Hui Lin and Ming Li

Introduction to Data Science

Contents

List of Tables

List of Figures

Preface

During the first couple years of our career as data scientists, we were bewildered by all kinds of data science hype. There is a lack of definition of many basic terminologies such as "Big Data" and "Data Science." How big is big? If someone ran into you asked what data science was all about, what would you tell them? What is the difference between the sexy role "Data Scientist" and the traditional "Data Analyst"? How suddenly came all kinds of machine algorithms? All those struck us as confusing and vague as real-world data scientists! But we always felt that there was something real there. After applying data science for many years, we explored it more and had a much better idea about data science. And this book is our endeavor to make data science to a more legitimate field.

Goal of the Book

This is an introductory book to data science with a specific focus on the application. Data Science is a cross-disciplinary subject involving hands-on experience and business problem-solving exposures. The majority of existing introduction books on data science are about the modeling techniques and the implementation of models using R or Python. However, they fail to introduce data science in a context of the industrial environment. Moreover, a crucial part, the art of data science in practice, is often missing. This book intends to fill the gap.

Some key features of this book are as follows:

- It is comprehensive. It covers not only technical skills but also soft skills and big data environment in the industry.

- It is hands-on. We provide the data and repeatable R and Python code. You can repeat the analysis in the book using the data and code provided. We also suggest you perform the analyses with your data whenever possible. You can only learn data science by doing it!
- It is based on context. We put methods in the context of industrial data science questions.
- Where appropriate, we point you to more advanced materials on models to dive deeper

Who This Book Is For

Non-mathematical readers will appreciate the emphasis on problem-solving with real data across a wide variety of applications and the reproducibility of the companion R and python code.

Readers should know basic statistical ideas, such as correlation and linear regression analysis. While the text is biased against complex equations, a mathematical background is needed for advanced topics.

What This Book Covers

Based on industry experience, this book outlines the real world scenario and points out pitfalls data science practitioners should avoid. It also covers big data cloud platform and the art of data science such as soft skills. We use R as the main tool and provide code for both R and Python.

Conventions

Acknowledgements

1

Introduction

Interest in data science is at an all-time high and has exploded in popularity in the last couple of years. Data scientists today are from various backgrounds. If someone ran into you asked what data science was all about, what would you tell them? It is not easy to answer. Data science is one of the areas where if you ask ten people you get ten different answers. It is not well-defined as an academic subject but broadly used in the industry. Media has been hyping about "Data Science" "Big Data" and "Artificial Intelligence" over the fast few years. With the data science hype picking up stream, many professionals changed their titles to "Data Scientist" without any of the necessary qualifications. Your first reaction to all of this might be some combination of skepticism and confusion. We want to address this up front that: we had that exact reaction. To make things clear, let's start with the fundamental question.

1.1 What is data science?

David Donoho (Donoho, 2015) summarizes in "50 Years of Data Science" the main recurring "Memes" about data sciences:

1. The 'Big Data' Meme
2. The 'Skills' Meme
3. The 'Jobs' Meme

Everyone should have heard about big data. Data science trainees now need the skills to cope with such big data sets. What are those skills? You may hear about: Hadoop, a system using Map/Reduce to process large data sets distributed across a cluster of computers. The new skills are for

dealing with organizational artifacts of large-scale cluster computing but not for better solving the real problem. A lot of data on its own is worthless. It isn't the size of the data that's important. It's what you do with it. The big data skills that so many are touting today are not skills for better solving the real problem of inference from data.

We are transiting to universal connectivity with a deluge of data filling telecom servers. But these facts don't immediately create a science. The statisticians and computer scientists have been laying the groundwork for data science for at least 50 years. Today's data science is an enlargement and combination of statistics and computer science rather than a brand new discipline.

Data Science doesn't come out of the blue. Its predecessor is data analysis. Back in 1962, John Tukey wrote in "The Future of Data Analysis":

> For a long time I have thought I was a statistician, interested in inferences from the particular to the general. But as I have watched mathematical statistics evolve, I have had cause to wonder and to doubt. ...All in all, I have come to feel that my central interest is in data analysis, which I take to include, among other things: procedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analyzing data.

It deeply shocked his academic readers. Aren't you supposed to present something mathematically precise, such as definitions, theorems, and proofs? If we use one sentence to summarize what John said, it is:

> data analysis is more than mathematics.

In September 2015, the University of Michigan made plans to invest $100 million over the next five years in a new Data Science Initiative (DSI) that will enhance opportunities for student and faculty researchers across the university to tap into the enormous potential of big data. How does DSI define Data science? Their website gives us an idea:

> "This coupling of scientific discovery and practice involves the collection, management, processing, analysis, visualization, and interpretation of vast amounts of heterogeneous data associated with a diverse array of scientific, translational, and interdisciplinary applications."

How about data scientist? Here is a list of somewhat whimsical definitions for a "data scientist":

- "A data scientist is a data analyst who lives in California"
- "A data scientist is someone who is better at statistics than any software engineer and better at software engineering than any statistician."
- "A data scientist is a statistician who lives in San Francisco."
- "Data Science is statistics on a Mac."

There is lots of confusion between Data Scientist, Statistician, Business/Financial/Risk(etc.) Analyst and BI professional due to the apparent intersections among skillsets. We see data science as a discipline to make sense of data. The techniques and methodologies of data science stem from the fields of computer science and statistics. One of the most well-cited diagrams describing the area comes from Drew Conway where he suggested data science is the intersection of hacking skills, math and stats knowledge, and substantial expertise. This diagram might be a bit of an oversimplification, but it's a great start.

There are almost as many definitions of data science as there are data scientists. Instead of listing some of these definitions, it might be more informative to let the subject matter define the field.

Let's start from a brief history of data science. If you hit up the Google Trends website which shows search keyword information over time and

check the term "data science," you will find the history of data science goes back a little further than 2004. From the way media describes it, you may feel machine learning algorithms were just invented last month, and there was never "big" data before Google. That is not true. There are new and exciting developments of data science, but many of the techniques we are using are based on decades of work by statisticians, computer scientists, mathematicians and scientists of all types.

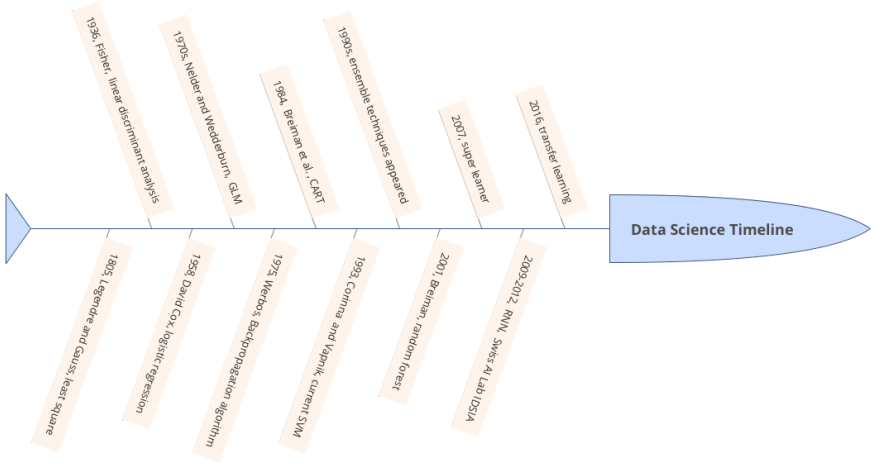


FIGURE 1.1: Data Science Timeline

In the early 19th century when Legendre and Gauss came up the least squares method for linear regression, only physicists would use it to fit linear regression. Now, even non-technical people can fit linear regressions using excel. In 1936 Fisher came up with linear discriminant analysis. In the 1940s, we had another widely used model – logistic regression. In the 1970s, Nelder and Wedderburn formulated "generalized linear model (GLM)" which:

> "generalized linear regression by allowing the linear model to be related to the response variable via a link function and by allowing the magnitude of the variance of each measurement to be a function of its predicted value." [from Wikipedia]

By the end of the 1970s, there was a range of analytical models and most of them were linear because computers were not powerful enough to fit non-linear model until the 1980s.

In 1984 Breiman et al. introduced classification and regression tree (CART) which is one of the oldest and most utilized classification and regression techniques. After that Ross Quinlan came up with more tree algorithms such as ID3, C4.5, and C5.0. In the 1990s, ensemble techniques (methods that combine many models' predictions) began to appear. Bagging is a general approach that uses bootstrapping in conjunction with any regression or classification model to construct an ensemble. Based on the ensemble idea, Breiman came up with random forest in 2001. Later, Yoav Freund and Robert Schapire came up with the AdaBoost.M1 algorithm. Benefiting from the increasing availability of digitized information, and the possibility to distribute that via the internet, the toolbox has been expanding fast. The applications include business, health, biology, social science, politics, etc.

John Tukey identified four forces driving data analysis (there was no "data science" then):

1. The formal theories of math and statistics
2. Acceleration of developments in computers and display devices
3. The challenge, in many fields, of more and ever larger bodies of data
4. The emphasis on quantification in an ever wider variety of disciplines

Tukey's 1962 list is surprisingly modern. Let's inspect those points in today's context. There is always a time gap between a theory and its application. We had the theories much earlier than application. Fortunately, for the past 50 years, statisticians have been laying the theoretical groundwork for constructing "data science" today. The development of computers enables us to calculate much faster and deliver results in a friendly and intuitive way. The striking transition to the internet of things generates vast amounts of commercial data. Industries have also sensed the

value of exploiting that data. Data science seems certain to be a major preoccupation of commercial life in coming decades. All the four forces John identified exist today and have been driving data science.

1.2 What kind of questions can data science solve?

1.2.1 Prerequisites

Data science is not a panacea, and data scientists are not magicians. There are problems data science can't help. It is best to make a judgment as early in the analytical cycle as possible. Tell your clients honestly and clearly when you figure data analytics can't give the answer they want. What kind of questions can data science solve? What are the requirements for our question?

1. Your question needs to be specific enough

Look at two examples:

- Question 1: How can I increase product sales?
- Question 2: Is the new promotional tool introduced at the beginning of this year boosting the annual sales of P1197 in Iowa and Wisconsin? (P1197 is an impressive corn seed product from DuPont Pioneer)

It is easy to see the difference between the two questions. Question 1 is a grammatically correct question, but it is proper for data analysis to answer. Why? It is too general. What is the response variable here? Product sales? Which product? Is it annual sales or monthly sales? What are the candidate predictors? You nearly can't get any useful information from the questions. In contrast, question 2 is much more specific. From the analysis point of view, the response variable is clearly "annual sales of P1197 in Iowa and Wisconsin". Even we don't know all the predictors, but the variable of interest is "the new promotional tool introduced early this year." We want to study the impact of the promotion of the sales. You can

start from there and move on to figure out other variables need to include in the model by further communication.

As a data scientist, you may start with something general and unspecific like question 1 and eventually get to question 2. Effective communication and in-depth domain knowledge about the business problem are essential to convert a general business question into a solvable analytical problem. Domain knowledge helps data scientist communicate with the language the other people can understand and obtain the required information.

However, defining the question and variables involved don't guarantee that you can answer it. I have encountered a well-defined supply chain problem. My client asked about the stock needed for a product in a particular area. Why can not this question be answered? I did fit a Multivariate Adaptive Regression Spline (MARS) model and thought I found a reasonable solution. But it turned out later that the data they gave me was inaccurate. In some areas, only estimates of past supply figures were available. The lesson lends itself to the next point.

2. You need to have sound and relevant data

One cannot make a silk purse out of a sow's ear. Data scientists need data, sound and relevant data. The supply problem is a case in point. There was relevant data, but not sound. All the later analytics based on that data was a building on sand. Of course, data nearly almost have noise, but it has to be in a certain range. Generally speaking, the accuracy requirement for the independent variables of interest and response variable is higher than others. In question 2, it is data related to the "new promotion" and "sales of P1197".

The data has to be helpful for the question. If you want to predict which product consumers are most likely to buy in the next three months, you need to have historical purchasing data: the last buying time, the amount of invoice, coupons and so on. Information about customers' credit card number, ID number, the email address is not going to help.

Often the quality of the data is more important than the quantity, but the quantity cannot be overlooked. In the premise of guaranteeing qual-

ity, usually the more data, the better. If you have a specific and reasonable question, also sound and relevant data, then congratulations, you can start playing data science!

1.2.2 Problem type

Many of the data science books classify the various models from a technical point of view. Such as supervised vs. unsupervised models, linear vs. nonlinear models, parametric models vs. non-parametric models, and so on. Here we will continue on "problem-oriented" track. We first introduce different groups of real problems and then present which models can be used to answer the corresponding category of questions.

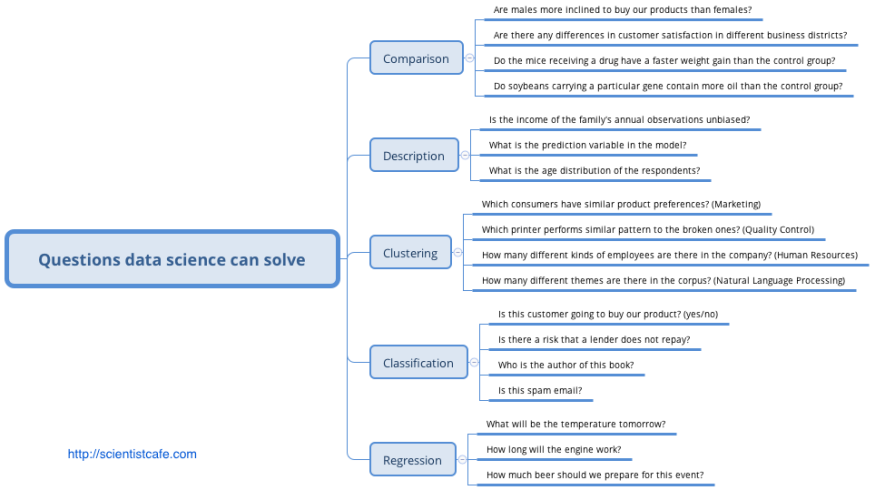


FIGURE 1.2: Data Science Questions

1. Comparison

The first common problem is to compare different groups. Such as: Is A better in some way than B? Or more comparisons: Is there any difference among A, B, C in a certain aspect? Here are some examples:

- Are the purchasing amounts different between consumers receiving coupons and those without coupons?

- Are males more inclined to buy our products than females?
- Are there any differences in customer satisfaction in different business districts?
- Do the mice receiving a drug have a faster weight gain than the control group?
- Do soybeans carrying a particular gene contain more oil than the control group?

For those problems, it is usually to start exploring from the summary statistics and visualization by groups. After a preliminary visualization, you can test the differences between treatment and control group statistically. The commonly used statistical tests are chi-square test, t-test, and ANOVA. There are also methods using Bayesian methods. In biology industry, such as new drug development, crop breeding, mixed effect models are the dominant technique.

2. Description

In the problem such as customer segmentation, after you cluster the sample, the next step is to figure out the profile of each class by comparing the descriptive statistics of the various variables. Questions of this kind are:

- Is the income of the family's annual observations unbiased?
- What is the mean/variance of the monthly sales volume of a product in different regions?
- What is the difference in the magnitude of the variable? (Decide whether the data needs to be standardized)
- What is the prediction variable in the model?
- What is the age distribution of the respondents?

Data description is often used to check data, find the appropriate data preprocessing method, and demonstrate the model results.

3. Clustering

Clustering is a widespread problem, which is usually related to classification. Clustering answers questions like:

- Which consumers have similar product preferences? (Marketing)

- Which printer performs similar pattern to the broken ones? (Quality Control)
- How many different kinds of employees are there in the company? (Human Resources)
- How many different themes are there in the corpus? (Natural Language Processing)

Note that clustering is unsupervised learning. The most common clustering algorithms include K-Means and Hierarchical Clustering.

4. Classification

Usually, a labeled sample set is used as a training set to train the classifier. Then the classifier is used to predict the category of a future sample. Here are some example questions:

- Is this customer going to buy our product? (yes/no)
- Is there a risk that a lender does not repay?
- Who is the author of this book?
- Is this spam email?

There are hundreds of classifiers. In practice, we do not have to try all the models as long as we fit in several of the best models in most cases.

5. Regression

In general, regression deals with the problem of "how much is it?" and return a numerical answer. In some cases, it is necessary to coerce the model results to be 0, or round the result to the nearest integer. It is the most common problem.

- What will be the temperature tomorrow?
- What will be the company's sales in the fourth quarter of this year?
- How long will the engine work?
- How much beer should we prepare for this event?

1.3 Data Scientist Skill Set

We talked about the bewildering definitions of data scientist. What are the required skills for a data scientist?

- Educational Background

Most of the data scientists today have undergraduate or higher degree from one of the following areas: computer science, electronic engineering, mathematics or statistics. According to a 2017 survey, 25% of US data scientists have a Ph.D. degree, 64% have a Master's degree, and 11% are Bachelors.

- Database Skills

Data scientists in the industry need to use SQL to pull data from the database. So it is necessary to be familiar with how data is structured and how to do basic data manipulation using SQL. Many statistics/mathematics students do not have experience with SQL in school. Don't worry. If you are proficient in one programming language, it is easy to pick up SQL. The main purpose of graduate school should be to develop the ability to learn and analytical thinking rather than the technical skills. Even the technical skills are necessary to enter the professional area. Most of the skills needed at work are not taught in school.

- Programming Skills

Programming skills are critical for data scientists. According to a 2017 survey from Burtch Works[1], 97% of the data scientists today using R or Python. We will focus on R in this book, but both are great tools for data science. There is not one "have-to-use" tool. The goal is to solve the problem not which tool to choose. However, a good tool needs to be flexible and scalable.

- Modeling Skills

Data scientists need to know statistical and machine learning models. There is no clear line separating these two. Many statistical models are

[1] http://www.burtchworks.com/2017/06/19/2017-sas-r-python-flash-survey-results/

also machine learning models, vice versa. Generally speaking, a data scientist is familiar with basic statistical tests such as t-test, chi-square test, and analysis of variance. They can explain the difference between Spearman rank correlation and Pearson correlation, be aware of basic sampling schemes, such as Simple Random Sampling, Stratified Random Sampling, and Multi-Stage Sampling. Know commonly used probability distributions such as Normal distribution, Binomial distribution, Poisson distribution, F distribution, T distribution, and Chi-square distribution. Experimental design plays a significant role in the biological study. Understanding the main tenants of Bayesian methods is necessary (at least be able to write the Bayes theorem on the whiteboard and know what does it mean). Know the difference between supervised and unsupervised learning. Understand commonly used cluster algorithms, classifiers, and regression models. Some powerful tools in predictive analytics are tree models (such as random forest and AdaBoost) and penalized model (such as lasso and SVM). Data scientist working on social science (such as consumer awareness surveys), also needs to know the latent variable model, such as exploratory factor analysis, confirmatory factor analysis, structural equation model.

Is the list getting a little scary? It can get even longer. Don't worry if you don't know all of them now. You will learn as you go. Standard mathematics, statistics or computer science training in graduate school can get you started. But you have to learn lots of new skills after school. Learning is happening increasingly outside of formal educational settings and in unsupervised environments. An excellent data scientist must be a lifetime learner. Fortunately, technological advantages provide new tools and opportunities for lifetime learners, MOOC, online data science workshops and various online tutorials. So above all, **self-learning ability** is the most critical skill.

- Soft Skills

In addition to technical knowledge, there are some critical soft skills. These include the ability to translate practical problems into data problems, excellent communication skill, attention to detail, storytelling and so on. We will discuss it in a later chapter in more detail.

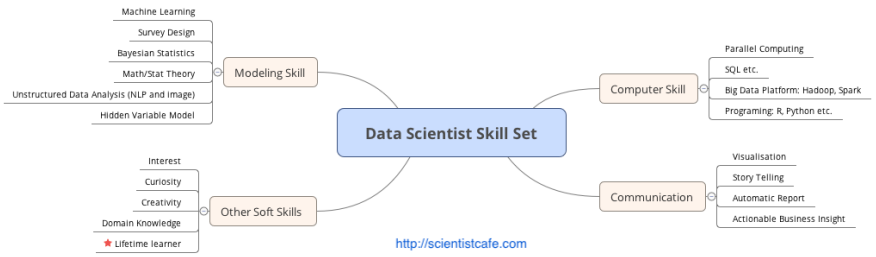


FIGURE 1.3: Data Scientist Skill Set

1.4 Types of Learning

There are three broad groups of styles: supervised learning, reinforcement learning, and unsupervised learning.

In supervised learning, each observation of the predictor measurement(s) corresponds to a response measurement. There are two flavors of supervised learning: regression and classification. In regression, the response is a real number such as the total net sales in 2017, or the yield of corn next year. The goal is to approximate the response measurement as much as possible. In classification, the response is a class label, such as dichotomous response such as yes/no. The response can also have more than two categories, such as four segments of customers. A supervised learning model is a function that maps some input variables with corresponding parameters to a response y. Modeling tuning is to adjust the value of parameters to make the mapping fit the given response. In other words, it is to minimize the discrepancy between given response and the model output. When the response y is a real value, it is intuitive to define discrepancy as the squared difference between model output and given the response. When y is categorical, there are other ways to measure the difference, such as AUC or information gain.

In reinforcement learning, the correct input/output pairs are not present. The model will learn from a sequence of actions and select the action maximizing the expected sum of the future rewards. There is a discount fac-

tor that makes future rewards less valuable than current rewards. Reinforcement learning is difficult for the following reasons:

(1) The rewards are not instant. If the action sequence is long, it is hard to know which action was wrong.

(2) The rewards are occasional. Each reward does not supply much information, so its impact of parameter change is limited. Typically, it is not likely to learn a large number of parameters using reinforcement learning. However, it is possible for supervised and unsupervised learning. The number of parameters in reinforcement learning usually range from dozens to maybe 1,000, but not millions.

In unsupervised learning, there is no response variable. For a long time, the machine learning community overlooked unsupervised learning except for one called clustering. Moreover, many researchers thought that clustering was the only form of unsupervised learning. One reason is that it is hard to define the goal of unsupervised learning explicitly. Unsupervised learning can be used to do the following:

(1) Identify a good internal representation or pattern of the input that is useful for subsequent supervised or reinforcement learning, such as finding clusters.

(2) It is a dimension reduction tool that is to provide compact, low dimensional representations of the input, such as factor analysis.

(3) Provide a reduced number of uncorrelated learned features from original variables, such as principal component regression.

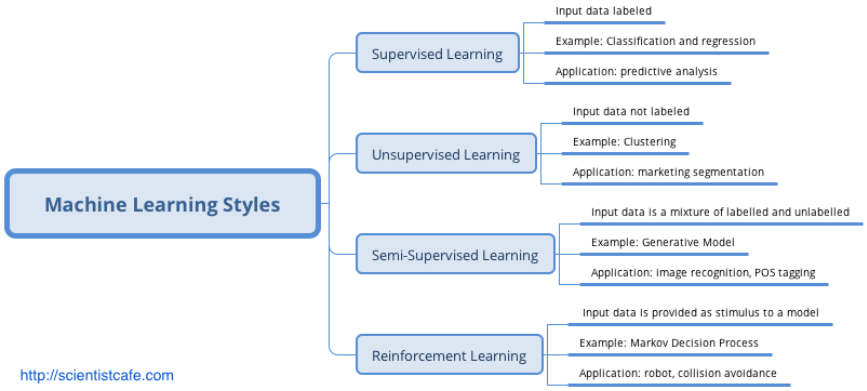


FIGURE 1.4: Machine Learning Styles

1.5 Types of Algorithm

The categorization here is based on the structure (such as tree model, Regularization Methods) or type of question to answer (such as regression).[2] It is far less than perfect but will help to show a bigger map of different algorithms. Some can be legitimately classified into multiple categories, such as support vector machine (SVM) can be a classifier, and can also be used for regression. So you may see other ways of grouping. Also, the following summary does not list all the existing algorithms (there are just too many).

1. Regression

Regression can refer to the algorithm or a particular type of problem. It is supervised learning. Regression is one of the oldest and most widely used statistical models. It is often called the statistical machine learning method. Standard regression models are:

[2]The summary of various algorithms for data science in this section is based on Jason Brownlee's blog "(A Tour of Machine Learning Algorithms)[http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/]." We added and subtracted some algorithms in each category and gave additional comments.

- Ordinary Least Squares Regression
- Logistic Regression
- Multivariate Adaptive Regression Splines (MARS)
- Locally Estimated Scatterplot Smoothing (LOESS)

The least squares regression and logistic regression are traditional statistical models. Both of them are highly interpretable. MARS is similar to neural networks and partial least squares (PLS) in the respect that they all use surrogate features instead of original predictors.

They differ in how to create the surrogate features. PLS and neural networks use linear combinations of the original predictors as surrogate features [3]. MARS creates two contrasted versions of a predictor by a truncation point. And LOESS is a non-parametric model, usually only used in visualization.

2. Similarity-based Algorithms

This type of model is based on a similarity measure. There are three main steps: (1) compare the new sample with the existing ones; (2) search for the closest sample; (3) and let the response of the nearest sample be used as the prediction.

- K-Nearest Neighbour [KNN]
- Learning Vector Quantization [LVQ]
- Self-Organizing Map [SOM]

The biggest advantage of this type of model is that they are intuitive. K-Nearest Neighbour is generally the most popular algorithm in this set. The other two are less common. The key to similarity-based algorithms is to find an appropriate distance metric for your data.

3. Feature Selection Algorithms

The primary purpose of feature selection is to exclude non-information

[3] To be clear on neural networks, the linear combinations of predictors are put through non-linear activation functions, deeper neural networks have many layers of non-linear transformation

or redundant variables and also reduce dimension. Although it is possible that all the independent variables are significant for explaining the response. But more often, the response is only related to a portion of the predictors. We will expand the feature selection in detail later.

- Filter method
- Wrapper method
- Embedded method

Filter method focuses on the relationship between a single feature and a target variable. It evaluates each feature (or an independent variable) before modeling and selects "important" variables.

Wrapper method removes the variable according to particular law and finds the feature combination that optimizes the model fitting by evaluating a set of feature combinations. In essence, it is a searching algorithm.

Embedding method is part of the machine learning model. Some model has built-in variable selection function such as lasso, and decision tree.

4. Regularization Method

This method itself is not a complete model, but rather an add-on to other models (such as regression models). It appends a penalty function on the criteria used by the original model to estimate the variables (such as likelihood function or the sum of squared error). In this way, it penalizes model complexity and contracts the model parameters. That is why people call them "shrinkage method." This approach is advantageous in practice.

- Ridge Regression
- Least Absolute Shrinkage and Selection Operator (LASSO)
- Elastic Net

5. Decision Tree

Decision trees are no doubt one of the most popular machine learning algorithms. Thanks to all kinds of software, implementation is a no-brainer which requires nearly zero understanding of the mechanism. The followings are some of the common trees:

- Classification and Regression Tree (CART)
- Iterative Dichotomiser 3 (ID3)
- C4.5
- Random Forest
- Gradient Boosting Machines (GBM)

6. Bayesian Models

People usually confuse Bayes theorem with Bayesian models. Bayes theorem is an implication of probability theory which gives Bayesian data analysis its name.

$$Pr(\theta|y) = \frac{Pr(y|\theta)Pr(\theta)}{Pr(y)}$$

The actual Bayesian model is not identical to Bayes theorem. Given a likelihood, parameters to estimate, and a prior for each parameter, a Bayesian model treats the estimates as a purely logical consequence of those assumptions. The resulting estimates are the posterior distribution which is the relative plausibility of different parameter values, conditional on the observations. The Bayesian model here is not strictly in the sense of Bayesian but rather model using Bayes theorem.

- Naïve Bayes
- Averaged One-Dependence Estimators (AODE)
- Bayesian Belief Network (BBN)

7. Kernel Methods

The most common kernel method is the support vector machine (SVM). This type of algorithm maps the input data to a higher order vector space where classification or regression problems are easier to solve.

- Support Vector Machine (SVM)
- Radial Basis Function (RBF)
- Linear Discriminate Analysis (LDA)

8. Clustering Methods

Like regression, when people mention clustering, sometimes they mean a class of problems, sometimes a class of algorithms. The clustering algorithm usually clusters similar samples to categories in a centroidal or hierarchical manner. The two are the most common clustering methods:

- K-Means
- Hierarchical Clustering

9. Association Rule

The basic idea of an association rule is: when events occur together more often than one would expect from their rates of occurrence, such co-occurrence is an interesting pattern. The most used algorithms are:

- Apriori algorithm
- Eclat algorithm

10. Artificial Neural Network

The term neural network has evolved to encompass a repertoire of models and learning methods. There has been lots of hype around the model family making them seem magical and mysterious. A neural network is a two-stage regression or classification model. The basic idea is that it uses linear combinations of the original predictors as surrogate features, and then the new features are put through non-linear activation functions to get hidden units in the 2nd stage. When there are multiple hidden layers, it is called deep learning, another over hyped term. Among varieties of neural network models, the most widely used "vanilla" net is the single hidden layer back-propagation network.

- Perceptron Neural Network
- Back Propagation
- Hopield Network
- Self-Organizing Map (SOM)
- Learning Vector Quantization (LVQ)

11. Deep Learning

The name is a little misleading. As mentioned before, it is multilayer neu-

ral network. It is hyped tremendously especially after AlphaGO defeated Li Shishi at the board game Go. We don't have too much experience with the application of deep learning and are not in the right position to talk more about it. Here are some of the common algorithms:

- Restricted Boltzmann Machine (RBN)
- Deep Belief Networks (DBN)
- Convolutional Network
- Stacked Autoencoders
- Long short-term memory (LSTM)

12. Dimensionality Reduction

Its purpose is to construct new features that have significant physical or statistical characteristics, such as capturing as much of the variance as possible.

- Principle Component Analysis (PCA)
- Partial Least Square Regression (PLS)
- Multi-Dimensional Scaling (MDS)
- Exploratory Factor Analysis (EFA)

PCA attempts to find uncorrelated linear combinations of original variables that can explain the variance to the greatest extent possible. EFA also tries to explain as much variance as possible in a lower dimension. MDS maps the observed similarity to a low dimension, such as a two-dimensional plane. Instead of extracting underlying components or latent factors, MDS attempts to find a lower-dimensional map that best preserves all the observed similarities between items. So it needs to define a similarity measure as in clustering methods.

13. Ensemble Methods

Ensemble method made its debut in the 1990s. The idea is to build a prediction model by combining the strengths of a collection of simpler base models. Bagging, originally proposed by Leo Breiman, is one of the earliest ensemble methods. After that, people developed Random Forest (T, 1998; Y and D, 1997) and Boosting method (L, 1984; M and L, 1989). This is a class of powerful and effective algorithms.

- Bootstrapped Aggregation (Bagging)
- Random Forest
- Gradient Boosting Machine (GBM)

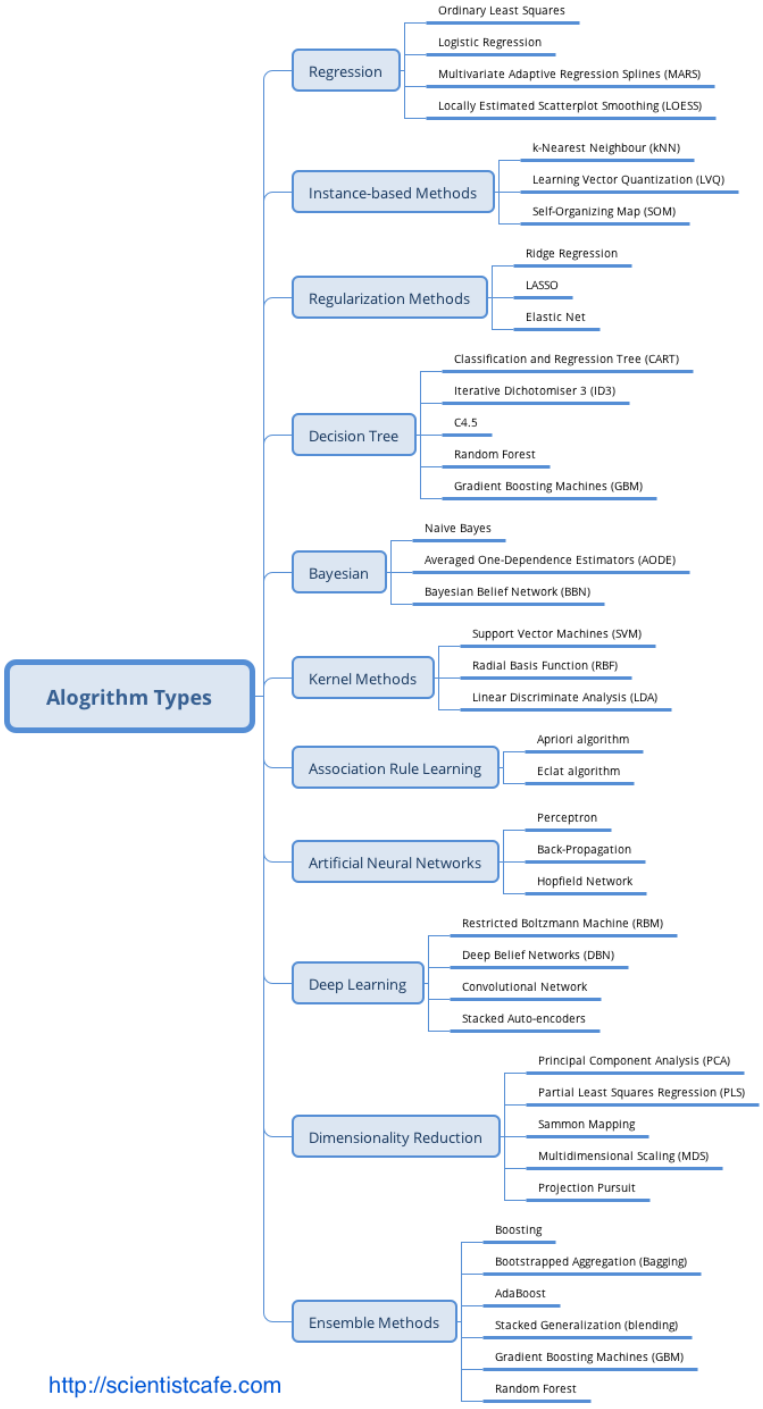


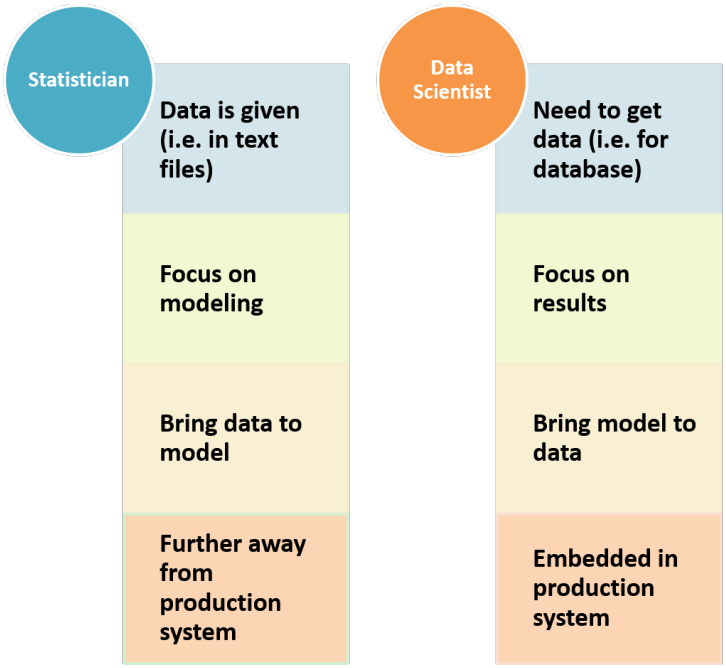
FIGURE 1.5: Machines Learning Algorithms

2

Soft Skills for Data Scientists

2.1 Comparison between Statistician and Data Scientist

Statistics as a scientific area can be traced back to 1749 and statistician as a career has been around for hundreds of years with well-established theory and application. Data Scientist becomes an attractive career for only a few years along with the fact that data size and variety beyond the traditional statistician's toolbox and the fast-growing of computation power. Statistician and data scientist have a lot of common backgrounds, but there are also some significant differences.



Both statistician and data scientist work closely with data. For the tradi-

tional statistician, the data is usually well-formatted text files with numbers and labels. The size of the data usually can be fitted in a PC's memory. Comparing to statisticians, data scientists need to deal with more varieties of data: well-formatted data stored in a database system with size much larger than a PC's memory or hard-disk; huge amount of verbatim text, voice, image, and video; real-time streaming data and other types of records. One particular power of statistics is that statistician can fit model and make an inference based on limited data. It is quite common that once the data is given and cleaned, the majority of the work is developed different models around the data. Today, data is relatively abundant, and modeling is just part of the overall effort. The focus is to deliver actionable results. Different from statisticians, data scientists, sometimes need to fit model on the cloud instead of reading data in since the data size is too large. From the entire problem-solving cycle, statisticians are usually not well integrated with the production system where data is obtained in real time; while data scientists are more embedded in the production system and closer to the data generation procedures.

2.2 Where Data Science Team Fits?

During the past decade, a huge amount of data has become available and readily accessible for analysis in many companies across different business sectors. The size, complexity, and speed of increment of data suddenly beyond the traditional scope of statistical analysis or BI reporting as mentioned above. To leverage the big data, many companies have established new data science organizations. Companies have gone through different paths to create their data science and machine learning organizations. There are three major formats of data science teams:

(1) independent of any current organizations and the team report directly to senior leadership;

(2) within each business unit and the team report to business unit

leaders;

(3) within in the traditional IT organizations and the team report to IT leaders.

Companies are different in many aspects, but in general, the most efficient option to mine big data is a team of data scientist independent of business units and IT organizations. The independence enables the data science team to collaborate across business units and IT organizations more efficiently and the independence also provides flexibility and potential to solve corporate level strategic big data problems. For each business units, there are many business unit specific data science related problems and embedding data scientist within each business units is also an efficient way to solve business unit specific data science problems. The full cycle of data science projects from data to decision (i.e. Data -> Information -> Knowledge -> Insight -> Decision) is relatively difficult to achieve if the data science team is part of traditional IT organizations.

2.3 Beyond Data and Analytics

Data scientists usually have a good sense of data and analytics, but data scientist project is more than just data and analytics. A data science project may involve people with many different roles:

- a business owner or leader to identify opportunities in business value; program managers to ensure each data science project fit into the overall technical program development;
- data owners and computation resource and infrastructure owners from IT department;
- dedicated team to make sure the data and model are under model governance and privacy guidelines;
- a team to implement, maintain and refresh the model;
- project managers to coordinate all parties to set periodical tasks so that the project meets the preset milestones and delivery results;

- multiple rounds of discussion of resource allocation (i.e. who will pay for the data science project).

Effective communication and in-depth domain knowledge about the business problem are essential requirements for a successful data scientist. A data scientist will interact with people at various levels ranging from senior leaders who are setting the corporate strategies to front-line employees who are doing the daily work. A data scientist needs to have the capability to view the problem from 10,000 feet above ground, as well as down to the detail to the very bottom. To convert a business question into a data problem, a data scientist needs to communicate using the language the other people can understand and obtain the required information.

2.4 Data Scientist as a Leader

During the entire process of data science project defining, planning, executing and implementation, the data scientist lead needs to be involved in every step to ensure the business problem is defined correctly and the business value and success metric are evaluated reasonably. Corporates are investing heavily in data science and machine learning with a very high expectation of big return. There are too many opportunities to introduce unrealistic goal and business impact for a particular data science project. The leading data scientist need to be the leader in these discussions to define the goal backed by data and analytics. Many data science projects over promise in deliverables and too optimistic on the timeline and these projects eventually fail by not delivering the preset business impact within the timeline. As the data scientist in the team, we need to identify these issues early in the stage and communicate to the entire team to make sure the project has a realistic deliverable and timeline. The data scientist team also need to work closely with data owners to identify relevant internal and external data source and evaluate the quality of the data; as well as working closely with the computation infrastructure team to understand the computation resources (i.e. hardware and software) available for the data science project.

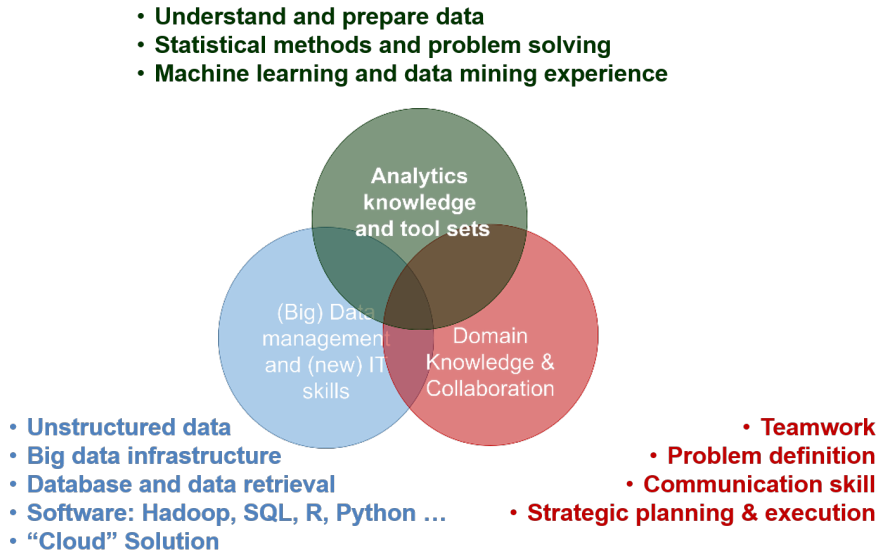
2.5 Three Pillars of Knowledge

The following picture summarizes the needed three pillars of knowledge to be a successful data scientist.

(1) A successful data scientist needs to have a strong technical background in data mining, statistics and machine learning. The in-depth understanding of modeling with the insight about data enable a data scientist to convert a business problem to a data science problem.

(2) A successful data scientist needs some domain knowledge to understand business problem. For any data science project, the data scientist need to collaborate with other team members and effective communication and leadership skills are critical, especially when you are the only data person in the room and you need to decide with uncertainty.

(3) The last pillar is about computation environment and model implementation in big data platform. This used to be the most difficult one for a data scientist with statistics background (i.e. lack computer science or programming skills). The good news is that with the rise of cloud computation big data platform, this barrier is getting easier for a statistician to overcome and we will discuss in more detail in next chapter.



2.6 Common Pitfalls of Data Science Projects

Data science projects are usually complicated, and many of these data science projects eventually fail due to various reasons. We will briefly discuss a few common pitfalls in data science projects and how to avoid them.

- **Solve the wrong problem:** data science project usually starts with a very vague description and a few rounds of detailed discussion with all stakeholders involved are needed to define the busses problem. There will be lots of opportunities to introduce misalignment when mapping the business problem into specific data science methods. Especially when the quality and availability of the data are not as good as what is expected at the first place. If not well-communicated during the project, the final data science solution may not be the right one to solve the business problem. As the data scientist (sometimes the only data scientist) in the room, we must understand the business problem thoroughly and communicate regularly to business partners especially there is a change

of status to make sure everyone is aligned with the progress and final deliverables.

- **Over promise on business value:** business leaders usually have high expectation on data science projects and the goal of business value and deliverables sometimes are set unrealistic and eventually beyond the scope of available data and computation resource. As the data scientist (sometimes the only data scientist) in the room, we must have our voice heard based on fact (i.e. data, analytics, and resources) instead of wishful thinking. Backed with fact-based evidence, it is easier to communicate what is a realistic goal for the entire team.
- **Too optimistic about the timeline:** there are lots of uncertainties in data science projects such as the data source availability and data quality, computation hardware and software, resource availability in the business team, implementation team and IT department, as well as project direction change which may delay the final delivery date. To have a better-estimated timeline, get as much detail as possible for all the needed tasks and estimated each task individually and reach out to each team member to confirm their availability. Most importantly, communicate with the entire team if there are blocking factors for the project in a prompt way such that everyone aware of the situation and potential impact on the timeline.
- **Too optimistic about data availability and quality:** the most important asset in data science project is data. Even though we are at the big data age, often there is not enough relevant data for the data science projects. The data quality is also a general problem for data science projects. A thorough data availability and quality check are needed at the beginning of the data science project to estimate the needed effort to obtain data as well as data cleaning.
- **Model cannot be scaled:** be careful if you use a subset of data to fit the model and then scale it to the entire dataset. When developing the model using a smaller dataset, we must keep in mind how much computation resources needed for the whole dataset. With limited computation resource, it is important to maximize the computation time in production to a reasonable level based on the business application when fits the model with a sample dataset.

- **Take too long to fail:** data science projects usually are trying to push the boundary of current applications to new territory, people do not expect all data science projects to be successful. Fail fast is good practice such that we can quickly find a better way to solve the problem. A data scientist needs to have an open mindset to not stuck with one idea or one approach for a long time to avoid taking too long to fail.

3

Introduction to the data

Before tackling analytics problem, we start by introducing data to be analyzed in later chapters.

3.1 Customer Data for Clothing Company

Our first data set represents customers of a clothing company who sells products in stores and online. This data is typical of what one might get from a company's marketing data base (the data base will have more data than the one we show here). This data includes 1000 customers for whom we have 3 types of data:

1. Demography
    - `age`: age of the respondent
    - `gender`: male/female
    - `house`: 0/1 variable indicating if the customer owns a house or not
2. Sales in the past year
    - `store_exp`: expense in store
    - `online_exp`: expense online
    - `store_trans`: times of store purchase
    - `online_trans`: times of online purchase
3. Survey on product preference

It is common for companies to survey their customers and draw insights to guide future marketing activities. The survey is as below:

How strongly do you agree or disagree with the following statements:

1. Strong disagree
2. Disagree
3. Neither agree nor disagree
4. Agree
5. Strongly agree

- Q1. I like to buy clothes from different brands
- Q2. I buy almost all my clothes from some of my favorite brands
- Q3. I like to buy premium brands
- Q4. Quality is the most important factor in my purchasing decision
- Q5. Style is the most important factor in my purchasing decision
- Q6. I prefer to buy clothes in store
- Q7. I prefer to buy clothes online
- Q8. Price is important
- Q9. I like to try different styles
- Q10. I like to make a choice by myself and don't need too much of others' suggestions

There are 4 segments of customers:

1. Price
2. Conspicuous
3. Quality
4. Style

Let's check it:

```
str(sim.dat,vec.len=3)
```

```
## 'data.frame':    1000 obs. of  19 variables:
##  $ age         : int  57 63 59 60 51 59 57 57 ...
##  $ gender      : Factor w/ 2 levels "Female","Male": 1 1 2 2 2 2 2 2 ...
##  $ income      : num  120963 122008 114202 113616 ...
##  $ house       : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 ...
##  $ store_exp   : num  529 478 491 348 ...
##  $ online_exp  : num  304 110 279 142 ...
##  $ store_trans : int  2 4 7 10 4 4 5 11 ...
##  $ online_trans: int  2 2 2 2 4 5 3 5 ...
##  $ Q1          : int  4 4 5 5 4 4 4 5 ...
```

```
##  $ Q2          : int  2 1 2 2 1 2 1 2 ...
##  $ Q3          : int  1 1 1 1 1 1 1 1 ...
##  $ Q4          : int  2 2 2 3 3 2 2 3 ...
##  $ Q5          : int  1 1 1 1 1 1 1 1 ...
##  $ Q6          : int  4 4 4 4 4 4 4 4 ...
##  $ Q7          : int  1 1 1 1 1 1 1 1 ...
##  $ Q8          : int  4 4 4 4 4 4 4 4 ...
##  $ Q9          : int  2 1 1 2 2 1 1 2 ...
##  $ Q10         : int  4 4 4 4 4 4 4 4 ...
##  $ segment     : Factor w/ 4 levels "Conspicuous",..: 2 2 2 2 2 2 2 2 ...
```

3.2 Customer Satisfaction Survey Data from Airline Company

This data set is from a customer satisfaction survey for three airline companies. There are `N=1000` respondents and 15 questions. The market researcher asked respondents to recall the experience with different airline companies and assign a score (1-9) to each airline company for all the 15 questions. The higher the score, the more satisfied the customer to the specific item. The 15 questions are of 4 types (the variable names are in the parentheses):

- How satisfied are you with your_____?

    1. Ticketing
        - •Ease of making reservation Easy_Reservation
        - •Availability of preferred seats Preferred_Seats
        - •Variety of flight options Flight_Options
        - •Ticket prices Ticket_Prices
    2. Aircraft
        - •Seat comfort Seat_Comfort
        - •Roominess of seat area Seat_Roominess
        - •Availability of Overhead Overhead_Storage
        - •Cleanliness of aircraft Clean_Aircraft
    3. Service
        - •Courtesy of flight attendant Courtesy

- Friendliness Friendliness
- Helpfulness Helpfulness
- Food and drinks Service

4. General
   - Overall satisfaction Satisfaction
   - Purchase again Fly_Again
   - Willingness to recommend Recommend

Now check the data frame we have:

```
str(rating,vec.len=3)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    3000 obs. of  17 variables:
##  $ Easy_Reservation: int  6 5 6 5 4 5 6 4 ...
##  $ Preferred_Seats : int  5 7 6 6 5 6 6 6 ...
##  $ Flight_Options  : int  4 7 5 5 3 4 6 3 ...
##  $ Ticket_Prices   : int  5 6 6 5 6 5 5 5 ...
##  $ Seat_Comfort    : int  5 6 7 7 6 6 6 4 ...
##  $ Seat_Roominess  : int  7 8 6 8 7 8 6 5 ...
##  $ Overhead_Storage: int  5 5 7 6 5 4 4 4 ...
##  $ Clean_Aircraft  : int  7 6 7 7 7 7 6 4 ...
##  $ Courtesy        : int  5 6 6 4 2 5 5 4 ...
##  $ Friendliness    : int  4 6 6 6 3 4 5 5 ...
##  $ Helpfulness     : int  6 5 6 4 4 5 5 4 ...
##  $ Service         : int  6 5 6 5 3 5 5 5 ...
##  $ Satisfaction    : int  6 7 7 5 4 6 5 5 ...
##  $ Fly_Again       : int  6 6 6 7 4 5 3 4 ...
##  $ Recommend       : int  3 6 5 5 4 5 6 5 ...
##  $ ID              : int  1 2 3 4 5 6 7 8 ...
##  $ Airline         : chr  "AirlineCo.1" "AirlineCo.1" "AirlineCo.1" ...
```

4

Data Pre-processing

Many data analysis related books focus on models, algorithms and statistical inferences. However, in practice, raw data is usually not directly used for modeling. Data preprocessing is the process of converting raw data into clean data that is proper for modeling. A model fails for various reasons. One is that the modeler doesn't correctly preprocess data before modeling. Data preprocessing can significantly impact model results, such as imputing missing value and handling with outliers. So data preprocessing is a very critical part.

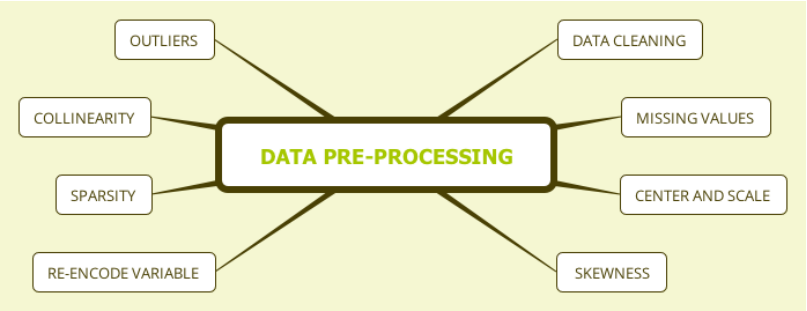


FIGURE 4.1: Data Pre-processing Outline

In real life, depending on the stage of data cleanup, data has the following types:

1. Raw data
2. Technically correct data
3. Data that is proper for the model
4. Summarized data
5. Data with fixed format

The raw data is the first-hand data that analysts pull from the database,

market survey responds from your clients, the experimental results collected by the R & D department, and so on. These data may be very rough, and R sometimes can't read them directly. The table title could be multiline, or the format does not meet the requirements:

- Use 50% to represent the percentage rather than 0.5, so R will read it as a character;
- The missing value of the sales is represented by "-" instead of space so that R will treat the variable as character or factor type;
- The data is in a slideshow document, or the spreadsheet is not ".csv" but ".xlsx"
- ...

Most of the time, you need to clean the data so that R can import them. Some data format requires a specific package. Technically correct data is the data, after preliminary cleaning or format conversion, that R (or another tool you use) can successfully import it.

Assume we have loaded the data into R with reasonable column names, variable format and so on. That does not mean the data is entirely correct. There may be some observations that do not make sense, such as age is negative, the discount percentage is greater than 1, or data is missing. Depending on the situation, there may be a variety of problems with the data. It is necessary to clean the data before modeling. Moreover, different models have different requirements on the data. For example, some model may require the variables are of consistent scale; some may be susceptible to outliers or collinearity, some may not be able to handle categorical variables and so on. The modeler has to preprocess the data to make it proper for the specific model.

Sometimes we need to aggregate the data. For example, add up the daily sales to get annual sales of a product at different locations. In customer segmentation, it is common practice to build a profile for each segment. It requires calculating some statistics such as average age, average income, age standard deviation, etc. Data aggregation is also necessary for presentation, or for data visualization.

The final table results for clients need to be in a nicer format than what used in the analysis. Usually, data analysts will take the results from data scientists and adjust the format, such as labels, cell color, highlight. It is

important for a data scientist to make sure the results look consistent which makes the next step easier for data analysts.

It is highly recommended to store each step of the data and the R code, making the whole process as repeatable as possible. The R markdown reproducible report will be extremely helpful for that. If the data changes, it is easy to rerun the process. In the remainder of this chapter, we will show the most common data preprocessing methods.

Load the R packages first:

```
source("https://raw.githubusercontent.com/happyrabbit/CE_JSM2017/master/Rcode/00-course-setup.
```

4.1 Data Cleaning

After you load the data, the first thing is to check how many variables are there, the type of variables, the distributions, and data errors. Let's read and check the data:

```
sim.dat <- read.csv("https://raw.githubusercontent.com/happyrabbit/DataScientistR/master/Data/
summary(sim.dat)
```

```
##       age            gender        income
##  Min.   : 16.00   Female:554   Min.   : 41776
##  1st Qu.: 25.00   Male  :446   1st Qu.: 85832
##  Median : 36.00                Median : 93869
##  Mean   : 38.84                Mean   :113543
##  3rd Qu.: 53.00                3rd Qu.:124572
##  Max.   :300.00                Max.   :319704
##                                NA's   :184
##  house       store_exp         online_exp
##  No :432   Min.   : -500.0   Min.   :  68.82
##  Yes:568   1st Qu.:  205.0   1st Qu.: 420.34
##            Median :  329.0   Median :1941.86
##            Mean   : 1356.8   Mean   :2120.18
##            3rd Qu.:  597.3   3rd Qu.:2440.78
```

```
##            Max.   :50000.0   Max.   :9479.44
##
##   store_trans     online_trans         Q1
##  Min.   : 1.00   Min.   : 1.00   Min.   :1.000
##  1st Qu.: 3.00   1st Qu.: 6.00   1st Qu.:2.000
##  Median : 4.00   Median :14.00   Median :3.000
##  Mean   : 5.35   Mean   :13.55   Mean   :3.101
##  3rd Qu.: 7.00   3rd Qu.:20.00   3rd Qu.:4.000
##  Max.   :20.00   Max.   :36.00   Max.   :5.000
##
##        Q2              Q3              Q4
##  Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:2.000
##  Median :1.000   Median :1.000   Median :3.000
##  Mean   :1.823   Mean   :1.992   Mean   :2.763
##  3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.:4.000
##  Max.   :5.000   Max.   :5.000   Max.   :5.000
##
##        Q5              Q6              Q7
##  Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:1.750   1st Qu.:1.000   1st Qu.:2.500
##  Median :4.000   Median :2.000   Median :4.000
##  Mean   :2.945   Mean   :2.448   Mean   :3.434
##  3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :5.000   Max.   :5.000   Max.   :5.000
##
##        Q8              Q9             Q10
##  Min.   :1.000   Min.   :1.000   Min.   :1.00
##  1st Qu.:1.000   1st Qu.:2.000   1st Qu.:1.00
##  Median :2.000   Median :4.000   Median :2.00
##  Mean   :2.396   Mean   :3.085   Mean   :2.32
##  3rd Qu.:3.000   3rd Qu.:4.000   3rd Qu.:3.00
##  Max.   :5.000   Max.   :5.000   Max.   :5.00
##
##         segment
##  Conspicuous:200
##  Price      :250
```

```
##  Quality    :200
##  Style      :350
##
##
##
```

Are there any problems? Questionnaire response Q1-Q10 seem reasonable, the minimum is 1 and maximum is 5. Recall that the questionnaire score is 1-5. The number of store transactions (store_trans) and online transactions (store_trans) make sense too. Things need to pay attention are:

- There are some missing values.
- There are outliers for store expenses (`store_exp`). The maximum value is 50000. Who would spend $50000 a year buying clothes? Is it an imputation error?
- There is a negative value (-500) in `store_exp` which is not logical.
- Someone is 300 years old.

How to deal with that? Depending on the real situation, if the sample size is large enough, it will not hurt to delete those problematic samples. Here we have 1000 observations. Since marketing survey is usually expensive, it is better to set these values as missing and impute them instead of deleting the rows.

```
# set problematic values as missings
sim.dat$age[which(sim.dat$age>100)]<-NA
sim.dat$store_exp[which(sim.dat$store_exp<0)]<-NA
# see the results
summary(subset(sim.dat,select=c("age","income")))
```

```
##       age            income
##  Min.   :16.00   Min.   : 41776
##  1st Qu.:25.00   1st Qu.: 85832
##  Median :36.00   Median : 93869
##  Mean   :38.58   Mean   :113543
##  3rd Qu.:53.00   3rd Qu.:124572
##  Max.   :69.00   Max.   :319704
##  NA's   :1       NA's   :184
```

Now we will deal with the missing values in the data.

4.2 Missing Values

You can write a whole book about missing value. This section will only show some of the most commonly used methods without getting too deep into the topic. Chapter 7 of the book by De Waal, Pannekoek and Scholtus (de Waal et al., 2011) makes a concise overview of some of the existing imputation methods. The choice of specific method depends on the actual situation. There is no best way.

One question to ask before imputation: Is there any auxiliary information? Being aware of any auxiliary information is critical. For example, if the system set customer who did not purchase as missing, then the real purchasing amount should be 0. Is missing a random occurrence? If so, it may be reasonable to impute with mean or median. If not, is there a potential mechanism for the missing data? For example, older people are more reluctant to disclose their ages in the questionnaire, so that the absence of age is not completely random. In this case, the missing values need to be estimated using the relationship between age and other independent variables. For example, use variables such as whether they have children, income, and other survey questions to build a model to predict age.

Also, the purpose of modeling is important for selecting imputation methods. If the goal is to interpret the parameter estimate or statistical inference, then it is important to study the missing mechanism carefully and to estimate the missing values using non-missing information as much as possible. If the goal is to predict, people usually will not study the absence mechanism rigorously (but sometimes the mechanism is obvious). If the absence mechanism is not clear, treat it as missing at random and use mean, median, or k-nearest neighbor to impute. Since statistical inference is sensitive to missing values, researchers from survey statistics have conducted in-depth studies of various imputation schemes which focus on valid statistical inference. The problem of miss-

ing values in the prediction model is different from that in the traditional survey. Therefore, there are not many papers on missing value imputation in the prediction model. Those who want to study further can refer to Saar-Tsechansky and Provost's comparison of different imputation methods (M and F, 007b) and De Waal, Pannekoek and Scholtus' book (de Waal et al., 2011).

4.2.1 Impute missing values with median/mode

In the case of missing at random, a common method is to impute with the mean (continuous variable) or median (categorical variables). You can use `impute()` function in `imputeMissings` package.

```
# save the result as another object
demo_imp<-impute(sim.dat,method="median/mode")
# check the first 5 columns, there is no missing values in other columns
summary(demo_imp[,1:5])
```

```
##       age           gender        income
##  Min.   :16.00   Female:554   Min.   : 41776
##  1st Qu.:25.00   Male  :446   1st Qu.: 87896
##  Median :36.00                Median : 93869
##  Mean   :38.58                Mean   :109923
##  3rd Qu.:53.00                3rd Qu.:119456
##  Max.   :69.00                Max.   :319704
##  house       store_exp
##  No :432   Min.   :  155.8
##  Yes:568   1st Qu.:  205.1
##            Median :  329.8
##            Mean   : 1357.7
##            3rd Qu.:  597.3
##            Max.   :50000.0
```

After imputation, `demo_imp` has no missing value. This method is straightforward and widely used. The disadvantage is that it does not take into account the relationship between the variables. When there is a significant proportion of missing, it will distort the data. In this case, it is better to consider the relationship between variables and study the missing

mechanism. In the example here, the missing variables are numeric. If the missing variable is a categorical/factor variable, the `impute()` function will impute with the mode.

You can also use `preProcess()` function, but it is only for numeric variables, and can not impute categorical variables. Since missing values here are numeric, we can use the `preProcess()` function. The result is the same as the `impute()` function. `PreProcess()` is a powerful function that can link to a variety of data preprocessing methods. We will use the function later for other data preprocessing.

```
imp<-preProcess(sim.dat,method="medianImpute")
demo_imp2<-predict(imp,sim.dat)
summary(demo_imp2[,1:5])
```

```
##       age           gender        income
##  Min.   :16.00   Female:554   Min.   : 41776
##  1st Qu.:25.00   Male  :446   1st Qu.: 87896
##  Median :36.00                Median : 93869
##  Mean   :38.58                Mean   :109923
##  3rd Qu.:53.00                3rd Qu.:119456
##  Max.   :69.00                Max.   :319704
##  house       store_exp
##  No :432   Min.   :  155.8
##  Yes:568   1st Qu.:  205.1
##            Median :  329.8
##            Mean   : 1357.7
##            3rd Qu.:  597.3
##            Max.   :50000.0
```

4.2.2 K-nearest neighbors

K-nearest neighbor (KNN) will find the k closest samples (Euclidian distance) in the training set and impute the mean of those "neighbors."

Use `preProcess()` to conduct KNN:

```
imp<-preProcess(sim.dat,method="knnImpute",k=5)
```

```
# need to use predict() to get KNN result
demo_imp<-predict(imp,sim.dat)
```

```
Error in `[.data.frame`(old, , non_missing_cols, drop = FALSE) :
  undefined columns selected
```

Now we get an error saying "undefined columns selected." It is because `sim.dat` has non-numeric variables. The `preProcess()` in the first line will automatically ignore non-numeric columns, so there is no error. However, there is a problem when using `predict()` to get the result. Removing those variable will solve the problem.

```
# find factor columns
imp<-preProcess(sim.dat,method="knnImpute",k=5)
idx<-which(lapply(sim.dat,class)=="factor")
demo_imp<-predict(imp,sim.dat[,-idx])
summary(demo_imp[,1:3])
```

```
##       age                income
##  Min.   :-1.5910972   Min.   :-1.43989
##  1st Qu.:-0.9568733   1st Qu.:-0.53732
##  Median :-0.1817107   Median :-0.37606
##  Mean   : 0.0000156   Mean   : 0.02389
##  3rd Qu.: 1.0162678   3rd Qu.: 0.21540
##  Max.   : 2.1437770   Max.   : 4.13627
##    store_exp
##  Min.   :-0.43345
##  1st Qu.:-0.41574
##  Median :-0.37105
##  Mean   :-0.00042
##  3rd Qu.:-0.27437
##  Max.   :17.52734
```

`lapply(data,class)` can return a list of column class. Here the data frame is `sim.dat`, and the following code will give the list of column class:

```
# only show the first three elements
lapply(sim.dat,class)[1:3]
```

```
## $age
```

```
## [1] "integer"
##
## $gender
## [1] "factor"
##
## $income
## [1] "numeric"
```

Comparing the KNN result with the previous median imputation, the two are very different. This is because when you tell the `preProcess()` function to use KNN (the option `method =" knnImpute"`), it will automatically standardize the data. Another way is to use Bagging tree (in the next section). Note that KNN can not impute samples with the entire row missing. The reason is straightforward. Since the algorithm uses the average of its neighbors if none of them has a value, what does it apply to calculate the mean? Let's append a new row with all values missing to the original data frame to get a new object called `temp`. Then apply KNN to `temp` and see what happens:

```
temp<-rbind(sim.dat,rep(NA,ncol(sim.dat)))
imp<-preProcess(sim.dat,method="knnImpute",k=5)
idx<-which(lapply(temp,class)=="factor")
```

```
demo_imp<-predict(imp,temp[,-idx])
```

```
Error in FUN(newX[, i], ...) :
  cannot impute when all predictors are missing in the new data point
```

There is an error saying "cannot impute when all predictors are missing in the new data point". It is easy to fix by finding and removing the problematic row:

```
idx<-apply(temp,1,function(x) sum(is.na(x)) )
as.vector(which(idx==ncol(temp)))
```

```
## [1] 1001
```

It shows that row 1001 is problematic. You can go ahead to delete it.

4.2.3 Bagging Tree

Bagging (Bootstrap aggregating) was originally proposed by Leo Breiman. It is one of the earliest ensemble methods (L, 966a). When used in missing value imputation, it will use the remaining variables as predictors to train a bagging tree and then use the tree to predict the missing values. Although theoretically, the method is powerful, the computation is much more intense than KNN. In practice, there is a trade-off between computation time and the effect. If a median or mean meet the modeling needs, even bagging tree may improve the accuracy a little, but the upgrade is so marginal that it does not deserve the extra time. The bagging tree itself is a model for regression and classification. Here we use `preProcess()` to impute `sim.dat`:

```
imp<-preProcess(sim.dat,method="bagImpute")
demo_imp<-predict(imp,sim.dat)
summary(demo_imp[,1:5])
```

```
      age           gender        income       house       store_exp
 Min.   :16.00   Female:554   Min.   : 41776   No :432   Min.   :  155.8
 1st Qu.:25.00   Male  :446   1st Qu.: 86762   Yes:568   1st Qu.:  205.1
 Median :36.00                Median : 94739             Median :  329.0
 Mean   :38.58                Mean   :114665             Mean   : 1357.7
 3rd Qu.:53.00                3rd Qu.:123726             3rd Qu.:  597.3
 Max.   :69.00                Max.   :319704             Max.   :50000.0
```

4.3 Centering and Scaling

It is the most straightforward data transformation. It centers and scales a variable to mean 0 and standard deviation 1. It ensures that the criterion for finding linear combinations of the predictors is based on how much variation they explain and therefore improves the numerical stability. Models involving finding linear combinations of the predictors to explain response/predictors variation need data centering and scaling,

such as PCA (Jolliffe, 2002), PLS (Geladi P, 1986) and EFA (Mulaik, 2009). You can quickly write code yourself to conduct this transformation.

Let's standardize the variable `income` from `sim.dat`:

```
income<-sim.dat$income
# calculate the mean of income
mux<-mean(income,na.rm=T)
# calculate the standard deviation of income
sdx<-sd(income,na.rm=T)
# centering
tr1<-income-mux
# scaling
tr2<-tr1/sdx
```

Or the function `preProcess()` in package `caret` can apply this transformation to a set of predictors.

```
sdat<-subset(sim.dat,select=c("age","income"))
# set the "method" option
trans<-preProcess(sdat,method=c("center","scale"))
# use predict() function to get the final result
transformed<-predict(trans,sdat)
```

Now the two variables are in the same scale:

```
summary(transformed)
```

```
##       age              income
##  Min.   :-1.5911   Min.   :-1.4399
##  1st Qu.:-0.9569   1st Qu.:-0.5560
##  Median :-0.1817   Median :-0.3947
##  Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.0163   3rd Qu.: 0.2213
##  Max.   : 2.1438   Max.   : 4.1363
##  NA's   :1         NA's   :184
```

Sometimes you only need to scale the variable. For example, if the model adds a penalty to the parameter estimates (such as L_2 penalty is ridge regression and L_1 penalty in LASSO), the variables need to have a similar scale to ensure a fair variable selection. I am a heavy user of this kind of

penalty-based model in my work, and I used the following quantile transformation:

$$x_{ij}^* = \frac{x_{ij} - quantile(x_{.j}, 0.01)}{quantile(x_{.j} - 0.99) - quantile(x_{-j}, 0.01)}$$

The reason to use 99% and 1% quantile instead of maximum and minimum values is to resist the impact of outliers.

It is easy to write a function to do it:

```
qscale<-function(dat){
  for (i in 1:ncol(dat)){
    up<-quantile(dat[,i],0.99)
    low<-quantile(dat[,i],0.01)
    diff<-up-low
    dat[,i]<-(dat[,i]-low)/diff
  }
  return(dat)
}
```

In order to illustrate, let's apply it to some variables from 'demo_imp2:

```
demo_imp3<-qscale(subset(demo_imp2,select=c("income","store_exp","online_exp")))
summary(demo_imp3)
```

```
##      income           store_exp
##  Min.   :-0.05776   Min.   :-0.003407
##  1st Qu.: 0.15736   1st Qu.: 0.003984
##  Median : 0.18521   Median : 0.022704
##  Mean   : 0.26009   Mean   : 0.176965
##  3rd Qu.: 0.30456   3rd Qu.: 0.062849
##  Max.   : 1.23857   Max.   : 7.476996
##    online_exp
##  Min.   :-0.006023
##  1st Qu.: 0.042719
##  Median : 0.253691
##  Mean   : 0.278417
##  3rd Qu.: 0.322871
##  Max.   : 1.298845
```

After transformation, most of the variables are between 0-1.

4.4 Resolve Skewness

Skewness[1] is defined to be the third standardized central moment. The formula for the sample skewness statistics is:

$$skewness = \frac{\sum(x_i - \bar{x})^3}{(n-1)v^{3/2}}$$

$$v = \frac{\sum(x_i - \bar{x})^2}{(n-1)}$$

Skewness=0 means that the destribution is symmetric, i.e. the probability of falling on either side of the distribution's mean is equal.

```
# need skewness() function from e1071 package
set.seed(1000)
par(mfrow=c(1,2),oma=c(2,2,2,2))
# random sample 1000 chi-square distribution with df=2
# right skew
x1<-rchisq(1000,2, ncp = 0)
# get left skew variable x2 from x1
x2<-max(x1)-x1
plot(density(x2),main=paste("left skew, skewnwss =",round(skewness(x2),2)), xlab="X2")
plot(density(x1),main=paste("right skew, skewness =",round(skewness(x1),2)), xlab="X1")
```

You can easily tell if a distribution is skewed by simple visualization(Figure4.2). There are different ways may help to remove skewness such as log, square root or inverse. However, it is often difficult to determine from plots which transformation is most appropriate for correcting skewness. The Box-Cox procedure automatically identified a transformation from the family of power transformations that are indexed by a parameter λ(Box G, 1964).

[1] https://en.wikipedia.org/wiki/Skewness

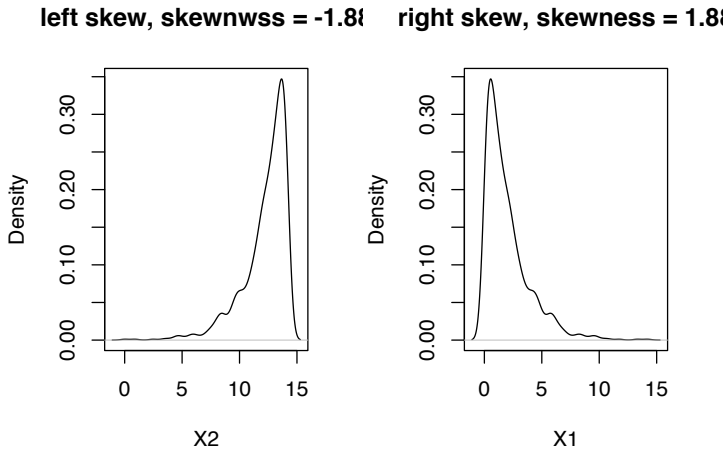


FIGURE 4.2: Shewed Distribution

$$x^* = \begin{cases} \frac{x^\lambda - 1}{\lambda} & if\ \lambda \neq 0 \\ log(x) & if\ \lambda = 0 \end{cases}$$

It is easy to see that this family includes log transformation ($\lambda = 0$), square transformation ($\lambda = 2$), square root ($\lambda = 0.5$), inverse ($\lambda = -1$) and others in-between. We can still use function `preProcess()` in package `caret` to apply this transformation by chaning the `method` argument.

```
describe(sim.dat)
```

```
##              vars    n      mean       sd   median
## age             1  999     38.58    14.19    36.00
## gender*         2 1000      1.45     0.50     1.00
## income          3  816 113543.07 49842.29 93868.68
## house*          4 1000      1.57     0.50     2.00
## store_exp       5  999   1358.71  2775.17   329.80
## online_exp      6 1000   2120.18  1731.22  1941.86
## store_trans     7 1000      5.35     3.70     4.00
## online_trans    8 1000     13.55     7.96    14.00
## Q1              9 1000      3.10     1.45     3.00
## Q2             10 1000      1.82     1.17     1.00
## Q3             11 1000      1.99     1.40     1.00
## Q4             12 1000      2.76     1.16     3.00
```

```
## Q5             13 1000      2.94     1.28     4.00
## Q6             14 1000      2.45     1.44     2.00
## Q7             15 1000      3.43     1.46     4.00
## Q8             16 1000      2.40     1.15     2.00
## Q9             17 1000      3.08     1.12     4.00
## Q10            18 1000      2.32     1.14     2.00
## segment*       19 1000      2.70     1.15     3.00
##                trimmed       mad      min       max
## age              37.67    16.31    16.00     69.00
## gender*           1.43     0.00     1.00      2.00
## income       104841.94 28989.47 41775.64 319704.34
## house*            1.58     0.00     1.00      2.00
## store_exp       845.14   197.47   155.81  50000.00
## online_exp     1874.51  1015.21    68.82   9479.44
## store_trans       4.89     2.97     1.00     20.00
## online_trans     13.42    10.38     1.00     36.00
## Q1                3.13     1.48     1.00      5.00
## Q2                1.65     0.00     1.00      5.00
## Q3                1.75     0.00     1.00      5.00
## Q4                2.83     1.48     1.00      5.00
## Q5                3.05     0.00     1.00      5.00
## Q6                2.43     1.48     1.00      5.00
## Q7                3.54     0.00     1.00      5.00
## Q8                2.36     1.48     1.00      5.00
## Q9                3.23     0.00     1.00      5.00
## Q10               2.27     1.48     1.00      5.00
## segment*          2.75     1.48     1.00      4.00
##                  range  skew kurtosis      se
## age              53.00  0.47    -1.18    0.45
## gender*           1.00  0.22    -1.95    0.02
## income       277928.70  1.69     2.57 1744.83
## house*            1.00 -0.27    -1.93    0.02
## store_exp     49844.19  8.08   115.04   87.80
## online_exp     9410.63  1.18     1.31   54.75
## store_trans      19.00  1.11     0.69    0.12
## online_trans     35.00  0.03    -0.98    0.25
## Q1                4.00 -0.12    -1.36    0.05
```

```
## Q2                4.00  1.13    -0.32    0.04
## Q3                4.00  1.06    -0.40    0.04
## Q4                4.00 -0.18    -1.46    0.04
## Q5                4.00 -0.60    -1.40    0.04
## Q6                4.00  0.11    -1.89    0.05
## Q7                4.00 -0.90    -0.79    0.05
## Q8                4.00  0.21    -1.33    0.04
## Q9                4.00 -0.68    -1.10    0.04
## Q10               4.00  0.39    -1.23    0.04
## segment*          3.00 -0.20    -1.41    0.04
```

It is easy to see the skewed variables. If `mean` and `trimmed` differ a lot, there is very likely outliers. By default, `trimmed` reports mean by dropping the top and bottom 10%. It can be adjusted by setting argument `trim=`. It is clear that `store_exp` has outliers.

As an example, we will apply Box-Cox transformation on `store_trans` and `online_trans`:

```
# select the two columns and save them as dat_bc
dat_bc<-subset(sim.dat,select=c("store_trans","online_trans"))
(trans<-preProcess(dat_bc,method=c("BoxCox")))
```

```
## Created from 1000 samples and 2 variables
##
## Pre-processing:
##   - Box-Cox transformation (2)
##   - ignored (0)
##
## Lambda estimates for Box-Cox transformation:
## 0.1, 0.7
```

The last line of the output shows the estimates of λ for each variable. As before, use `predict()` to get the transformed result:

```
transformed<-predict(trans,dat_bc)
par(mfrow=c(1,2),oma=c(2,2,2,2))
hist(dat_bc$store_trans,main="Before Transformation",xlab="store_trans")
hist(transformed$store_trans,main="After Transformation",xlab="store_trans")
```

Before the transformation, the `stroe_trans` is skewed right. The situation

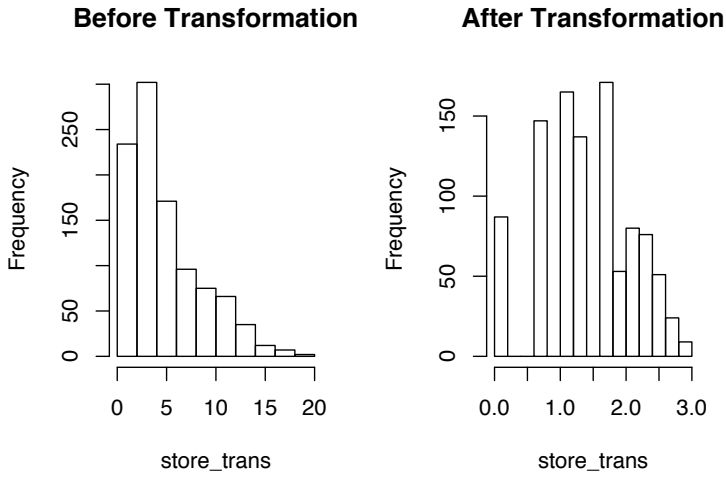


FIGURE 4.3: Box-Cox Transformation

is significantly improved after (figure4.3). `BoxCoxTrans ()` can also conduct Box-Cox transform. But note that `BoxCoxTrans ()` can only be applied to a single variable, and it is not possible to transform difference columns in a data frame at the same time.

```
(trans<-BoxCoxTrans(dat_bc$store_trans))
```

```
## Box-Cox Transformation
##
## 1000 data points used to estimate Lambda
##
## Input data summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    3.00    4.00    5.35    7.00   20.00
##
## Largest/Smallest: 20
## Sample Skewness: 1.11
##
## Estimated Lambda: 0.1
## With fudge factor, Lambda = 0 will be used for transformations
```

```
transformed<-predict(trans,dat_bc$store_trans)
skewness(transformed)
```

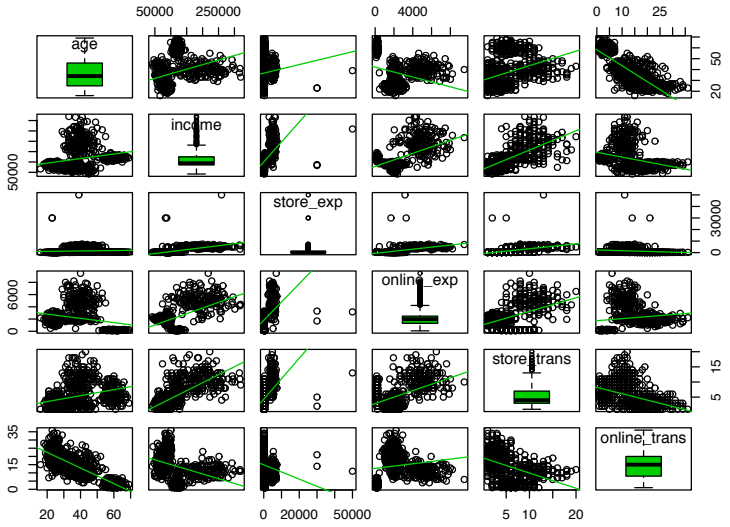


FIGURE 4.4: Use basic visualization to check outliers

```
## [1] -0.2154708
```

The estimate of λ is the same as before (0.1). The skewness of the original observation is 1.1, and -0.2 after transformation. Although it is not strictly 0, it is greatly improved.

4.5 Resolve Outliers

Even under certain assumptions we can statistically define outliers, it can be hard to define in some situations. Box plot, histogram and some other basic visualizations can be used to initially check whether there are outliers. For example, we can visualize numerical non-survey variables in `sim.dat`:

```
# select numerical non-survey data
sdat<-subset(sim.dat,select=c("age","income","store_exp","online_exp","store_trans","online_tr
# use scatterplotMatrix() function from car package
par(oma=c(2,2,1,2))
scatterplotMatrix(sdat,diagonal="boxplot",smoother=FALSE)
```

As figure 4.4 shows, `store_exp` has outliers. It is also easy to observe the pair relationship from the plot. `age` is negatively correlated with `online_trans` but positively correlated with `store_trans`. It seems that older people tend to purchase from the local store. The amount of expense is positively correlated with income. Scatterplot matrix like this can reveal lots of information before modeling.

In addition to visualization, there are some statistical methods to define outliers, such as the commonly used Z-score. The Z-score for variable $\mathbf{Y}$ is defined as:

$$Z_i = \frac{Y_i - \bar{Y}}{s}$$

where $\bar{Y}$ and s are mean and standard deviation for Y. Z-score is a measurement of the distance between each observation and the mean. This method may be misleading, especially when the sample size is small. Iglewicz and Hoaglin proposed to use the modified Z-score to determine the outlier(Iglewicz and Hoaglin, 1993)

$$M_i = \frac{0.6745(Y_i - \bar{Y})}{MAD}$$

Where MAD is the median of a series of $|Y_i - \bar{Y}|$, called the median of the absolute dispersion. Iglewicz and Hoaglin suggest that the points with the Z-score greater than 3.5 corrected above are possible outliers. Let's apply it to `income`:

```
# calculate median of the absolute dispersion for income
ymad<-mad(na.omit(sdat$income))
# calculate z-score
zs<-(sdat$income-mean(na.omit(sdat$income)))/ymad
# count the number of outliers
sum(na.omit(zs>3.5))
```

```
## [1] 59
```

According to modified Z-score, variable income has 59 outliers. Refer to (Iglewicz and Hoaglin, 1993) for other ways of detecting outliers.

The impact of outliers depends on the model. Some models are sensitive to outliers, such as linear regression, logistic regression. Some are pretty robust to outliers, such as tree models, support vector machine. Also, the outlier is not wrong data. It is real observation so cannot be deleted at will. If a model is sensitive to outliers, we can use *spatial sign transformation* (Serneels S, 2006) to minimize the problem. It projects the original sample points to the surface of a sphere by:

$$x_{ij}^* = \frac{x_{ij}}{\sqrt{\sum_{j=1}^{p} x_{ij}^2}}$$

where x_{ij} represents the i^{th} observation and j^{th} variable. As shown in the equation, every observation for sample i is divided by its square mode. The denominator is the Euclidean distance to the center of the p-dimensional predictor space. Three things to pay attention here:

1. It is important to center and scale the predictor data before using this transformation
2. Unlike centering or scaling, this manipulation of the predictors transforms them as a group
3. If there are some variables to remove (for example, highly correlated variables), do it before the transformation

Function `spatialSign()` `caret` package can conduct the transformation. Take `income` and `age` as an example:

```
# KNN imputation
sdat<-sim.dat[,c("income","age")]
imp<-preProcess(sdat,method=c("knnImpute"),k=5)
sdat<-predict(imp,sdat)
transformed <- spatialSign(sdat)
transformed <- as.data.frame(transformed)
par(mfrow=c(1,2),oma=c(2,2,2,2))
plot(income ~ age,data = sdat,col="blue",main="Before")
plot(income ~ age,data = transformed,col="blue",main="After")
```

Some readers may have found that the above code does not seem to stan-

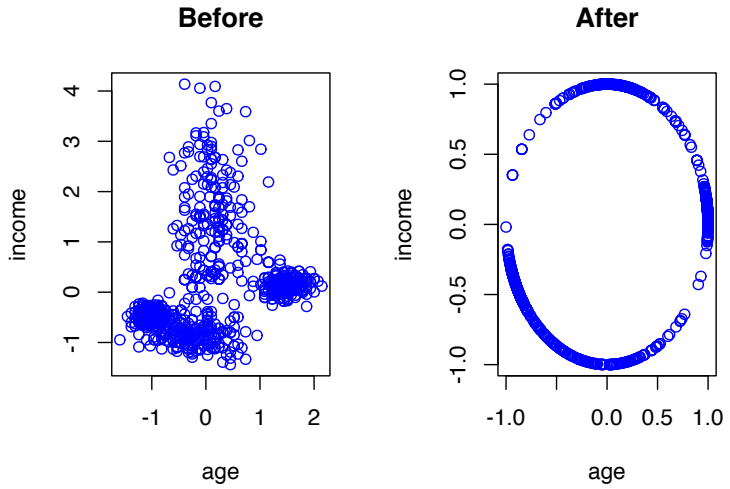


FIGURE 4.5: Spatial sign transformation

dardize the data before transformation. Recall the introduction of KNN, `preProcess()` with `method="knnImpute"` by default will standardize data.

4.6 Collinearity

It is probably the technical term known by the most un-technical people. When two predictors are very strongly correlated, including both in a model may lead to confusion or problem with a singular matrix. There is an excellent function in `corrplot` package with the same name `corrplot()` that can visualize correlation structure of a set of predictors. The function has the option to reorder the variables in a way that reveals clusters of highly correlated ones.

```
# select non-survey numerical variables
sdat<-subset(sim.dat,select=c("age","income","store_exp","online_exp","store_trans","online_tr
# use bagging imputation here
imp<-preProcess(sdat,method="bagImpute")
sdat<-predict(imp,sdat)
# get the correlation matrix
```

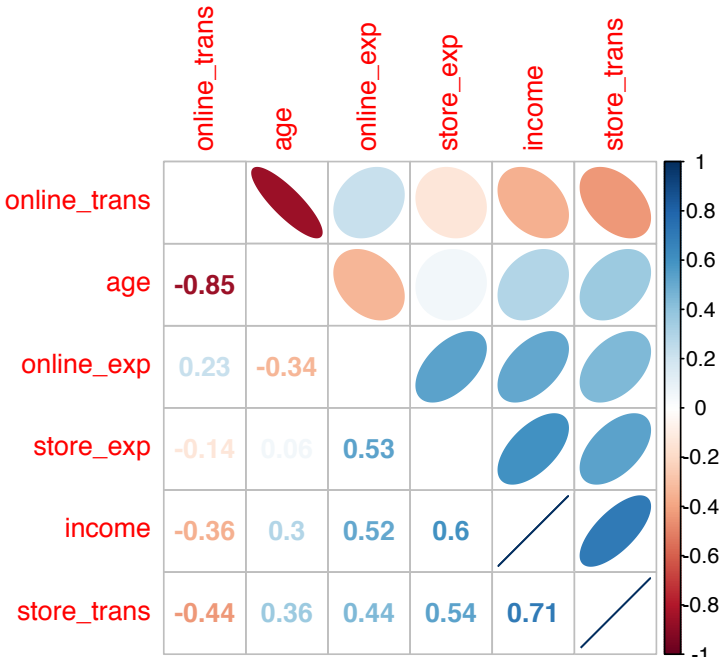


FIGURE 4.6: Correlation Matrix

```
correlation<-cor(sdat)
# plot
par(oma=c(2,2,2,2))
corrplot.mixed(correlation,order="hclust",tl.pos="lt",upper="ellipse")
```

Here use `corrplot.mixed()` function to visualize the correlation matrix (figure 4.6). The closer the correlation is to 0, the lighter the color is and the closer the shape is to a circle. The elliptical means the correlation is not equal to 0 (because we set the `upper = "ellipse"`), the greater the correlation, the narrower the ellipse. Blue represents a positive correlation; red represents a negative correlation. The direction of the ellipse also changes with the correlation. The correlation coefficient is shown in the lower triangle of the matrix. The variables relationship from previous scatter matrix (figure @ref(fig: scm)) are clear here: the negative correlation between age and online shopping, the positive correlation between income and amount of purchasing. Some correlation is very strong (such

as the correlation between `online_trans` and`age` is -0.85) which means the two variables contain duplicate information.

Section 3.5 of "Applied Predictive Modeling" (Kuhn and Johnston, 2013) presents a heuristic algorithm to remove a minimum number of predictors to ensure all pairwise correlations are below a certain threshold:

---

(1) Calculate the correlation matrix of the predictors.
(2) Determine the two predictors associated with the largest absolute pairwise correlation (call them predictors A and B).
(3) Determine the average correlation between A and the other variables. Do the same for predictor B.
(4) If A has a larger average correlation, remove it; otherwise, remove predictor B.
(5) Repeat Step 2-4 until no absolute correlations are above the threshold.

The `findCorrelation()` function in package `caret` will apply the above algorithm.

```
(highCorr<-findCorrelation(cor(sdat),cutoff=.75))
```

```
## [1] 1
```

It returns the index of columns need to be deleted. It tells us that we need to remove the first column to make sure the correlations are all below 0.75.

```
# delete highly correlated columns
sdat<-sdat[-highCorr]
# check the new correlation matrix
cor(sdat)
```

```
##                 income  store_exp online_exp
## income       1.0000000  0.6004006  0.5198623
## store_exp    0.6004006  1.0000000  0.5349527
## online_exp   0.5198623  0.5349527  1.0000000
```

```
## store_trans   0.7069595  0.5399121  0.4420638
## online_trans -0.3572884 -0.1367411  0.2256370
##              store_trans online_trans
## income         0.7069595   -0.3572884
## store_exp      0.5399121   -0.1367411
## online_exp     0.4420638    0.2256370
## store_trans    1.0000000   -0.4367544
## online_trans  -0.4367544    1.0000000
```

The absolute value of the elements in the correlation matrix after removal are all below 0.75. How strong does a correlation have to get, before you should start worrying about multicollinearity? There is no easy answer to that question. You can treat the threshold as a tuning parameter and pick one that gives you best prediction accuracy.

4.7 Sparse Variables

Other than the highly related predictors, predictors with degenerate distributions can cause the problem too. Removing those variables can significantly improve some models' performance and stability (such as linear regression and logistic regression but the tree based model is impervious to this type of predictors). One extreme example is a variable with a single value which is called zero-variance variable. Variables with very low frequency of unique values are near-zero variance predictors. In general, detecting those variables follows two rules:

- The fraction of unique values over the sample size
- The ratio of the frequency of the most prevalent value to the frequency of the second most prevalent value.

`nearZeroVar()` function in the `caret` package can filter near-zero variance predictors according to the above rules. In order to show the useage of the function, let's arbitaryly add some problematic variables to the origional data `sim.dat`:

```
# make a copy
zero_demo<-sim.dat
# add two sparse variable
# zero1 only has one unique value
# zero2 is a vector with the first element 1 and the rest are 0s
zero_demo$zero1<-rep(1,nrow(zero_demo))
zero_demo$zero2<-c(1,rep(0,nrow(zero_demo)-1))
```

The function will return a vector of integers indicating which columns to remove:

```
nearZeroVar(zero_demo,freqCut = 95/5, uniqueCut = 10)
```

As expected, it returns the two columns we generated. You can go ahead to remove them. Note the two arguments in the function `freqCut =` and `uniqueCut =` are corresponding to the previous two rules.

- `freqCut`: the cutoff for the ratio of the most common value to the second most common value
- `uniqueCut`: the cutoff for the percentage of distinct values out of the number of total samples

4.8 Re-encode Dummy Variables

A dummy variable is a binary variable (0/1) to represent subgroups of the sample. Sometimes we need to recode categories to smaller bits of information named "dummy variables." For example, some questionnaires have five options for each question, A, B, C, D, and E. After you get the data, you will usually convert the corresponding categorical variables for each question into five nominal variables, and then use one of the options as the baseline.

Let's encode `gender` and `house` from `sim.dat` to dummy variables. There are two ways to implement this. The first is to use `class.ind()` from `nnet` package. However, it only works on one variable at a time.

```
dumVar<-nnet::class.ind(sim.dat$gender)
head(dumVar)
```

```
##      Female Male
## [1,]      1    0
## [2,]      1    0
## [3,]      0    1
## [4,]      0    1
## [5,]      0    1
## [6,]      0    1
```

Since it is redundant to keep both, we need to remove one of them when modeling. Another more powerful function is `dummyVars()` from `caret`:

```
dumMod<-dummyVars(~gender+house+income,
                  data=sim.dat,
                  # use "origional variable name + level" as new name
                  levelsOnly=F)
head(predict(dumMod,sim.dat))
```

```
##   gender.Female gender.Male house.No house.Yes
## 1             1           0        0         1
## 2             1           0        0         1
## 3             0           1        0         1
## 4             0           1        0         1
## 5             0           1        0         1
## 6             0           1        0         1
##     income
## 1 120963.4
## 2 122008.1
## 3 114202.3
## 4 113616.3
## 5 124252.6
## 6 107661.5
```

`dummyVars()` can also use formula format. The variable on the right-hand side can be both categorical and numeric. For a numerical variable, the function will keep the variable unchanged. The advantage is that you can

apply the function to a data frame without removing numerical variables. Other than that, the function can create interaction term:

```
dumMod<-dummyVars(~gender+house+income+income:gender,
                  data=sim.dat,
                  levelsOnly=F)
head(predict(dumMod,sim.dat))
```

```
##   gender.Female gender.Male house.No house.Yes
## 1             1           0        0         1
## 2             1           0        0         1
## 3             0           1        0         1
## 4             0           1        0         1
## 5             0           1        0         1
## 6             0           1        0         1
##     income gender.Female:income gender.Male:income
## 1 120963.4             120963.4                0.0
## 2 122008.1             122008.1                0.0
## 3 114202.3                  0.0           114202.3
## 4 113616.3                  0.0           113616.3
## 5 124252.6                  0.0           124252.6
## 6 107661.5                  0.0           107661.5
```

If you think the impact income levels on purchasing behavior is different for male and female, then you may add the interaction term between `income` and `gender`. You can do this by adding `income: gender` in the formula.

4.9 Python Computing

Environmental Setup

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import numpy as np
```

```
import scipy as sp
import pandas as pd
import math

from sklearn.preprocessing import Imputer
from sklearn.preprocessing import StandardScaler

from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
```

4.9.1 Data Cleaning

5

Model Tuning Technique

5.1 Systematic Error and Random Error

Assume $\mathbf{X}$ is $n \times p$ observation matrix and $\mathbf{y}$ is response variable, we have:

$$\mathbf{y} = f(\mathbf{X}) + \epsilon$$

where ϵ is the random error with a mean of zero. The function $f(\cdot)$ is our modeling target, which represents the information of Y that can be explained by X. The main goals of estimating $f(\cdot)$ are inference and prediction, or sometimes both. In general, there is a trade-off between flexibility and interpretability of the model. So data scientists need to comprehend the delicate balance between the two.

Depending on the modeling purposes, the requirement for interpretability varies. If the prediction is the only goal, then as long as the prediction is accurate enough, the interpretability is not under consideration. In this case, people often use "black box" model, such as random forest, boosting tree, neural network, support vector machine and so on. These models are very flexible but nearly impossible to explain. Their predictive accuracy is usually higher on the training set, but not necessary when it predicts. It is not surprising since those models have a huge number of parameters and high flexibility that they can memorize the entire training data. A paper by Chiyuan Zhang et. al. in 2017 pointed out that "Deep neural networks (even just two-layer net) easily fit random labels" (Zhang et al., 2017). The traditional forms of regularization, such as weight decay, dropout, and data augmentation, fail to control generalization error. It poses a conceptual challenge to statistical theory and also calls our attention when we use such black-box models.

Assume we have $\hat{f}$ which is an estimator of f. Then we can further get $\hat{\mathbf{y}} = \hat{(}f)((X))$. The predicted error is divided into two parts, systematic error and random error:

$$E(\mathbf{y} - \hat{\mathbf{y}})^2 = E[f(\mathbf{X}) + \epsilon - \hat{f}(\mathbf{X})]^2 = \underbrace{E[f(\mathbf{X}) - \hat{f}(\mathbf{X})]^2}_{(1)} + \underbrace{Var(\epsilon)}_{(2)}$$

In the above equation, (1) is the systematic error, It exists because $\hat{f}$ usually does not completely model the "systematic relation" between X and y, where the system relation refers to the stable relationship that exists on different samples. This part of the error can be improved by improving the model; (2) is the random error which represents the part of the response that can not be explained by current input data. A more complex model does not reduce the error. The biggest problem with black-box models is to fit random error as well, i.e., over-fitting. The notable feature of random error is that it can not be repeated on different samples. So a way to determine whether there is overfitting is to reserve a part of the sample as a test set and then check the performance of the trained model on the test data. Note that overfitting is a general problem, and any model may be overly fitted. Because black-box model usually has a large number of parameters, it is more suspectable to over-fitting.

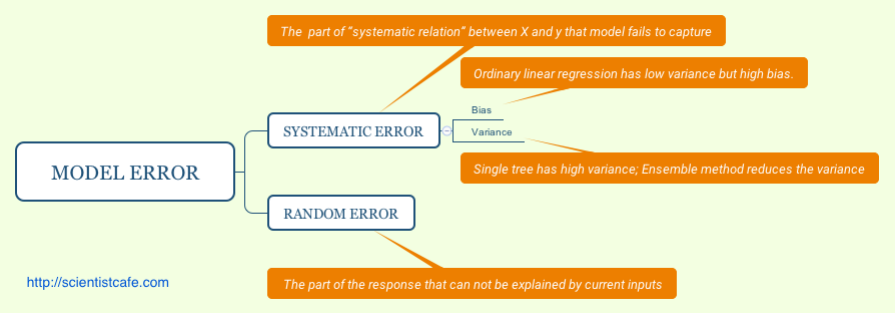


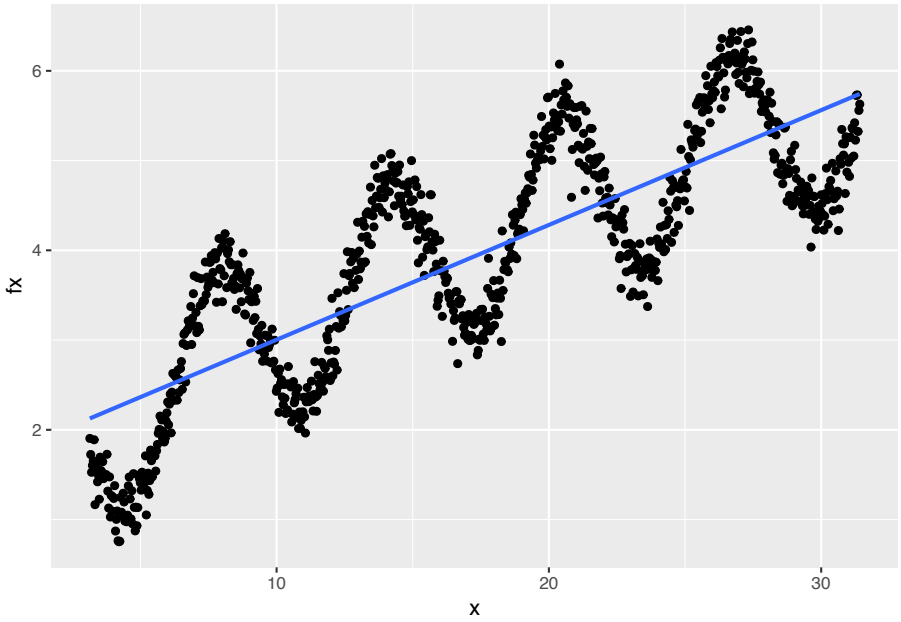
FIGURE 5.1: Types of Model Error

The systematic error can be further decomposed as:

$$\begin{aligned} E[f(\mathbf{X}) - \hat{f}(\mathbf{X})]^2 &= E\left(f(\mathbf{X}) - E[\hat{f}(\mathbf{X})] + E[\hat{f}(\mathbf{X})] - \hat{f}(\mathbf{X})\right)^2 \\ &= E\left(E[\hat{f}(\mathbf{X})] - f(\mathbf{X})\right)^2 + E\left(\hat{f}(\mathbf{X}) - E[\hat{f}(\mathbf{X})]\right)^2 \\ &= [Bias(\hat{f}(\mathbf{X}))]^2 + Var(\hat{f}(\mathbf{X})) \end{aligned}$$

The systematic error consists of two parts, $Bias(\hat{f}(\mathbf{X}))$ and $Var(\hat{f}(\mathbf{X}))$. To minimize the systematic error, we need to minimize the bias and variance. The bias represents the error caused by the approximation of the reality of the model, which may be very complex. For example, linear regression assumes that there is a linear relationship between the independent variable and the response, but the perfect linear relationship in real life is not common. The relationship between x and fx in the following figure is non-linear. Therefore, despite a large sample size, linear regression can not give the accurate prediction. In other words, in this case, the prediction of the linear regression model has a high bias.

```
library(grid)
library(lattice)
library(ggplot2)
source("https://raw.githubusercontent.com/happyrabbit/DataScientistR/master/R/multiplot.r")
# randomly simulate some non-linear samples
x=seq(1,10,0.01)*pi
e=rnorm(length(x),mean=0,sd=0.2)
fx<-sin(x)+e+sqrt(x)
dat=data.frame(x,fx)
# plot fitting result
ggplot(dat,aes(x,fx))+
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```

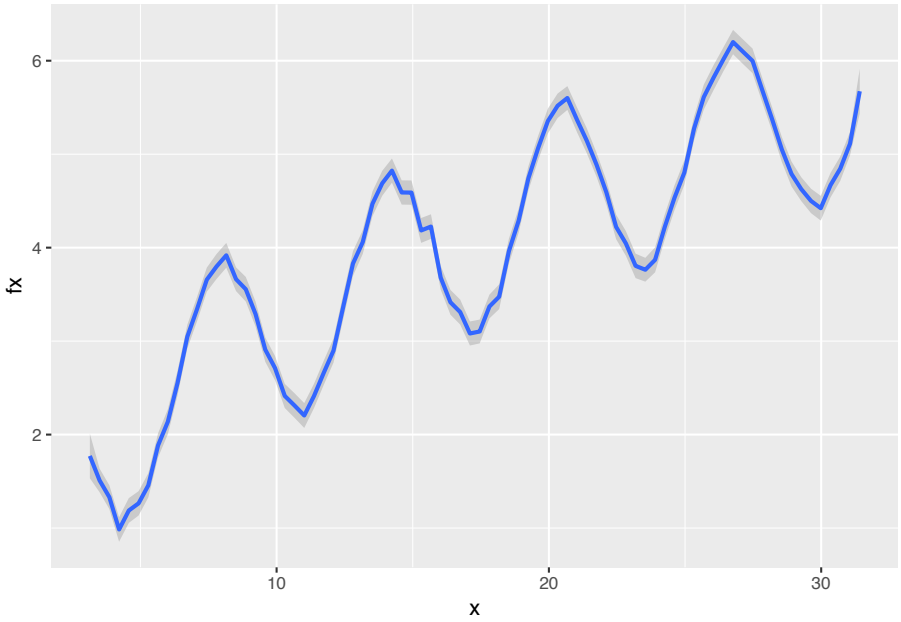


The estimated parameters will be different for the different training data. Intuitively, the estimated variance indicates that if we fit the same model with different data sets, how much the estimated results will change. Ideally, the change is small. For high variance models, small changes in the training data result in very different estimates. In general, a model with high flexibility also has high variance., such as the CART tree, and the initial boosting method. The Random Forest and Gradient Boosting Model aim to reduce the variance by summarizing the results obtained on different samples.

The blue curve in the figure below is obtained by fitting the above nonlinear observations by a smoothing method, which is highly flexible and can fit the current data tightly:

```
ggplot(dat,aes(x,fx))+geom_smooth(span = 0.03)
```

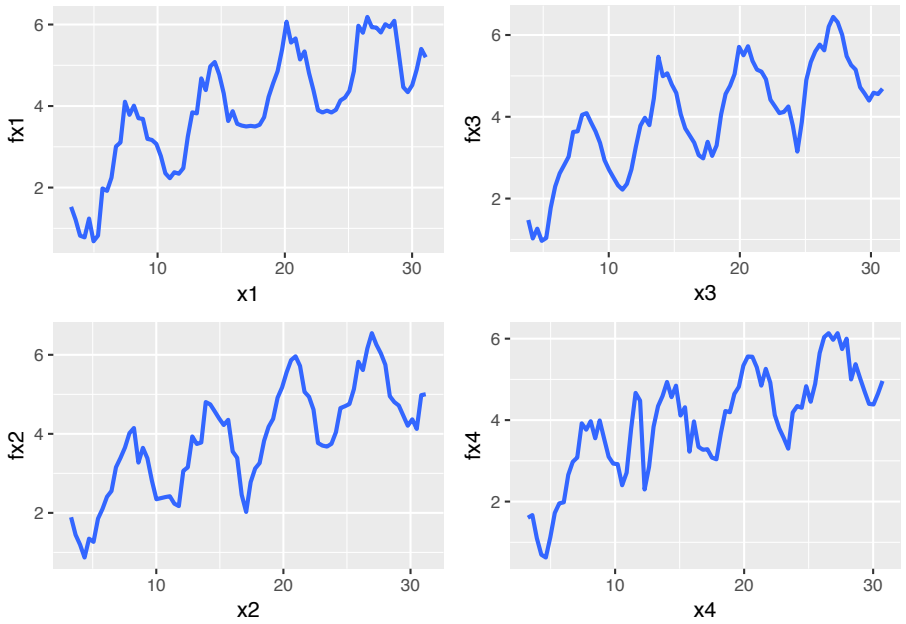
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



However, this method has a high variance, and if we simulate different subsets of the sample, the result curve will change significantly:

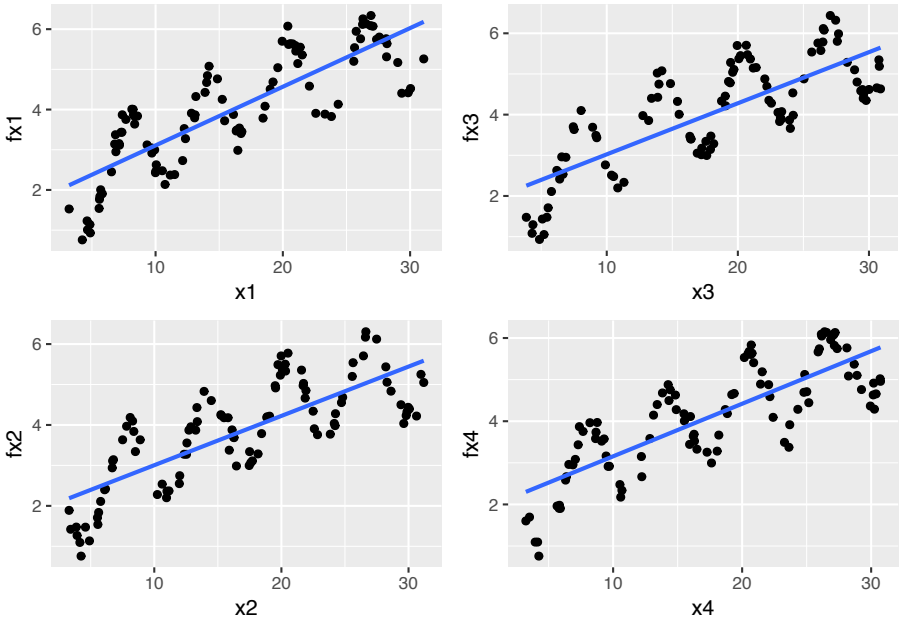
```
# set random seed
set.seed(2016)
# sample part of the data to fit model
# sample 1
idx1=sample(1:length(x),100)
dat1=data.frame(x1=x[idx1],fx1=fx[idx1])
p1=ggplot(dat1,aes(x1,fx1))+geom_smooth(span = 0.03)
# sample 2
idx2=sample(1:length(x),100)
dat2=data.frame(x2=x[idx2],fx2=fx[idx2])
p2=ggplot(dat2,aes(x2,fx2))+geom_smooth(span = 0.03)
# sample 3
idx3=sample(1:length(x),100)
dat3=data.frame(x3=x[idx3],fx3=fx[idx3])
p3=ggplot(dat3,aes(x3,fx3))+geom_smooth(span = 0.03)
# sample 4
idx4=sample(1:length(x),100)
dat4=data.frame(x4=x[idx4],fx4=fx[idx4])
```

```
p4=ggplot(dat4,aes(x4,fx4))+geom_smooth(span = 0.03)
multiplot(p1,p2,p3,p4,cols=2)
```



Fitting the linear model using the same four subsets, the result barely changes:

```
p1=ggplot(dat1,aes(x1,fx1))+
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
p2=ggplot(dat2,aes(x2,fx2))+
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
p3=ggplot(dat3,aes(x3,fx3))+
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
p4=ggplot(dat4,aes(x4,fx4))+
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
multiplot(p1,p2,p3,p4,cols=2)
```



In general, the variance ($Var(\hat{f}(\mathbf{X}))$) **increases** and the bias ($Bias(\hat{f}(\mathbf{X}))$) **decreases** as the model flexibility increases. Variance and bias together determine the systematic error (or mean square error, MSE). As we increase the flexibility of the model, at first the rate at which $Bias(\hat{f}(\mathbf{X}))$ decreases is faster than $Var(\hat{f}(\mathbf{X}))$, so the MSE decreases. However, to some degree, higher flexibility has little effect on $Bias(\hat{f}(\mathbf{X}))$ but $Var(\hat{f}(\mathbf{X}))$ increases significantly, so the MSE increases.

5.1.1 Measurement Error in the Response

The random error (ϵ) reflects the measurement error in the response. This part of the error is irreducible, and so it makes the root mean square error (RMSE) and R^2 have the corresponding upper and lower limits. RMSE and R^2 are commonly used performance measures for the regression model which we will talk in more detail later. Therefore, the random error term not only represents the part of fluctuations the model cannot explain but also contains measurement error in the response variables. Section 20.2 of Applied Predictive Modeling (Kuhn and Johnston, 2013)

has an example that shows the effect of the measurement error in the response variable on the model performance (RMSE and R^2).

The authors increased the error in the response proportional to a base level error which was gotten using the original data without introducing extra noise. Then fit a set of models repeatedly using the "contaminated" data sets to study the change of $RMSE$ and R^2 as the level of noise. Here we use clothing consumer data for a similar illustration. Suppose many people do not want to disclose their income and so we need to use other variables to establish a model to predict income. We set up the following model:

```
# load data
sim.dat <- read.csv("https://raw.githubusercontent.com/happyrabbit/DataScientistR/master/Data/
ymad<-mad(na.omit(sim.dat$income))
# calculate z-score
zs<-(sim.dat$income-mean(na.omit(sim.dat$income)))/ymad
# which(na.omit(zs>3.5)): identify outliers
# which(is.na(zs)): identify missing values
idex<-c(which(na.omit(zs>3.5)),which(is.na(zs)))
# delete rows with outliers and missing values
sim.dat<-sim.dat[-idex,]
fit<-lm(income~store_exp+online_exp+store_trans+online_trans,data=sim.dat)
```

The output shows that without additional noise, the root mean square error (RMSE) of the model is 29567, R^2 is 0.6.

Let's add various degrees of noise (0 to 3 times the RMSE) to the variable `income`:

$$RMSE \times (0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0)$$

```
noise<-matrix(rep(NA,7*nrow(sim.dat)),nrow=nrow(sim.dat),ncol=7)
for (i in 1:nrow(sim.dat)){
noise[i,]<-rnorm(7,rep(0,7),summary(fit)$sigma*seq(0,3,by=0.5))
}
```

We then examine the effect of noise intensity on R^2 for models with different complexity. The models with complexity from low to high are: ordinary linear regression, partial least square regression(PLS), multivari-

ate adaptive regression spline (MARS), support vector machine (SVM, the kernel function is radial basis function), and random forest.

```
# fit ordinary linear regression
rsq_linear<-rep(0,ncol(noise))
for (i in 1:7){
withnoise<-sim.dat$income+noise[,i]
fit0<-lm(withnoise~store_exp+online_exp+store_trans+online_trans,data=sim.dat)
rsq_linear[i]<-summary(fit0)$adj.r.squared
}
```

PLS is a method of linearizing nonlinear relationships through hidden layers. It is similar to the principal component regression (PCR), except that PCR does not take into account the information of the dependent variable when selecting the components, and its purpose is to find the linear combinations (i.e., unsupervised) that capture the most variance of the independent variables. When the independent variables and response variables are related, PCR can well identify the systematic relationship between them. However, when there exist independent variables not associated with response variable, it will undermine PCR's performance. And PLS maximizes the linear combination of dependencies with the response variable. In the current case, the more complicated PLS does not perform better than simple linear regression.

```
# pls: conduct PLS and PCR
library(pls)
rsq_pls<-rep(0,ncol(noise))
# fit PLS
for (i in 1:7){
withnoise<-sim.dat$income+noise[,i]
fit0<-plsr(withnoise~store_exp+online_exp+store_trans+online_trans,data=sim.dat)
rsq_pls[i]<-max(drop(R2(fit0, estimate = "train",intercept = FALSE)$val))
}
```

```
# earth: fit mars
library(earth)
rsq_mars<-rep(0,ncol(noise))
for (i in 1:7){
withnoise<-sim.dat$income+noise[,i]
```

```
fit0<-earth(withnoise~store_exp+online_exp+store_trans+online_trans,data=sim.dat)
rsq_mars[i]<-fit0$rsq
}
```

```
# caret: awesome package for tuning predictive model
library(caret)
rsq_svm<-rep(0,ncol(noise))
# Need some time to run
for (i in 1:7){
idex<-which(is.na(sim.dat$income))
withnoise<-sim.dat$income+noise[,i]
trainX<-sim.dat[,c("store_exp","online_exp","store_trans","online_trans")]
trainY<-withnoise
fit0<-train(trainX,trainY,method="svmRadial",
            tuneLength=15,
            trControl=trainControl(method="cv"))
rsq_svm[i]<-max(fit0$results$Rsquared)
}
```

```
# randomForest: random forest model
library(randomForest)
rsq_rf<-rep(0,ncol(noise))
# ntree=500 number of trees
# na.action = na.omit  ignore missing value
for (i in 1:7){
withnoise<-sim.dat$income+noise[,i]
fit0<-randomForest(withnoise~store_exp+online_exp+store_trans+online_trans,data=sim.dat,ntree=
rsq_rf[i]<-tail(fit0$rsq,1)
}
library(reshape2)
rsq<-data.frame(cbind(Noise=c(0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0),rsq_linear,rsq_pls,rsq_mars,r
rsq<-melt(rsq,id.vars="Noise",measure.vars=c("rsq_linear","rsq_pls","rsq_mars","rsq_svm","rsq_
```

```
library(ggplot2)
ggplot(data=rsq, aes(x=Noise, y=value, group=variable, colour=variable)) +
    geom_line() +
    geom_point()+
  ylab("R2")
```

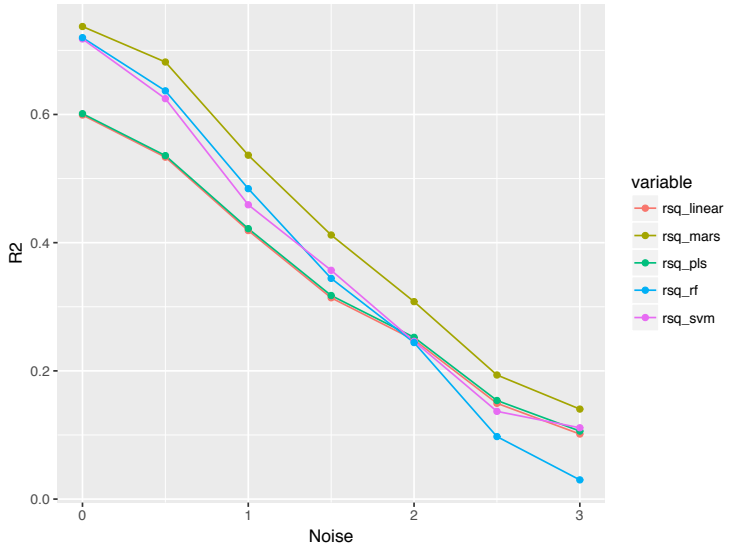


FIGURE 5.2: Test set R^2 profiles for income models when measurement system noise increases.

6

References

Bibliography

Box G, C. D. (1964). An analysis of transformations. *Journal of the Royal Statistical Society*, pages 211–252.

de Waal, T., Pannekoek, J., and Scholtus, S. (2011). *Handbook of Statistical Data Editing and Imputation*. John Wiley and Sons.

Donoho, D. (2015). 50 years of data science.

Geladi P, K. B. (1986). Partial least squares regression: A tutorial. *Analytica Chimica Acta*, (185):1–17.

Iglewicz, B. and Hoaglin, D. (1993). How to detect and handle outliers. *The ASQC Basic References in Quality Control: Statistical Techniques*, 16.

Jolliffe, I. (2002). *Principla component analysis*. Springer, 2nd edition.

Kuhn, M. and Johnston, K. (2013). *Applied Predictive Modeling*. Springer.

L, B. (1966a). Bagging predictors. *Machine Learning*, 24(2):123–140.

L, V. (1984). A theory of the learnable. *Communications of the ACM*, 27:1134–1142.

M, K. and L, V. (1989). Cryptographic limitations on learning boolean formulae and finite automata. In *In "Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*.

M, S. T. and F, P. (2007b). Handling missing values when applying classification models. *Journal of Machine Learning Research*, 8:1625–1657.

Mulaik, S. (2009). *Foundations of factor analysis*. Boca Raton: Chapman & Hall/CRC, 2nd edition.

Serneels S, Nolf ED, E. P. (2006). Spatial sign preprocessing: A simple way to impart moderate robustness to multivariate estimators. *Journal of Chemical Information and Modeling*, 46(3):1402–1409.

T, H. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:340–354.

Y, A. and D, G. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. *arXiv :1611.03530*.