it has three levels. However, the variable y is numeric and I convert it into a factor by using as.factor() function. A tag variable is added to split the dataset into training and test sets. Next we can examine the dataset by graphical presentation.

```
> library(ggplot2)
> qplot(x1,x2, data=df1, colour=y,shape=tag)
```

As you can see in *Figure 2*, different categories are denoted by red, green and blue colors. The whole dataset is split in 150:50 ratio for training and test datasets. Dots represent test data and triangles are training data.

Performing kNN algorithm with R

The R package class contains very useful function for the purpose of kNN machine learning algorithm (7). Firstly one needs to install and load the class package to the working space.

```
> install.packages("class")
> library(class)
```

Then we divide the original dataset into the training and test datasets. Note that the training and test data frames contain the predictor variable only. The response variable is stored in an other vector.

```
> train<-df1[1:150,1:2]
> train.label<-df1[1:150,3]
> test<-df1[151:200,1:2]
> test.label<-df1[151:200,3]
```

Up to now, datasets are well prepared for the kNN model building. Because kNN is a non-parametric algorithm, we will not obtain parameters for the model. The kNN() function returns a vector containing factor of classifications of test set. In the following code, I arbitrary choose a k value of 6. The results are stored in the vector *pred*.

```
> pred<-knn(train=train,test=test,cl=train.label,k=6)
```

The results can be viewed by using CrossTable() function in the *gmodels* package.

```
> install.packages("gmodels")
> library(gmodels)
> CrossTable(x = test.label, y = pred,prop.chisq=FALSE)
```

```
 Cell Contents
                                 N
                 N / Row Total
                 N / Col Total
               N / Table Total

Total Observations in Table:  50

                 Pred
test.label |         1 |         2 |         3 | Row Total |
-----------|-----------|-----------|-----------|-----------|
         1 |        29 |         0 |         0 |        29 |
           |     1.000 |     0.000 |     0.000 |     0.580 |
           |     0.935 |     0.000 |     0.000 |           |
           |     0.580 |     0.000 |     0.000 |           |
-----------|-----------|-----------|-----------|-----------|
         2 |         2 |         6 |         2 |        10 |
           |     0.200 |     0.600 |     0.200 |     0.200 |
           |     0.065 |     0.857 |     0.167 |           |
           |     0.040 |     0.120 |     0.040 |           |
```

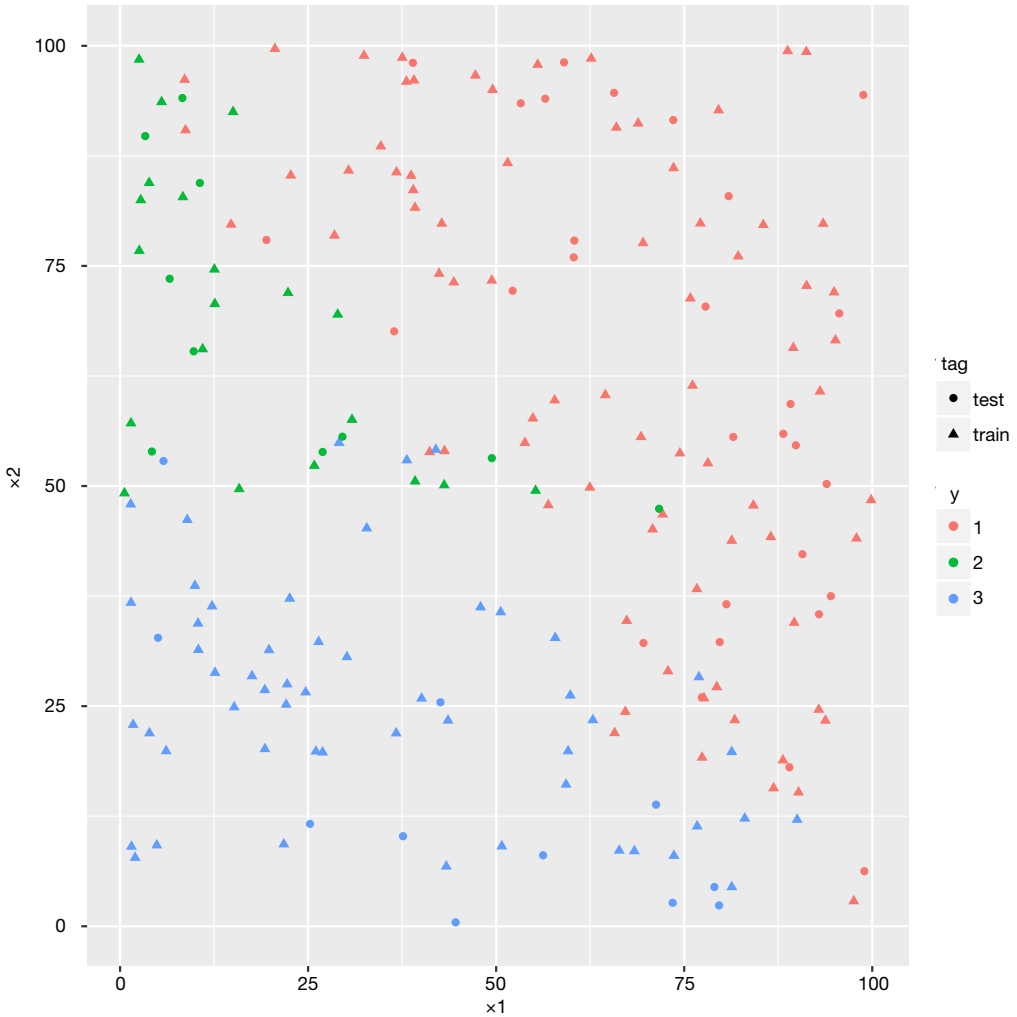



Figure 2 Visual presentation of simulated working example. The class 1, 2 and 3 are denoted by red, green and blue colors, respectively. Dots represent test data and triangles are training data.

3	0	1	10	11
	0.000	0.091	0.909	0.220
	0.000	0.143	0.833	
	0.000	0.020	0.200	
Column Total	31	7	12	50
	0.620	0.140	0.240	

Diagnostic performance of the model

The kNN algorithm assigns a category to observations in the test dataset by comparing them to the observations in the training dataset. Because we know the actual category of observations in the test dataset, the performance of the kNN model can be evaluated. One of the most commonly used parameters is the average accuracy that is defined by the following equation (8):

$$Average\ Accuracy = \sum_{i=1}^{l} \frac{TP_i + TN_i}{TP_i + FN_i + FP_i + TN_i} / l \quad [2]$$

where TP is the true positive, TN is the true negative, FP is the false positive and FN is the false negative. The subscript i indicates category, and l refers to the total category.

```
> table<-CrossTable(x = test.label,
y = pred,prop.chisq=TRUE)
```



```
> tp1<-table$t[1,1]
> tp2<-table$t[2,2]
> tp3<-table$t[3,3]
> tn1<-table$t[2,2]+table$t[2,3]+table$t[3,2]+
table$t[3,3]
> tn2<-table$t[1,1]+table$t[1,3]+table$t[3,1]+
table$t[3,3]
> tn3<-table$t[1,1]+table$t[1,2]+table$t[2,1]+
table$t[2,2]
> fn1<-table$t[1,2]+table$t[1,3]
> fn2<-table$t[2,1]+table$t[2,3]
> fn3<-table$t[3,1]+table$t[3,2]
> fp1<-table$t[2,1]+table$t[3,1]
> fp2<-table$t[1,2]+table$t[3,2]
> fp3<-table$t[1,3]+table$t[2,3]
> accuracy<-(((tp1+tn1)/
(tp1+fn1+fp1+tn1))+((tp2+tn2)/
(tp2+fn2+fp2+tn2))+((tp3+tn3)/(tp3+fn3+fp3+tn3)))/3
> accuracy
[1] 0.9333333
```

The CrossTable() function returns the result of cross tabulation of predicted and observed classifications. The number in each cell can be used for the calculation of four basic parameters true positive (TP), true negative (TN), false negative (FN) and false positive (FP). The process repeated for each category. Finally, the accuracy is 0.93.

Sensitivity and specificity

Sensitivity is a measure of the proportion of positives that are correctly identify positive observations. Specificity is a measure of the proportion of negatives that are truly negative. They are commonly used to measure the diagnostic performance of a test (9). In evaluation of a prediction model, they can be used to reflect the performance of the model. Imaging a perfectly fitted model that can predict outcomes with 100% accuracy, both sensitivity and specificity are 100%. In multiclass situation as in our example, sensitivity and specificity are calculated separately for each class. The equations are as follows.

$$Sen_i = TP_i / \left(TP_i + FN_i\right) \quad [3]$$

$$Sp_i = TN_i / \left(TN_i + FP_i\right) \quad [4]$$

where TP is the true positive, TN is the true negative, FP is the false positive and FN is the false negative. The subscript i indicates category.

```
> sen1<-tp1/(tp1+fn1)
> sp1<-tn1/(tn1+fp1)
> sen1
[1] 1
> sp1
[1] 0.9047619
```

Multiclass area under the curve (AUC)

A receiver operating characteristic (ROC) curve measures the performance of a classifier to correctly identify positives and negatives. The AUC ranges between 0.5 and 1. An AUC of 0.5 indicates a random classifier that it has no value. Multiclass AUC is well described by Hand and coworkers (10). The multiclass.roc() function in pROC package is able to do the task.

```
> install.packages("pROC")
> library(pROC)
> multiclass.roc(response=test.label,
predictor=as.ordered(pred))

Call:
multiclass.roc.default(response = test.label, predictor =
as.ordered(pred))

Data: as.ordered(pred) with 3 levels of test.label: 1, 2, 3.
Multi-class area under the curve: 0.9212
```

As you can see from the output of the command, the multi-class AUC is 0.9212.

Kappa statistic

Kappa statistic is a measurement of the agreement for categorical items (11). Its typical use is in the assessment of the inter-rater agreement. Here kappa can be used to assess the performance of kNN algorithm. Kappa can be formally expressed by the following equation:

$$kappa = \frac{P\left(A\right) - P(E)}{1 - P(E)} \quad [5]$$

where P(A) is the relative observed agreement among raters,

and P(E) is the proportion of agreement expected between the classifier and the ground truth by chance. In our example the tabulation of predicted and observed classes are as follows:

```
> table<-table(test.label,pred)
> table
            pred
test.label  1  2  3
1           29 0  0
2           2  6  2
3           0  1  10
```

The relative observed agreement can be calculated as

$$P(A) = (29+6+10)/50 = 0.9 \quad [6]$$

the kNN algorithm predicts 1, 2 and 3 for 31, 7, and 12 times. Thus, the probability that kNN says for 1, 2 and 3 are 0.62, 0.14 and 0.24, respectively. Similarly, the probabilities that 1, 2 and 3 are observed are 0.58, 0.2 and 0.22, respectively. Then, the probability that both classifier say 1, 2 and 3 are 0.62×0.58=0.3596, 0.14×0.2=0.028 and 0.24×0.22=0.0528. The overall probability of random agreement is:

$$P(E) = 0.3596 + 0.028 + 0.0528 = 0.4404 \quad [7]$$

and the kappa statistic is:

$$kappa = \frac{P(A)-P(E)}{1-P(E)} = \frac{0.9-0.4404}{1-0.4404} \approx 0.82 \quad [8]$$

Fortunately, the calculation can be performed by the cohen.kappa() function in the psych package. I present the calculation process here for readers to better understand the concept of kappa.

```
> install.packages("psych")
> library(psych)
> cohen.kappa(x=cbind(test.label,pred))
Call: cohen.the kappa1(x = x, w = w, n.obs = n.obs, alpha
= alpha)

Cohen Kappa and Weighted Kappa correlation coeffi-
cients and confidence boundaries
                    lower  estimate  upper
unweighted kappa    0.68   0.82      0.96
weighted kappa      0.87   0.93      0.99
Number of subjects = 50
```

Tuning k for kNN

The parameter k is important in kNN algorithm. In the last section I would like to tune the k values and examine the change of the diagnostic accuracy of the kNN model. Custom-made R function is helpful in simplifying the calculation process. Here I write a function named "accuracyCal" to calculate a series of average accuracies. There is only one argument for the function. That is the maximum number of k you would like to examine. There is a for loop within the function which calculates accuracies repeatedly from one to N. When you run the function, the results may not exactly the same for each time. That is because the knn() function breaks ties at random. To explain, if we have 4 nearest neighbors and two are classified as A and 2 are classified as B, then A and B are randomly chosen as the predicted result.

```
> accuracyCal<-function(N) {
      accuracy<-1
      for (x in 1:N) {
          pred<-knn(train=train,test=test,
cl=train.label,k=x)
          table<- table(test.label,pred)
          tp1<-table[1,1]
          tp2<-table[2,2]
          tp3<-table[3,3]
          tn1<-table[2,2]+table[2,3]+table[3,2]+table[3,3]
          tn2<-table[1,1]+table[1,3]+table[3,1]+table[3,3]
          tn3<-table[1,1]+table[1,2]+table[2,1]+table[2,2]
          fn1<-table[1,2]+table[1,3]
          fn2<-table[2,1]+table[2,3]
          fn3<-table[3,1]+table[3,2]
          fp1<-table[2,1]+table[3,1]
          fp2<-table[1,2]+table[3,2]
          fp3<-table[1,3]+table[2,3]
          accuracy<-c(accuracy, (((tp1+tn1)/
(tp1+fn1+fp1+tn1))+((tp2+tn2)/
(tp2+fn2+fp2+tn2))+((tp3+tn3)/(tp3+fn3+fp3+tn3)))/3)
```

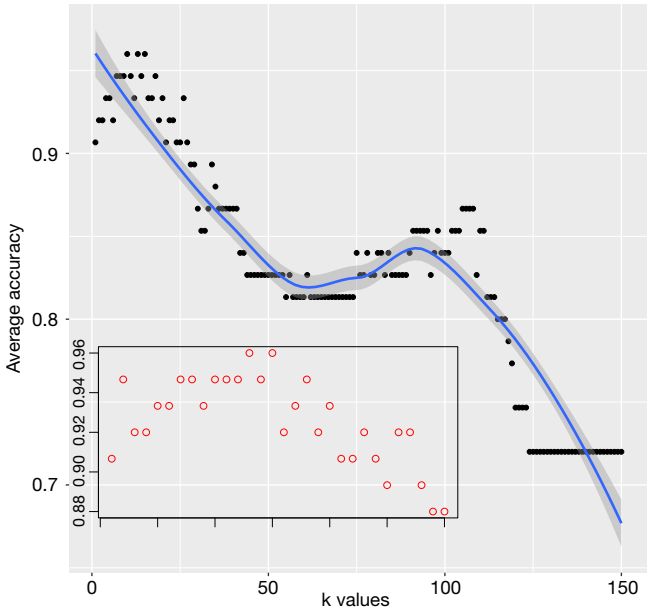



Figure 3 Graphical presentation of average accuracy with different k values. The inset zooms in at k range between 0 and 30.

```
        }
return(accuracy[-1])
}
```

The following code creates a visual display of the results. An inset plot is created to better visualize how the accuracy changes within the k range between 5 and 20. The subplot() function contained in the TeachingDemos package is helpful in drawing such an inset. It is interesting to adjust graph parameters to make the figure have a better appearance (*Figure 3*). The figure shows that the average accuracy is the highest at k=15. At a large k value (150 for example), all observations in the training dataset are included and all observations in the test dataset are assigned to the class with the largest number of subjects in the training dataset. Of course, this is not.

```
> install.packages("TeachingDemos")
> library(TeachingDemos)
> qplot(seq(1:150),accuracyCal(150),
xlab="k values",ylab="Average accuracy",
geom = c("point", "smooth"))
> subplot(
    plot(seq(1:30),accuracyCal(30), col=2,xlab='',
ylab='',cex.axis=0.8),
    x=grconvertX(c(0,0.75), from='npc'),
    y=grconvertY(c(0,0.45), from='npc'),
    type='fig', pars=list( mar=c(0,0,1.5,1.5)+0.1) )
```

Summary

The article introduces some basic ideas underlying the kNN algorithm. The dataset should be prepared before running the knn() function in R. After prediction of outcome with kNN algorithm, the diagnostic performance of the model should be checked. Average accuracy is the most widely used statistic to reflect the performance the kNN algorithm. Factors such as the k value, distance calculation and the choice of appropriate predictors all have significant impact on the model performance.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Short RD, Fukunaga K. The optimal distance measure for nearest neighbor classification. IEEE Transactions on Information Theory 1981;27:622-627.
2. Weinberger KQ, Saul LK. Distance metric learning for large margin nearest neighbor classification. The Journal of Machine Learning Research 2009;10:207-244.
3. Cost S, Salzberg S. A weighted nearest neighbor algorithm for learning with symbolic features. Machine Learning 1993;10:57-78.
4. Breiman L. Random forests. Machine Learning. 2001;45:5-32.
5. Zhang Z. Too much covariates in a multivariable model may cause the problem of overfitting. J Thorac Dis 2014;6:E196-E197.
6. Lantz B. Machine learning with R. 2nd ed. Birmingham: Packt Publishing; 2015:1.
7. Venables WN, Ripley BD. Modern applied statistics with S-PLUS. 3rd ed. New York: Springer; 2001.
8. Hernandez-Torruco J, Canul-Reich J, Frausto-Solis J, et al. Towards a predictive model for Guillain-Barré syndrome.

Conf Proc IEEE Eng Med Biol Soc 2015;2015:7234-7.
9. Linden A. Measuring diagnostic and predictive accuracy in disease management: an introduction to receiver operating characteristic (ROC) analysis. J Eval Clin Pract 2006;12:132-139.
10. Hand DJ, Till RJ. A simple generalisation of the area under the ROC curve for multiple class classification problems. Machine Learning 2001;45:171-186.
11. Thompson JR. Estimating equations for kappa statistics. Stat Med 2001;20:2895-2906.

Cite this article as: Zhang Z. Introduction to machine learning: k-nearest neighbors. Ann Transl Med 2016;4(11):218. doi: 10.21037/atm.2016.03.37

Naïve Bayes classification in R

Zhongheng Zhang

Abstract: Naïve Bayes classification is a simple probabilistic classification method based on the Bayes' theorem with the assumption of independence between features. The model is trained on training dataset to make predictions by predict() function. This article introduces two functions naiveBayes() and train() for the performance of Naïve Bayes classification.

Keywords: Machine learning; R; naïve Bayes; classification; average accuracy; kappa

Submitted Jan 25, 2016. Accepted for publication Feb 24, 2016.
doi: 10.21037/atm.2016.03.38
View this article at: http://dx.doi.org/10.21037/atm.2016.03.38

Introduction to naïve Bayes classification

Bayes' theorem can be used to make predictions based on prior knowledge and current evidence (1). With accumulating evidence, the prediction is ever changing. In technical terms, the prediction is the posterior probability that investigators are interested in. The prior knowledge is termed prior probability that reflects the most probable guess on the outcome without additional evidence. The current evidence is expressed as likelihood which reflects the probability of a predictor given a certain outcome. The training dataset is used to derive likelihood (2,3). Bayes' theorem is formally expressed by the following equation.

$$P(A|B)=\frac{P(B|A)\times P(A)}{P(B)} \quad [1]$$

where P(A) and P(B) are probability of events A and B without regarding each other. P(A|B) is the probability of A conditional on B and P(B|A) is the probability of B conditional on A. In naïve Bayes classification, A is categorical outcome events and B is a series of predictors. The word "naïve" indicates that the predictors are independent on each other conditional on the same outcome value. Therefore $P(b1,b2,b3|A)$ can be written as $P(b1|A)\times P(b2|A)\times P(b3|A)$, which makes the calculation process much easier.

I will use an example to illustrate how the naïve Bayes classification works. The example of sepsis diagnosis is employed and the algorithm is simplified. Suppose there are two predictors of sepsis, namely, the respiratory rate and mental status. Septic patients are defined as those with fast respiratory rate and altered mental status (4-6). The likelihood table is shown in *Table 1*. The table is obtained from the training dataset. In Bayes' theorem terms, the likelihood of fast respiratory rate given sepsis is 15/20=0.75, and the likelihood of altered mental status given non-sepsis is 3/80=0.0375. Suppose we have a patient with slow respiratory rate and altered mental status, and we want to make a classification of this patient to either sepsis or non-sepsis.

The prior probabilities of sepsis and non-sepsis are:

P(sepsis)=20/100=0.2

P(non-sepsis)=80/100=0.8

The probabilities of likelihood are:

P(fast RR|sepsis)=15/20=0.75

P(slow RR|sepsis)=5/20=0.25

P(fast RR|non-sepsis)=5/80=0.0625

P(slow RR|non-sepsis)=75/80=0.9375

P(altered mental status| sepsis)=17/20=0.85

P(normal mental status | sepsis)=3/20=0.15

P(altered mental status|non-sepsis)=3/80=0.0375

P(normal mental status |non-sepsis)=77/80=0.9625

By applying the maximum a posteriori classification rule (7,8), only the numerator of Bayes' equation needs to be calculated. The denominators of each classification are the same. The likelihood of sepsis given slow respiratory rate and altered mental status are:

$$\begin{aligned}&P(sepsis|slow\,RR\cap altered\,mental\,status)\\&=\frac{P(slow\,RR|sepsis)\times P(altered\,mental\,status|sepsis)\times P(sepsis)}{P(slow\,RR)\times P(altered\,mental\,status)}\\&=\frac{0.25\times 0.85\times 0.2}{P}=\frac{0.0425}{P}\end{aligned} \quad [2]$$

The probability of non-sepsis can be calculated in a similar fashion:

$$P(non-sepsis|slow\,RR \cap altered\ mental\ status) = \frac{P(slow\,RR|non-sepsis)\times P(altered\ mental\ status|non-sepsis)\times P(non-sepsis)}{P(slow\,RR)\times P(altered\ mental\ status)} = \frac{0.9375\times 0.0375\times 0.8}{P} = \frac{0.028125}{P} \quad [3]$$

Since the likelihood of sepsis is greater than non-sepsis (0.0425>0.028125), we classify it into the class of sepsis.

Working example

We employed the Titanic dataset to illustrate how naïve Bayes classification can be performed in R.

```
> data(Titanic)
> str(Titanic)
 table [1:4, 1:2, 1:2, 1:2] 0 0 35 0 0 0 17 0 118 154 ...
 - attr(*, "dimnames")=List of 4
  ..$ Class   : chr [1:4] "1st" "2nd" "3rd" "Crew"
  ..$ Sex     : chr [1:2] "Male" "Female"
  ..$ Age     : chr [1:2] "Child" "Adult"
  ..$ Survived: chr [1:2] "No" "Yes"
```

The dataset is a 4-dimensional array resulting from cross-tabulating 2,201 observations on 4 variables. Because the NaiveBayes() function can pass both data frame and tables, I would like to convert the 4-dimensional array into a data frame with each row represents a passenger. This is also the format with which original data are collected.

```
> countsToCases <- function(x, countcol = "Freq") {
   # Get the row indices to pull from x
   idx <- rep.int(seq_len(nrow(x)), x[[countcol]])
   # Drop count column
   x[[countcol]] <- NULL
   # Get the rows from x
   x[idx, ]
}
> caseTita<-countsToCases(as.data.frame(Titanic))
> head(caseTita)
    Class  Sex   Age   Survived
3   3rd    Male  Child No
3.1 3rd    Male  Child No
3.2 3rd    Male  Child No
3.3 3rd    Male  Child No
3.4 3rd    Male  Child No
3.5 3rd    Male  Child No
> nrow(caseTita)
[1] 2201
```

Table 1 Likelihood table to make a diagnosis of sepsis

Likelihood	Respiratory rate		Mental status		
	Fast	Slow	Altered	Normal	Total
Sepsis	15/20	5/20	17/20	3/20	20
Non-sepsis	5/80	75/80	3/80	77/80	80
Total	20/100	80/100	20/100	80/100	100

The as.data.frame() function converts the array into a data fame with each row representing the unique combinations of all variables. Then the custom-made function countsToCases() (http://www.cookbook-r.com) is employed to expand rows with more than one observations. The new dataset *caseTita* contains 2201 rows and four columns, which is exactly what we want.

Naïve Bayes classification with e1071 package

The e1071 package contains a function named naiveBayes() which is helpful in performing Bayes classification (9). The function is able to receive categorical data and contingency table as input. It returns an object of the class "naiveBayes". This object can be passed to predict() to predict outcomes of unlabeled subjects.

```
> install.packages("e1071")
> library(e1071)
> model <- naiveBayes(Survived ~ ., data = caseTita)
> predict(model, caseTita[sample(1:2201,10,
replace=FALSE),])
 [1] No  No  No  No  No  No  No  No  No  Yes
Levels: No Yes
```

In the above example, a training model is created by using the naiveBayes() function. The model is used to predict the survival status of a random sample of ten passengers. Here we use the sample() function to select 10 passengers without replacement. The predicted survival status of them is "No" for nine passengers and "yes" for the last one. If you want to examine the conditional a-posterior probabilities, a character

value "raw" should be assigned to the type argument.

```
> predict(model, caseTita[sample(1:2201,10,
replace=FALSE),],type="raw")
                No          Yes
[1,]  0.72478200  0.2752180
[2,]  0.72478200  0.2752180
[3,]  0.84661708  0.1533829
[4,]  0.09927006  0.9007299
[5,]  0.85522172  0.1447783
[6,]  0.72478200  0.2752180
[7,]  0.84661708  0.1533829
[8,]  0.85522172  0.1447783
[9,]  0.52792424  0.4720758
[10,] 0.52792424  0.4720758
```

The naiveBayes() function receives contingency table as well. The example below is also presented in the help file of the naiveBayes() function.

```
> m <- naiveBayes(Survived ~ ., data = Titanic)
> m
Naive Bayes Classifier for Discrete Predictors
Call:
naiveBayes.formula(formula = Survived ~ ., data = Titanic)
A-priori probabilities:
Survived
No                              Yes
0.676965                        0.323035

Conditional probabilities:
          Class
Survived  1st         2nd         3rd         Crew
No        0.08187919  0.11208054  0.35436242  0.45167785
Yes       0.28551336  0.16596343  0.25035162  0.29817159

          Sex
Survived  Male        Female
No        0.91543624  0.08456376
Yes       0.51617440  0.48382560

          Age
Survived  Child       Adult
No        0.03489933  0.96510067
Yes       0.08016878  0.91983122
```

The a-priori probabilities are prior probability in Bayes' theorem. That is, how frequently each level of class occurs in the training dataset. The rationale underlying the prior probability is that if a level is rare, it is unlikely that such level will occur in the test dataset. In other words, the prediction of an outcome is not only influenced by the predictors, but also by the prevalence of the outcome. Conditional probabilities are calculated for each variable. It is actually the likelihood table as shown in *Table 1*. For example, the likelihood of male given survival P(Male|Survived) equals to 0.51617440. Similarly, the predict() function can be applied for new passengers with predictors of age, sex and class.

Naïve Bayes classification with caret package

The caret package contains train() function which is helpful in setting up a grid of tuning parameters for a number of classification and regression routines, fits each model and calculates a resampling based performance measure. Let's first install and load the package.

```
> install.packages("caret")
> library(caret)
```

Then the data frame caseTita should be split into the predictor data frame and outcome vector. Remember to convert the outcome variable into a vector instead of a data frame. The later format will result in an error message.

```
> x<-caseTita[,-4]
> y<-caseTita$Survived
```

The model is trained by using train() function.

```
> model1 <- train(x,y,'nb',
trControl=trainControl(method='cv',number=10))
> model1
Naive Bayes
2201 samples
  3 predictor
  2 classes: 'No', 'Yes'
No pre-processing
Resampling: Cross-Validated (10 fold)
```

```
Summary of sample sizes: 1981, 1981, 1980, 1981, 1981, 1981,
...
Resampling results across tuning parameters:
usekernel Accuracy  Kappa     Accuracy SD  Kappa SD
FALSE     0.7782826 0.4441458 0.01614583   0.04959141
TRUE      0.7782826 0.4441458 0.01614583   0.04959141
Tuning parameter 'fL' was held constant at a value of 0
Accuracy was used to select the optimal model using the
largest value.
The final values used for the model were fL = 0 and
usekernel = FALSE.
```

The first argument is a data frame where samples are in rows and features are in columns. The second argument is a vector containing outcomes for each sample. 'nb' is a string specifying that the classification model is naïve Bayes. The trainController argument tells the trainer to use cross-validation ('cv') with 10 folds. Specifically, the original dataset is randomly divided into 10 equally sized subsamples. Of the 10 subsamples, 9 subsamples are used as training data, and the remaining one subsample is used as the validation data. The cross-validation process is then repeated for 10 times, with each of the 10 subsamples used once as the validation data. The process results in 10 estimates which then are averaged (or otherwise combined) to produce a single estimation (10,11). The output shows a kappa value of 0.4, which is not very good. It doesn't matter since this is only an illustration example. The next task is to use the model for prediction.

```
> predict(model1$finalModel,caseTita[sample(1:2201,10,
replace=FALSE),])$class
12.316 11.225 29.116 11.298 27.19 29.35 28.159 12.78 12.331 11.2
No     No     Yes    No     No    Yes   No     No    No     No
Levels: No Yes
```

The 10 subjects are randomly selected from the original dataset. The first line of the output indicates the row names of the subjects. The second line indicates the survival status by prediction with the model. To compare the predicted results with the observed results, a confusion matrix can be useful.

```
> table(predict(model1$finalModel,x)$class,y)
     y
     No  Yes
No   1364 362
Yes  126  349
```

This time the whole dataset was used. It is obvious that the error rate is not low as indicated by off-diagonal numbers.

Summary

The article introduces some basic ideas behind the naïve Bayes classification. It is a sample method in machine learning but can be useful in some instances. The training is easy and fast that just requires considering each predictor in each class separately. There are two packages *e1071* and *caret* that can be used for the performance of naïve Bayes classification. Key parameters within these packages are introduced.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Efron B. Mathematics. Bayes' theorem in the 21st century. Science 2013;340:1177-1178.
2. Medow MA, Lucey CR. A qualitative approach to Bayes' theorem. Evid Based Med 2011;16:163-167.
3. López Puga J, Krzywinski M, Altman N. Points of significance: Bayes' theorem. Nat Methods 2015;12:277-278.
4. Zhang Z, Chen L, Ni H. Antipyretic therapy in critically ill patients with sepsis: an interaction with body temperature. PLoS One 2015;10:e0121919.
5. Cohen J, Vincent JL, Adhikari NK, et al. Sepsis: a roadmap for future research. Lancet Infect Dis 2015;15:581-614.
6. Drewry AM. Hotchkiss RS1. Sepsis: Revising definitions of sepsis. Nat Rev Nephrol 2015;11:326-328.
7. Murphy KP. Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series). 1st ed. London: The MIT Press; 2012:1.
8. Kononenko I. Machine learning for medical diagnosis: history, state of the art and perspective. Artif Intell Med

2001;23:89-109.
9. Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien [R package e1071 version 1.6-7]. Comprehensive R Archive Network (CRAN); 2014. Available online: https://CRAN.R-project.org/package=e1071
10. Hadorn DC, Draper D, Rogers WH, et al. Cross-validation performance of mortality prediction models. Stat Med 1992;11:475-489.
11. Schumacher M, Holländer N, Sauerbrei W. Resampling and cross-validation techniques: a tool to reduce bias caused by model building? Stat Med 1997;16:2813-2827.

Cite this article as: Zhang Z. Naïve Bayes classification in R. Ann Transl Med 2016;4(12):241. doi: 10.21037/atm.2016.03.38

Machine Learning

Decision tree modeling using R

Zhongheng Zhang

Abstract: In the field of machine learning, decision tree learner is powerful and easy to interpret. It employs recursive binary partitioning algorithm that splits the sample in partitioning variable with the strongest association with the response variable. The process continues until some stopping criteria are met, In the example. I focus on conditional inference tree, which incorporates tree-structured regression models into conditional inference procedures. While growing a single tree is subject to small changes in the training data, random forests procedure is introduced to address this problem. The sources of diversity for random forests come from the random sampling and restricted set of input variables to be selected. Finally, I introduce R functions to perform model based recursive partitioning. This method incorporates recursive partitioning into conventional parametric model building.

Keywords: Machine learning; R; decision trees; recursive partitioning; conditional inference; random forests

Submitted Feb 05, 2016. Accepted for publication Mar 10, 2016.
doi: 10.21037/atm.2016.05.14
View this article at: http://dx.doi.org/10.21037/atm.2016.05.14

Understanding the decision tree

Decision tree learner is a technique of machine learning. As its name implies, the prediction or classification of outcomes is made going from root to leaves. The tree is made up of decision nodes, branches and leaf nodes. The tree is placed upside down, so the root is at the top and leaves indicating an outcome category is at the bottom. At the root, all classifications are mixed, representing the original dataset. Then the tree grows to the first node where a certain feature variable is used to split the population into sub-populations. Because the parent population can be split into numerous patterns, we are interested in the one with the greatest purity. In technical terminology, purity can be described by entropy.

$$\text{entropy}=\sum_{i=1}^{c}-p_i \log 2(p_i)$$

where c is the number of different classifications. For the research of mortality outcome, c equals to two. p_i is the proportion of observations falling into class i. From the formula, we can see that entropy is zero when the population is completely homogenous; and 1 indicates the maximum degree of impurity. Other statistics such as Gini index, Chi-square statistics and gain ratio are also employed to describe the purity and to decide the best splitting (1).

The tree will stop growing by the following three criteria: (I) all leaf nodes are pure with the entropy of zero; (II) a prespecified minimum change in purity cannot be made with any splitting methods; and (III) the number of observations in the leaf node reaches the prespecified minimum one (2). There are algorithms allowing the tree to grow and then the tree is pruned. Pruning technique aims to address the problem of overfitting as can occur in other parametric regression models (3).

Recursive partitioning within conditional inference framework, a type of decision tree learner, was developed by Hothorn and coworkers (2,4). It has many attractive features. Mathematical details of the technical are out of the scope of this paper. The word "recursive" refers to the fact that the decision nodes are repeatedly split. The algorithm works in the following steps:

(I) Test the independence between any of the input variables and the response. Stop if there is no dependence between variables and response variable. Otherwise select the input variable with the strongest association with the response. Permutation test is employed for the test of dependence (5). This step can prevent overgrowth of the tree (model overfitting) by statistical test;

(II) Split observations in the decision node in the selected input variable;

(III) Recursively repeat steps (I) and (II) until any of the stop criteria is reached.

Motivating example

We use the *airquality* dataset for the illustration of how to work with decision tree using R. The dataset contains information on New York air quality measurement in 1973. We obtain the dataset and examine the data frame with the following codes.

```
> data(airquality)
> str(airquality)
'data.frame': 153 obs. of 6 variables:
$ Ozone: int 41 36 12 18 NA 28 23 19 8 NA ...
$ Solar.R: int 190 118 149 313 NA NA 299 99 19 194 ...
$ Wind: num 7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
$ Temp: int 67 72 74 62 56 66 65 59 61 69 ...
$ Month: int 5 5 5 5 5 5 5 5 5 5 ...
$ Day: int 1 2 3 4 5 6 7 8 9 10 ...
```

The data frame contains 153 observations and 6 variables. In the example, we will set the *Ozone* as response variable. However, there are missing values in the variable. We first have to impute these missing values, or simply exclude observations with missing values in *Ozone*. Because dropping observations result in information loss, imputation method is used to address this problem (6). We impute missing *Ozone* values with values randomly selected from the non-missing ones. The sample() function allows taking a sample of the specified size from given elements. Here the size of the sample is 37, and the given element is a vector containing non-missing Ozone values.

```
> set.seed(888)
> airquality[is.na(airquality$Ozone),1]<-
sample(airquality[!is.na(airquality$Ozone),1],37)
> summary(airquality$Ozone)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.00	18.00	29.00	40.82	61.00	168.00

From the output of the summary() function, it is apparent that Ozone contains no missing value any more.

Conditional inference tree

In this section we will use the imputed dataset to build a conditional inference tree. Owning to the ctree() function in the *party* package, the code for building a tree is fairly simple (4).

```
> airct <- ctree(Ozone ~ ., data = airquality,controls =
ctree_control(maxsurrogate = 3))
```

The ctree() function created a conditional inference tree and returned an object of class *BinaryTree-class*. The first argument of the function is a formula defining the response and input variables. The format of the equation is similar to that used in building a generalized linear model. "data" argument defines the data frame which the model is built upon. The ctree_control() function tunes various parameters to control aspects of the conditional inference tree fit. In the example, we adjusted the number of surrogate splits to evaluate to 3. We can take a look at the tree structure as follows.

```
> airct

        Conditional inference tree with 5 terminal nodes

Response: Ozone
Inputs: Solar.R, Wind, Temp, Month, Day
Number of observations: 153

1) Temp <= 82; criterion = 1, statistic = 41.908
 2) Wind <= 6.3; criterion = 1, statistic = 15.879
  3)* weights = 8
 2) Wind > 6.3
  4) Solar.R <= 81; criterion = 0.977, statistic = 7.997
   5)* weights = 26
  4) Solar.R > 81
   6)* weights = 71
1) Temp > 82
 7) Wind <= 8; criterion = 0.977, statistic = 8.043
  8)* weights = 29
 7) Wind > 8
  9)* weights = 19
```

The above output gives you a general glimpse of the tree structure. The response variable is *Ozone*, and input variables include *Solar.R*, *Wind*, *Temp*, *Month* and *Day*. We can visualize the tree graphically using the generic function

plot(), and it is more convenient to interpret the tree with graphics.

```
> plot(airct)
```

The decision nodes are represented by circles, and each circle is numbered. The input variable to split upon is shown in each circle, together with the P value of the dependence test. For example, the first decision node at the top shows that *Temp* is the variable that is most strongly associated with *Ozone* and thus is selected as the first node. The association is measured by a P<0.001. The left and right branches show that a cutoff value equal to 82 is the best to reduce impurity. Thereafter, the decision nodes are further divided by variables *Wind* and *Solar.R* sequentially. The leaf nodes show the classifications of *Ozone* by the decision tree. Because variable *Ozone* is a continuous variable, the result is displayed by box plot (*Figure 1*).

Decision tree plot can be modified by setting graphical parameters. For example, if you want to display boxplot, instead of statistics, in inner nodes (decision nodes). The following code can be used:

```
> plot(airct, inner_panel = node_boxplot, edge_panel =
function(...) invisible(),tnex = 1)
```

As you can see in *Figure 2*, all inner nodes show boxplot of observations.

Sometimes investigators are interested in examining the split statistics based on which the split point is derived (4). Remember that the split input variable is continuous variable and there are numerous split points to choose from. For a given split point, there is a statistic measuring how good the splitting is. A greater statistic value is associated with a better splitting. Split statistics can be derived from *splitstatistic* element of a split. The following syntax draws scatter plots of split statistics for inner nodes 1, 2, 4 and 7. The leaf nodes have no split statistics because they do not split any more (the tree stops growing at leaf node, *Figure 3*).

```
> inner <- nodes(airct, c(1, 2, 4,7))
> layout(matrix(1:length(inner), ncol = length(inner)/2))
> out <- sapply(inner, function(i) {
    splitstat <- i$psplit$splitstatistic
    x <- airquality[[i$psplit$variableName]][splitstat >0]
                                    plot(x, splitstat[splitstat> 0], main =
paste("Node",i$nodeID), xlab = i$psplit$variableName,
ylab = "Statistic",ylim = c(0, 10), cex.axis = 1.2, cex.lab =
1.2,cex.main = 1.2)
    abline(v = i$psplit$splitpoint, lty = 4)
     })
```

Prediction with decision tree

The purpose of building a decision tree model is to predict responses in future observations. A subset of the *airquality* data frame is employed as a new cohort of observations.

```
> ind <- sample(2, nrow(airquality), replace=TRUE,
prob = c(0.7,0.3))
> newData <- airquality[ind==2,]
> newpred <- predict(airct, newdata= newData)
> plot(newpred,newData$Ozone,
xlab="Ozone value predicted by decision tree",
ylab="Observed ozone value")
```

The sample() function returns indicator vector of the same length to the row number of the *airquality* data frame. There are two levels of "1" and "2" in this vector. Observations in the *airquality* data frame denoted by "2" are selected to constitute the *newData*. Then the predict() function is used to make a prediction based on input variables. Because the decision tree results in five classifications (5 leaf nodes), the predict() function returns a vector with five values. Each value is the mean value of observations in each leaf nodes (*Figure 4*). We plot predicted value against observed value in a scatter plot. Not surprisingly, the predicted values take only five values.

Random forests

In the above sections, I grow a single tree for prediction. This method has been criticized for its instability to small changes in the training data. In recursive partitioning framework, the decision on which input variable to split in and the split point determine how observations are split into leaf nodes. The exact split position and input variable to be selected are highly dependent on the training data. A small change to the training data may alter the first variable to split in, and the appearance of tree can be completely altered. One approach

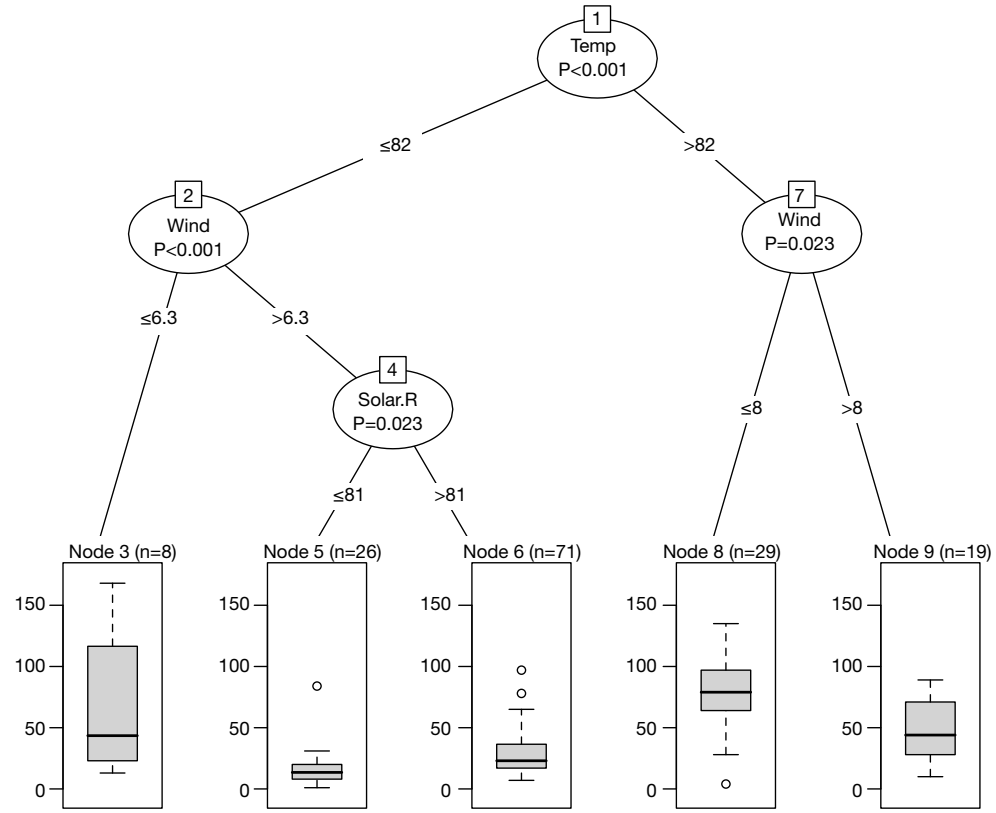


Figure 1 Conditional inference tree for the airquality data. For each inner node, input variable and P values are given, the boxplot of Ozone value is displayed for each terminal node.

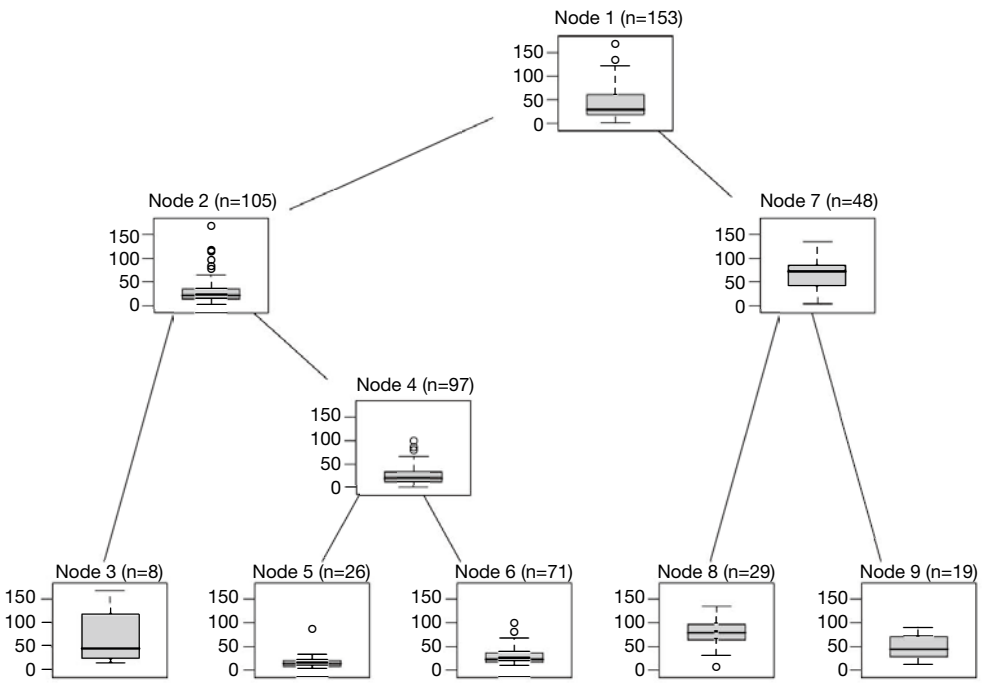


Figure 2 Conditional inference tree for the airquality data with the Ozone value displayed for both inner and terminal nodes.

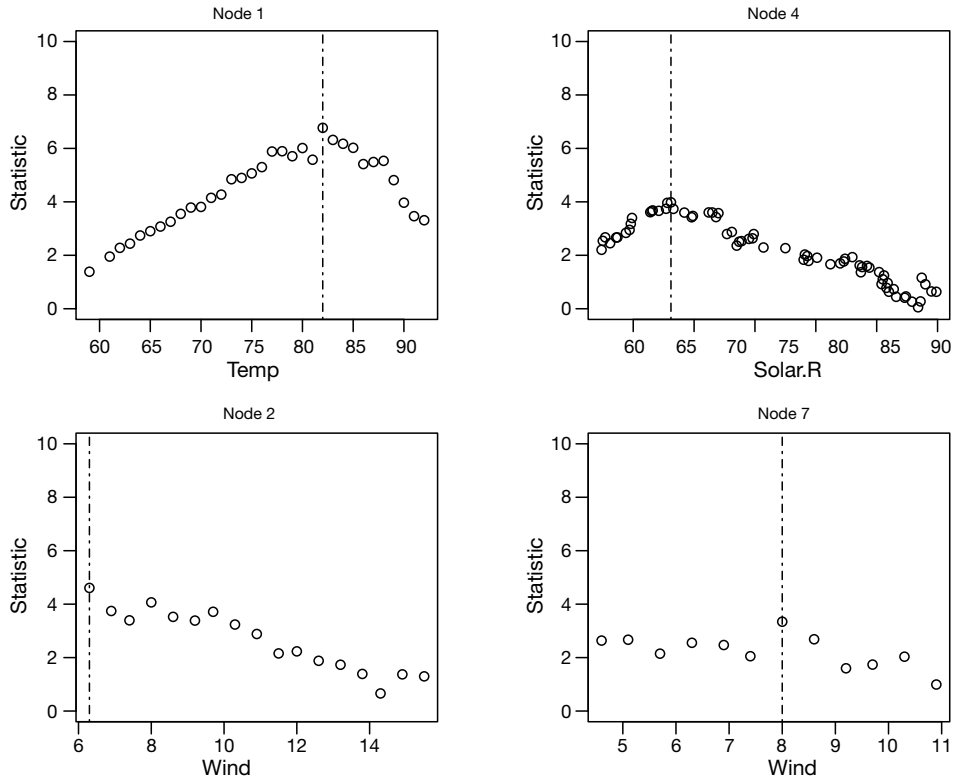


Figure 3 Split point estimation based on split statistics in inner nodes 1, 2, 4 and 7. Standardized two-sample test statistics are plotted against each possible split point in the selected input variable. The estimated split point is denoted by a vertical dotted line.

to this problem is to grow a forest instead of one tree (7). Diversity is introduced from two sources: (I) a set of trees is built on random samples of the training sample; and (II) the set of variables to be selected from in each decision node is restricted. Trees of the forest can be built on bootstrap samples or a subsample of the training dataset. If there are 10 variables to be selected from, we can restrict it to 5 at a given split node. The five variables are randomly selected and the pools of candidate variables can be different across forest trees. The final prediction model is the combination of all trees (8).

```
> aircf<-cforest(Ozone ~ ., data = airquality)
> aircf
        Random Forest using Conditional Inference Trees
Number of trees: 500
Response: Ozone
Inputs: Solar.R, Wind, Temp, Month, Day
Number of observations: 153
```

The syntax for fitting a random forests model is similar to a single conditional inference tree. The output of *aircf* shows that we have grown 500 trees. The response variable is *Ozone* and input variables are *Solar.R*, *Wind*, *Temp*, *Month* and *Day*. Tree plot cannot be drawn because these trees are different in structure. However, we can make prediction with the random forests model. Again, we use the predict() function, but this time the model is *aircf*.

```
> predforest<-predict(aircf, newdata= newData)
> plot(predforest,newData$Ozone,
ylab="Observed ozone value",
xlab="Predicted ozone value based on random forest")
```

In the graphical output (*Figure 5*), observed ozone values are plotted against predicted values by the random forests model. This time the predicted value is not restricted to five levels. Instead, they are evenly distributed across the ozone range, which is more likely to occur in the real world.

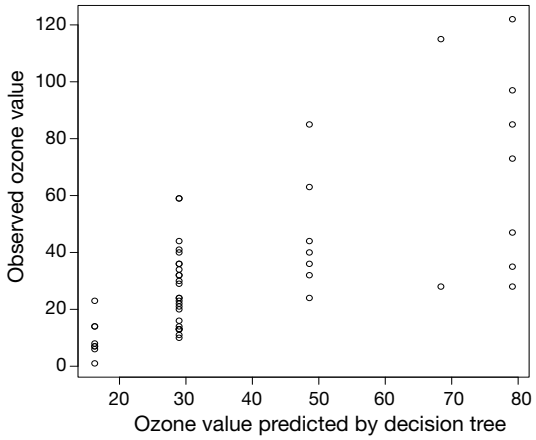


Figure 4 Scatter plot showing predicted value based on conditional inference tree against observed value. The predicted values are only allowed to take five values that are the mean values of each classification.

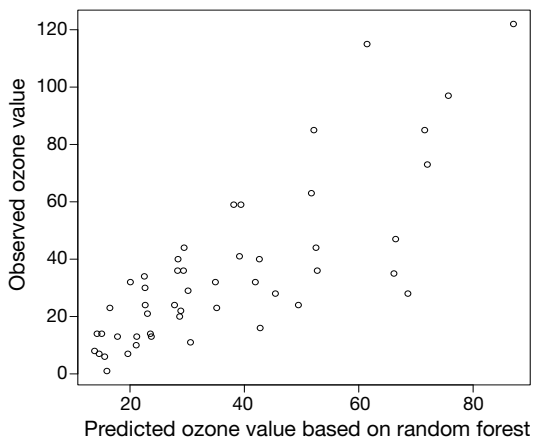


Figure 5 Scatter plot showing the observed ozone values against predicted values by random forests model. This time the predicted value is not restricted to five levels. Instead, they are evenly distributed across the ozone range, which is more likely to occur in the real world.

Model based recursive partitioning

Recursive partitioning can be incorporated into conventional parametric model building. Model based recursive partitioning algorithm consists of the following steps: (I) a parametric model (such as linear model and logistic regression model) is fit to current dataset; (II) parameter instability is tested over partitioning variables; (III) tree node is split by partitioning variable associated with the highest parameter instability; (IV) repeat above steps in each of the daughter nodes (9). Model based recursive partitioning algorithm can be easily performed using R.

```
> airmod<-mob(Ozone ~Temp+Day|
Solar.R+Wind+Month , data = airquality)
> plot(airmod)
```

In the above code, we used variables *Temp* and *Day* to construct a linear regression model with the *Ozone* as the response variable. Variables after symbol "|" are partitioning variables. The result is displayed with decision tree plot (*Figure 6*). Because Temp is used as a predictor in regression model, it no longer participates in partitioning. Thus the first partition is made on the variable *Month*. There seems no strong association between *Day* and *Ozone* as shown in the Figure. Therefore, we need to take a close look at statistics of the model.

```
> airmod
1) Month <= 6; criterion = 1, statistic = 32.402
 2)* weights = 57
Terminal node model
Gaussian GLM with coefficients:
(Intercept)    Temp     Day
 -52.3217   0.9880   0.7325

1) Month > 6
 3) Wind <= 8; criterion = 0.952, statistic = 15.979
  4)* weights = 36
Terminal node model
Gaussian GLM with coefficients:
(Intercept)    Temp     Day
 -200.2079   3.1813   -0.1323

 3) Wind > 8
  5)* weights = 53
Terminal node model
Gaussian GLM with coefficients:
(Intercept)    Temp     Day
 -149.8501   2.1784   0.5631
```

The above output displays coefficient of the fitted linear regression model. The coefficients of *Temp* vary remarkably across leaf nodes, indicating a good partitioning. The coefficient of each leaf nodes can be extracted using the coef() function.

```
> coef(airmod)
```

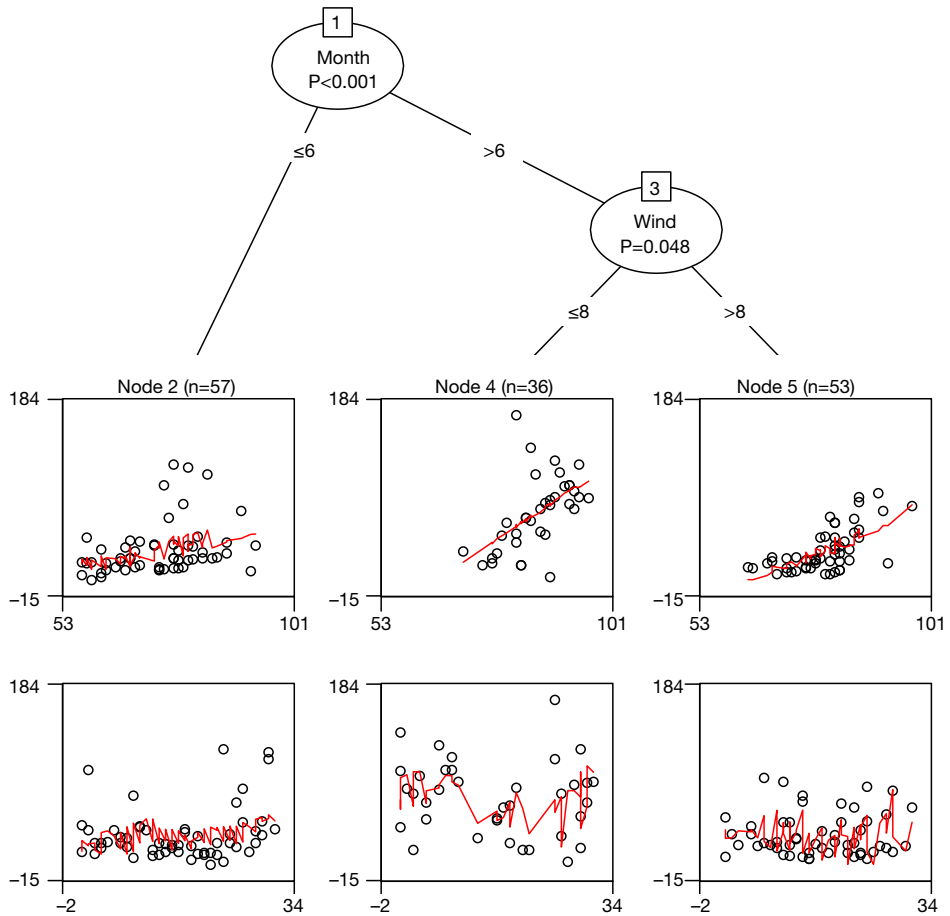


Figure 6 Linear-regression-based tree for the airquality data. The leaf nodes give partial scatter plots for *Temp* (upper panel) and *Day* (lower panel).

	(Intercept)	Temp	Day
2	-52.32172	0.9880186	0.7325495
4	-200.20787	3.1812577	-0.1323226
5	-149.85014	2.1783656	0.5630807

The output of coef() function displays the model coefficient for each leaf node. Test for parameter stability can be examined with the sctest() function by passing a model based tree and node number to the function.

```
> sctest(airmod, node =1)
```

	Solar.R	Wind	Month
statistic	7.3785883	3.016852e+01	3.240226e+01
p.value	0.8682267	1.012705e-04	3.405009e-05

In the above output, we can see that the P value for *Month* is the smallest and thus split of the first decision node is made on *Wind* (*Figure 6*).

Summary

The article introduces some basic knowledges behind the decision tree learning. It employs recursive binary partitioning algorithm that splits the sample in partitioning variable with the strongest association with the response variable. The process continues until some certain criteria are met. In the example, I focused on conditional inference tree, which incorporated tree-structured regression models into conditional inference procedures. While growing a single tree is subject to small changes in the training data, random forests procedure is introduced. The sources of diversity for random forest come from the random sampling and restricted set of input variables to be selected. Finally, I introduce R functions to perform model based

recursive partitioning. This method incorporates recursive partitioning into the conventional parametric model building.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Lantz B, editor. Machine learning with R. 2nd ed. Birmingham: Packt Publishing, 2015.
2. Hothorn T, Hornik K, Zeileis A. Unbiased Recursive Partitioning: A Conditional Inference Framework. Journal of Computational and Graphical Statistics 2006;15:651-674.
3. Zhang Z. Too much covariates in a multivariable model may cause the problem of overfitting. J Thorac Dis 2014;6:E196-E197.
4. Hothorn T, Hornik K, Strobl C, et al. Party: A laboratory for recursive partytioning. 2010.
5. Strasser H, Weber C. On the Asymptotic Theory of Permutation Statistics. SFB Adaptive Information Systems and Modelling in Economics and Management Science, WU Vienna University of Economics and Business, 1999.
6. Zhang Z. Missing data imputation: focusing on single imputation. Ann Transl Med 2016;4:9.
7. Strobl C, Malley J, Tutz G. An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. Psychol Methods 2009;14:323-348.
8. Breiman L, editor. Random Forests. Machine Learning. Dordrecht: Kluwer Academic Publishers, 2001:5-32.
9. Zeileis A, Hothorn T, Hornik K. Model-Based Recursive Partitioning. Journal of Computational and Graphical Statistics 2012;17:492-514.

Cite this article as: Zhang Z. Decision tree modeling using R. Ann Transl Med 2016;4(15):275. doi: 10.21037/atm.2016.05.14

A gentle introduction to artificial neural networks

Zhongheng Zhang

Abstract: Artificial neural network (ANN) is a flexible and powerful machine learning technique. However, it is under utilized in clinical medicine because of its technical challenges. The article introduces some basic ideas behind ANN and shows how to build ANN using R in a step-by-step framework. In topology and function, ANN is an analogue to the human brain. There are input and output signals transmitting from input to output nodes. Input signals are weighted before reaching output nodes according to their respective importance. Then the combined signal is processed by activation function. I simulated a simple example to illustrate how to build a simple ANN model using the nnet() function. This function allows for one hidden layer with varying number of units in that layer. The basic structure of ANN can be visualized with the plug-in plot.nnet() function. The plot function is powerful that it allows for a variety of adjustments to the appearance of the neural networks. Prediction with ANN can be performed with the predict() function, similar to that of conventional generalized linear models. Finally, the prediction power of ANN is examined using confusion matrix and average accuracy. It appears that ANN is slightly better than the conventional linear model.

Keywords: Machine learning; R; neural networks; recursive partitioning; conditional inference; random forests

Submitted Feb 25, 2016. Accepted for publication Mar 21, 2016.
doi: 10.21037/atm.2016.06.20
View this article at: http://dx.doi.org/10.21037/atm.2016.06.20

Introduction

Artificial neural networks (ANN) mimic human brain in processing input signals and transform them into output signals (1). It provides powerful modeling algorithm that allows for non-linearity between feature variables and output signals. ANN is a kind of non-parametric modeling technique, which is suitable for complex phenomenon that investigators do not know the underlying functions. In other words, ANN is able to learn from data without specific function assumptions. The article firstly provides some basic knowledges on the ANN, and then shows how to conduct ANN modeling with simulated data. Predicting performance of the ANN is compared with that of the generalized linear model.

Understanding ANN

ANN works in a very similar way to human brain. In structure, human brain is made up of neurons and there are approximately 85 billion neurons in human brain (2). The dendrites of a neuron receive input signals from environmental stimulation or up-stream neurons. Signal is processed in the cell body and transmitted along axon to the output terminal. The output signal may be received by down-stream neurons or by the function organs such as muscles to make reaction. A single artificial neuron works in a similar way. Feature variables, also known as predictors, input variables and covariates, are input signals that provide information for pattern recognition. Each feature variable is weighted according to its importance (3). This work is done by dendrites in the biological nervous system. The weighted signals are summed and processed by activation function. The signal processing procedure can be mathematically expressed as:

$$y(x) = \Phi\left(\sum_{i=1}^{n} w_i \cdot x_i\right)$$

where y is the output signal, Φ() is the activation function, x is a input variable and w is a weight assigned to each input variable. Suppose there are n input variables. To better understand ANN, it can be compared to the regression model. Each input variable is in analogue to the predictors of a regression model. Weight is actually the coefficient of each predictor.

In ANN topology, input nodes receive feature variables from raw data and the output node applies activation functions to combined information from input nodes. The

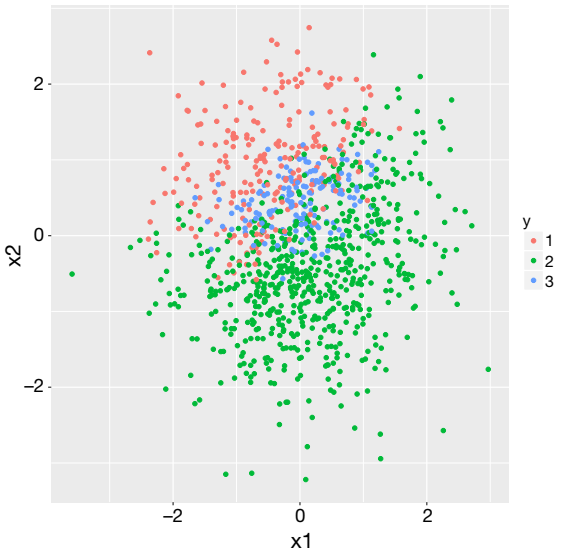


Figure 1 Distribution of observations in the two-dimension feature space. Classifications are denoted with dot colors. The three classifications are not linearly separable, which is as expected because output classifications are constructed with quadratic terms.

nodes are arranged in layers (3). For example, all input nodes constitute one layer. If a network only contains input and output nodes, it is termed single-layer network, or skip-layer units. Such network is commonly used for simple classification in which the outcome pattern is linearly separable. More complex tasks can be done with a multilayer network that adds one or more hidden layers to the single-layer network. In our example, we allow for one hidden layer to be added to the network.

Working example

To illustrate the process of building ANN with R, I created a simple example. The dataset is made up of two input variables x1 and x2, and one output variable y with three levels. The dataset is generated by the following syntax.

```
> set.seed(888) # set a seed for replication
> x1<-rnorm(1000,0)
> set.seed(666) # a different seed to allow different x1
#and x2 vectors
> x2<-rnorm(1000,0)
> logit1<-2+3*x1+x1^2-4*x2
> logit2<-1.5+2*x1-3*x2^2+x2
> Denominator<- 1+exp(logit1)+exp(logit2) #
#denominator for probability calculation
> vProb <- cbind(1/Denominator, exp(logit1)/
Denominator, exp(logit2)/Denominator) #Calculating
#the matrix of probabilities for three choices
> mChoices <-t(apply(vProb, 1, rmultinom, n = 1, size
= 1)) # Assigning value 1 to maximum probability and
#0 for the rest to get the appropriate choices for the
#combinations of x1 and x2
> data<- cbind.data.frame(y = as.factor(apply(mChoices,
1, function(x) which(x==1))), x1,x2) #response variable
#y and predictors x1 and x2 are combined together.
```

To simulate a multinomial outcome variable, we created two logit functions, namely logit1 and logit2 (4). The three-level output contains one reference level, and the remaining two are compared with the reference level. Then the probability of each level can be computed from input features. Here quadratic terms of input variables are applied. For each observation, the level with the largest probability is assigned value one and the remaining two levels are coded by zero. The last line combines these variables into a data frame.The contents started with a "#" symbol are used for annotation, and they will not be executed.

To take a look at the dataset, I employ the qplot() function. If you have not installed ggplot2 package, install it before running the library() function (5).

```
> library(ggplot2)
> qplot(x1,x2,data=data,geom="point",color=y)
```

Figure 1 plots x1 against x2, which constitutes a two-dimension feature space. It appears that the three classifications are not linearly separable, which is as expected because I constructed output classifications with quadratic terms. In particular, class 3 is embedded between class 1 and 2.

ANN training and visualization

ANN training can be easily done by using the nnet() function. The original cohort is split into training and test sets. Because the observations in data are randomly arranged, the first 700 cases are used as training set and the remaining 300 are used as test set.

```
> train<-data[1:700,]
```

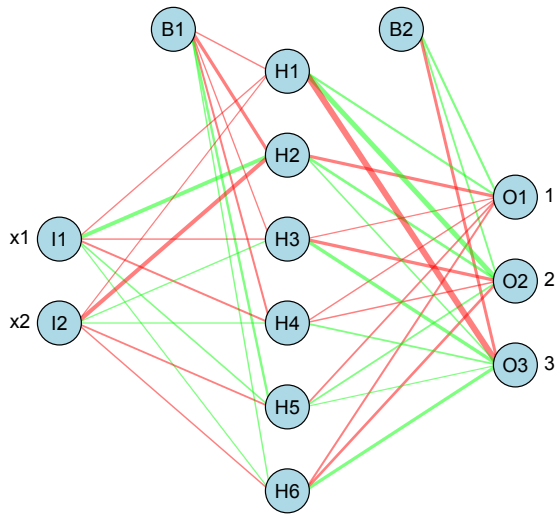



Figure 2 Artificial neural networks plot showing nodes and connections of the network. There are two input nodes named I1 and I2, transmitting information from feature variables x1 and x2. The green and red colors represent the positive and negative weights, respectively. B1 is the bias applied to hidden neurons.

```
> test<-data[701:1000,]
```

Up to now, the data are well prepared for ANN training. In most situations, the feature variables should be scaled before being passed to the nnet() function. However, this step is not necessary because our feature variables are created in the same scale.

```
> annmod<-nnet(y~.,train,size=6)
```

The above code creates an object of class "nnet" and stored it with the name of *annmod*. The first argument of nnet() function is a data frame of feature variables and the second argument is a vector of response variable. The size argument defines the number of units in the hidden layer.

The visualization function of ANN is not included in the *nnet* package. Fortunately, the function, plot.nnet(), is very powerful in plotting the neural network. It also allows for customization of the graph with a variety of options. To install the package into your workspace, you need to install the *devtools* package. It contains package development tools for R.

```
> install.packages("devtools")
> library(devtools)
> source_url('https://gist.githubusercontent.com/
Peque/41a9e20d6687f2f3108d/raw/85e14f3a292e126f14
54864427e3a189c2fe33f3/nnet_plot_update.r')
```

The plot.nnet() function runs in the following way.

```
> plot.nnet(annmod, alpha.val = 0.5, pos.col='green',
neg.col='red')
```

The first argument of the function is an object of "nnet". Transparency of connections is determined by the alpha.val argument whose value ranges between 0 and 1. The color of positive connections is determined by pos.col argument and here I make it green. Similarly, the negative connections are depicted with red color. *Figure 2* is the ANN plot showing nodes and connections of the network. There are two input nodes named I1 and I2, transmitting information from feature variables x1 and x2. Weights are assigned to each of the connections between input nodes and hidden nodes. The green and red colors represent the positive and negative weights, respectively. B1 is the bias applied to hidden neurons. Finally, signals are transmitted to the output neurons which are denoted by O1 through O3. Similarly, there are weights and bias applied to the output neurons.

Prediction with ANN

Training of the ANN is only the first step of a project. More important work is prediction on future observations with the trained model. Like the prediction with other models, the predict() function works well with ANN. The first argument is an object of "nnet". The second argument is a data frame containing feature variables. Because the response variable in our example is a factor variable with three levels, the type of output is "class".

```
> pred<-predict(annmod,test[,-1],type="class")
> table(test[,1],pred)
 pred
          1   2   3
1        44  12  11
2         7 163  13
3        12  17  21
```

To examine the predictive accuracy of the ANN model, I created a confusion matrix as shown in the output of table()

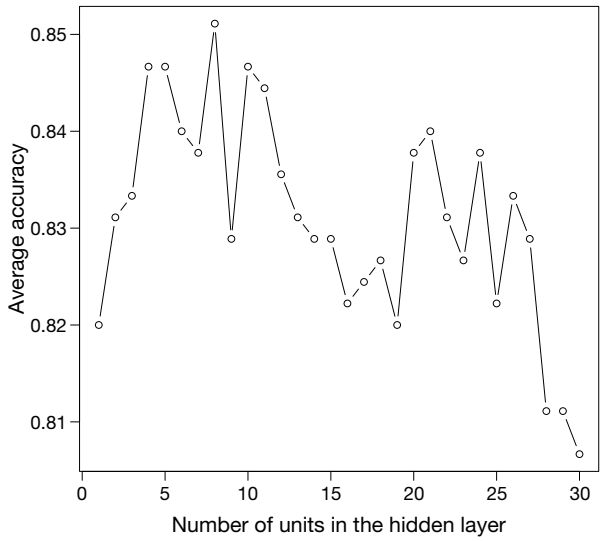


Figure 3 Plotting average accuracy against the number of units in the hidden layer. It appears that the accuracy reaches its largest value when the number of units is between 5 to 10.

function (6). The diagonal of the matrix shows correctly classified numbers. Alternatively, the classification of ANN model can be evaluated using average accuracy (7). I write a function called accuracyCal() to perform the calculation in the following syntax.

```
> accuracyCal<-function(N) {
 accuracy<-1
 for (x in 1:N) {
 annmod<-nnet(y~., data=train,
size=x,trace=FALSE,maxit=200)
 pred<-predict(annmod,test[,-1],type="class")
 table<- table(test[,1],pred)
 if (ncol(table)==3) {
 table<-table
 }
 else {
 table<-cbind(table,c(0,0,0))
 }
 tp1<-table[1,1]
 tp2<-table[2,2]
 tp3<-table[3,3]
 tn1<-table[2,2]+table[2,3]+table[3,2]+table[3,3]
 tn2<-table[1,1]+table[1,3]+table[3,1]+table[3,3]
 tn3<-table[1,1]+table[1,2]+table[2,1]+table[2,2]
 fn1<-table[1,2]+table[1,3]
 fn2<-table[2,1]+table[2,3]
 fn3<-table[3,1]+table[3,2]
 fp1<-table[2,1]+table[3,1]
 fp2<-table[1,2]+table[3,2]
 fp3<-table[1,3]+table[2,3]
 accuracy<-c(accuracy, (((tp1+tn1)/
(tp1+fn1+fp1+tn1))+((tp2+tn2)/
(tp2+fn2+fp2+tn2))+((tp3+tn3)/(tp3+fn3+fp3+tn3)))/3)
 }
 return(accuracy[-1])
 }
```

With this function, we can easily calculate a series of average accuracy by varying the number of units in the hidden layer. In the following syntax, a number of 30 is passed to the function, which in turn returns a series of average accuracies for ANN with the number of hidden layer units ranging from 1 to 30. Visualization of how average accuracy varies with the size parameter can be performed with generic plot() function.

```
> accuracySeri<-accuracyCal(30)
> plot(accuracySeri,type="b",xlab="Number of units in
the hidden layer.",ylab="Average Accuracy")
```

Figure 3 plots average accuracy against the number of units in hidden layer. It appears that the accuracy reaches its largest value when the number of units is between 5 to 10.

Comparison with generalized linear model

The same task can be done with the generalized linear model. For classification of outcome variable with three levels, the multinomial logistic regression model is a choice. The function multinom() shipped with *nnet* package can do the work.

```
> model.lin<-multinom(y~.,train)
> pred.lin<-predict(model.lin,test[,-1])
> table<-table(test[,1],pred.lin)
> table
 pred.lin
              1          2          3
1            51         14          2
2            12        168          3
3            17         31          2
```

The confusion matrix shows that the predictive performance of generalized linear model is not inferior

to the ANN. Next we continue to calculate the average accuracy.

```
> tp1<-table[1,1]
> tp2<-table[2,2]
> tp3<-table[3,3]
> tn1<-table[2,2]+table[2,3]+table[3,2]+table[3,3]
> tn2<-table[1,1]+table[1,3]+table[3,1]+table[3,3]
> tn3<-table[1,1]+table[1,2]+table[2,1]+table[2,2]
> fn1<-table[1,2]+table[1,3]
> fn2<-table[2,1]+table[2,3]
> fn3<-table[3,1]+table[3,2]
> fp1<-table[2,1]+table[3,1]
> fp2<-table[1,2]+table[3,2]
> fp3<-table[1,3]+table[2,3]
> accuracy<-(((tp1+tn1)/
(tp1+fn1+fp1+tn1))+((tp2+tn2)/
(tp2+fn2+fp2+tn2))+((tp3+tn3)/(tp3+fn3+fp3+tn3)))/3
> accuracy
[1] 0.8244444
```

The average accuracy is 0.82, which is slightly lower than that obtained by ANN with the size equal to 6.

Summary

The article introduces some basic ideas behind ANN. It is an analogue to the human brain. There are input and output signals in ANN. Input signals are weighted before reaching output nodes according to their respective importance. Then the combined signal is processed by activation function. I simulated a simple example to illustrate how to build a simple ANN model using the nnet() function. This function allows for one hidden layer with varying number of units in that layer. The basic structure of ANN can be visualized with the plug-in plot.nnet() function. The plot function is powerful that it allows for a variety of adjustments to the appearance of the neural networks. Prediction with ANN can be performed with the predict() function, similar to prediction with conventional models. Finally, the prediction power of ANN is examined using a confusion matrix and average accuracy. It appears that ANN is slightly better than the conventional linear model.

Cite this article as: Zhang Z. A gentle introduction to artificial neural networks. Ann Transl Med 2016;4(19):370. doi: 10.21037/atm.2016.06.20

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Wesolowski M, Suchacz B. Artificial neural networks: theoretical background and pharmaceutical applications: a review. J AOAC Int 2012;95:652-668.
2. Lantz B, editor. Machine learning with R. 2nd ed. Birmingham: Packt Publishing, 2015:1.
3. Haykin SO, editor. Neural networks and learning machines. 3rd ed. New Jersey: Prentice Hall, 2008.
4. Hosmer DW Jr, Lemeshow S, Sturdivant RX, editors. Applied logistic regression. 3rd ed. Hoboken, NJ: Wiley, 2013:1.
5. Wickham H, editor. ggplot2: elegant graphics for data analysis. New York: Springer-Verlag, 2009.
6. Stehman SV. Selecting and interpreting measures of thematic classification accuracy. Remote Sensing of Environment 1997;62:77-89.
7. Hernandez-Torruco J, Canul-Reich J, Frausto-Solis J, et al. Towards a predictive model for Guillain-Barré syndrome. Conf Proc IEEE Eng Med Biol Soc 2015;2015:7234-7.

Neural networks: further insights into error function, generalized weights and others

Zhongheng Zhang

Abstract: The section is the continuum of a previous one providing further insights into the structure of neural network (NN). Key concepts of NN including activation function, error function, learning rate and generalized weights are introduced. NN topology can be visualized with the generic plot() function by passing a "nn" class object. Generalized weights assist interpretation of the NN model with respect to the independent effect of individual input variables. A large variance of generalized weights for a covariate indicates non-linearity of its independent effect. If generalized weights of a covariate are approximately zero, the covariate is considered to have no effect on outcome. Finally, prediction of new observations can be performed using the compute() function. Make sure that the feature variables passed to the compute() function are in the same order to that in the training NN.

Keywords: Machine learning; R; neural networks (NNs); error function; generalized weights; activation function

Submitted Mar 20, 2016. Accepted for publication Apr 21, 2016.
doi: 10.21037/atm.2016.05.37
View this article at: http://dx.doi.org/10.21037/atm.2016.05.37

Introduction

Neural network (NN) has been introduced in the last article. In this tutorial, more insights will be provided to give readers a comprehensive understanding of how signals are processed in the neural networks. Specifically, key terms and concepts such as error function, generalized weights, multi-layer perceptron and learning rate are illustrated in the working example. I believe that concepts are more comprehensible when they are illustrated in a working example than in pure theoretical framework.

Working example

If you have installed *MASS* package, you can load the *birthwt* dataset directly into your workspace. Otherwise, you need to install the MASS package first.

```
> data(birthwt)
> str(birthwt)
'data.frame':	189 obs. of  10 variables:
 $ low  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ age  : int  19 33 20 21 18 21 22 17 29 26 ...
 $ lwt  : int  182 155 105 108 107 124 118 103 123 113 ...
 $ race : int  2 3 1 1 1 3 1 3 1 1 ...
 $ smoke: int  0 0 1 1 1 0 0 0 1 1 ...
 $ ptl  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ ht   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ ui   : int  1 0 0 1 1 0 0 0 0 0 ...
 $ ftv  : int  0 3 1 2 0 0 1 1 1 0 ...
 $ bwt  : int  2523 2551 2557 2594 2600 2622 2637 2637
2663 2665 ...
```

The dataset contains 189 observations and ten variables. The data were collected at Baystate Medical Center, Springfield, Mass in 1986. The first variable is the indicator of low birth weight (low). Mother's age (age), weight (lwt), race (race), smoking status during pregnancy (smoke), number of previous premature labors (ptl), history of hypertension (ht), presence of uterine irritability (ui), and the number of physician visits during the first trimester (ftv) were recorded. Birth weights of babies are measured and recorded in gram (bwt). The purpose of the study is to predict body weight of newborns with recorded covariates, and body weight has been recorded in gram as a continuous variable and in category as an indicator variable. Thus, the outcome can be either binary or continuous.

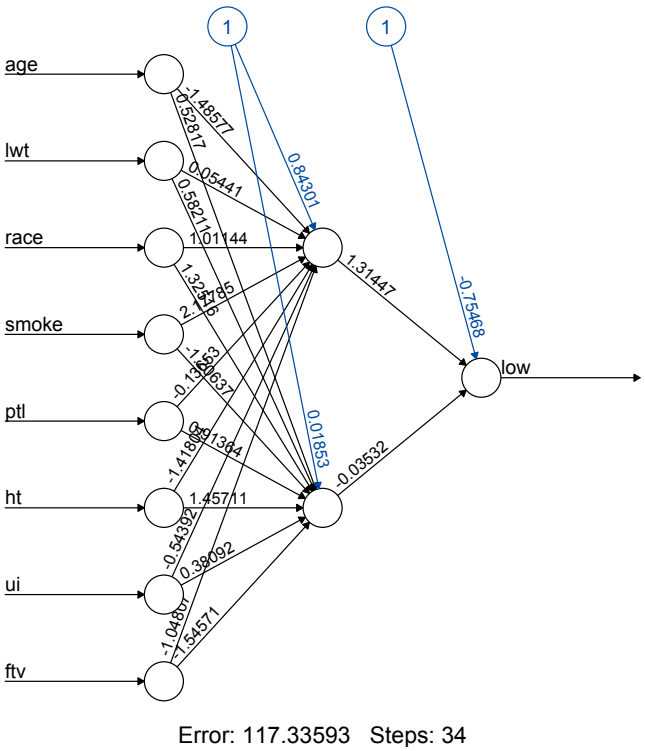


Figure 1 Visualization of neural network. Signal from each of the eight predictors is received by input node. The synaptic weights are displayed above each line. The blue circles indicate bias, corresponding the intercept in conventional regression model. There is one hidden layer consisting two units. The variable *low* is output neuron.

Training the neural network (NN)

With the help of the neuralnet() function contained in *neuralnet* package, the training of NN model is extremely easy (1).

```
> install.packages("neuralnet")
> library(neuralnet)
> nn <- neuralnet(
      low~age+lwt+race+smoke+ptl+ht+ui+ftv,
      data=birthwt, hidden=2, err.fct="ce",
      linear.output=FALSE)
> plot(nn)
```

The above codes firstly install the *neuralnet* package and then load it to the workspace. The neuralnet() function is powerful and flexible in training a NN model. I leave detailed explanations of its parameters to the next paragraph. After model training, the topology of the NN can be visualized using the generic function plot() with many options for adjusting the appearance of the plot. *Figure 1* displays topology of the NN. Signal from each of the eight predictors is received by input node. The synaptic weights are displayed above each line. The blue circles indicate bias, corresponding to the intercept in conventional regression model. There is one hidden layer consisting two units. The variable *low* is output neuron. The total error is 117 and 34 steps are needed for iterations to converge. They are shown at the bottom of the figure.

The first argument of the neuralnet() function is a formula describing the model to be fitted. As you can see from the example, the specification of the model is similar to that in building generalized linear model (2). The right-hand side specifies the response variable, and the left-hand side is predictors connected by "+" symbols. The "data" argument specifies the data frame on which the NN training is based. The parameter *hidden* is a vector specifying the number of hidden layers and the number of units in each layer. For example, a vector c(4,2,5) indicates a neural network with three hidden layers, and the numbers of neurons for the first, second and third layers are 4, 2 and 5, respectively. In our example, there is one hidden layer consisting two neurons.

In analogue to the link function in generalized linear model, NN requires to define an activation function. A differentiable activation function can be defined by "act.fct" argument. A string value of "logistic" or "tanh" is acceptable for logistic function or tangent hyperbolicus, respectively. The default is "logistic". Activation function transforms aggregated input signals, also known as induced local field, into output signal (3). In our example, the output is a binary outcome, and it is left to its default.

The above NN is a multilayer perceptron that contains one or more hidden layers. If a NN contains no hidden layer, it is reduced to the Rosenblatt's perceptron (4). Other features of a multilayer perceptron include (I) the activation function is differentiable; and (II) the network exhibits a high degree of connectivity.

The argument "err.fct" defines the error function, which can be either "sum of squared error (sse)" or "cross entropy (ce)". The default is "sum of squared error" and it can be expressed as:

$$E_{sse}=\frac{1}{2}\sum_{l=1}^{L}\sum_{h=1}^{H}(o_{lh}-y_{lh})^2 \quad [1]$$

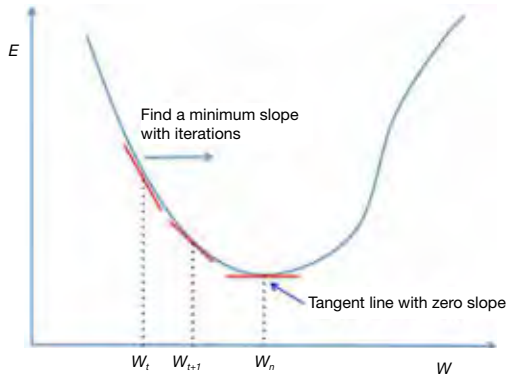


Figure 2 Univariate error function. The derivative of error function is negative at step t, then the next weight w_{t+1} should be greater than w_t in order to find a weight with a slope equal or close to zero.

where l=1,2,3,…,L indexes observations, h=1,2,…,H is the output nodes, and o is the predicted output and y is the observed output. Error function in this form is intuitively understandable. However, the comprehension of mathematical expression of cross entropy is a little more challenging:

$$E_{ce}=\frac{1}{2}\sum_{l=1}^{L}\sum_{h=1}^{H}[y_{lh}\log(o_{lh})+(1-y_{lh})\log(1-o_{lh})] \quad [2]$$

overall, the error function describes the deviation of predicted outcomes from the observed ones. Large deviation suggests a poorly fitted model and the synaptic weights should be adjusted. The algorithm for NN training is backpropagation. At the very beginning, each synaptic weight adopts a random value. This random model leads to a predicted outcome, which is then compared to the observed outcome. The comparison is made using error function. Absolute partial derivatives of the error function with respect to weight ($\partial E/\partial w$) are slopes used to guide us to find a minimum error (e.g., a slope of zero indicates the nadir).

A new weight (w_{t+1}) is calculated based on the present weight (w_t) and the partial derivative.

$$w_k^{(t+1)}=w_k^{(1)}-\eta\frac{\partial E^{(t)}}{\partial w_k^{(t)}} \quad [3]$$

where η is the learning rate, defining the magnitude of weight change in each iteration (3). In traditional backpropagation, the learning rate is fixed, but it can be changed during training process in resilient backpropagation (5,6). Weight update of resilient backpropagation in each iteration is written in the following equation:

$$w_k^{(t+1)}=w_k^{(t)}-\eta_k^{(t)}sign\left(\frac{\partial E^{(t)}}{\partial w_k^{(t)}}\right) \quad [4]$$

where the learning rate can be changed during training process according to the sign of the partial derivative. The learning rate η_k is increased when the partial derivative keeps its sign. In contrast, if partial derivative of the error function changes its sign, η_k should be decreased. That is because the changing sign suggests the optimal weight is jumped over. *Figure 2* is a univariate error function used for illustration. The derivative of error function is negative at step t, then the next weight w_{t+1} should be greater than w_t in order to find a weight with a slope equal or close to zero. In reality, because the learning rate is a value greater than 0, the nadir cannot be exactly arrived at but can be approached. Therefore, a threshold should be defined for convergence. By default, the neuralnet() function uses 0.01 as the threshold for partial derivative of error function to stop iteration.

The traditional backpropagation can be performed by assigning "backprop" to the argument "algorithm". Resilient backpropagation with and without weight backtracking can be specified by "rprop+" and "rprop−", respectively. Here, weight backtracking is to undo the last iteration and add a smaller value to the weight in the next step. The aim of the technique is to accelerate convergence (1).

Generalized weights

While NN have been proven to have good predictive power as compare to traditional models, its interpretability remains difficult. Interpretability of NN model requires understanding of the independent effect of each individual predictor on the prediction of the model. In 2001, Intrator and coworkers developed the concept of generalized weights for the interpretation of NN (7). The generalized weights are mathematically written as:

$$\tilde{w}_i=\frac{\partial\log\left[\frac{o(x)}{1-o(x)}\right]}{\partial x_i} \quad [5]$$

where i is the index for each covariate, $o(x)$ is the predicted outcome probability by covariate vector. Log-odds is the link function for logistic regression model. The partial derivative of the log-odds function with respect to covariate

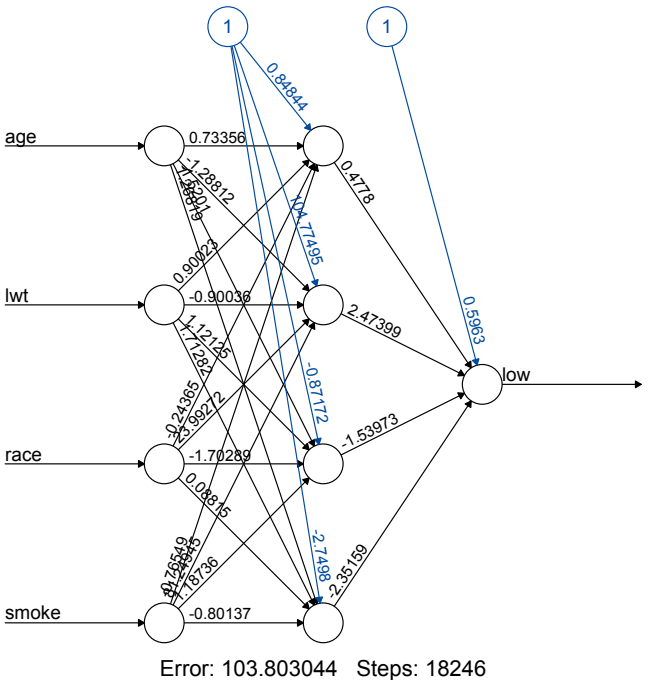


Figure 3 Newly fitted neural network with four input variables. The number of hidden neurons is increased to four. The error of the neural network is reduced to 104, but it requires 18,246 steps for convergence.

of interest is the coefficient for that covariate. However, if there are non-linear terms for the covariate, generalized weights for that covariate vary greatly over the entire covariate pattern. For linear terms as that in conventional logistic regression model, the generalized weights of a covariate are concentrated at one value. Generalized weights for all observations can be visited in "generalized. weights" element in the nn class object returned by the neuralnet() function. Graphical visualization of generalized weights is another way to examine the relative contribution of each covariate. The next NN model reduces the number of covariates and increases the number of hidden units.

```
> nn.limited <- neuralnet(
      low~age+lwt+race+smoke,
      data=birthwt, hidden=4, err.fct="ce",
      linear.output=FALSE)
> plot(nn.limited)
```

As you can see from *Figure 3*, there are four input nodes and four hidden units in the new NN model. With greater degree of freedom, the model reduces the error to 103. However, it requires 18,246 steps for convergence. Complexity of a model is measured by the degree of freedom in conventional models, and increasing complexity carries the risk of model over-fitting (8).

```
> par(mfrow=c(2,2))
> gwplot(nn.limited,selected.covariate="age")
> gwplot(nn.limited,selected.covariate="lwt")
> gwplot(nn.limited,selected.covariate="race")
> gwplot(nn.limited,selected.covariate="smoke")
```

The par() function is used to set graphical parameters. After setting mfrow=c(2,2), subsequent figures will be drawn in a 2-by-2 array on the device by rows (mfrow). The gwplot() graphical function is contained in the *neuralnet* package and it plots general weights against individual covariates (*Figure 4*). The variances of generalized weights for covariates *race* and *smoke* appear large, indicating non-linearity of their effects. If generalized weights of a covariate gather around zero, the covariate has no effect on outcome status.

Prediction with neural network (NN) model

While generalized weights help interpretation of the NN model allowing examination of independent effect of covariate, another purpose of NN is to assist prediction of future observations. The model is trained with both input and output signals, which is known as supervised learning in machine learning terminology. For predictions, only input signals are known and they are used to predict outcomes. In our example, suppose we have four mothers with known feature variables. The probability of having a baby with low birth weight can be calculated with compute() function.

```
> new.mother<-matrix(c(23,105,3,1,26,111,2,0,31,125,2,
1,35,136,1,0),byrow=TRUE,ncol=4)
> new.mother
      [,1]  [,2]  [,3]  [,4]
[1,]   23   105     3     1
[2,]   26   111     2     0
[3,]   31   125     2     1
[4,]   35   136     1     0
> pred<-compute(nn.limited,new.mother)
> pred$net.result
```

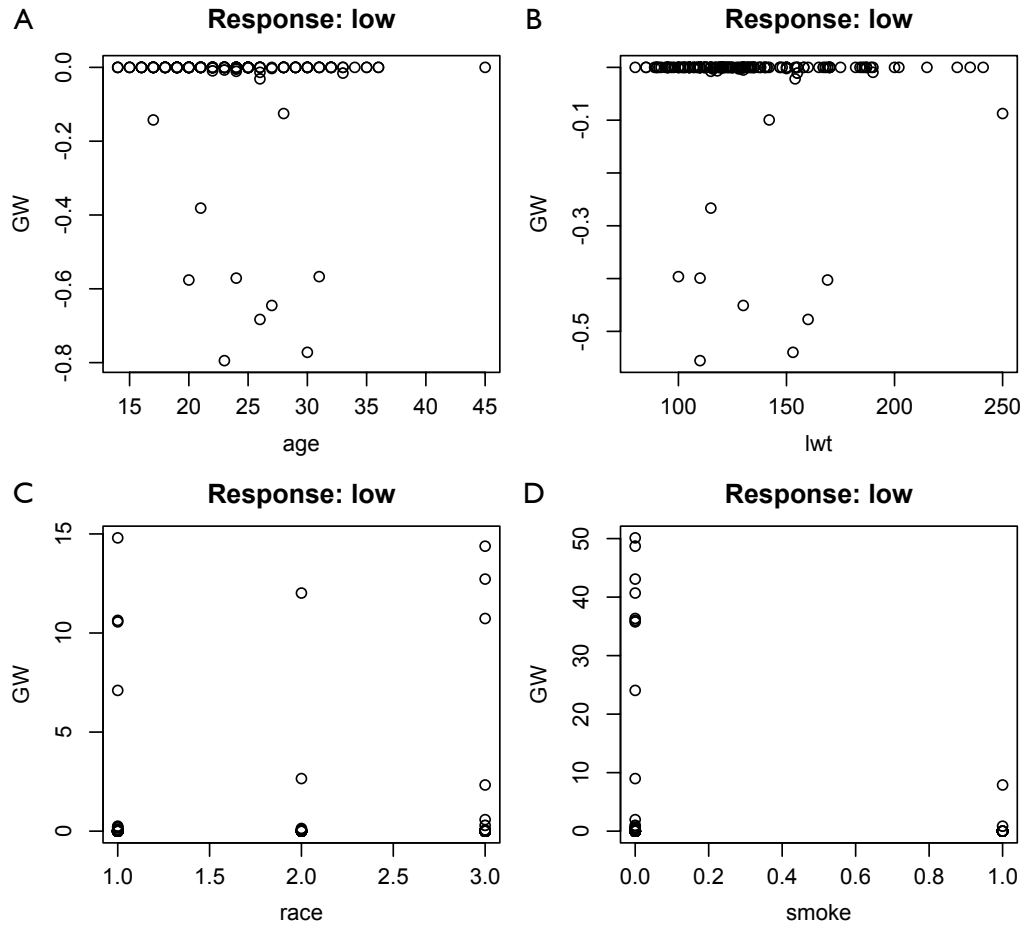



Figure 4 General weights for each covariate. The variances of generalized weights for covariates *race* and *smoke* appear large, indicating non-linearity of their effects.

```
          [,1]
[1,] 0.41502398595
[2,] 0.41502398353
[3,] 0.41502398595
[4,] 0.05640053409
```

In the above syntax, features of new mothers are stored in a matrix. Note that the order of feature variables should be the same as that in the training data frame. The first argument of the compute() function is the "nn" class object returned by neuralnet(). Next the feature matrix is passed to the function. A list of neurons' output of each layer and the net results of the neural network are returned by the compute() function. Typically, investigators are interested in the final result of the network. As shown in our example, the result shows the probability of having a low birth weight baby for each mother. The first three mothers have 41% probability, and the last one has 5% probability of having a low birth weight baby.

Summary

The article provides further insights into the structure of NN, covering concepts of activation function, error function, learning rate and generalized weights. NN topology can be visualized with the generic plot() function by passing a "nn" class object. Generalized weights assist interpretation of NN model with respect to the independent effect of individual input variables. A large variance of generalized weights for a covariate indicates non-linearity of its independent effect. If generalized weights of a covariate are approximately zero, the covariate is considered to have no effect on the outcome. Finally,

prediction of new observations can be performed using the compute() function. Make sure that the feature variables passed to the compute() function are in the same order as that in the training neural network.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Gunther F, Fritsch S. neuralnet: Training of neural networks. R J 2010;2010:30-38.
2. López-gonzález E. Data analysis from the Generalized Linear Model approach: An application using R | Análisis de datos con el Modelo Lineal Generalizado. Una aplicación con R. Revista Espanola de Pedagogia 2011;69:59-80.
3. Haykin SO. Neural Networks and Learning Machines (3rd Edition). Prentice Hall, 2008.
4. Engel I, Bershad NJ. A transient learning comparison of Rosenblatt, backpropagation, and LMS algorithms for a single-layer perceptron for system identification. IEEE Transactions on Signal Processing 1994;42:1247-1251.
5. Riedmiller M. Advanced Supervised Learning in Multi-layer Perceptrons - From Backpropagation to Adaptive Learning Algorithms. Computer Standards & Interfaces 1994;16:265-278.
6. Riedmiller M, Braun H. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In IEEE International Conference on Ural Networks 1993:586-91.
7. Intrator O, Intrator N. Interpreting neural-network results: A simulation study. Computational Statistics & Data Analysis 1999;37:373-393.
8. Zhang Z. Too much covariates in a multivariable model may cause the problem of overfitting. J Thorac Dis 2014;6:E196-E197.

Cite this article as: Zhang Z. Neural networks: further insights into error function, generalized weights and others. Ann Transl Med 2016;4(16):300. doi: 10.21037/atm.2016.05.37

Hierarchical cluster analysis in clinical research with heterogeneous study population: highlighting its visualization with R

Zhongheng Zhang, Fionn Murtagh, Sven Van Poucke, Su Lin, Peng Lan

Abstract: Big data clinical research typically involves thousands of patients and there are numerous variables available. Conventionally, these variables can be handled by multivariable regression modeling. In this article, the hierarchical cluster analysis (HCA) is introduced. This method is used to explore similarity between observations and/or clusters. The result can be visualized using heat maps and dendrograms. Sometimes, it would be interesting to add scatter plot and smooth lines to the panels of the heat map. The inherent R *heatmap* package does not provide this function. A series of scatter plots can be created using *lattice* package, and then background color of each panel is mapped to the regression coefficient by using custom-made panel functions. This is the unique feature of the *lattice* package. Dendrograms and color keys can be added as the legend elements of the lattice system. The *latticeExtra* package provides some useful functions for the work.

Keywords: Hierarchical cluster analysis (HCA); dendrogram; clinical research; heat map

Submitted Sep 19, 2016. Accepted for publication Jan 18, 2017.
doi: 10.21037/atm.2017.02.05
View this article at: http://dx.doi.org/10.21037/atm.2017.02.05

Introduction

Hierarchical cluster analysis (HCA), also known as hierarchical clustering, is a popular method for cluster analysis in big data research and data mining aiming to establish a hierarchy of clusters (1-3). As such, HCA attempts to group subjects with similar features into clusters. There are two types of strategies used in HCA: the agglomerative and the divisive strategy. With agglomerative clustering directing from "the leaves" to "the root" of a cluster tree, the approach is called a "bottom up" approach (4). Divisive clustering is considered a "top down" approach directing from the root to the leaves. All observations are initially considered as one cluster, and then splits are performed recursively as one moves down the hierarchy.

Clinical research is usually characterized by heterogeneous patient populations despite the use of long list of inclusion/exclusion criteria (5,6). For instance, sepsis and/or septic shock are typically treated as a disease entity in clinical trials. However, there are significant heterogeneities in patients with sepsis with respect to infection sites, coexisting comorbidities, inflammatory responses and timing of treatment (7,8). Traditionally, these factors are considered as confounding factors and can be addressed by multivariable regression modeling (9). However, such a method primarily focuses on prediction and adjustment, and fails to classify a mixed population into a more homogeneous one. Clustering analysis aims to classify mixed population into more homogenous groups based on available features. Each cluster has its own signature for identification (10,11). For instance, investigators may be interested in how physiological signals predict differently on the occurrence of subacute events (e.g., sepsis, hemorrhage and intubation) in intensive care unit (ICU). For instance, physiological signatures of hemorrhage were found to be similar in patients from surgical and medical ICU, indicating similarity between these two subgroups (12). In this article, we aim to provide some basic knowledge on the use of HCA and its visualization by dendrograms and heat maps.

Understanding HCA

Suppose the data consist of four observations (x1 to x4) and each contains two feature variables (a, b).

```
> df<-matrix(c(1,2,4,3,2,1,7,9),nrow=4)
```

Table 1 Methods to calculate distance between two observations

Names	Equations
Euclidean distance	$\left\|x_i - x_j\right\|_2 = \sqrt{\left(a_i - a_j\right)^2 + \left(b_i - b_j\right)^2}$
Manhattan distance	$\left\|x_i - x_j\right\|_1 = \left\|a_i - a_j\right\| + \left\|b_i - b_j\right\|$
Maximum distance	$\left\|x_i - x_j\right\|_\infty = \max\left\{\left\|a_i - a_j\right\|, \left\|b_i - b_j\right\|\right\}$
Mahalanobis distance	$\sqrt{\left(x_i - x_j\right)^T S^{-1} \left(x_i - x_j\right)}$, where S is the covariance matrix and x_i and x_j are variable vectors of x_i and x_j

x_i and x_j are ith and jth observations, where i and j are indices. a and b are feature variables.

```
> rownames(df)<-c("x1","x2","x3","x4")
> colnames(df)<-c("a","b")
> df
   a b
x1 1 2
x2 2 1
x3 4 7
x4 3 9
```

The matrix df is consistent with the output of a case report form (CRF) where each row represents an observation, and a column represents a variable (features). To facilitate a clear understanding, we assigned two-dimensional features to the observations. During the first step, the distances between the observations are calculated.

```
> dist(df)
          x1       x2       x3
x2 1.414214
x3 5.830952 6.324555
x4 7.280110 8.062258 2.236068
```

The dist() function calculates the distance between each pair of observations. There is a variety of methods to calculate the distance (*Table 1*) (13,14). By default, dist() function uses Euclidean distance, and this can be modified using the *method* argument. Next, Euclidean distance is checked between x2 and x3:

$$\left\|x_2 - x_3\right\|_2 = \sqrt{\left(a_2 - a_3\right)^2 + \left(b_2 - b_3\right)^2} = \sqrt{\left(2-4\right)^2 + \left(1-7\right)^2} = 6.32, \quad [1]$$

which is exactly the value displayed in the above tabular output.

From the dist() output table, it appears that x1 and x2 are the closest to each other and they are merged at the first step, leaving clusters {x1, x2}, {x3} and {x4} to be merged further. At each step, clusters/observations with the shortest distance are merged. Distance between clusters should be defined. Additionally, there are different methods (also called linkage criteria) to define the distance between two clusters (*Table 2*). The default method in hclust() function is the complete linkage clustering, in which the distance between two clusters is the distance between those two elements (one in each cluster) that are farthest away from each other (15). The minimum distance between the remaining set of observations/clusters was the one between x3 and x4 (d=2.24). The distances between other pairs of observations/clusters are: d({x1, x2}, x3) is 6.32, d({x1, x2}, x4) is 8.06. After the combination of x3 and x4, there are only two clusters {x1, x2} and {x3, x4}, which are merged as the latest. The results can be visualized with generic plot() function.

```
> plot(hclust(dist(df)))
```

The height axis displays the distance between observations and/or clusters (*Figure 1*). The horizontal bars indicate the point at which two clusters/observations are merged. For example, x1 and x2 are merged at a distance of 1.41, which is the minimum distance among all other distances. Observations x3 and x4 are merged at the value of 2.24. Finally, {x1, x2} and {x3, x4} are merged at a distance of 8.06. This easy example has illustrated the basic principles underlying HCA.

Worked example

To illustrate how to perform HCA using R, we simulated a worked example. In the example, there are five variables

Table 2 Methods to calculate distance between two clusters

Names	Equations
Maximum (complete linkage clustering)	$Max\{d(a,b), a \in A, b \in B\}$
Minimum (single linkage clustering)	$Min\{d(a,b), a \in A, b \in B\}$
Mean linkage clustering	$\frac{1}{\|A\|\|B\|}\sum_{a\in A}\sum_{b\in B} d(a,b)$
Centroid linkage clustering	$\|c_s - c_t\|$ where c_s and c_t are the centroids of clusters s and t, respectively.

a and b are elements belonging to clusters A and B, respectively.

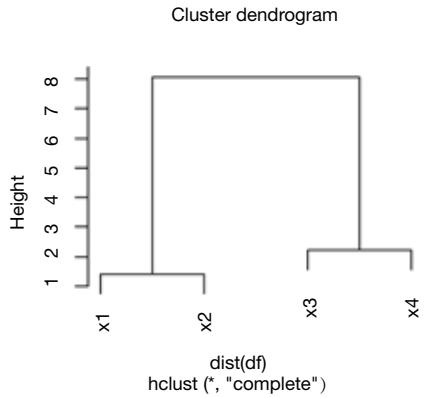


Figure 1 A simple cluster dendrogram. The height axis displays the distance between observations and/or clusters. The horizontal bars indicate the point at which two clusters/observations are merged. For example, x1 and x2 are merged at a distance of 1.41, which is the minimum one among all other distances. Also, x3 and x4 are merged at the value of 2.24. Finally, {x1, x2} and {x3, x4} are merged and their distance is 8.06.

(x1 to x5) represented by columns. Each row represents a patient. There is a factor variable named "diag" to categorize patients into different subgroups.

```
> nvar=5
> data<-data.frame(diag=factor(rep(c("Sepsis",
"AECOPD","Surgery","MODS","Poisoning"),50)))
> for (i in 1:nvar) {
data[[paste("x",i,sep="")]]<-rnorm(250)
}
> attach(data)
> data$y<-3*x1+2*x2-2*x3+x3^2-x4+x5^3-2*x5
> detach()
```

In real clinical research, the variables x1 to x5 can be any continuous variable such as blood pressure, heart rate, temperature and laboratory measurements. They are centered by mean and scaled by standard deviation, resulting in a normal distribution. The variable y can be an outcome variable such as cost, length of stay in ICU and hospital. If the outcome variable is binary, transformation to an appropriate scale is required, e.g., the logit scale.

Statistical quantity

A variety of statistical quantities can be explored. In its original design, HCA analyzes at individual level. Each patient takes one row and each column represents one feature variable. Such analysis provides information on the similarity between individual patients. However, in big data mining, typically thousands of patients are involved and it is more feasible to explore features in subgroups. Summary statistics such as median, mean, variance, correlation and regression coefficients can be explored. In the present example, suppose we are interested in the regression coefficient of each feature variable for the outcome y. We do not attempt to adjust these models. As a result, we need to fit regression models for each combination of feature variables and subgroups (5×5=25). Fitting these models one by one would be time-consuming and error-prone, therefore an R syntax is needed which is able to repeat the same regression model function. In R, it is not wise to use loop functions, instead the lapply() can apply a user-defined function across variables. Let's see how it works.

```
> library(lme4)
> coeff<-lapply(data[,2:6],function(x) {
```



```
coef(lmList(y~x|diag,data=data.frame(x=x,
y=data$y,diag=data$diag)))[2]
})
```

While the lapply() repeats the regression function across variables x1 to x5, lmList() is employed to perform regression analysis across subgroups (16). Note that the first formula argument of lmList() allows a grouping factor specifying the partitioning of the data according to which different lm fits will be performed. The *data* argument specifies the data frame containing the variables named in the formula. Here we vary the *data* argument in each cycle, ensuring each *lm* fit employs different feature variables. The index [2] extracts regression coefficient of *lm* models. Because the lapply() function returns a list, we need to transform it into a data frame for further analysis.

```
> coefficient<-t(as.data.frame(coeff))
> varlist<-names(data[,2:6])
> row.names(coefficient)<-varlist
```

Also, the t() function is used to transpose the data frame, making the rows represent variables and the columns represent the subgroups. Next, we rename the row names by using x1 to x5.

Heat map

A heat map is a graphical representation of data where the individual values contained in a matrix are represented as colors (17). The orders of columns and rows are reordered to facilitate better presentation of dendrograms. Dendrograms are used to describe the similarity between clusters and/or observations. There are a variety of heat map packages in R. heatmap() is a base function shipped with R installation. Other heat map packages include *d3heatmap* to create interactive heat maps, *fheatmap* to plot high quality, elegant heat map using 'ggplot2' graphics, *heatmap.plus* to allow non-identical X- and Y-dimensions, *heatmap3* to provide more powerful and convenient features, and *pheatmap* to offer more control over dimensions and appearance. In this case we use heatmap.2() function contained in *gplots* package. It provides good control over annotations and labels, and also draws a color key to map data values to colors.

```
> library("gplots")
> heatmap.2(coefficient,
ColSideColors=rainbow(ncol(coefficient)),
RowSideColors=rainbow(nrow(coefficient)),srtCol=45)
```

The heatmap.2() function first takes a numeric matrix of the values to be plotted. The method used to calculate distance can be specified using *distfun* for distance (dissimilarity) between both rows and columns, and *hclustfun* for computing the hierarchical clustering. Suppose one attempts to use "minkowski" method for distance calculation and "mcquitty" method for computing clustering, the following code can do the task:

```
> heatmap.2(coefficient,
hclust(dist(coefficient,method="minkowski"),
method="mcquitty"),
ColSideColors=rainbow(ncol(coefficient)),
RowSideColors=rainbow(nrow(coefficient)),
srtCol=45)
```

ColSideColors argument takes a character vector of length ncol(x) containing the color names for a horizontal side bar that may be used to annotate the columns of x. Here we used the rainbow color style for annotating the columns. *RowSideColors* is used for vertical side bar with the same usage as that of *ColSideColors*. *srtCol* argument is used to control the angle of column labels in degrees from horizontal. The result is shown in *Figure 2*. As indicated by the color key, more negative values are represented by more dark red color and positive values are represented by light yellow. The histogram shows the number of values in each color strip. The dendrogram shows the dissimilarity between columns and rows. The results show that surgery and poisoning patients are the most similar subgroups. Variable x3 is negatively correlated with y across subgroups and x1 is positively correlated with y.

Add scatter plot to the heat map

To better illustrate how variables correlate with outcome y, it would be interesting to visualize scatter plots in heat map. However, the aforementioned heat map packages do not provide this function. One approach is to draw each scatter plot within a panel with the *lattice*

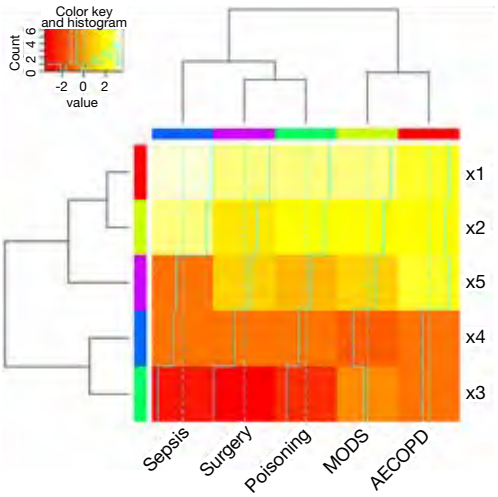


Figure 2 Heat map produced by heatmap.2() function with color key. The light blue solid lines in the heat map correspond to the value of coefficient. The dashed lines were the reference value zero. There is a histogram in the color key showing the number of coefficient values within each color bar (i.e., one color bar represents a range of coefficient values). The orders of rows and columns are rearranged to avoid intersecting of dendrogram lines. It appears that surgery and poisoning patients are the most similar subgroups.

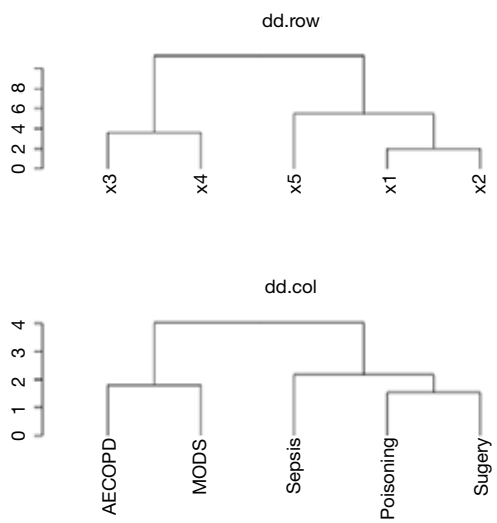


Figure 3 Dendrograms for rows and columns of the *coefficient* data frame. Note that dd.row corresponds to the variables x1 to x5, and dd.col corresponds to the subgroups.

package (18). Then background of each panel is filled with colors corresponding to coefficient values. Also, the dendrogram can be passed to the legend of xyplot() function using *dendrogramGrob* function (19). The order of lattice panels should be rearranged to the order that is consistent with HCA. Finally, the strip labels can be moved to the left and top of the plot. Next, let's take a close look at how each step is carried out and readers can adapt these codes into their own needs.

Firstly, the data frame needs to be reshaped to be utilized by the xyplot() function. The *reshape2* package can do this task perfectly. Because we change the wide format to long format, only the melt() function is used.

```
> library(lattice)
> library(reshape2)
> m.data<-melt(data, id.vars=c("diag", "y"))
> head(m.data)
```

	diag	y	variable	value
1	Sepsis	2.8858013	x1	-0.3937166
2	AECOPD	3.3040682	x1	1.3065381
3	Surgery	-1.2972116	x1	-0.5979787
4	MODS	-0.2088301	x1	0.1037778
5	Poisoning	-0.1218218	x1	0.0134908
6	Sepsis	10.9084870	x1	-0.3384164

Note that the variables x1 to x5 disappear and their values are stacked on the value column. A new variable named "variable" is created to denote the original variable names from x1 to x5. ID variables *diag* and *y* remain unchanged.

Next, we define the order of rows and columns according to the HCA. With the help of the dist() and hclust() function, this task can be easily done with several lines of code. Furthermore, the users can visualize the dendrograms and compare them with the results produced by the heatmap.2() function.

```
> dd.row <- as.dendrogram(hclust(dist(coefficient)))
> row.ord <- order.dendrogram(dd.row)
> dd.col <- as.dendrogram(hclust(dist(t(coefficient))))
> col.ord <- order.dendrogram(dd.col)
> par(mfrow=c(2,1))
> plot(dd.row)
> plot(dd.col)
```

Note that the dd.row and row.order correspond to the

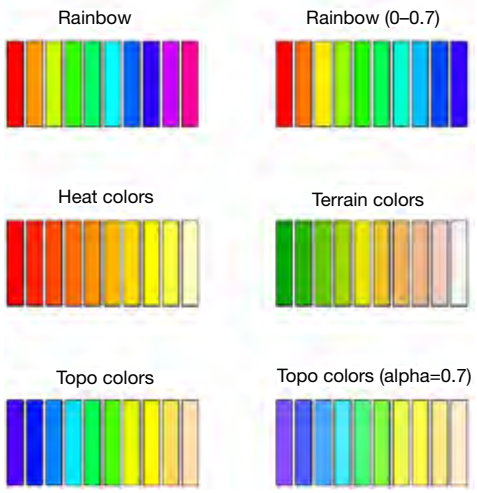


Figure 4 Illustration of how palette works in R, by varying color styles and relevant parameters such as range of hue and alpha.

variables x1 to x5, and the dd.col and col.ord correspond to the subgroups (*Figure 3*). This is important for reordering rows and columns of the *coefficient* data frame. The next code reorders the rows and columns according to the HCA order. The data frame *coeff.order* is then rescaled to ensure that its values are integers. Such values can help to map themselves to colors as defined by palette.

```
> coeff.order<-coefficient[row.ord,col.ord]
> scale.coef<-as.vector(round((coeff.order
-min(coefficient))*10+1))
```

One attractive feature of heat map is the use of colors to highlight differences between individual elements. Therefore, the color style is important to make the heat map attractive and informative. In R colors can be represented by indexing into palette, color name and hex constant. When a *col* argument is assigned a vector of numeric indices, each numeric value represents one color in the vector of colors defined by the palette. To help readers better understand how palette works in R, a series of simple examples are used by varying color styles and some relevant arguments.

```
> par(mfrow=c(3,2))
> palette(rainbow(10))
> barplot(rep(1,10), yaxt="n",main="rainbow", col=1:10)
> palette(rainbow(10,start=0,end=0.7))
> barplot(rep(1,10), yaxt="n",main="rainbow (0-0.7)",
col=1:10)
>  palette(heat.colors(10))
>  barplot(rep(1,10), yaxt="n",main="heat colors",
col=1:10)
>  palette(terrain.colors(10))
>  barplot(rep(1,10), yaxt="n",main="terrain colors",
col=1:10)
>  palette(topo.colors(10))
>  barplot(rep(1,10), yaxt="n",main="topo colors",
col=1:10)
>  palette(topo.colors(10,alpha=0.7))
>  barplot(rep(1,10), yaxt="n",
main="topo colors (alpha=0.7)", col=1:10)
```

The output is shown in *Figure 4*. There are a variety of color styles to select. In the figure we show rainbow, heat, terrain and topo colors. The *start* and *end* arguments are used to define the range of hue. The *alpha* argument takes a number in between 0 and 1 to specify the transparency. With the understanding of palette colors, we proceed to define the palette for our heat map.

```
> palette(rainbow(round((max(coefficient)-
min(coefficient))*10)+1,start=0,end=0.7))
```

In the example, rainbow color is added to the palette. The number of colors is determined by the range of coefficients. A numeral 1 is added to make sure that the minimum value refers to the first color in the palette. At this stage, it is well prepared to draw a heat map with scatter plot in each panel. We need another package called "latticeExtra" which provides several new high-level functions and methods, as well as additional utilities such as panel and axis annotation functions. *Figure 5* is produced by the following codes.

```
> library(latticeExtra)
> plot<-xyplot(y~value|variable+diag,data=m.data,
  par.strip.text = list(cex = 0.6),
  key=list(space="left",
    lines=list(col=seq(1,round((max(coefficient)-
    min(coefficient))*10)+1,4),lwd=4,size=1),
```

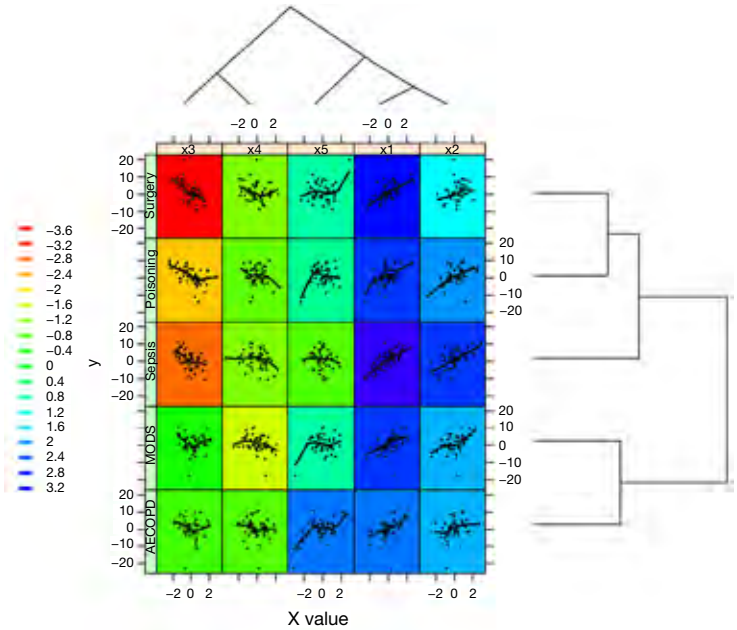



Figure 5 Heat map produced by xyplot() function, with background color of each panel mapping to coefficient values. For instance, the regression coefficient of x3 is −3.57 in the subgroup Surgery, thus the background color of the first panel (x3 and Surgery) is red. One can check the link between colors and values on the left legend.

```
  text=list(as.character(round((seq(1,round((
  max(coefficient)-min(coefficient))*10)+1,4)-
  1)/10+min(coefficient),1)))
  ),
legend =
  list(right =
    list(fun = dendrogramGrob,
    args =
    list(x = dd.col, ord = col.ord,
    side = "right",
    size = 10)),
  top =
    list(fun = dendrogramGrob,
    args =
    list(x = dd.row,
    side = "top",
    type = "triangle"))),
mycolors =scale.coef,
panel = function(x, y,col,mycolors) {
panel.fill(col=mycolors[panel.number()])
  panel.xyplot(x, y,cex=0.2,col="black")
  panel.loess(x, y, col="black",lwd=2)
},
index.cond=list(row.ord,col.ord),
xlab="x value"
)
> useOuterStrips(plot)
```

The first argument of xyplot() is a formula indicating that plots of y (on the y-axis) versus value (on the x-axis) will be produced conditioned on variables *variable* and *diag*. Remember that *variable* and *diag* are factor variables indicting x variables and subgroups, respectively. This formula produces one panel for each unique combination of these two factor variables. The *data* argument passes a data frame containing values for any variables in the formula. Here, m.data contains all variables specified in the formula. The size of strip text can be controlled with *par*: *strip.text* argument. *Key* takes a list that defines a legend to be drawn on the plot. In the example, we want the key to show the corresponding coefficient values of colors, and

this key is displayed on the left. The key is composed of lines and texts, where each line has a color and each text represents the coefficient value corresponding to the line color in the same row. The *legend* argument allows the use of arbitrary "grob"s (grid objects) as legends. Here we use the dendrogramGrob function to create a grob (a grid graphics object) that can be manipulated as such. The first argument of dendrogramGrob should be an object of dendrogram. Recall that *dd.col* corresponds to the subgroups and thus we assign it to the right argument. *Ord* argument takes *col.ord*. By default, dendrogram is displayed in that a child node is joined to its parent as a "stair" with two lines ("rectangle"). If one wants to join child node to parent directly with a straight line, the type should be assigned "triangle", as we have done for top dendrogram. A panel function is defined to allow for customized output. In the example, we need to display lowess smooth lines, scatter points and background in each panel. Of note, the background of each panel is different, which is determined by the coefficient values. Here, panel.number() is used to extract corresponding index number of mycolors. Each numeric value of the vector mycolors refers to a color in the palette that has been defined previously. By default, the xyplot() function ranges panels alphabetically by levels of each conditioning variable. In order to avoid lines of dendrograms intersecting with each other, we need to reorder the panels. This is done by the use of *index.cond* argument. In our example, the index.cond is a list. It is as long as the number of conditioning variables, and the i-th component is a valid indexing vector for levels(g_i), where g_i is the i-th conditioning variable in the plot. The second component of of *index.cond* list is *col.ord* which corresponds to the second conditioning variable *diag*.

```
> row.ord
[1] 3 4 5 1 2
```

As shown above, the order of row.ord is {3, 4, 5, 1, 2}, which is consistent with the order of x variables in *Figure 5* {x3, x4, x5, x1, x2}. The last line uses *useOuterStrips* function from the *latticeExtra* package, which moves strips to the top and left boundaries when printed, instead of in every panel as usual. When there are two conditioning variables, it seems redundant to display strips in every panel.

With binary outcome variable

In clinical research, there are more situations when researchers have to deal with binary outcome variables such as the occurrence of an event of interest, and mortality. As such, we can display probability of outcome in the vertical axis and the values of x on the horizontal axis. Here we created a new binary variable y.bin

```
> data$y.bin = 1/(1+exp(-data$y)) > 0.5
```

Again we need to extract coefficients of logistic regression models for every unique combination of subgroups and diagnosis. Here, a string "binomial" indicating the error distribution is assigned to the family argument, and the link function is "logit". This is the standard argument for logistic regression model. Coefficient obtained in this way has no direct clinical relevance, but its exponentiation gives the odds ratio.

```
> coeff.bin<-lapply(data[,2:6],
  function(x) {
    coef(lmList(y~x|diag,
    family=binomial(link="logit"),
    data=data.frame(x=x,y=data$y.bin,
    diag=data$diag)))[2]
})
> coeff.bin<-as.matrix(as.data.frame(coeff.bin))
> colnames(coeff.bin)<-varlist
> heatmap.2(coeff.bin,
    ColSideColors=rainbow(ncol(coeff.bin)),
    RowSideColors=rainbow(nrow(coeff.bin)),
    srtRow=45)
```

Here, we created a heat map with similar argument to that displayed in *Figure 2*, except that values are coefficients estimated from logistic regression models.

```
> models<-lapply(data[,2:6],
function(x) {
lmList(y~x|diag,
family=binomial(link="logit"),
data=data.frame(x=x,y=data$y.bin,diag=data$diag))
})
```

Then, the predicted probability can be estimated using predict.lmList() function. The function returns a vector whose order should be given more attention. If one intends to build also the confidence interval, the se.fit argument should be "TRUE" to allow estimation of standard error for each point estimate.

```
> prob<-as.vector(NULL)
> for (i in 1:5) {
prob<-c(prob,predict(models[[i]]))
}
```

Because the order of melted data.bin and prob are not consistent, we need some lines of code to arrange them. Alternatively, one may save predicted probability and standard error as a data frame, and the order can be different. Users can try it.

```
data.bin<-melt(data[,-7], id.vars=c("diag", "y.bin"))
> diaglist<-data[1:5,]$diag
> data.sort<-data.bin[0,]
> for (var in varlist) {
  for (dia in diaglist) {
  data.sort<-rbind(data.sort,
  data.bin[data.bin$variable==var&data.bin$
  diag==dia,])
  }
}
> data.pred<-cbind(data.sort,prob)
```

The following codes are similar to that described in the above example with minor adaptations.

```
> dd.row.bin <- as.dendrogram(hclust(dist(coeff.bin)))
> row.ord.bin <- order.dendrogram(dd.row.bin)
> dd.col.bin <- as.dendrogram(hclust(dist(t(coeff.bin))))
> col.ord.bin <- order.dendrogram(dd.col.bin)

> coeff.order.bin<-coeff.bin[row.ord.bin, col.ord.bin]
> scale.coef.bin<-as.vector(round((t(coeff.order.bin)-
min(coeff.bin))*10+1))
> palette(rainbow(round((max(coeff.bin)-
min(coeff.bin))*10)+1,start=0,end=0.7))
> plot.bin<-xyplot(prob~value|variable+diag,
data=data.pred,
  par.strip.text = list(cex = 0.6),
  key=list(space="left",
    lines=list(col=seq(1,round((max(coeff.bin)-
    min(coeff.bin))*10)+1,2),lwd=4,size=1),
    text=list(as.character(round((seq(1,
    round((max(coeff.bin)-min(coeff.bin))*10)+
    1,2)-1)/10+min(coeff.bin),1)))
    ),
  legend =
    list(right =
      list(fun = dendrogramGrob,
      args =
      list(x = dd.col.bin, ord = col.ord.bin,
      side = "right",
      size = 10)),
    top =
      list(fun = dendrogramGrob,
      args =
      list(x = dd.row.bin,
      side = "top",
      type = "triangle"))),
  colors.bin =scale.coef.bin,
  panel = function(x, y,col,colors.bin) {
  panel.fill(col=colors.bin[panel.number()])
  panel.xyplot(x, y,cex=0.2,col="black")
  panel.loess(x, y, col="black",lwd=1)
  },
  index.cond=list(col.ord.bin, row.ord.bin),
  xlab="x value",
  ylab="probability"
  )
> useOuterStrips(plot.bin)
```

The output is shown in *Figure 6*. This time the vertical axis represents the probability of the binary outcome. The black dot is the predicted probability, and thus each x value corresponds to one probability value.

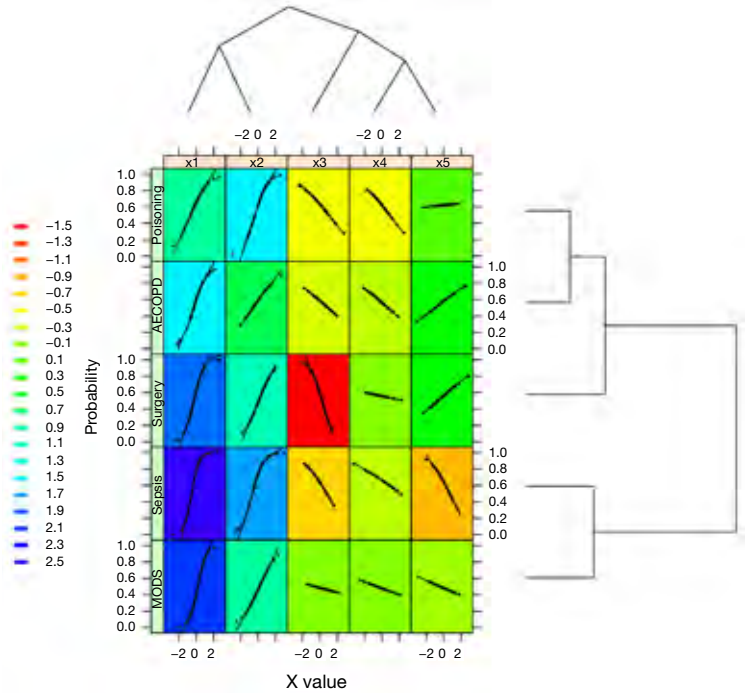


Figure 6 Heat map produced by xyplot() function, with vertical axis representing the estimated probability of outcome events.

Acknowledgements

None.

Footnote

Conflicts of Interest: The authors have no conflicts of interest to declare.

References

1. Muntaner C, Chung H, Benach J, et al. Hierarchical cluster analysis of labour market regulations and population health: a taxonomy of low- and middle-income countries. BMC Public Health 2012;12:286.
2. Petushkova NA, Pyatnitskiy MA, Rudenko VA, et al. Applying of hierarchical clustering to analysis of protein patterns in the human cancer-associated liver. PLoS One 2014;9:e103950.
3. Murtagh F. Hierarchical Clustering. In: Lovric M. editor. International Encyclopedia of Statistical Science. Berlin, Heidelberg: Springer; 2014:633-5.
4. Gil-Garcia RJ, Badia-Contelles JM, Pons-Porrata A. A General Framework for Agglomerative Hierarchical Clustering Algorithms. IEEE 2006:569-72.
5. Iwashyna TJ, Burke JF, Sussman JB, et al. Implications of Heterogeneity of Treatment Effect for Reporting and Analysis of Randomized Trials in Critical Care. Am J Respir Crit Care Med 2015;192:1045-1051.
6. Ruan SY, Lin HH, Huang CT, et al. Exploring the heterogeneity of effects of corticosteroids on acute respiratory distress syndrome: a systematic review and meta-analysis. Crit Care 2014;18:R63.
7. Kalil AC, Florescu DF. Severe sepsis: are PROWESS and PROWESS-SHOCK trials comparable? A clinical and statistical heterogeneity analysis. Crit Care 2013;17:167.
8. Ma PL, Peng XX, Du B, et al. Sources of Heterogeneity in Trials Reporting Hydroxyethyl Starch 130/0.4 or 0.42 Associated Excess Mortality in Septic Patients: A Systematic Review and Meta-regression. Chin Med J (Engl) 2015;128:2374-2382.
9. Zhang Z. Model building strategy for logistic regression: purposeful selection. Ann Transl Med 2016;4:111.
10. Murtagh F, Contreras P. Algorithms for hierarchical clustering: an overview. WIREs Data Mining Knowl Discov 2012;2:86-97.
11. Blasius J. Greenacre M. editors. Visualization and Verbalization of Data. Boca Raton: Chapman and Hall/CRC, 2014:xlii+350.

12. Moss TJ, Lake DE, Calland JF, et al. Signatures of Subacute Potentially Catastrophic Illness in the ICU: Model Development and Validation. Crit Care Med 2016;44:1639-1648.
13. Shahid R, Bertazzon S, Knudtson ML, et al. Comparison of distance measures in spatial analytical modeling for health service planning. BMC Health Serv Res 2009;9:200.
14. Murtagh F. A Survey of Recent Advances in Hierarchical Clustering Algorithms. Comput J 1983;26:354-359.
15. Defays D. An efficient algorithm for a complete link method. Comput J 1977;20:364-366.
16. Bates D, Mächler M, Bolker B, et al. Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software 2015;67:1-48.
17. Toddenroth D, Ganslandt T, Castellanos I, et al. Employing heat maps to mine associations in structured routine care data. Artif Intell Med 2014;60:79-88.
18. Sarkar D. Lattice: Multivariate Data Visualization with R. New York: Springer; 2008:1.
19. Maindonald J, Braun WJ. editors. Data Analysis and Graphics Using R – an Example-Based Approach (Cambridge Series in Statistical and Probabilistic Mathematics). 3rd ed. Cambridge: Cambridge University Press; 2009:538.

Cite this article as: Zhang Z, Murtagh F, Van Poucke S, Lin S, Lan P. Hierarchical cluster analysis in clinical research with heterogeneous study population: highlighting its visualization with R. Ann Transl Med 2017;5(4):75. doi: 10.21037/atm.2017.02.05

Causal mediation analysis in the context of clinical research

Zhongheng Zhang, Cheng Zheng, Chanmin Kim, Sven Van Poucke, Su Lin, Peng Lan

Abstract: Clinical researches usually collect numerous intermediate variables besides treatment and outcome. These variables are often incorrectly treated as confounding factors and are thus controlled using a variety of multivariable regression models depending on the types of outcome variable. However, these methods fail to disentangle underlying mediating processes. Causal mediation analysis (CMA) is a method to dissect total effect of a treatment into direct and indirect effect. The indirect effect is transmitted via mediator to the outcome. The mediation package is designed to perform CMA under the assumption of sequential ignorability. It reports average causal mediation effect (ACME), average direct effect (ADE) and total effect. Also, the package provides visualization tool for these estimated effects. Sensitivity analysis is designed to examine whether the results are robust to the violation of the sequential ignorability assumption since the assumption has been criticized to be too strong to be satisfied in research practice.

Keywords: Causal mediation analysis (CMA); mediator; clinical research; sensitivity analysis

Submitted Aug 02, 2016. Accepted for publication Sep 25, 2016.
doi: 10.21037/atm.2016.11.11
View this article at: http://dx.doi.org/10.21037/atm.2016.11.11

Introduction

In clinical research, a large number of variables may be collected and the relationship between each variable can be complex. The traditional method to explore the relationships between explanatory variables and outcomes is by using multivariable regression model. However, this method fails to disentangle mechanisms underlying the association between explanatory variable and the outcome, while investigators are sometimes interested in such underlying mediating pathophysiological processes (1,2). In these situations, the investigators may have prior knowledge that an explanatory variable exerts its effect on outcome via direct and indirect pathways. In the indirect pathway, there is a mediator that transmits the causal effect. For example, suppose that corticosteroids are effective for reducing mortality rate of patients with acute respiratory distress syndrome (ARDS), and this effect is partly mediated by suppressing inflammatory response (3). In this scenario, corticosteroid treatment is an exposure, mortality is a binary outcome and inflammatory response is a mediator. It is interesting to explore how much of the total effect is transmitted via the inflammatory response. The example motivates the causal mediation analysis (CMA). In this article, we will introduce a commonly used method for CMA proposed by Imai *et al*. (4). Its implementation with R package will be described in a step-by-step approach.

Understanding CMA

Suppose we have variables X and Y indicating the explanatory variable (or treatment variable) and the outcome variable, respectively (*Figure 1*). Mediation in its simplest form is to add a mediator M between X and Y (5). Model-based mediation analysis is implemented in the *mediation* package (6). The sequential ignorability assumption should be satisfied for validity of the method used for this package. The assumption states that "*the treatment (explanatory variable X) is first assumed to be ignorable given the pre-treatment covariates, and then the mediator variable (M) is assumed to be ignorable given the observed value of the treatment as well as the pre-treatment covariates.*" (7). The first part is often satisfied by randomization while the second part implies that there are no unmeasured confounding between mediator and outcome, which is often questionable even in randomized trials. Moreover, this assumption is

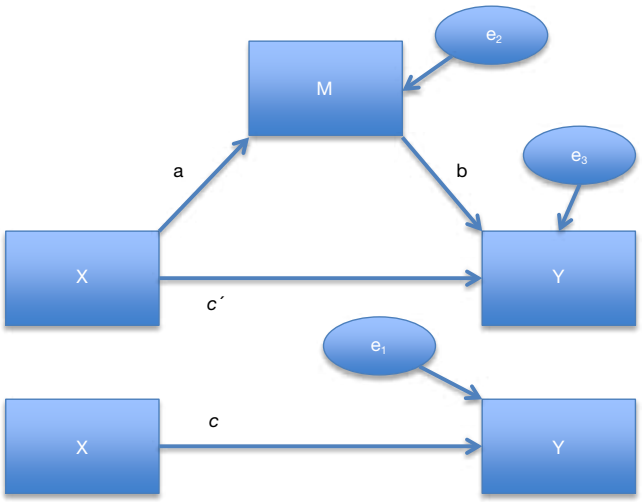


Figure 1 Causal mediation analysis in its simplest form. X, Y and M are treatment, outcome and mediator variables. c is coefficient linking X and Y (total causal effect), c' is the coefficient for the effect of X on Y adjusting for M (direct effect), b is the effect of M on Y adjusting for explanatory variable, a is the coefficient relating to the effect of X on M. e_1, e_2 and e_3 are residuals

not verifiable in practice and sensitivity analysis should be performed to quantify the degree to which violation of the assumption would change the results. Sensitivity analysis will be described in detail in following sections. This section focuses on the model based mediation analysis with continuous outcome, which is based on three linear equations:

$$Y = i_1 + cX + e_1 \quad [1]$$

$$Y = i_2 + c'X + bM + e_2 \quad [2]$$

$$M = i_3 + aX + e_3 \quad [3]$$

where i_1, i_2 and i_3 denote intercepts, Y is the outcome variable, X is the explanatory variable, M is the mediator, c is the coefficient linking X and Y (total causal effect), c' is the coefficient for the effect of X on Y adjusting for M (direct effect), b is the effect of M on Y adjusting for explanatory variable, a is the coefficient relating to the effect of X on M. e_1, e_2 and e_3 are residuals (7) that are uncorrelated with the variables in the right hand side of the equation and are independent to each other. Under this specific model, the causal mediation effect (CME) is represented by the product coefficient of ab. Of note, Eq. [3] can be substituted into Eq. [2] to eliminate the term M:

$$Y = i_2 + bi_3 + (c' + ab)X + e_2 + be_3 \quad [4]$$

It appears that the parameters related to direct (c') and indirect effect (ab) of X on Y is different from that of its total effect. That is, to testing the null hypothesis c=0 is unnecessary since CME can be nonzero even when the total causal effect is zero (i.e., direct and indirect effects can be opposite) (4,7), which reflects the effect cancellation from different pathways.

However, the above-mentioned equations within linear structural equation modeling (LSEM) framework are not directly applicable to binary outcome. LSEM equation needs extension to accommodate non-linear settings (8,9). In this article we will focus on Imai's method that is applicable to non-linear settings (4).

Worked example

Next, a dataset is simulated using the motivating example described in the introduction section. Suppose investigators want to explore the effect of corticosteroids on mortality outcome, and it is valuable to notice if part of the total effect is mediated via reducing inflammatory response. C-reactive protein (CRP) is a biomarker of inflammatory response and there is evidence that higher CRP is associated with adverse clinical outcomes (10). Note that the example is created to demonstrate the use of CMA into the clinical context, and there is no clinical relevance of the results.

```
> set.seed(888)
> treat_flg<-rbinom(1000,1,0.3) #binary treatment
#variable with binomial distribution
> crp<- round(abs(40*treat_flg+rnorm(1000,100,30)),1)
#regress crp on treatment
> lp<-10*treat_flg+0.02*crp-5 # linear prediction
> link_lp <- exp(lp)/(1 + exp(lp)) #link function
> mort<-(runif(1000) < link_lp)
> df<-data.frame(crp=crp,treat_flg=treat_flg,mort=mort)
```

CMA

This article implements CMA using the R *mediation* package (6). This package also contains functions for CMA with certain designs or conditions such as multiple outcome/treatment/mediator combinations, multiple causal mechanisms, treatment non-compliance, and dataset with missing values. In addition, this package allows researchers to implement sensitivity analysis for certain parametric models. In the following example, the authors only applied

the simplest form of CMA. Let's first install the package and then perform CMA with a few lines of codes.

```
> install.packages("mediation")
> library(mediation)
> model.m<-lm(crp~treat_flg,data=df[1:100,])
> model.y<-glm(mort~treat_flg+crp,family =
binomial,data=df[1:100,])
> med.out <- mediate(model.m, model.y,
treat = "treat_flg", mediator = "crp",
robustSE = TRUE, sims = 100)
```

The mediate() function requires two fitted model objects: one is for the mediator and the other is for the outcome. In the example, the mediator model is a linear model that regresses *crp* on *treat_flg*. The outcome model is a logistic regression model that regresses *mort* on *treat_flg* and *crp*. Note that these two regression models allow other confounding factors to be added if they are observed. In the example, only the first 100 observations are used for analysis. The *treat* argument takes a character string indicating the name of the treatment variable (*treat_flg*) used in the models. Similarly, the *mediator* argument takes a character string indicating the name of the mediator variable (*crp*) used in the model.

Understanding summary output with binary outcome variable

The results of CMA can be accessed with summary() and plot() function.

```
> summary(med.out)
Causal Mediation Analysis
Quasi-Bayesian Confidence Intervals
                    Estimate 95% CI Lower 95% CI Upper p-value
ACME (control)      1.84e-01 1.41e-02     4.01e-01     0.02
ACME (treated)      1.13e-08 7.48e-10     4.74e-08     0.02
ADE (control)       9.40e-01 8.78e-01     9.80e-01     0.00
ADE (treated)       7.55e-01 5.33e-01     9.36e-01     0.00
Total Effect        9.40e-01 8.78e-01     9.80e-01     0.00
Prop. Mediated      1.81e-01 1.47e-02     4.30e-01     0.02
(control)
Prop. Mediated      7.92e-09 8.09e-10     5.27e-08     0.02
(treated)
ACME                9.21e-02 7.03e-03     2.01e-01     0.02
(average)
ADE (average)       8.48e-01 7.06e-01     9.52e-01     0.00
Prop. Mediated      9.06e-02 7.35e-03     2.15e-01     0.02
(average)
Sample Size Used: 100
Simulations: 100
```

The results show the ACME and average direct effect (ADE) for the treated and control groups. For a single index, researchers can use the average ACME and ADE to determine the average mediation strength. Note that the ACMEs are different for the treated and control groups. This can be understood in the counterfactual framework. Let's first define ACME with mathematical equation according to Imai and colleagues (7):

$$\delta_i(t) \equiv Y_i[t, Mi(1)] - Y_i[t, Mi(0)] \quad [5]$$

$\delta_i(t)$ is the CME under treatment t, which represents the indirect effect of treatment on outcome Y_i(t) through mediator. The subscript i indicates the individual patient. The counterfactual framework can be understood with the following question: what changes would occur to the outcome if the mediator is changed from the value under treatment t=1 [M_i(1)] to the value that would be observed under treatment t=0 [M_i(0)], while holding treatment at t. While the outcome of the form $Y_i[t, Mi(t)]$ is observable, $Y_i[t, Mi(1-t)]$ is unobservable for a given patient. In the example, CME (treated, $\delta_i(1)$) is the difference between two potential mortality outcomes for patient i who received corticosteroids. For the patient, $Y_i[1, Mi(1)]$ is observed outcome if he or she receives corticosteroids, whereas $Y_i[1, Mi(0)]$ represents the mortality outcome under the condition that the patient still receives corticosteroids but change the mediator value that would result without treatment. This definition with counterfactual framework reflects the intuitive notion of mediation that the treatment indirectly influences outcome via mediator. Similarly, direct effect can be defined as:

$$\zeta_i(t) \equiv Y_i[1, Mi(t)] - Y_i[0, Mi(t)] \quad [6]$$

In the example, $\zeta_i(1)$represents the direct effect of corticosteroids on patient i's mortality outcome while holding the *crp* at the level that would be observed under treatment. The causal effect of the treatment on the outcome for unit *i* now can be defined as the following

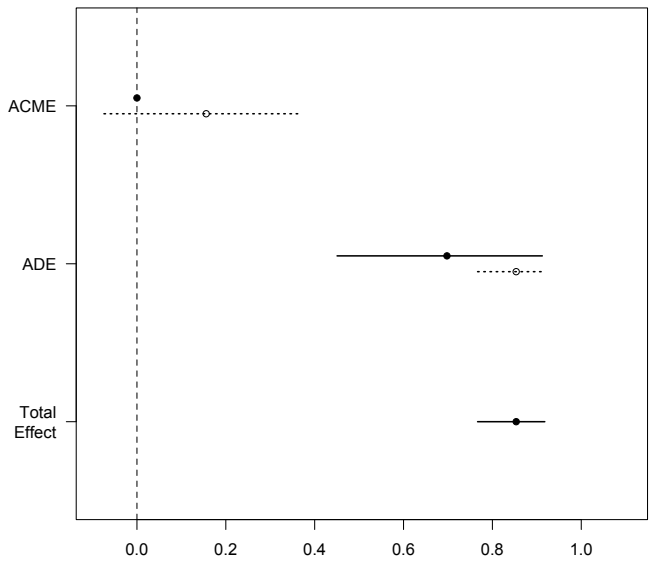


Figure 2 Visualization of results from mediate() function. Estimates for both treated and control group were depicted because they are different in the example. The dashed line represents the control and solid line represents the treated. It appears that most of the total effect is explained by the direct effect

relationship

$$Total \equiv \delta_i(t) - \zeta_i(1-t) \quad [7]$$

While CME is at the individual level, the ACME is the expected value for the whole study population:

$$\bar{\delta}(t) \equiv \mathrm{IE}\left\{Y_i\left[t, M_i(1)\right] - Y_i\left[t, M_i(0)\right]\right\} \quad [8]$$

It appears that ACMEs of both the treated and the control are statistically non-significant. This may be due to the limited sample size. Recall that we only used the first 100 patients for analysis. The results can be visualized with plot() function.

```
> plot(med.out)
```

The simple function automatically produces a plot of mediation analysis. Estimates for both treated and control group were depicted because they are different in the example. The dashed line represents the control and solid line represents the treated. It appears that most of the total effect is explained by the direct effect (*Figure 2*).

The next question that often confuses people is the interpretation of coefficients. In normal linear model, all variables are linked in the same scale and the interpretation is straightforward. However, in CMA with binary outcome variables, the estimated coefficients from above links to the probability scale, rather than that obtained from logistic regression model that its exponentiation gives the odds ratio (OR). Next, let's take a look at how estimation of mediation effects can be different using different methods: the standard LSEM approach proposed by Baron (11) and the method by Imai.

Imai's method consists of two steps which are based on Eqs. [5] and [6]. The first step is to fit regression models for mediator and outcome, which has been done in the above R codes. The second is to simulate the unobserved potential mediator and then compute the average potential outcome from the fitted model.

```
> #average over different simulation runs to obtain
#mediation effects, direct effects and total effect.
> #M is simulation times, this is sims=100 in Imai's
#method, when M is larger, the result is less sensitive to
#the seed use.
> M<-100
> acme0<-acme1<-rep(NA,M)
> total<-rep(NA,M)
> direct1<-direct0<-rep(NA,M)
> for (i in 1:M){
 #impute potential mediators
  m0<-predict(model.m,data.frame(treat_flg=0))+
summary(model.m)$sigma*rnorm(100,0,1)
  m1<-predict(model.m,data.frame(treat_flg=1))+
summary(model.m)$sigma*rnorm(100,0,1)
   #impute potential outcomes
  y0_m0<-predict(model.y,data.frame(treat_flg=0,
crp=m0),type="response")
y0_m1<-predict(model.y,data.frame(treat_flg=0,
crp=m1),type="response")
  y1_m0<-predict(model.y,data.frame(treat_flg=1,
crp=m0),type="response")
  y1_m1<-predict(model.y,data.frame(treat_flg=1,
crp=m1),type="response")
  acme0[i]<-mean(y0_m1-y0_m0)
  acme1[i]<-mean(y1_m1-y1_m0)
  total[i]<-mean(y1_m1-y0_m0)
   direct1[i]<-mean(y1_m1-y0_m1)
   direct0[i]<-mean(y1_m0-y0_m0)
```



```
}
> #point estimate for ACME0, ACME1 compare to
#mediation package output
> c(mean(acme0),med.out$d0)
 [1] 0.1929078 0.1842675
> c(mean(acme1),med.out$d1)
 [1] 7.032967e-09 1.131077e-08
> #point estimate for total effect, compare to mediation
package output
> c(mean(total),med.out$tau.coef)
 [1] 0.9447887 0.9396576
> #point estimate for direct effects, compare to
#mediation package output
> c(mean(direct0),med.out$z0)
 [1] 0.9447887 0.9396576
> c(mean(direct1),med.out$z1)
 [1] 0.7518808 0.7553901
```

The value 0.193 approximate the ACME(0) reported in tabular output by summary() function. A small diffidence is that we perform limited number of simulations. If we compute acme0 with our code 1,000 times and take the average, the value would be more approximate. The coefficients estimated with mediate() function can be better understood with the above explicit computations. Because the type argument takes a string variable "response" within the predict() function, y0_m0 is the probability of death for a given patient in the control group and a *crp* level being observed. y0_m1 is the probability of death for a given patient in the control group and a *crp* level that would be observed supposing the patient had received treatment. Thus, y0_m1 is unobservable in reality but only in counterfactual framework. The difference between y0_m0 and y0_m1 gives the ACME in the control group that can be interpreted as the probability of death being changed by varying *crp* values, while holding the treatment status constant. The total effect of 0.94 can be interpreted as the increased probability of death comparing treatment versus control group. The model coefficients estimated from Imai's nonparametric method have a direct link to the probability scale and thus is more clinically relevant.

Now we will show that if we ignore the fact that some of the equations are fit with nonlinear link and still use the formula derived from linear setting, i.e., the total effect c = c'+a*b, we will not be able to yield correct result. Eqs. [2] with logit link and [3] have been fitted and stored in objects *model.y* and *model.m*, respectively. We only have to fit the Eq. [1] with logit link and store it in an object called *model.t*.

```
> model.t<-glm(mort~treat_flg,family =
binomial,data=df[1:100,])
> c<-model.t$coef[[2]]
> c.prime<-model.y$coef[[2]]
> b<-model.y$coef[[3]]
> a<-model.m$coef[[2]]
> c(c,c.prime+a*b)
[1] 23.38447 24.45443
```

The results show that the total effect estimated from the regression model.t approximate that estimated from regression models.y and model.m. For nonlinear case, the two quantities might approximately the same under some special cases, for example, weak indirect effect (a*b), log link function or rare outcome. Our example is the one with weak indirect effect. However, the values of these estimated coefficients are not equal to that obtained via mediate() function, because the later coefficients are identified with methods as described above (12). The coefficients estimated from logistic regression model are in the linear prediction function scale that its exponentiation gives the odd ratio (OR).

```
> exp(c(a*b,c.prime,c))

[1]  7.082856e+00  5.891355e+09  1.431347e+10
```

The total OR is the sum of direct effect OR and indirect effect OR. The results show that there is negligible indirect effect, and the direct effect takes nearly all of the total effect.

Sensitivity analysis

CMA described above is performed under sequential ignorability assumption. However, this assumption is untestable in practice. One approach is to perform sensitivity analysis to examine whether the results are robust to the violation of the assumption. Violation to assumption indicates that there exists an unmeasured confounder that is related to both the mediator and the outcome. Note that this unmeasured confounder is not affected by the

treatment. Sensitivity analysis uses certain statistics to quantify how strong the confounder would have to be to change the conclusion being drawn about the direct and indirect effect (13). Imai and colleagues proposed a correlation parameter (ρ) reflecting the existence of omitted variables that were related to the mediator and outcome even after conditioning on treatment, and the parameter was added to the calculations of ACME. Sensitivity analysis is to vary ρ values and compute corresponding ACMEs. There are two methods to calculating ρ. One is to compute ρ based on residual correlation:

$$\rho \equiv \text{cor}(e_2, e_3) \quad [9]$$

The other way is to calculate correlation parameter ρ based on coefficients of determination (i.e., a number that indicates the proportion of the variance in the dependent variable that is predictable from the independent variable):

$$\rho = \text{sgn}\left(\lambda_2\lambda_3\right) R_M^* R_Y^* = \text{sgn}\left(\lambda_2\lambda_3\right) \tilde{R}_M \tilde{R}_Y / \sqrt{\left(1-R_M^2\right)\left(1-R_Y^2\right)} \quad [10]$$

$$e_2 = \lambda_2 U + e_2'$$
$$e_3 = \lambda_3 U + e_3'$$

where R_M^{2*} and R_Y^{2*} represent the proportion of previously unexplained variance (mediator and outcome) that is explained by the unobserved confounders. $\tilde{R}_M{}^2$ and $\tilde{R}_Y{}^2$ are the proportion of variance that is explained by the unobserved confounders. U is unobserved confounding variable. R_M^2 and R_Y^2 are the coefficients of determination for the mediation and outcome regression models. R_M^2 represents the proportion of explained variance by covariates in the mediation model (Eq. [3]). λ_2 and λ_3 are coefficients for unobserved variable in the equation regressing residual e_2 or e_3 on the unobserved variable. We only care about the direction (sign) of the effect. In this framework, investigators can specify values of (R_M^{2*}, R_Y^{2*}) and ($\tilde{R}_M{}^2$, $\tilde{R}_Y{}^2$), and the sign of ($\lambda_2\lambda_3$) to estimate ACME (4,14).

Next I proceed to perform sensitivity analysis with medsens() function. Because the function in its current version only support probit link, the mediation model needs to be updated.

```
> probit.y<-glm(mort~treat_flg+crp,family = binomial(
probit),data=df[1:100,])
> med.out1 <- mediate(model.m, probit.y, treat =
"treat_flg", mediator = "crp",robustSE = TRUE, sims =
100)
> sens.out <- medsens(med.out1, rho.by = 0.1,
effect.type = "indirect", sims = 100)
```

The medsens() function first takes an object of class "mediate" which is an output object from mediate() function. The argument *rho.by* takes a numeric value ranging from 0 to 1 indicating the increment for sensitivity parameter ρ. The *effect.type* argument indicates which effect to be examined.

```
> summary(sens.out)
Mediation Sensitivity Analysis: Average Mediation Effect
Sensitivity Region: ACME for Control Group
     Rho ACME      95% CI    95% CI    R^2_      R^2_
         (control) Lower     Upper     M*R^2_    M~R^2_
                                       Y*        Y~
[1,] 0.2 0.0098    -0.0020   0.0246    0.04      0.0013
[2,] 0.3 0.0020    -0.0119   0.0096    0.09      0.0029
[3,] 0.4 -0.0035   -0.0208   0.0021    0.16      0.0051
Rho at which ACME for Control Group = 0: 0.3
R^2_M*R^2_Y* at which ACME for Control Group = 0: 0.09
R^2_M~R^2_Y~ at which ACME for Control Group = 0:
0.0029

Sensitivity Region: ACME for Treatment Group
     Rho ACME      95% CI    95% CI    R^2_      R^2_
         (treated) Lower     Upper     M*R^2_    M~R^2_
                                       Y*        Y~
[1,] 0.2 0.0017    0e+00     0.0071    0.04      0.0013
[2,] 0.3 0.0008    0e+00     0.0035    0.09      0.0029
[3,] 0.4 0.0001    -6e-04    0.0008    0.16      0.0051
Rho at which ACME for Treatment Group = 0: 0.4
R^2_M*R^2_Y* at which ACME for Treatment Group = 0:
0.16
R^2_M~R^2_Y~ at which ACME for Treatment Group = 0:
0.0051
```

Because we assigned "indirect" for the argument *effect.type*, sensitivity analysis for ACME is performed. The summary output displays the computed ACMEs by varying ρ values for the treatment and control groups. Two tables are similar in interpretation, so we only take a look at the first table (control group). The first column is the ρ values with an increment of 0.1. The second column shows the corresponding ACMEs, and the third and forth columns are

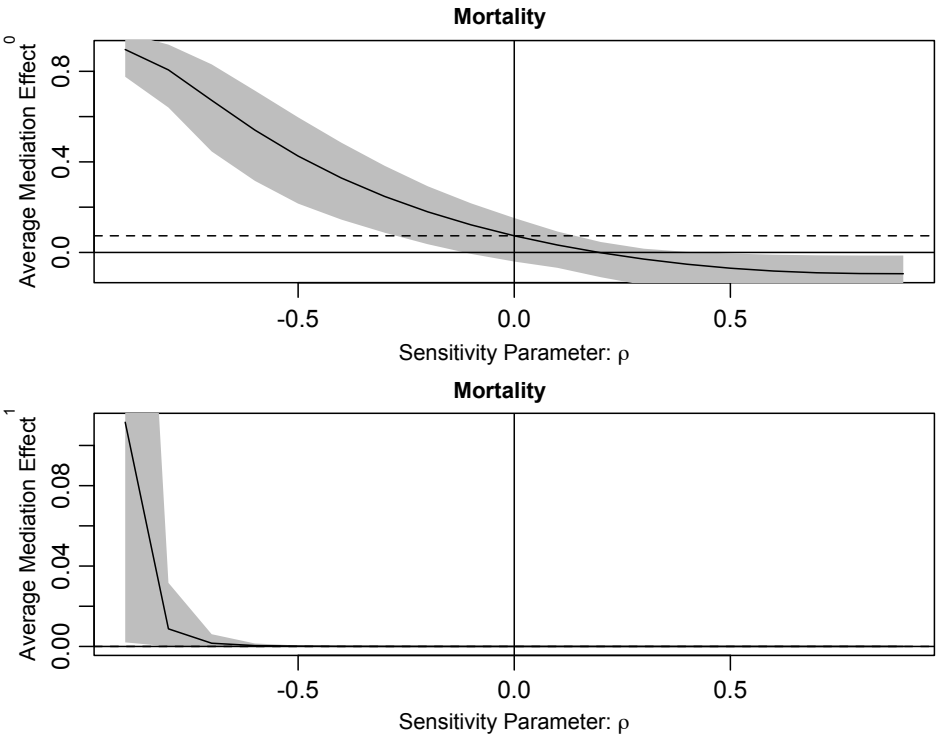


Figure 3 Sensitivity analysis based on residual correlation. The dashed line represents the ACME (0.0735) without correlation (ρ=0), which is computed under the assumption of sequential ignorability. Note that ρ≥0.2 is required for the ACME to become negative.

confidence intervals. The last two columns are quantities based on coefficients of determination. Because the ρ value at which ACME becomes zero is of interest, it is displayed following the table. A graphical display of the results can be more intuitive and helpful in sensitivity analysis.

```
> par(mfrow=c(2,1))
> plot(sens.out, sens.par = "rho", main = "Mortality")
```

The *sens.par* argument specifies which sensitivity parameter, residual correlation or coefficients of determination, to be displayed. The dashed line represents the ACME value without correlation (ρ=0), which is computed under the assumption of sequential ignorability. We need to set $\rho \geq 0.3$ for the ACME to become negative (*Figure 3*). A large critical ρ value is needed to reverse the sign of ACME indicates the robustness of the result to the violation of ignorability assumption. It requires subject knowledge to judge whether the value of 0.3 is large or not. In practice, observed confounding variables are often used to determine the upper bound of sensitivity parameters.

Sensitivity analysis based on R^2 statistic is performed by assign "R2" to sens.par. The type of R^2 to be used is specified in *r.type* argument. A string variable "residual" indicates that the effects are plotted against the proportions of the residual variances that are explained by the unobserved confounder. On the other hand, a value of "total" indicates that the proportions of the total variances are used as sensitivity parameters. When sensitivity analysis is performed with R^2 statistic, the user must specify whether the unobserved confounder affects the mediator (λ_3) and outcome (λ_2) in the same direction (sgn[$\lambda_2\lambda_3$]=1) or opposite direction (sgn[$\lambda_2\lambda_3$]=−1). If the effects are in the same direction, users assign sign.prod = "positive". Otherwise, the *sign.prod* argument takes the string variable "negative".

```
> par(mfrow=c(2,2))
> plot(sens.out, sens.par = "R2", r.type = "residual",
sign.prod = "negative")
> plot(sens.out, sens.par = "R2", r.type = "total",
sign.prod = "positive")
```

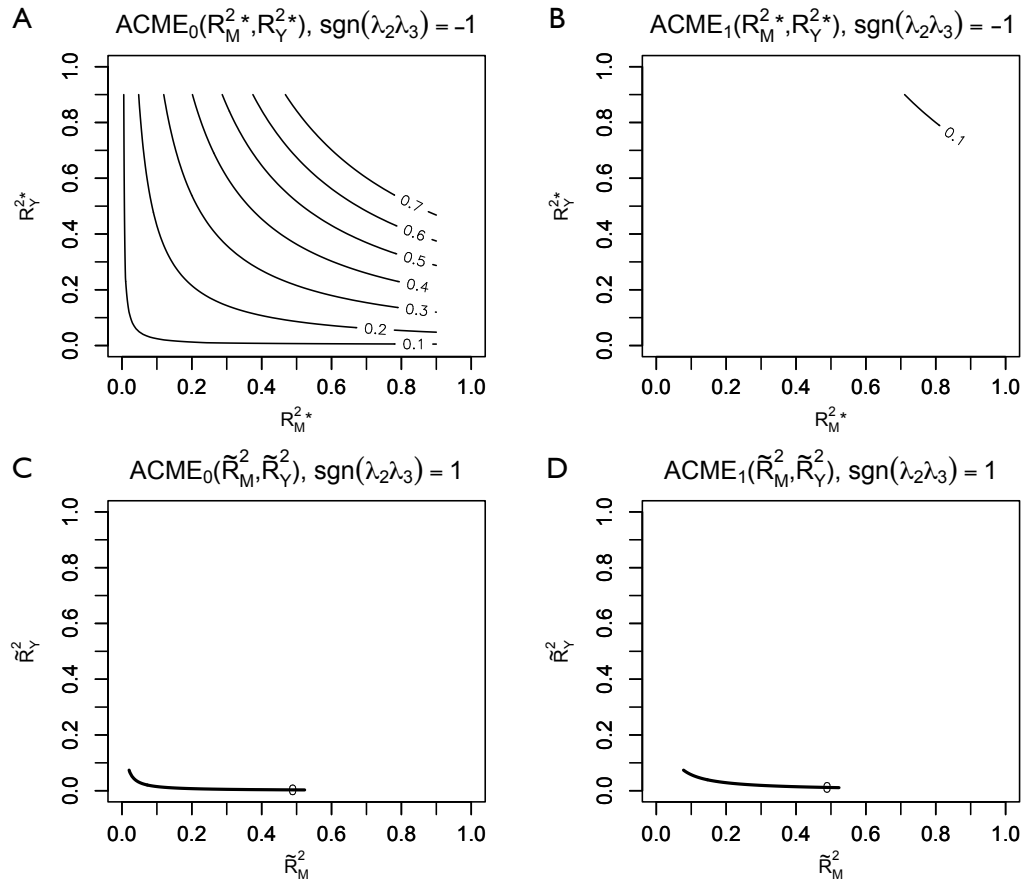



Figure 4 Sensitivity analysis based on coefficient of determination. Four plots are produced by varying *r.type* and *sign.prod* arguments within plot() function. Note that only the first plot that has displayed a series of ACME values. Each of the other plots has displayed the contour of one ACME value, because the native R contour function automatically adjusts the displaying values.

Four plots are produced by varying *r.type* and *sign.prod* arguments (*Figure 4*). The first panel shows the sensitivity analysis for ACME in the control group by setting $\text{sgn}(\lambda_2\lambda_3)=-1$. However, it appears that except for the first plot that displayed many values of ACME, each of the other plots only displays the contour of one ACME value. The reason is that the native R contour function automatically adjusts the displaying values, and thus we need to adjust the *levels* argument to allow displaying of more contours. The ACME values across a range of degrees of confounding can be examined in the values returned by the medsens() function.

```
> sens.out$d0

[1]         0.913575804   0.661963822   0.453954736
[4]         0.314137898   0.220546426   0.156108478
[7]         0.110405639   0.077178510   0.052570140
[10]        0.034124673   0.020227810   0.009787342
[13]        0.002044243   -0.003544247  -0.007382268
[16]        -0.009793831  -0.011076604  -0.011559293
[19]        -0.011636731

> sens.out$d1

[1]         2.545749e-01  9.531345e-02  5.024505e-02
[4]         3.087918e-02  2.067147e-02  1.455696e-02
[7]         1.054603e-02  7.727958e-03  5.639027e-03
[10]        4.023900e-03  2.733704e-03  1.678442e-03
[13]        8.033842e-04  7.707901e-05  -5.145001e-04
```

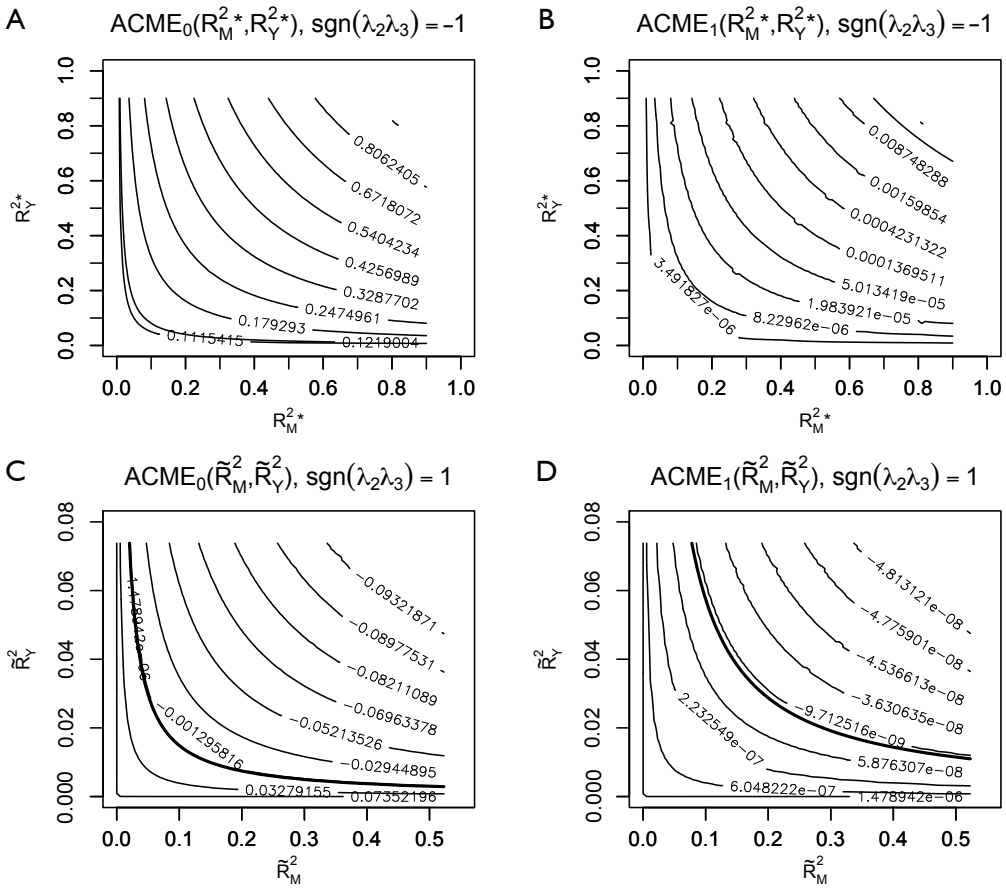



Figure 5 Sensitivity analysis based on coefficient of determination. Series of ACME values are displayed by adjusting *levels* argument in the plot() function.

```
[16]      -9.706079e-04 -1.278730e-03 -1.428202e-03
[19]      -1.456307e-03
```

Note that the ACME values (i.e., the first 9 values of the sens.out$d0 output) for the control group when $\text{sgn}(\lambda_2\lambda_3)=-1$ is increased at a step approximating 0.1 and thus can be displayed without specifying levels argument. However, the ACME values (i.e., the first 9 values of the sens.out$d1 output) for the treated group when $\text{sgn}(\lambda_2\lambda_3)=-1$ is changed at steps significantly different from 0.1, thus we need to specify the levels for displaying more contours.

```
> par(mfrow=c(2,2))
> plot(sens.out, sens.par = "R2", r.type = "residual",
sign.prod = "negative",levels=c(sens.out$d0[1:10],
sens.out$d1[1:10]))
> plot(sens.out, sens.par = "R2", r.type = "total",
sign.prod = "positive",levels=c(sens.out$d0[10:19],
sens.out$d1[10:19]),ylim=c(0,0.08),xlim=c(0,0.55))
```

The bold line represents various combinations of R^2 statistics where ACME takes the value zero (*Figure 5*). In the example, the value of ACME under sequential ignorability assumption is not statistically significant, thus the degree of confounding where ACME would be zero is of limited interest. In other situations, where ACME is statistically significant, the critical point of the degree of confounding where ACME takes zero is of great interest because beyond this point the effect size is in opposite direction (i.e., the sign is changed). However, there is no cutoff value for the statistics of confounding (i.e., ρ and combinations of R^2 statistics) to judge the robustness of CMA results estimated under ignorability assumption.

Acknowledgements

None.

Footnote

Conflicts of Interest: The authors have no conflicts of interest to declare.

References

1. MacKinnon DP, Pirlott AG. Statistical approaches for enhancing causal interpretation of the M to Y relation in mediation analysis. Pers Soc Psychol Rev 2015;19:30-43.
2. Linden A, Karlson KB. Using mediation analysis to identify causal mechanisms in disease management interventions. Health Serv Outcomes Res Method 2013;13:86-108.
3. Zhang Z, Chen L, Ni H. The effectiveness of Corticosteroids on mortality in patients with acute respiratory distress syndrome or acute lung injury: a secondary analysis. Sci Rep 2015;5:17654.
4. Imai K, Keele L, Tingley D. A general approach to causal mediation analysis. Psychol Methods 2010;15:309-334.
5. MacKinnon DP, Fairchild AJ, Fritz MS. Mediation analysis. Annu Rev Psychol 2007;58:593-614.
6. Tingley D., Yamamoto T, Hirose K, et al. mediation: R Package for Causal Mediation Analysis. J Stat Softw 2014;59:1-38.
7. Imai K, Keele L, Yamamoto T. Identification, Inference and Sensitivity Analysis for Causal Mediation Effects. Statistical Science 2010;25:51-71.
8. Li Y, Schneider JA, Bennett DA. Estimation of the mediation effect with a binary mediator. Stat Med 2007;26:3398-3414.
9. MacKinnon DP, Lockwood CM, Brown CH, et al. The intermediate endpoint effect in logistic and probit regression. Clin Trials 2007;4:499-513.
10. Zhang Z, Ni H. C-reactive protein as a predictor of mortality in critically ill patients: a meta-analysis and systematic review. Anaesth Intensive Care 2011;39:854-861.
11. Baron RM, Kenny DA. The moderator-mediator variable distinction in social psychological research: conceptual, strategic, and statistical considerations. J Pers Soc Psychol 1986;51:1173-1182.
12. Imai K, Keele L, Tingley D, et al. Unpacking the Black Box of Causality: Learning about Causal Mechanisms from Experimental and Observational Studies. American Political Science Review 2011;105:765-789.
13. VanderWeele TJ. Mediation Analysis: A Practitioner's Guide. Annu Rev Public Health 2016;37:17-32.
14. Imbens GW. Sensitivity to Exogeneity Assumptions in Program Evaluation. American Economic Review 2003;93:126-132.

Cite this article as: Zhang Z, Zheng C, Kim C, Van Poucke S, Lin S, Lan P. Causal mediation analysis in the context of clinical research. Ann Transl Med 2016;4(21):425. doi: 10.21037/atm.2016.11.11

Structural equation modeling in the context of clinical research

Zhongheng Zhang

Abstract: Structural equation modeling (SEM) has been widely used in economics, sociology and behavioral science. However, its use in clinical medicine is quite limited, probably due to technical difficulties. Because SEM is particularly suitable for analysis of complex relationships among observed variables, it must have potential applications to clinical medicine. The article introduces some basic ideas of SEM in the context of clinical medicine. A simulated dataset is employed to show how to do model specification, model fit, visualization and assessment of the goodness-of-fit. The first example fits a SEM with continuous outcome variable using sem() function, and the second explores the binary outcome variable using lavaan() function.

Keywords: Structural equation modeling (SEM); latent variable; endogenous variable; exogenous variable

Submitted Jun 06, 2016. Accepted for publication Jul 19, 2016.
doi: 10.21037/atm.2016.09.25
View this article at: http://dx.doi.org/10.21037/atm.2016.09.25

Introduction

Structural equation modeling (SEM) combines various types of regression models to describe relationships among observed variables, aiming to provide a quantitative test of a theoretical model hypothesized by the investigators (1). A set of observed variables may be used to define a construct (measurement model), and these constructs are related to each other (structural model). Constructs (latent variables or factors) are variables that are not directly observed or measured, but are defined by other observed variables. The indicators (observed or measured) are a set of variables that define or infer a construct (2). In terms of the relationship, variables can be defined as either independent or dependent variables. These terms will be referred to in the following example, which may help readers to better understand them.

The article will discuss SEM in the context of clinical research. Basic ideas and terminologies will be introduced along with the example, which may give readers a better understanding than tutorials full of mathematical details. There is a variety of R packages for SEM and its visualization, which will be discussed in the article (3).

Worked example

For the purpose of illustration, I create a dataset containing patients from intensive care unit (ICU). The research setting is employed to give readers an understanding of how to perform SEM in clinical medicine. Of note, the dataset is created by simulation technique and bears no practical interpretation. Suppose the study is designed to investigate the predictors of financial cost and mortality for ICU patients. Because there are numerous laboratory measurements being obtained for ICU patients, the complex relationships among them are difficult to disentangle. However, these laboratory measurements can be divided into broad categories. For example, c-reactive protein (crp), procalcitonin (pct) and white blood cell (wbc) are biomarkers of inflammation. Bilirubin (bil), serum creatinin (scr) and oxygen index (oxyindex) are biomarkers of organ dysfunction. However, there are no direct measurements of inflammation and organ dysfunction, thus these two are designated as latent variables. It is rational to hypothesize that inflammation causes organ dysfunction, and medical cost (cost) is increased by inflammation and organ dysfunction.

```
> set.seed(888)
> inflammation<-abs(rnorm(1000,100,20)) # some continuous
#variables
> crp<-round(abs(inflammation+rnorm(1000,0,20)),1)
> pct<-round(abs(0.15*inflammation+rnorm(1000,0,5)),1)
> wbc<-round(abs(0.1*inflammation+rnorm(1000,0,6)),1)
```

```
> orgdys<-abs(0.4*inflammation+rnorm(1000,0,5))
> bil<-round(abs(orgdys+rnorm(1000,0,8)),1)
> scr<-round(abs(3*orgdys+rnorm(1000,0,20)),1)
> oxyindex<- round(abs(7*orgdys+rnorm(1000,0,40)))
> cost<-abs(round(50*inflammation+100*orgdys+
rnorm(1000,100,100))) #cost can never have negative values
> z<-inflammation+orgdys-150   # linear combination with a
#bias
> pr<-1/(1+exp(-z))        # pass through an inv-logit function
> mort<-factor(rbinom(1000,1,pr), labels=c("survivor",
"nonsurvivor"))     # bernoulli response variable
> data<-data.frame(crp=crp,pct=pct,wbc=wbc,bil=bil,scr=scr,
oxyindex=oxyindex,cost=cost,mort=mort)
```

The above codes firstly set a seed [888] to allow readers to reproduce the results. Then inflammation is created as in crp scale with normal distribution. Inflammation is correlated with crp. The error term for crp has a mean of 0 and standard error of 20. Other variables are created in the same manner. All variables are forced to be positive by the abs() function. Cost is determined by inflammation and organ dysfunction with an error term. Because mortality is a binary outcome, it is assumed to follow Bernoulli distribution and is created by using rbinom() function. Finally, all observed variables are combined into a data frame. Note that the latent variables inflammation and orgdys are excluded because in the real world they are not observable.

Fitting the structural equation model

The first step in fitting the SEM is to setup environment for sem() function. Furthermore, the DiagrammeR package should be installed and loaded to the workspace for the purpose of drawing SEM diagram.

```
> install.packages("sem")
> library(sem)
> install.packages("DiagrammeR")
> library(DiagrammeR)
```

Before estimation for parameters, the structure of the model should be specified. Model specification is primarily based on subject knowledge and previous studies reporting the association between variables. The model can be specified using specifyEquations() function. Other functions such as specifyModel() can also be used. The example uses specifyEquations().

```
> model.cost <- specifyEquations()
   bil = 1*orgdys
   scr = lam2*orgdys
   oxyindex=lam3*orgdys
   crp = 1*inflammation
   pct = lam1*inflammation
   wbc=lam4*inflammation
   orgdys = gamma11*inflammation
   cost = gamma21*inflammation + beta21*orgdys
   v(inflammation) = phi
```

In specifyEquations(), each line specifies either a regression equation or variance or covariance. Variable on the right side of the equation is exogenous variable that it has no arrow points to it but only has arrows point out. In other terminology, exogenous variable is explanatory variable that explains changes of other variables. The left side of equation shows the parameter and endogenous variables. Endogenous variable is dependent variable that arrows point to it. The parameter is required to be estimated. If parameters are given fixed values (numeral 1 for orgdys in the example) it is treated as fixed. Otherwise, parameters are constrained if two equations use the same name for their parameters. Variances of a variable that cannot be explained by its exogenous variables are specified in the form V(variable) = parameter. Covariance of two variables are represented in the form C(variable 1, variable 2) = parameter. The symbols “v” and “c” can be in either lower- or upper-case. By default, variance for an endogenous variable will be calculated by sem() without explicitly specifying it. However, variance of an exogenous variable should be specified. In the example, inflammation is an exogenous variable and I assign phi as its variance. Next, let’s take a look at the structure of the model.

```
> model.cost
```

	Path			Parameter	StartValue
1	orgdys	->	bil	lam2	
2	orgdys	->	scr	<fixed>	1
3	orgdys	->	oxyindex	lam3	
4	inflammation	->	crp	<fixed>	1
5	inflammation	->	pct	lam1	
6	inflammation	->	wbc	lam4	

7	inflammation	->	orgdys	gamma11
8	inflammation	->	cost	gamma21
9	orgdys	->	cost	beta21
10	inflammation	<->	inflammation	phi
11	orgdys	<->	orgdys	V[orgdys]
12	bil	<->	bil	V[bil]
13	scr	<->	scr	V[scr]
14	oxyindex	<->	oxyindex	V[oxyindex]
15	crp	<->	crp	V[crp]
16	pct	<->	pct	V[pct]
17	wbc	<->	wbc	V[wbc]
18	cost	<->	cost	V[cost]

The above output displays the model in reticular action model (RAM) format via single- and double-headed arrows. Single-head arrow specifies a coefficient between variables. Double-head arrow specifies variance if two variable names are the same and covariance if corresponding variable names are different. The parameter column displays names of parameters. Because variances of endogenous variables are not explicitly specified in the example, their names take the form V[var]. The model can be fit with simple code.

```
> sem.cost<-sem(model.cost,data=data)
```

Model identification

A model can be identified if there exists enough information for solution for all of the model's parameters. Consider the equation x+2y=6, there is an infinite number of pairs of values for x and y to serve as a solution to the equation. The model is underidentified because there are fewer "knowns" than "unknowns". However, when I add another equation 3x+y=4, there is only one set of x and y satisfying both equations. Thus, the model is just identified because there are as many "knowns" as "unknowns". In this simple example, x and y are parameters to be estimated and each equation represents an observation. When there are more parameters than observations, the model is underidentified. When there are as many parameters as observations, the model is just identified. When there are more observations than parameters (e.g., add another equation like x+y=3 to the model), the model is overidentified. The solution to overidentified model is to find a set value of x and y that the sum of squared differences between the observations (3,4) and these totals is as small as possible (4).

The SEM is comprised of structural and measurement models. In the example the measurement model describes the relationship between latent variable inflammation and observed variables crp, pct and wbc. The structural model describes the relationship between variables that we are interested in. For example, are inflammation and organ dysfunction the causes of increased financial cost in ICU patients? The order condition is the necessary requirement for a model to be identified. In order condition, the number of free parameters to be estimated must be less than or equal to the number of distinct values in matrix S. the number of distinct values in matrix S can be determined by Eq. [1]:

$$p\times(p+1)/2 \quad [1]$$

where p is the number of observed values in the model. In our example, the number of distinct values is 7×(7+1)/2=28, and the number of free parameters is 18–2=16. Note there are 18 lines in the "model.cost" output and two parameters are fixed, leaving 16 free parameters to be estimated. The model satisfies the order condition. Because there are more observations than parameters, the model is overidentified. The degree of freedom for the SEM is the difference between number of distinct values in matrix S and the number of free parameters df=28–16=12. You may want to take a look at the matrix S by the following code.

```
> round(sem.cost$S)
```

	crp	pct	wbc	bil	scr	oxyindex	cost
crp	780	56	37	152	459	1039	34062
pct	56	33	3	22	63	154	4917
wbc	37	3	35	13	36	92	3102
bil	152	22	13	151	283	615	16329
scr	459	63	36	283	1227	1887	49570
oxyindex	1039	154	92	615	1887	5867	113688
cost	34062	4917	3102	16329	49570	113688	3299097

Order condition is the necessary but not the sufficient condition for model identification. Other useful conditions can aid model identification. In measurement model, the parameter is also called factor loading. The latent variable is called a construct and observed variables are indicators. Scaling the latent variable is to add a nonzero fix factor loading, which can facilitate model identification. In the example, I add a fixed factor loading 1 for observed

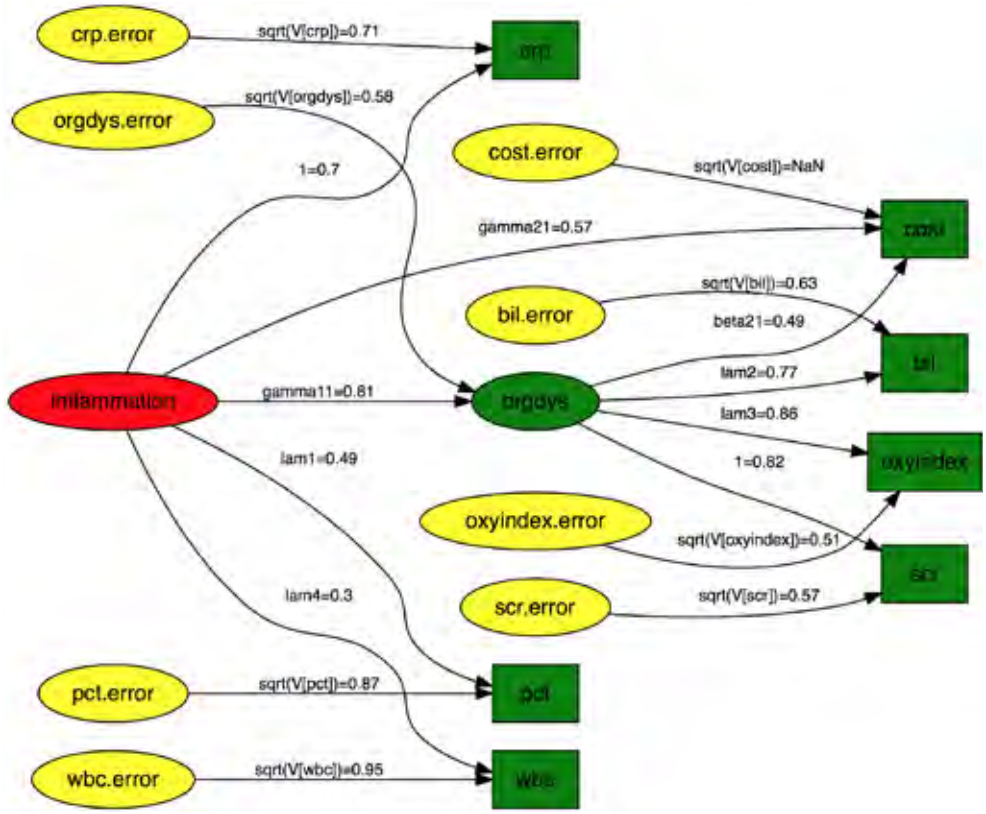


Figure 1 Diagram of structural equation model in traditional style with error variables displayed. Note that latent variables and errors are represented by ellipse and observed variables are represented by rectangles. Exogenous variable is in red color. Endogenous variables are in green color and errors are in yellow color. The arrows represent parameters to be estimated with names and values displayed above each edge. crp, c-reactive protein; pct, procalcitonin; wbc, white blood cell; bil, bilirubin; scr, serum creatinin; oxyindex, oxygen index.

variables crp and scr. The purpose is to fix the unit of latent measurement. The "three measure rule" states that one latent construct has at least three indicators whose errors are uncorrelated with each other. The "two measure rule" states that every latent construct is associated with at least two indicators AND every construct is correlated with at least one other construct (5-8). However, technical details of model identification is very complex and beyond the scope of this article. Next, I will use the summary() function to print the parameter estimates of the SEM, as well as statistics for model fit.

The non-significant P value of 0.45 indicates that the model cannot be rejected. Theoretically, any over-identified model can be rejected in a large sample size. Because the sample size in the example is large but there is still no evidence of under-fitting, the model can be accepted. Bayesian information criterion (BIC) is another criterion for the judgment of model fit. Negative values of BIC indicate that a model has greater support from the data than the just-identified model. The just-identified model has a BIC value of 0. BICs of alternative models can be used to compare their fits to data. It is suggested that a difference of five in BIC is a strong evidence that one model is superior to the other (9). Similarly, AIC is an alternative information criterion for model selection. Parameter estimates are of primary interests in SEM. The results show that all parameters and variances are statistically significant. That is because the data were simulated in the way the model was specified. Graphical presentation of the model can be obtained using pathDiagram() function. Note that this function requires DiagrammeR package.

```
> pathDiagram(sem.cost,standardize=TRUE,
edge.labels="both",ignore.self=TRUE,ignore.double=TRUE,
style="traditional",node.colors=c("red","green","yellow"))
```

There are numerous options to customize the graphical

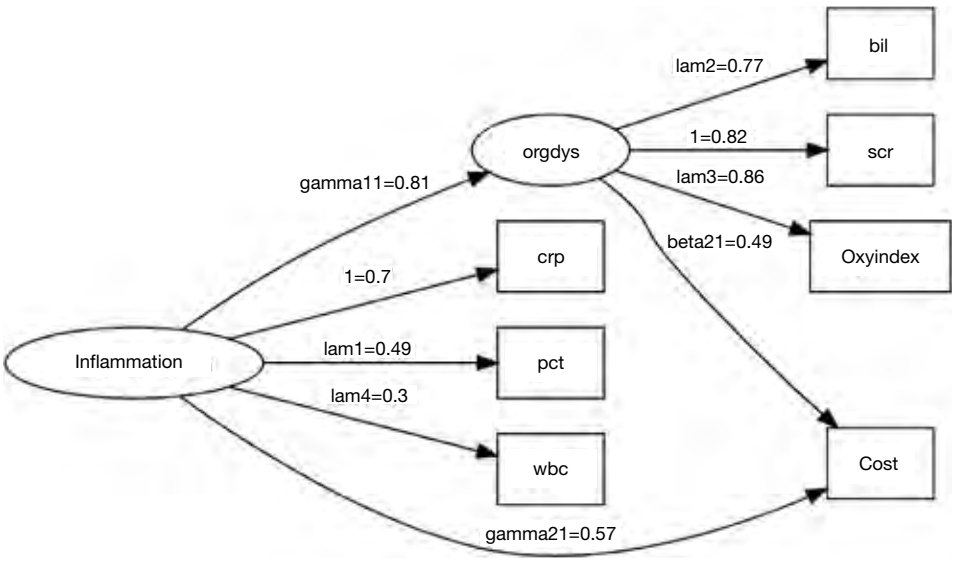


Figure 2 Diagram of structural equation model in reticular action model (RAM) style, which is default to the pathDiagram() function. Note there is no error variables being displayed, making the appearance of the diagram simpler. crp, c-reactive protein; pct, procalcitonin; wbc, white blood cell; bil, bilirubin; scr, serum creatinin; oxyindex, oxygen index.

display. In the example, I present the SEM in traditional style, which includes nodes for error variables (*Figure 1*). Note that latent variables and errors are represented by ellipse and observed variables are represented by rectangles. Exogenous variable is in red color. Endogenous variables are in green color and errors are in yellow color. The arrows represent parameters to be estimated with names and values displayed above each edge. One may notice that the parameter estimates displayed above the edge are not parameter estimates as shown in the summary() output. For example, gamma21, parameter for the estimate of the effect of inflammation on cost, is 0.57 in the figure. The value is 52.8 in the summary() output. Recall that I have assigned the value of 50 for this coefficient in simulation. Then how can we interpret the parameter estimates displayed in the diagram? The diagram displays standardized parameter estimates, instead of the original ones. Also note the standardized estimates for fixed parameter is not 1s. Alternatively, one may wish to draw a RAM path diagram without displaying errors (*Figure 2*). The default of style argument is "ram", thus I leave it unspecified.

```
> pathDiagram(sem.cost,standardize=TRUE,
edge.labels="both")
```

However, The pathDiagram() function provides limited options for diagram appearance. The semPaths() function shipped with semPlot package provides an alternative to draw SEM diagram after fitting the model with sem() function (10). Furthermore, this function also takes SEM object produced by other R functions such as lavaan(). Now let's take a look at how this function works.

```
> install.packages("semPlot")
> library(semPlot)
> semPaths(sem.cost,whatLabels="est",layout="spring",
nCharNodes=0,edge.color="red")
```

The semPaths() function takes the sem object sem.cost. The whatLabels argument specifies what the edge label should indicate. The "est" argument displays the parameter estimate in edge labels, whereas the "stand" displays the standardized parameter estimate. There are several options for the layout of the diagram. Here I use the "spring" option and the appearance is shown in *Figure 3*. While the solid edges represent free parameters, the dashed edges represent the fixed parameters. Other options include "tree" (the default), "circle", "tree2" and "circle2". The color of edges can be specified using edge.color argument.

Interpretation of parameter estimates: direct and indirect effects

A SEM can be useful for clinicians only when its parameter

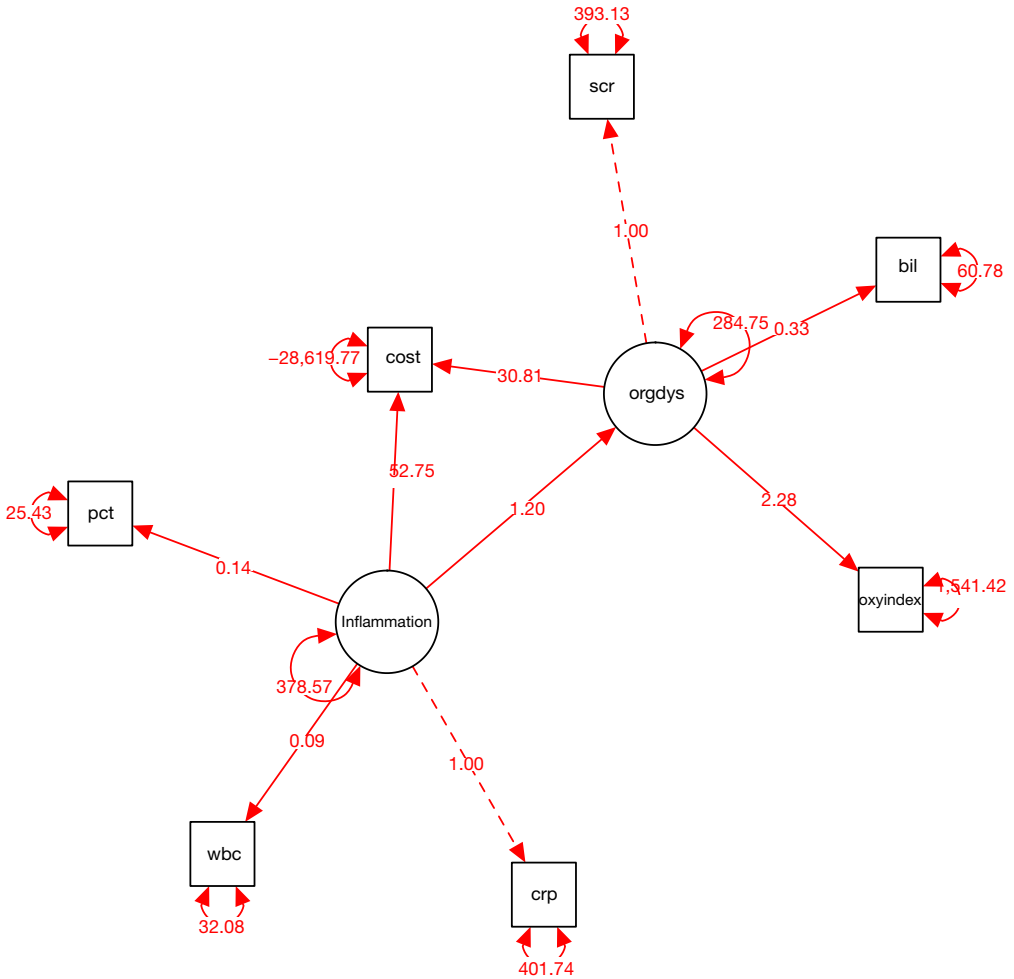


Figure 3 Diagram of structural equation model in spring style produced by semPaths() function. Note that parameter estimates are displayed above the edges. The values of these parameters are consistent with that produced by summary() function. crp, c-reactive protein; pct, procalcitonin; wbc, white blood cell; bil, bilirubin; scr, serum creatinin; oxyindex, oxygen index.

interpretation is related to subject-matter knowledge. The effects() function provides estimates of the direct and indirect effects.

Total Effects (column on row)

	orgdys	inflammation
orgdys	0.000000	1.2048350
bil	0.328352	0.3956100
scr	1.000000	1.2048350
oxyindex	2.277118	2.7435519
crp	0.000000	1.0000000
pct	0.000000	0.1442269
wbc	0.000000	0.0927691
cost	30.806195	89.8682182

Direct Effects

	orgdys	inflammation
orgdys	0.000000	1.2048350
bil	0.328352	0.0000000
scr	1.000000	0.0000000
oxyindex	2.277118	0.0000000
crp	0.000000	1.0000000
pct	0.000000	0.1442269
wbc	0.000000	0.0927691
cost	30.806195	52.7518363

Indirect Effects

	orgdys	inflammation
orgdys	0	0.000000
bil	0	0.395610
scr	0	1.204835
oxyindex	0	2.743552
crp	0	0.000000
pct	0	0.000000
wbc	0	0.000000
cost	0	37.116382

It is noted that the total effect of inflammation on cost is 89.9, which is the sum of the direct (52.8) and indirect effect (37.1). The direct effect of inflammation is its coefficient in the equation for cost, which describes the change in cost attributable to a unit change in inflammation, conditional on all other variables in the equation. This effect ignores any other simultaneous effect. The total effect of inflammation is the change in endogenous variable cost attributable to a unit change in inflammation after accounting for all the simultaneity in the system. The indirect effect acts via the latent variable organ dysfunction.

RAM

To better understand the underlying mechanisms of SEM, I would like to discuss more details on the RAM model. Furthermore, some elements of RAM may help to better understand the modification index (MI) that will be discussed in the next section. The RAM model is expressed by the equation Eq. [2]:

$$v=Av+u \quad [2]$$

where v contains indicator variables, directly observed exogenous variables and the latent exogenous and endogenous variables. u contains directly observed exogenous variables, measurement-error variables, and structural disturbances. The matrix A contains structural coefficients and factor loadings.

As expected, the matrix A is sparse with many 0 s.

```
> summary(sem.cost)

Model Chisquare =11.98179   Df = 12 Pr(>Chisq) =0.4471436
AIC =43.98179
BIC = -70.91128

Normalized Residuals
Min.      1st Qu.      Median      Mean      3rd Qu.   Max.
-1.604    -0.1515      -1E-07      -0.07734  0.0742    0.5318

R-square for Endogenous Variables
orgdys  bil     scr     oxyindex  crp     pct     wbc     cost
0.6587  0.5968  0.6797  0.7373    0.4852  0.2365  0.0922  1.0087

Parameter Estimates
        Estimate   StdError  zvalue    Pr(>|z|)
lam2    3.28E-01   1.15E-02  28.46266  3.40E-178  bil<---orgdys
lam3    2.28E+00   6.82E-02  33.3749   3.17E-244  oxyindex<---orgdys
lam1    1.44E-01   9.61E-03  15.00758  6.55E-51   pct<---inflammation
lam4    9.28E-02   9.85E-03  9.415566  4.71E-21   wbc<---inflammation
gamma11 1.20E+00   8.67E-02  13.89751  6.56E-44   orgdys<---inflammation
gamma21 5.28E+01   8.13E+00  6.491563  8.50E-11   cost<---inflammation
beta21  3.08E+01   4.60E+00  6.694512  2.16E-11   cost<---orgdys
```


phi	3.79E+02	3.55E+01	10.67342	1.36E-26	inflammation<-->inflammation
V[orgdys]	2.85E+02	4.33E+01	6.571925	4.97E-11	orgdys<-->orgdys
V[bil]	6.08E+01	3.02E+00	20.11305	5.67E-90	bil<-->bil
V[scr]	3.93E+02	2.07E+01	18.95356	4.13E-80	scr<-->scr
V[oxyindex]	1.54E+03	8.77E+01	17.57318	3.95E-69	oxyindex<--> oxyindex
V[crp]	4.02E+02	2.62E+01	15.34663	3.73E-53	crp<-->crp
V[pct]	2.54E+01	1.21E+00	21.06921	1.52E-98	pct<-->pct
V[wbc]	3.21E+01	1.45E+00	22.15517	9.30E-109	wbc<-->wbc
V[cost]	-2.86E+04	5.49E+04	-0.52149	6.02E-01	cost<-->cost

Iterations =234

Another component of RAM is the matrix P of u, which can be obtained using the following code.

```
> round(sem.cost$P,1)
```

	crp	pct	wbc	bil	scr	oxyindex	cost	orgdys	inflammation
crp	401.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
pct	0.0	25.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0
wbc	0.0	0.0	32.1	0.0	0.0	0.0	0.0	0.0	0.0
bil	0.0	0.0	0.0	60.8	0.0	0.0	0.0	0.0	0.0
scr	0.0	0.0	0.0	0.0	393.1	0.0	0.0	0.0	0.0
oxyindex	0.0	0.0	0.0	0.0	0.0	1541.4	0.0	0.0	0.0
cost	0.0	0.0	0.0	0.0	0.0	0.0	-28619.8	0.0	0.0
orgdys	0.0	0.0	0.0	0.0	0.0	0.0	0.0	284.7	0.0
inflammation	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	378.6

Model modification

A MI tells the difference in the goodness-of-fit (as measured in Chi-squares) between an existing model and a modified model in which a fixed parameter is free to be estimated. For example, if a parameter is incorrectly fixed to 1, then the test statistic for this parameter should be large. MI is a chi-square statistic with one degree of freedom; therefore a value of 3.84, which is the statistical significance threshold, requires attention. The output of modIndices() lists the five largest MI in A and P matrix (11).

```
> modIndices(sem.cost)
 5 largest modification indices, A matrix (regression coefficients):
```

scr<-bil	bil<-scr	wbc<-pct	pct<-wbc	cost<-oxyindex
4.649549	4.649546	4.127541	4.127541	3.210411

```
 5 largest modification indices, P matrix (variances/covariances):
```

scr<->bil	wbc<->pct	cost<->oxyindex	orgdys<->oxyindex	inflammation<->oxyindex
4.649548	4.127541	3.210408	2.360746	2.360742

The results show that if we add an arrow from bil to scr, the chi-square of the modified model would be reduced by 4.65. The MIs in P matrix suggest the addition of covariance between observed variables. Let's update our model under the guidance of MI. In the example, I add a covariance between scr and bil.

```
> sem.cost$A
```

	crp	pct	wbc	bil	scr	oxyindex	cost	orgdys	inflammation
crp	0	0	0	0	0	0	0	0.000000	1.0000000
pct	0	0	0	0	0	0	0	0.000000	0.1442269
wbc	0	0	0	0	0	0	0	0.000000	0.0927691
bil	0	0	0	0	0	0	0	0.328352	0.0000000
scr	0	0	0	0	0	0	0	1.000000	0.0000000
oxyindex	0	0	0	0	0	0	0	2.277118	0.0000000
cost	0	0	0	0	0	0	0	30.806195	52.7518363
orgdys	0	0	0	0	0	0	0	0.000000	1.2048350
inflammation	0	0	0	0	0	0	0	0.000000	0.0000000

```
> model.cost.1 <- update(model.cost)
 add, scr<->bil, theps
> sem.cost.1<-sem(model.cost.1, data=data)
> summary(sem.cost.1)
> anova(sem.cost,sem.cost.1)
LR Test for Difference Between Models
           Model Df  Model Chisq  Df   LR Chisq  Pr(>Chisq)
sem.cost   12        11.9818
sem.cost.1 11        7.4134       1    4.5684    0.03257 *
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The Chi-square is 7.4 for the new model, yielding a difference of 4.6 comparing to the original model. The value is consistent with the MI of "scr<->bil". The anova() compares the difference between the two models, and the results show that there is statistical difference between the two models. The sem.cost.1 model fits better to data than the sem.cost model.

SEM with binary endogenous variable

In clinical research, binary data are common such as mortality, gender, and occurrence of an event of interest. Therefore, I would like to introduce how to model SEM with binary outcome variable using lavaan package (12). Besides, the package contains more postestimation functions that can be used to assess fitness of the model.

```
> install.packages("lavaan")
> library(lavaan)
> model.mort<- '
 #latent variable definition
 inflammation=~1*crp+pct+wbc
 orgdys=~scr+1*bil+oxyindex
 #regressions
orgdys~inflammation
mort~orgdys+inflammation
# variances
inflammation~~ inflammation
#intercept
orgdys~1;mort~1
'
> sem.mort<-lavaan(model.mort,data=data,ordered="mort",
auto.var=TRUE)
```

The model structure is specified with formula like expressions. It is specified as a literal string enclosed by single quotes as in the example above. The "=~" symbol links latent variable and indicators, which can be read as "is manifested by". For the structural model, regression equations are written for each dependent variable. The regression equation is similar to that in ordinary linear regression and is specified by the "~" operator. Independent variables are linked by "+" operator on the right side of the equation. Variance and covariance are specified using "~~" operator. In the example, I set auto.var=TRUE in the lavaan() function, letting residual variances and the variances of exogenous latent variables be included in the model and set free. Intercepts are specified in special case of regression equation that there is only the number "1" on the right of the equation. Intercepts represent the mean value of a dependent variable. Because the underlying structure is known in the example, model specification can be easy. Again we can draw a SEM diagram.

```
> semPaths(sem.mort,whatLabels="est")
```

The effect of inflammation on organ dysfunction is 0.39 (*Figure 4*), which is consistent with the value of 0.4 that we used for simulation. The coefficients of inflammation and orgdys on mortality are 0.03 and 0.05, respectively. By exponentiation, they approximate one as specified in the simulation. The parameters of interest (parameters of structural model), as well as corresponding statistics can be examined in the following way.

```
> Est <- parameterEstimates(sem.mort)
> subset(Est,op=="~")
  lhs   op rhs         est    se     z      pval-  ci.lower ci.upper
                                             ue
7 org-  ~  inflamma-  0.388  0.030  12.965 0      0.330    0.447
  dys      tion
8 mort  ~  orgdys     0.051  0.010  4.904  0      0.031    0.072
9 mort  ~  inflamma-  0.028  0.006  4.728  0      0.016    0.040
           tion
```

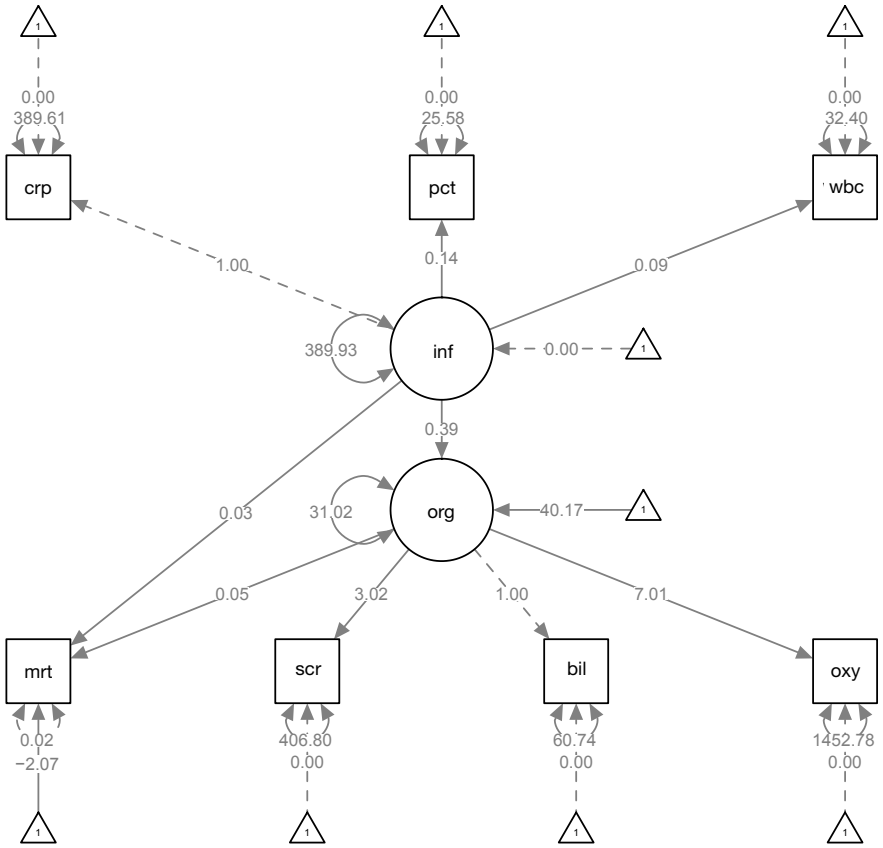


Figure 4 Diagram produced by passing a lavaan object to semPaths() function. The dependent outcome is binary. The effect of inflammation on organ dysfunction is 0.39, which is consistent with the value of 0.4 that we used for simulation. The coefficients of inflammation and orgdys on mortality are 0.03 and 0.05, respectively. By exponentiation, they approximate one as specified in the simulation. crp, c-reactive protein; pct, procalcitonin; wbc, white blood cell; bil, bilirubin; scr, serum creatinin; oxyindex, oxygen index.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. MacCallum RC, Austin JT. Applications of structural equation modeling in psychological research. Annu Rev Psychol 2000;51:201-226.
2. Schumacker RE, Lomax RG. A Beginner's Guide to Structural Equation Modeling. 3 edition. Abingdon, OX: Routledge, 2012.
3. Qiu H, Song Y, Zhao T. An overview on R packages for structural equation modeling. Res J Appl Sci Eng Technol 2014;7:4182-4186.
4. Kline RB. Principles and Practice of Structural Equation Modeling. New York: Guilford Press, 2011.
5. Bollen KA. Structural Equations with Latent Variables. 1 edition. Hoboken, NJ: Wiley-Interscience,1989.
6. Davis WR. The FC1 Rule of Identification for Confirmatory Factor Analysis: A General Sufficient Condition. Sociological Methods & Research 1993;21:403-437.
7. Rigdon EE. Identification of structural equation models with latent variables: A review of contributions by Bekker, Merckens, and Wansbeek. Structural Equation Modeling: A Multidisciplinary Journal 1997;4:80-85.
8. Rigdon EE. A Necessary and Sufficient Identification Rule for Structural Models Estimated in Practice. Multivariate Behav Res 1995;30:359-383.
9. Bollen KA, Harden JJ, Ray S, et al. BIC and Alternative Bayesian Information Criteria in the Selection of

Structural Equation Models. Structural Equation Modeling: A Multidisciplinary Journal 2014;21:1-19.
10. Epskamp S. semPlot: Unified Visualizations of Structural Equation Models. Structural Equation Modeling: A Multidisciplinary Journal 2015;22:474-483.
11. Fox J. TEACHER'S CORNER: Structural Equation Modeling With the sem Package in R. Structural Equation Modeling: A Multidisciplinary Journal 2006;13:465-486.
12. Rosseel Y. lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software 2012.48.

Cite this article as: Zhang Z. Structural equation modeling in the context of clinical research. Ann Transl Med 2017;5(5):102. doi: 10.21037/atm.2016.09.25

Case-crossover design and its implementation in R

Zhongheng Zhang

Abstract: Case-crossover design is a variation of case-control design that it employs persons' history periods as controls. Case-crossover design can be viewed as the hybrid of case-control study and crossover design. Characteristic confounding that is constant within one person can be well controlled with this method. The relative risk and odds ratio, as well as their 95% confidence intervals (CIs), can be estimated using Cochran-Mantel-Haenszel method. R codes for the calculation are provided in the main text. Readers may adapt these codes to their own task. Conditional logistic regression model is another way to estimate odds ratio of the exposure. Furthermore, it allows for incorporation of other time-varying covariates that are not constant within subjects. The model fitting per se is not technically difficult because there is well developed statistical package. However, it is challenging to convert original dataset obtained from case report form to that suitable to be passed to the clogit() function. R code for this task is provided and explained in the text.

Keywords: Case crossover; R; transient effect; risk ratio; conditional logistic regression; odds ratio

Submitted Mar 23, 2016. Accepted for publication Apr 21, 2016.
doi: 10.21037/atm.2016.05.42
View this article at: http://dx.doi.org/10.21037/atm.2016.05.42

Introduction

Case-control study is a basic study design in epidemiology. It includes all incident cases and a sample of non-cases. Thus, as compared to the cohort study that includes all cases and controls during study period, case-control study is suitable for studying rare disease. However, it is also criticized for its difficulty in controlling between-person confounding (1). Furthermore, case-control study investigates the cumulative effect of an exposure and it is difficult to disentangle acute transient effect from chronic effect. In response to these limitations, the case-crossover design was first developed by Maclure in 1991 (2). The same idea was introduced in a later paper (3). Since then, the case-crossover design has become increasingly popular in medical literature. By searching PubMed in April 2016 [searching strategy: case crossover (title/abstract)], a total of 1,044 citations were identified. The number of publications with case-crossover design increases exponentially in recent years (*Figure 1*). To assist clinicians become familiar with this design, this paper introduces some basic ideas and rationales behind the case-crossover design. R codes for calculations of risk ratio and its variance are present in the main text.

Understanding the case-crossover design

Case-crossover design uses all cases for study, and non-cases contribute nothing to the analysis. Because the effect of an exposure is transient, it defines a time window during which the risk of event is transiently elevated. After this window, the risk returns to the baseline level. The history preceding the event of interest is used as the controls. In this regard, case-crossover design can be viewed as matched case control design that controls are the same person before event occurs. Within each person, the person-time of exposure can be estimated by multiplying the frequency of exposure by a effect time window. Unexposed person-time can be estimated by subtracting exposed person-time from the total person-time (4). Schematic illustration of the case-crossover design is shown in *Figure 2*. All patients have the event of interest being observed. All patients have intermittent exposure of risks, and there is a transient effect time window during which the risk is altered. Only patient 2 has the event occurring within the effect time window. In reality, the triggers can be coffee intake, sexual activity, environmental temperature and PM2.5 air pollution. The outcome events can be myocardial infarction (MI), emergency room visit, and intracranial hemorrhage (5-7).

The Cochran-Mantel-Haenszel risk ratio can be written as:

$$RR_{CMH} = \frac{\sum_i \frac{a_i N_{0i}}{N_i}}{\sum_i \frac{c_i N_{1i}}{N_i}} \quad [1]$$

and the Cochran-Mantel-Haenszel odds ratio can be written as:

$$\mathrm{OR}_{\mathrm{CMH}} = \frac{\sum_i \frac{a_i d_i}{N_i}}{\sum_i \frac{c_i b_i}{N_i}} \quad [2]$$

where i is an indicator of the ith stratum, and a, b, c, d, N_1, N_0 and N are number of participants as shown in *Table 1*.

Because each case typically experiences one episode of events, either a or c is equal to 1; and the other is equal to zero. That is, this one episode of event occurs during either exposed or unexposed person-time.

The Cochran-Mantel-Haenszel risk ratio for case crossover study can be written as:

$$RR_{crossover} = \frac{\sum_i \frac{a_i N_{0i}}{T_i}}{\sum_i \frac{c_i N_{1i}}{T_i}} \quad [3]$$

where i is the indicator of the ith case. Either a or c is equal to 1; and the other is equal to zero. N_1 is exposed time, and N_0 is the unexposed time. T is the total time (*Table 2*). Because T is typically the same for all participants, it can be eliminated from both numerator and denominator.

$$RR_{crossover} = \frac{\sum_i a_i N_{0i}}{\sum_i c_i N_{1i}} \quad [4]$$

In the case-crossover design, we usually know the frequency of trigger activity (frq), total observation time (T), time from last trigger activity to the event (t).

The log variance of Cochran-Mantel-Haenszel RR can be written as (8):

$$\mathrm{var}\left[\ln\left(RR_{CMH}\right)\right] = \frac{\sum_i \left[\frac{(a_i + c_i)N_{1i}N_{0i}}{T^2} - \frac{a_i c_i}{T}\right]}{\left(\sum_i \frac{a_i N_{0i}}{T}\right)\left(\sum_i \frac{c_i N_{1i}}{T}\right)} \quad [5]$$

Because either a or c is 0, the last term of the numerator can be eliminated. The T^2 can be dropped from numerator and denominator. $a_i + c_i$ is deleted because it equals to one. The equation can be rewritten as:

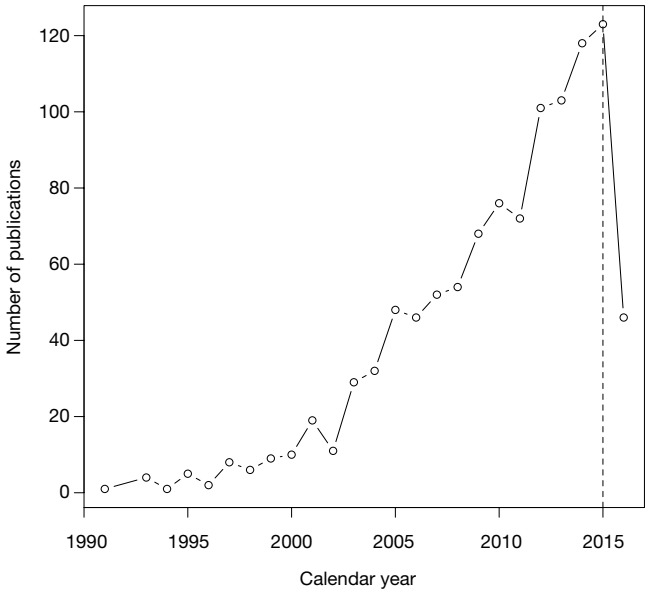


Figure 1 The number of publications with case-crossover design increases exponentially in recent years.

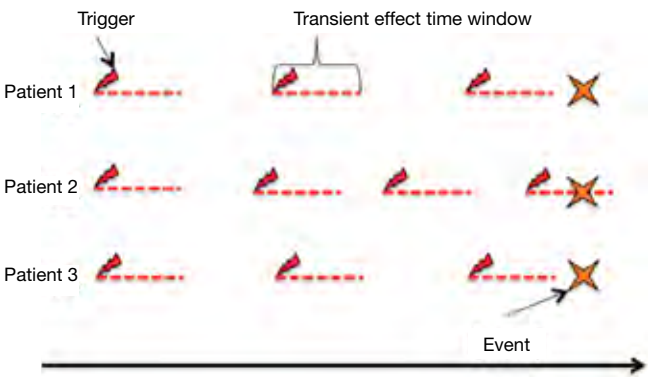


Figure 2 Schematic illustration of the case-crossover design. The effect of exposure to a risk factor is short lived. Risk factor or trigger is represented by the thunder symbol. The transient effect time is represented by the dashed line. A star denotes the event. Note that only patient 2 has event occurring within the effect time.

$$\mathrm{var}\left[\ln\left(RR_{CMH}\right)\right] = \frac{\sum_i N_{1i}N_{0i}}{\left(\sum_i a_i N_{0i}\right)\left(\sum_i c_i N_{1i}\right)} \quad [6]$$

The Cochran-Mantel-Haenszel odds ratio for case-crossover design can be written as:

$$OR_{crossover} = \frac{\sum_i \frac{a_i d_i}{T_i}}{\sum_i \frac{c_i b_i}{T_i}} \quad [7]$$

Table 1 Cross tabulation of risk factor exposure and outcome for the ith stratum

Variable	Events	Non-events	Total
Exposed	a	b	$c + d = N_1$
Unexposed	c	d	$c + d = N_0$
Total			N

Table 2 Cross tabulation of exposure time and events for the ith case

Variable	Events	Non-events	Total
Exposed time	a (efftime)	b (frq-efftime)	$c + d = N_1$ (frq)
Unexposed time	c (1-efftime)	d (T-frq-1+efftime)	$c + d = N_0$ (T-frq)
Total	[1]	(T-1)	T

Note: annotations within the parenthesis were variable names used in our calculations.

Similarly, this equation can be simplified as:

$$OR_{crossover} = \frac{\sum_i a_i d_i}{\sum_i c_i b_i} \quad [8]$$

The log variance of Cochran-Mantel-Haenszel OR can be written as:

$$\text{var}\left[\ln\left(RR_{CMH}\right)\right] = \frac{\sum_i a_i d_i (a_i + d_i)}{2\left(\sum_i a_i d_i\right)^2} + \frac{\sum_i a_i d_i (c_i + b_i) + c_i b_i (a_i + d_i)}{2\sum_i a_i d_i \sum_i c_i b_i} + \frac{\sum_i c_i b_i (c_i + b_i)}{2\left(\sum_i c_i b_i\right)^2} \quad [9]$$

Working example

We adapted the study by La Vecchia and colleagues investigating the association between coffee intake and MI (9). It is assumed that the transient effect of coffee lasts for one hour. The variable vectors for ten patients are generated by the following syntax:

```
> t<-c(9,1/3,3,22,6,7,12,5,0.5,24)
> frq<-c(730,365,36,1820,2920,24,730,730,3650,365)
> T<-365*24
> efftime<-ifelse(t>=1,0,1)
> rr<-sum(efftime*(T-frq))/sum((1-efftime)*frq)
> rr
[1] 1.836166
```

The first line generates a variable *t*, which represents the interval between the last time of coffee intake and MI (hour). The frequency of coffee intake is recorded as counts in the preceding year. The first one has 730 coffee intakes per year, corresponding to twice per day. *T* is the total number of hours in a year. Vector *efftime* is a tag variable denoting whether MI occurs within one hour after coffee intake. The last line calculates the risk ratio of MI for periods with coffee effect versus those without coffee effect. Then, we proceed to estimate 95% confidence interval (CI) for RR. The variance in log scale is calculated and then transformed to the original scale.

```
> var.log<-sum(frq*(T-frq))/(sum(efftime*(T-
frq))*sum((1-efftime)*frq))
> se.log<-sqrt(var.log)
> lo.log<-log(rr)-1.96*se.log
> hi.log<-log(rr)+1.96*se.log
> lo<-exp(lo.log)
> hi<-exp(hi.log)
```

The Cochran-Mantel-Haenszel OR and its variance can be calculated using the following R syntax:

```
> or<-sum(efftime*(T-frq-(1-efftime)))/sum((1-efftime)*(frq-
efftime))
> var.log.or<-(sum(efftime*(T-frq-1+efftime)*(efftime+
T-frq-1+efftime))/(2*(sum(efftime*(T-frq-
1+efftime)))^2))+(sum(efftime*(T-frq-1+efftime)*(1-
efftime+frq-efftime)+(1-efftime)*(frq-efftime)*(efftime+(T-
frq-1+efftime)))/(2*sum(efftime*(T-frq-1+efftime))*sum((1-
efftime)*(frq-efftime))))+(sum((1-efftime)*(frq-efftime)*(1-
efftime+frq-efftime))/(2*(sum((1-efftime)*(frq-efftime)))^2))
> se.log.or<-sqrt(var.log.or)
> lo.log.or<-log(or)-1.96*se.log.or
> hi.log.or<-log(or)+1.96*se.log.or
> lo.or<-exp(lo.log.or)
> hi.or<-exp(hi.log.or)
> matrix<-matrix(round(c(rr,lo,hi,or,lo.or,
hi.or),2),nrow=2,byrow=TRUE)
```

```
> rownames(matrix)<-c("RR","OR")
> colnames(matrix)<-c("Value","low 95% CI",
"high 95% CI")
> matrix
```

	Value	low 95% CI	high 95% CI
RR	1.84	0.34	9.81
OR	1.84	0.33	10.09

Conditional logistic regression

Since the case-crossover design can be viewed as matched case-control design with 1:M matched pairs, conditional logistic regression model can be utilized for the estimation of OR of the exposure of interest (4,10). However, the format of data frame described above is not suitable for regression modeling. Therefore the first step is to change the format of data frame, making it suitable for conditional regression analysis. In this example, the clogit() function contained in survival package is employed. The function requires that all person-times, including the exposed and unexposed, be regarded as an observation (e.g., each person-time takes one row). An *id* variable is used to distinguish between individual patients.

```
mat<-matrix(, nrow = T-1, ncol = 0)
for (i in 1:10) {
      if (T%%frq[i]==0) {
        exposure<-c(rep(c(1,rep(0,T/frq[i]-1)),frq[i]))[-T]
       }  else {
        exposure<-c(rep(c(1,rep(0,trunc(T/frq[i])-
1)),frq[i]),rep(0,T-frq[i]*trunc(T/frq[i])))[-T]
  }
   mat<-cbind(mat,exposure)
}
```

The first line creates a matrix with T-I rows and 0 column. It is an empty matrix. Then I create a for() loop to generate a matrix of exposed and unexposed person-times. In the *mat* matrix, each column represents one person. Because the first exposed time and its relation to the occurrence of MI are obtained via interview, it is isolated from the *mat* matrix. The recalled coffee drinking frequency in the preceding year is used to create the *mat* matrix. There are two kinds of persons. For the first one, the total person-time T is divisible by the exposed person-time (frq). The exposed person time can be equally spaced during the past one year. For the second one, the total person-time T is not divisible by the exposed person-time (frq). The if-else statement is used to do the task.

```
> commat<-rbind(efftime,mat)
> library(reshape2)
> data.wide<-as.data.frame(commat)
> colnames(data.wide)<-c(1:10)
> data<-melt(data.wide,measure=c(1:10))
> colnames(data)<-c("id","exposure")
> data$case<-rep(c(1,rep(0,T-1)),5)
>  head(data)
```

	id	exposure	case
1	1	0	1
2	1	1	0
3	1	0	0
4	1	0	0
5	1	0	0
6	1	0	0

The above codes combine the person-time matrix with the first interviewed exposure. Then vectors of persons are stacked into one column using melt() function (11). The final data frame contains only thee variables including *id*, *exposure* and *case*. In conditional logistic model, *id* is used to indicate matched pairs. Here *id* variable identify persons. The variable exposure denotes exposed [1] and unexposed [0] person-time. The variable *case* represents the case period and control period. As expected, there is only ten case periods.

The following codes perform conditional logistic regression analysis and its summary output.

```
library(survival)
> mod<- clogit(case~exposure+strata(id),data)
> summary(mod)
Call:
coxph(formula = Surv(rep(1, 87600L), case) ~ exposure +
strata(id),
   data = data, method = "exact")

 n= 87600, number of events= 10
```

	coef	exp(coef)	se(coef)	z	Pr(>|z|)
exposure	0.6392	1.8950	0.8960	0.713	0.476


```
            exp(coef) exp(-coef) lower .95 upper .95
exposure    1.895     0.5277     0.3273    10.97

Rsquare= 0   (max possible= 0.002 )
Likelihood ratio test= 0.47  on 1 df,   p=0.4934
Wald test            = 0.51  on 1 df,   p=0.4756
Score (logrank) test = 0.52  on 1 df,   p=0.4705
```

The first argument of clogit() function specifies the model structure. Differently from that in generalized linear model, there is a strata argument at the end of the equation. The strata() argument passes the *id* variable. The output shows that the estimated OR is 1.895 (95% CI: 0.327–10.97), which is similar to that estimated by Cochran-Mantel-Haenszel method.

Time trend adjustment with conditional logistic regression model

Case-crossover design uses subjects as their own control, and thus it is able to eliminate confounding characteristics that are constant within subject. However, there is time trend confounding that cannot be avoided by this method (2). In other words, exposure distribution in any time periods is not globally exchangeable within a person. For example, there is evidence showing that MI risk follows a circadian pattern (12). That is, time periods in the morning are not exchangeable to that at night. Here I create a clock variable to show how to adjust time-varying confounders with conditional logistic regression model.

```
> data$clock<-c(rep(1,T/(365*4)),rep(2,T/
(365*4)),rep(3,T/(365*4)),rep(4,T/(365*4)))
```

One day is divided into four clock time periods. Morning (6:00–12:00), afternoon (12:00–18:00), evening (18:00–24:00) and night (0:00–6:00) are denoted by 1, 2, 3 and 4, respectively.

```
> mod.adj<- clogit(case~exposure+clock+strata(id),data)
Warning message:
In fitter(X, Y, strats, offset, init, control, weights = weights,  :
 Loglik converged before variable  2 ; beta may be infinite.
> summary(mod.adj)
Call:
coxph(formula = Surv(rep(1, 87600L), case) ~ exposure +
clock +
  strata(id), data = data, method = "exact")

 n= 87600, number of events= 10

              coef      exp(coef)  se(coef)  z      Pr(>|z|)
exposure  9.301e-02  1.097e+00 8.362e-01  0.111  0.911
clock     -1.982e+01 2.473e-09  6.389e+03 -0.003 0.998

          exp(coef)  exp(-coef) lower .95  upper .95
exposure  1.097e+00  9.112e-01  0.2131     5.652
clock     2.473e-09  4.044e+08  0.0000     Inf

Rsquare= 0   (max possible= 0.002 )
Likelihood ratio test= 27.74  on 2 df,   p=9.479e-07
Wald test            = 0.01  on 2 df,   p=0.9938
Score (logrank) test = 18.14  on 2 df,   p=0.0001152
```

There is a warning message after running the clogit() function. That is because we have no data on the clock time of the occurrence of MI, and I assigned 1 to the *clock* variable for the first person-time, which is of course not true. However, this doesn't interfere the illustration of how to adjust time-varying covariates in the model. Risk variance attributable to clock time is expressed by OR.

Summary

Case-crossover design is a variation of case-control design that it employs persons' history periods as controls. Case-crossover design can be viewed as the hybrid of case-control study and crossover design. Characteristic confounding that is constant within one person can be well controlled with this method. The relative risk and odds ratio, as well as their 95% CIs, can be estimated using the Cochran-Mantel-Haenszel method. R codes for the calculation are provided in the main text. Readers may adapt these codes to their own task. Conditional logistic regression model is another way to estimate odds ratio of exposure. Furthermore, it allows for incorporation of other time-varying covariates that are not constant within subjects. The model fitting per se is not technically difficult because there is well developed statistical package. However, it is challenging to convert original dataset from case report form to that suitable to be

passed to clogit() function. R code for this task is provided and explained in the text.

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Schneeweiss S, Stürmer T, Maclure M. Case-crossover and case-time-control designs as alternatives in pharmacoepidemiologic research. Pharmacoepidemiol Drug Saf 1997;6 Suppl 3:S51-S59.
2. Maclure M. The case-crossover design: a method for studying transient effects on the risk of acute events. Am J Epidemiol 1991;133:144-153.
3. Feldmann U. Epidemiologic assessment of risks of adverse reactions associated with intermittent exposure. Biometrics 1993;49:419-428.
4. Mittleman M. Control sampling strategies for case-crossover studies: An assessment of relative efficiency. Am J Epidemiol 1995;142:91-98.
5. Weichenthal S, Lavigne E, Evans G, et al. Ambient PM2.5 and risk of emergency room visits for myocardial infarction: impact of regional PM2.5 oxidative potential: a case-crossover study. Environ Health 2016;15:46.
6. Weichenthal SA, Lavigne E, Evans GJ, et al. PM2.5 and Emergency Room Visits for Respiratory Illness: Effect Modification by Oxidative Potential. Am J Respir Crit Care Med 2016;193:594-596.
7. Zheng D, Arima H, Sato S, et al. Low Ambient Temperature and Intracerebral Hemorrhage: The INTERACT2 Study. Wang X, editor. PLoS One 2016;11:e0149040.
8. Rothman KJ, Greenland S, Lash TL, editors. Modern epidemiology. Third, Mid-cycle revision edition. Philadelphia: Lippincott Williams and Wilkins, 2012:758.
9. La Vecchia C, Gentile A, Negri E, et al. Coffee consumption and myocardial infarction in women. Am J Epidemiol 1989;130:481-485.
10. Marshall RJ, Jackson RT. Analysis of case-crossover designs. Stat Med 1993;12:2333-2341.
11. Zhang Z. Reshaping and aggregating data: an introduction to reshape package. Ann Transl Med 2016;4:78.
12. Takeda N, Maemura K. Circadian clock and the onset of cardiovascular events. Hypertens Res 2016. [Epub ahead of print].

Cite this article as: Zhang Z. Case-crossover design and its implementation in R. Ann Transl Med 2016;4(18):341. doi: 10.21037/atm.2016.05.42

乐九医疗
LE9HEALTH

医疗大数据应用专家

核心竞争力

- 国内领先的医疗数据集成、融合、建模和算法技术
- 卓越的临床科研平台构建能力，百家三甲医院临床科研平台解决方案经验沉淀
- 全面提升临床研究的人群查找、数据收集效率，节省95%的时间和工作量

高效的专业团队

部分成果清单

- 临床决策系统

基于相似病人的高血压用药推荐系统

基于MedRank算法的药物推荐系统

- 临床科研成果

Preprocedural Prediction Model for Contrast-Induced Nephropathy Patients (IF=5.12)
Low Thyroid Hormone in Early Pregnancy Is Associated With an Increased Risk of Gestational Diabetes Mellitus (IF=5.53)
Value of Neutrophil Counts in Predicting Surgery-Related Acute Kidney Injury and the Interaction of These Counts With Diabetes in Chronic Kidney Disease Patients With Hypertension (IF=5.82)
A Predictive Model for Assessing Surgery Related Acute Kidney Injury Risk in Hypertensive Patients: A Retrospective Cohort Study (IF=3.06)

联系我们
021-54265596
sales@le9health.com
www.le9health.com
上海市徐汇区古美路1515号19单元凤凰大厦19层


Vol 9, No 6 June 2017
ISSN 2072-1439
2016
IMPACT FACTOR
2.365
JOURNAL of THORACIC DISEASE
jtd.amegroups.com
The Official Publication of
AME
Journal of Thoracic Disease


TRANSLATIONAL
CANCER RESEARCH
ISSN 2218-676X
VOL 6, NO 4
AUG 2017
2016
IMPACT FACTOR
1.167
AME
Translational Cancer Research

HBSN

Indexed by SCIE

AME Publishing Company

HEPATOBILIARY SURGERY AND NUTRITION

Editor-in-Chief:

Yilei Mao, MD, PhD

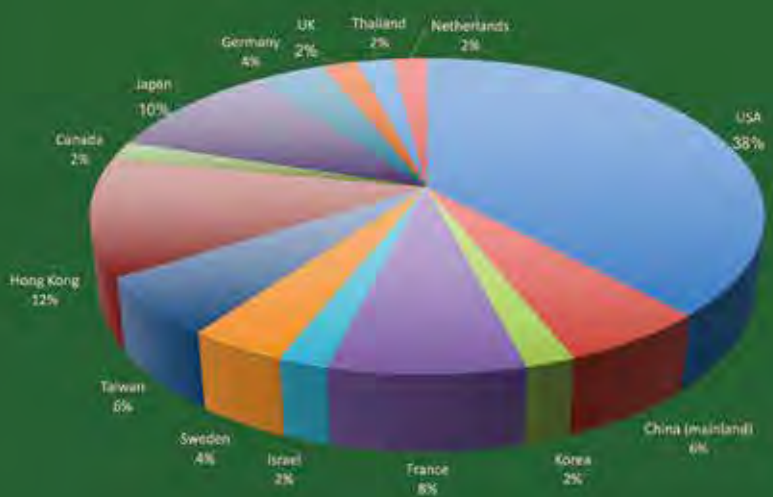
Department of Liver Surgery, Peking Union Medical College Hospital, Chinese Academy of Medical Sciences, Beijing, China

Open Access

Peer-reviewed

Bi-monthly Publication

Indexed in PubMed/PMC/SCIE



Distribution of Editorial Board Members of HBSN

hbsn.amegroups.com

QIMS Quantitative Imaging in Medicine and Surgery

Indexed by SCIE

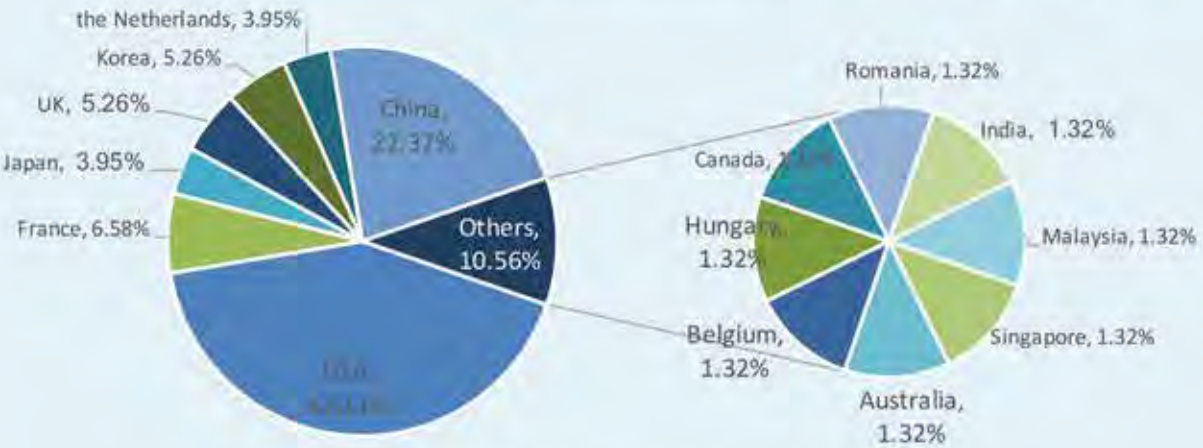
Editor-in-Chief:

Yi-Xiang Wang, MD

The Chinese University of Hong Kong, Hong Kong SAR. Beijing, China

Open access & Peer-reviewed
Bi-monthly Publication
Indexed in PubMed/Scopus/SCIE

QIMS is endorsed by Nanotechnology Cancer Asia-Pacific (NCAP) Network & Macau Radiology Association.



Geographical Distribution of Editorial Board Members

AME Publishing Company

qims.amegroups.com


Print ISSN 2305-5839
Online ISSN 2305-5847
Vol 3, No 20 Nov 2015
ANNALS OF TRANSLATIONAL MEDICINE
inspire2live