STA 321 E-pack: Advanced Statistics

Cheng Peng

West Chester University

Contents

```
install.packages("bookdown")
# or the development version
# devtools::install_github("rstudio/bookdown")
```

Chapter 1

Introduction

This *E-coursepack* (a.k.a. **E-Pack**) is a self-contained homegrown Ebook that contains all topics covered in current STA321 at WCU.

1.1 Why This E-Pack?

Since this is an advanced-topic course that covers three major topics in linear regression modeling, generalized linear regression modeling, and time series modeling. These topics are typically covered in three different textbook. This E-pack contains all topics and are delivered using parametric and non-parametric methods.

1.2 Components Statistical Reports

Since is a project-based modeling class. The assignments are building-blocks of about 3 projects that cover linear regression, generalized linear regression and time series. Every will use data sets that are real-world or close to the real-world data for all projects. All statistical reports must have the following key components.

A. Introduction

Provide some background on the problem. This includes the motivations and objectives of the analysis.

B. Materials

Some information about the data should be described here. For example, methods of data collection, variable names, and definitions, potential data challenges, etc. You could use subsections to organize your work.

C. Methodology

Describe all the methods (including justifications for using the methods) you used to gather and analyze the data here. You need to provide extensive details so that anyone can replicate your results.

D. Results and Conclusions

Show your audience all your results and conclusions with justifications. Write this section in a way that enables a non-statistician to understand the content. Be very specific.

E. General Discussion

Talk about results (with justifications) and link them to real-world implications. Pay attention to whether the research questions were well addressed.

F. References (if any)

Everything you used in the analysis including notes and blogs from the internet, textbook, journal articles, etc.

G. Appendices (if any)

Additional output tables and graphics that important but not fundamental to the report should be placed in this section.

1.3 Basics of RStudio

This brief note will introduce the basics of Rstudio, R Markdown, and R.

- **RStudio** is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging, and workspace management.
- **R Markdown** is a file format for making dynamic documents with R. An R Markdown document is written in markdown (an easy-to-write plain text format) and contains chunks of embedded R code and the output generated from the R code. This note is written in R Markdown. This is also a tutorial showing how to use R Markdown to write an R Markdown report. – RStudio documentation.
- **R** is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a

different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

1.3.1 RStudio GUI

The RStudio interface consists of several windows. I insert an image of a regular RStudio GUI.

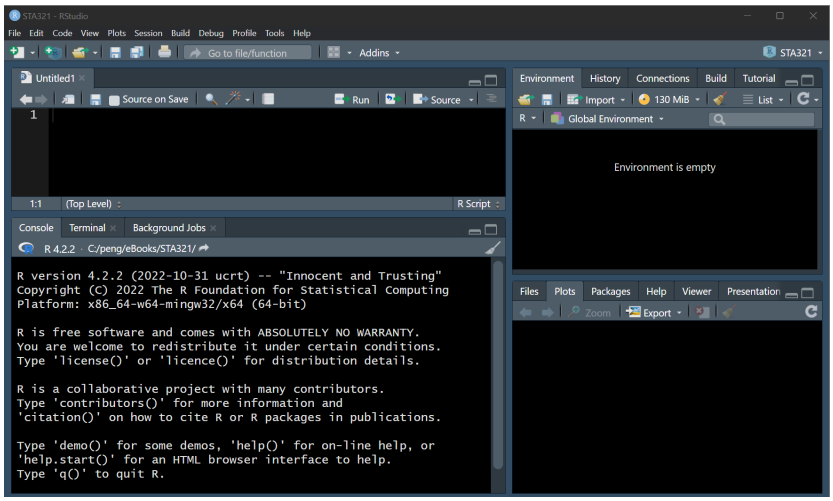


Figure 1.1: The GUI of RStudio

1.3.1.1 Console

We can type commands directly into the console, or write in a text file, and then send the command to the console. It is convenient to use the console if your task involves one line of code. Otherwise, we should always use an editor to write code and then run the code in the Console.

1.3.1.2 Source Editor

Generally, we will want to write programs longer than a few lines. The Source Editor can help you open, edit and execute these programs.

1.3.1.3 Environment Window

The Environment window shows the objects (i.e., data frames, arrays, values, and functions) in the environment (workspace). We can see the descriptive information such as types as the dimension of the objects in your environment. We also choose a data source from the environment to view in the source window like a spreadsheet.

1.3.1.4 System and Graphic files

The Files tab has a navigable file manager, just like the file system on your operating system. The Plot tab is where the graphics you create will appear. The Packages tab shows you the packages that are installed and those that can be installed (more on this just now). The Help tab allows you to search the R documentation for help and is where the help appears when you ask for it from the Console.

1.3.2 RMarkdown

An R Markdown document is a text-based file format that allows you to include both descriptive text, code blocks, and code output. It can be converted to other types of files such as PDF, HTML, and WORD that can include code, plots, outputs generated from the code chunks.

1.3.2.1 Code Chunk

In R Markdown, we can embed R code in the code chunk defined by the symbol ```` ```{} ```` and closed by ```` ``` ````. The symbol ‘, also called **backquote** or **backtick**, can be found on the top left corner of the standard keyboard as shown in the following.

Figure 1.2: The location of backquote on the standard keyboard

There are two code chunks: executable and non-executable chunks. The following code chunk is non-executable since is no argument specified in the `{}`.

````
```{}
This is a code chunk
```
````

Figure 1.3: Non-executable code chunk.

```
This is a code chunk
```

To write a code chunk that will be executed, we can simply put the letter `r` inside the curly bracket. If the code chunk is executable, you will the green arrow on the top-right corner of the chunk.

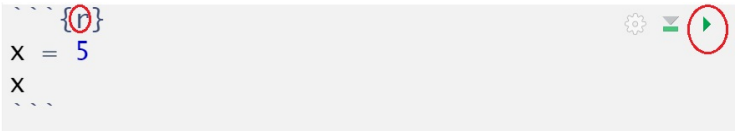


Figure 1.4: Executable code chunk.

We can define R objects with and without any outputs. In the above R code chunk, we define an R object under the name `x` and assign value 5 to `x` (the first line of the code). We also request an output that prints the value of `x`. The above executable code chunk gives output `[1] 5` in the Markdown document. The same output in the knit output files is in a box with a transparent background in the form `## [1] 5`.

```
x = 5
x
```

```
## [1] 5
```

We can also use an argument in the code chunk to control the output. For example, the following code chunk will be evaluated when kitting to other formats of files. But we can still click the green arrow inside the code chunk to evaluate the code.

```
x = 5
x
```

1.3.2.2 Graphics Generated from R Code Chunks

In the previous sub-sections, we include images from external image files. In fact, can use the R function to generate graphics (other than interacting with plots, etc.) in the markdown file & knit. For instance, we can generate the following image from R and include it in the Markdown document and the knitter output files.

Unlike the way of including an external image in to the R code chunk in which we use chunk option **out.width="80%"** or **out.height = "60%", out.width="80%"** to specify the dimension of the displayed image, The graphics generated from R need a different option to specify the dimension. The dimension of the following graph is specified by `{r, fig.align="center", fig.height=5, fig.width=5, fig.cap= "R Generated Graph"}`.

```
data(iris)
plot(iris[,-5])
```

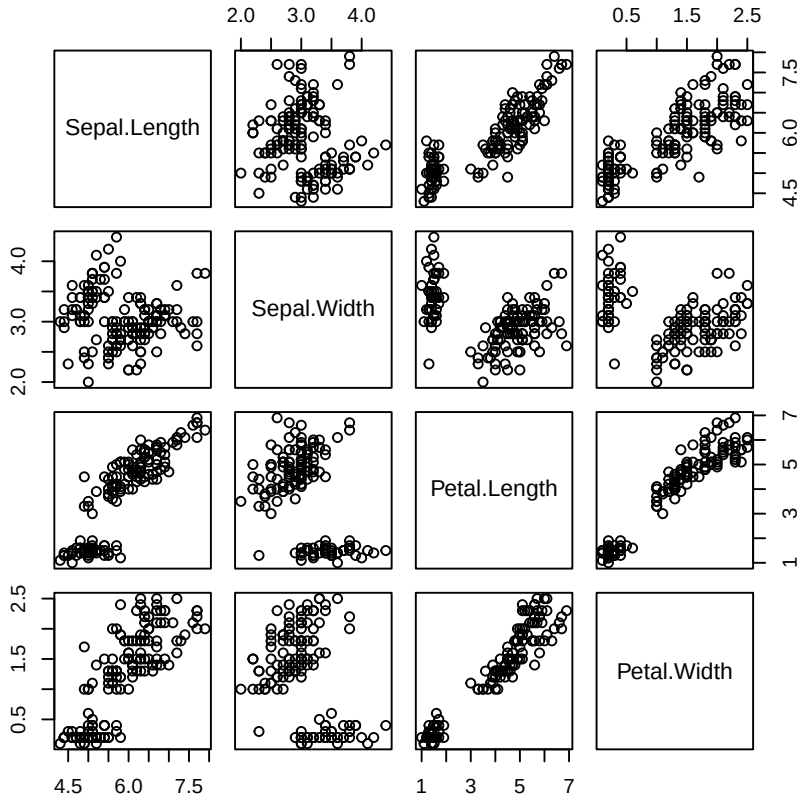


Figure 1.5: R Generated Graph

1.4 Collaborative Platforms

There are many platforms and technologies available for applied statisticians and data scientists. We will use RPubs (https://rpubs.com/) and GitHub Repository (https://github.com/) in this class.

1.4.1 RPubs

RPubs is a free web server provided by RStudio (recently changed to **Posit**) that you can use it to publish you analytic reports and code and share with your peers and friends worldwide.

To use this resource, you need to sign up an account with RPubs first. Onece you set up your RPubs account, you can then create reports via RMarkdown and publish them on RPubs in the HTML format. You can share your work with people by providing the hyperlink to them.

In this class, all preject reports are required to be published on RPubs so I can read your work directly from RPubs. You need to submit the links to you reports via D2L dropbox.

1.4.2 GitHub Repository

GitHub is an online software development platform. It's used for storing, tracking, and collaborating on software projects.

To use it, you need to create an account. After you set your GitHub account, you can upload your files (text, code, photos, videos, etc) to the repository. You can also use GitHub to host your personal web page (static).

In this class, all data sets you are going to use in your assignments and project are required to uploaded to your specific repository so you can read your data sets directly from GitHub repository.

1.5 Github

1.5.1 What is Github?

GitHub is a social networking site for programmers to share their code. Many companies and organizations use it to facilitate project management and collaboration. It is the most prominent source code host, with over 60 million new repositories.

Most importantly, it is free. We can also use this resource to host web pages. Many images and data sets that I used are stored on GitHub. You need to register a GitHub account (https://github.com/login) to use create GitHub repositories and download and install Git for version control (version control

is not required for this course, but is is extremely important in practice). The following Figure 1.5 shows the GitHub front page.

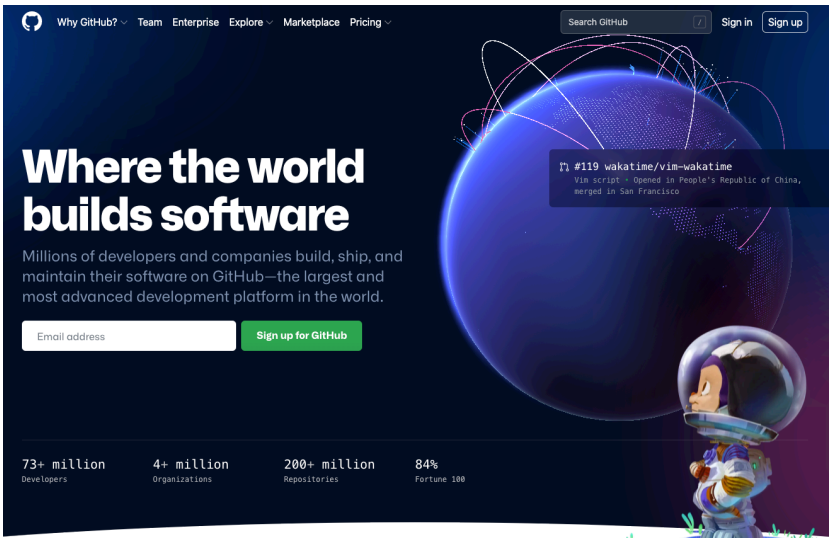


Figure 1.6: GitHub front page.

1.5.2 Getting Started with GitHub

We will use screenshots to demonstrate how to create a repository, folders, and files.

1. After you logged into your account, you click the "continue for free" button located at the bottom of the following page (screenshot, Figure 1.6)

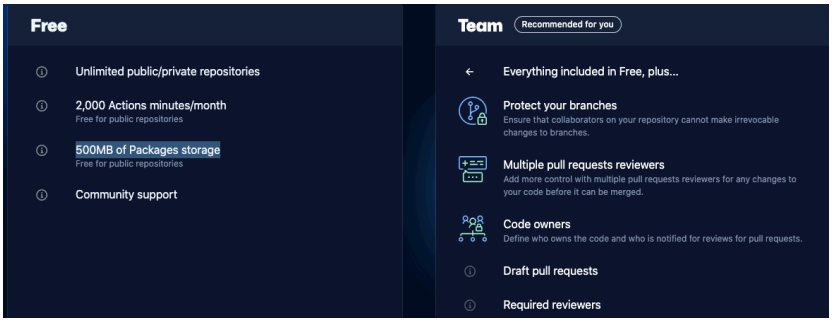


Figure 1.7: The first page after logging-on.

2. Now you see your Github front page. Click the green button "create repository" on the left panel. Our first repository is called "sta553" (Figure 1.7)

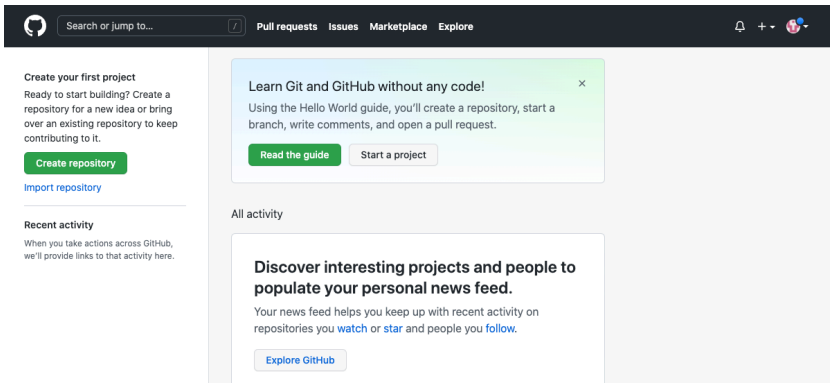


Figure 1.8: Starting creating repository.

3. To organize files in the repository `sta553`, We want folders for different files. To create a folder under `sta553`, click the hyperlink 'creating a new file (Figure 1.8)

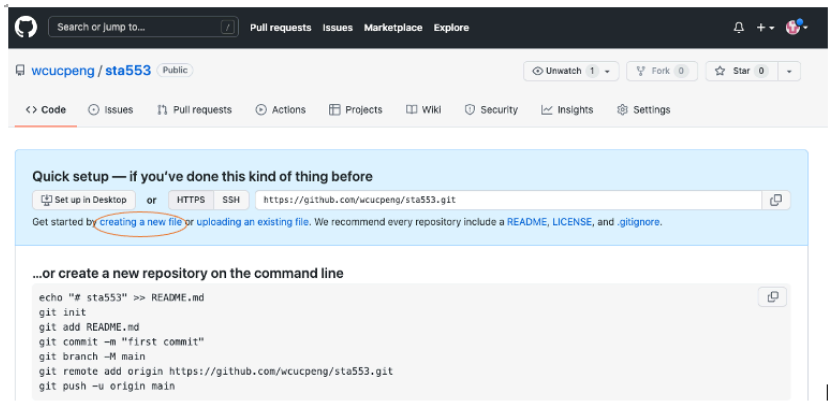


Figure 1.9: Creating new folders to organize your files.

4. The first folder to create is called the `data` folder which will be used to store data files. After typing "data/", a new box appears under the "data" folder, type the first file name - readme, and the content of the file (see the screenshot). In the end, click the green button "Commit new file" to complete the creation of the first folder in the repository `data` (Figure 1.9).
5. To load the data file to the `data` folder, we click the drop-down menu on the top right corner and select `upload files` (Figure 1.10)
6. To create other folders under `sta553`, we click `Creating New File`, and we can create a new folder `image` similarly (Figure 1.11).

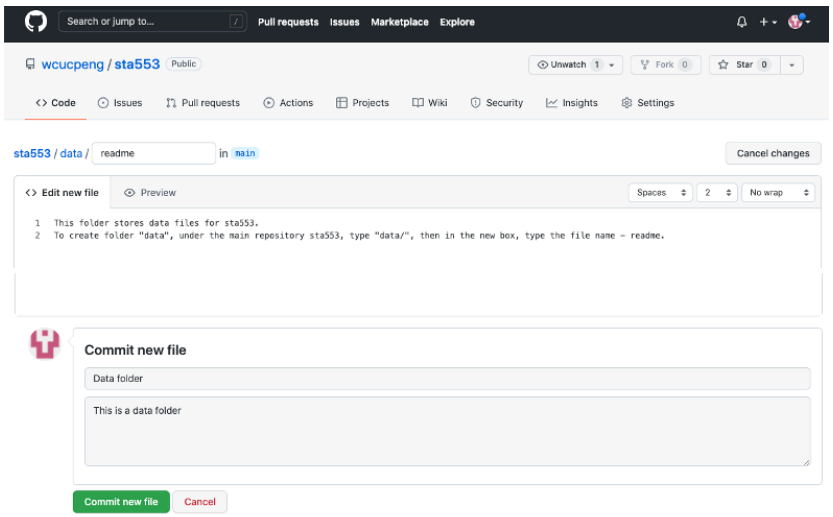


Figure 1.10: Creating new files in a folder created earlier.

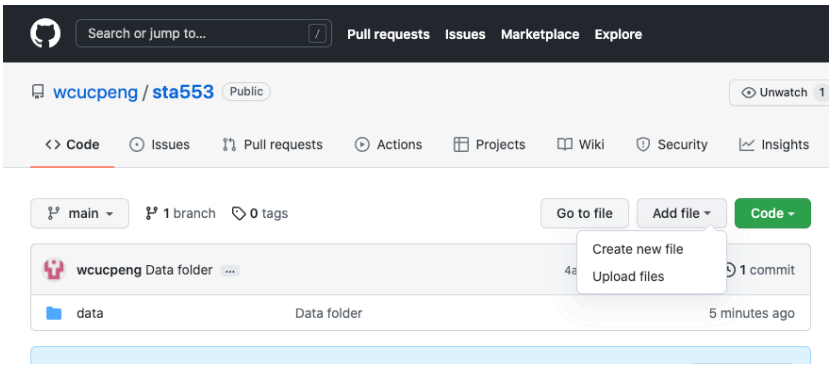


Figure 1.11: Creating another new folder.

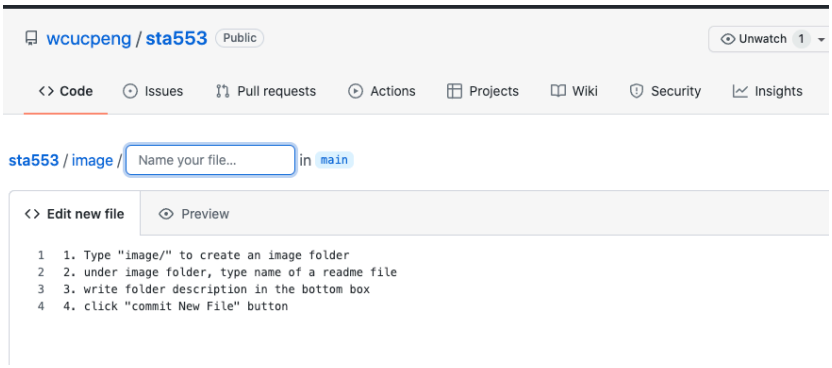


Figure 1.12: Creating new folders for specialized files such as image files.

7. To create a new repository, Click the drop-down menu on the top right corner and select `New repository` to create a new repository (Figure 1.12).

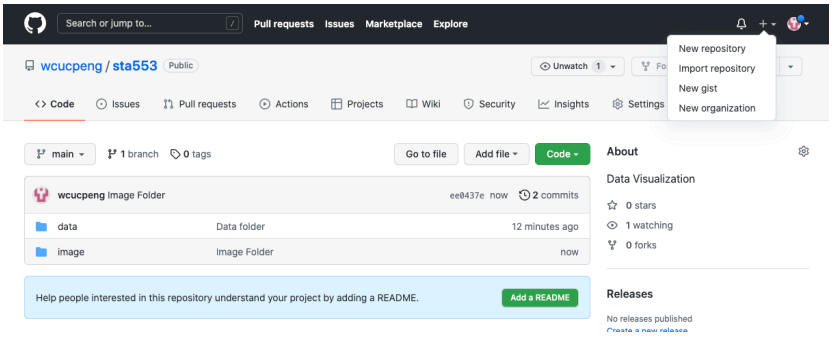


Figure 1.13: Creating new repositories for different projects

Chapter 2

Data Input and Output in R

Loading data into R can be quite frustrating since there so many different ways available for different types data files. We compile a short note to list some of these functions that commonly used to read the most commonly used formats of data in practice. We may also want to use some data sets built in some libraries (packages) in R for the purpose of illustration and practice.

2.1 Built-in Data

Build-in data are also ready in R format. To get the list of available data sets in **base R**, we can use data(). However, if we want to get the list of data sets available **in a package**, we first need to **load that package** then **data()** command shows the available data sets in that package.

For example, the library `datasets` in base R has a number of built-in data sets, we can use command `data()` or `ls("package:datasets")` to list all of these built-in data sets. When opening the R session, all built-in data sets in library `datasets` are all automatically loaded to the current working directory. We can simply use any of these data set The well-known `iris` is one of these built-in data sets. Next, we make a pair-wise scatter plot for all numerical variables in the iris data set in the following

```
plot(iris[,-5])  # the 5th column is species of iris flower.
```

To use a built-in data set that is not in the library `datasets` in base R, we need to load the library first. After the library is loaded, we can use command to list all data sets in that library. For example, `PimaIndiansDiabetes2` is a frequently used data set in logistic regression modeling. It is a built-in data set

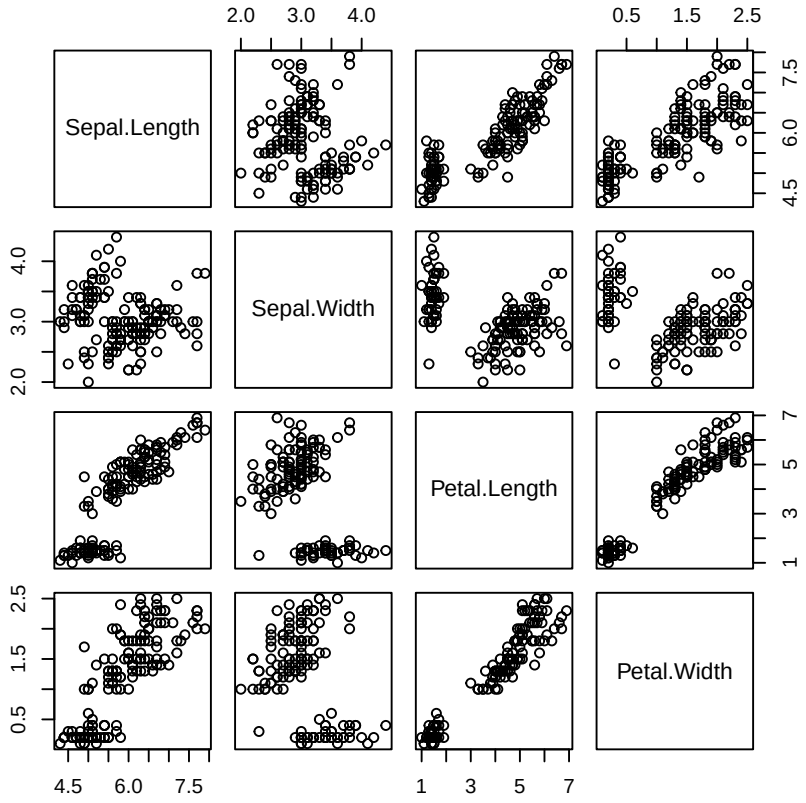


Figure 2.1: Pair-wise scatter plot of iris data

in the library `mlbench`. If we don't load library `mlbench` (Machine Learning Benchmark Problems), we will not be able to access that data set.

```
summary(PimaIndiansDiabetes2)
```

The message returned from the above code is `Error in summary(PimaIndiansDiabetes2): object 'PimaIndiansDiabetes2' not found`. However, if we load the library first, then generate the summary table, we will have

```
library(knitr)
if (!require("mlbench")) {      # check whether *mlbench* is installed in the machine
   install.packages("mlbench")  # if not, install it
   library(mlbench)             # then load the package!
}
data(PimaIndiansDiabetes2)      # load the specific data set to the working directory
summary(PimaIndiansDiabetes2)   # use the data set for analysis
```

```
##     pregnant         glucose         pressure         triceps
##  Min.   : 0.000   Min.   : 44.0   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.:22.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :29.00
##  Mean   : 3.845   Mean   :121.7   Mean   : 72.41   Mean   :29.15
##  3rd Qu.: 6.000   3rd Qu.:141.0   3rd Qu.: 80.00   3rd Qu.:36.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##                   NA's   :5       NA's   :35       NA's   :227
##     insulin            mass          pedigree           age        diabetes
##  Min.   : 14.00   Min.   :18.20   Min.   :0.0780   Min.   :21.00   neg:500
##  1st Qu.: 76.25   1st Qu.:27.50   1st Qu.:0.2437   1st Qu.:24.00   pos:268
##  Median :125.00   Median :32.30   Median :0.3725   Median :29.00
##  Mean   :155.55   Mean   :32.46   Mean   :0.4719   Mean   :33.24
##  3rd Qu.:190.00   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00
##  Max.   :846.00   Max.   :67.10   Max.   :2.4200   Max.   :81.00
##  NA's   :374      NA's   :11
```

We can also use `data()` to list all buit-in data set in the package

```
library(mlbench)              # load the package available in the machine
data(package = 'mlbench')     # list all data sets available in the package
```

Note: if the package name is not specified in the argument of `data()`, **all** built-in data sets in **all loaded packages** will be listed!

2.2 Loading External Data Sets

Loading an external data set to can be a challenge depending on the format and the structure of the data.

Before moving on and discover how to load data into R, it might be

useful to go over the following checklist that will make it easier to import the data correctly into R:

- If working with spreadsheets, the first row is usually reserved for the header, while the first column is used to identify the sampling unit;
- Avoid names, values, or fields with blank spaces. Otherwise, each word will be interpreted as a separate variable, resulting in errors that are related to the number of elements per line in your data set;
- If we want to concatenate words, inserting a `.` in between two words instead of a space; Short names are preferred over longer names;
- Try to avoid using names that contain symbols, such as ?, $,%, ^, &, *, (,),-,#, ?„,<,>, /, |, , [,] ,{, and };
- Delete any comments in the Excel file to avoid extra columns or NA's to be added to the file; and
- Make sure that any missing values in the data set are indicated with NA.

2.2.1 Preparing R Workspace

It is a good practice to empty all R objects defined in other R sessions to avoid miss-use (unintentional-use) of R objects with simple easy names that were defined in the other sessions.

We might want to start an environment that is **NOT** filled with data and values from other sessions. The simplest way is to delete all existing R objects before starting new analysis using the following line of code

```
rm(list=ls())
```

The `rm()` function removes objects from a specified environment. We can then use `ls()` to check whether all previously defined R objects were removed.

2.2.2 Setting-up Working Directory

It is dependent on whether you are writing R script or RMarkdown. If writing an R script, it is suggested to set-up a working directory for the specific analysis task using the following R function

```
setwd("<location of your dataset>")
```

If writing RMarkdown document, by the default, the folder we save the RMarkdown document is the automatically set as working directory (also called document directory). `setwd()` in the code chunk does not work in RMarkdown!

2.3 Read Common Files into R

The following basic R functions focus on getting spreadsheets into R, rather than Excel or other type of files. There are ways of importing other files into R using various R functions in different packages.

2.3.1 Read TXT files with read.table()

If you have a `.txt` or a `tab-delimited` text file, we can easily import it with the basic R function read.table().

- **Data with No Column Names**

In other words, the contents of the file will look similar to this

```
1   6   a
2   7   b
3   8   c
4   9   d
5   10  e
```

If the above data file is saved in a local folder, say, `C:\peng\eBooks\STA321\fakeDat.txt`, we use the following code to load this data to R

```
myFakeData <- read.table("C:\\peng\\eBooks\\STA321\\fakeDat.txt")
```

If the above data file is on a web server, say, `http://www.someserver.com/STA321/fakeDat.txt`, we use the following code to load this data to R

```
myFakeData <- read.table("http://www.someserver.com/STA321/fakeDat.txt")
```

- **Data with Good Column Names**

The above fake data file does not have column names, the loaded R data set will be automatically assigned names `V1`, `V2`, and `V3` to the three corresponding columns. If the data file has column names and we want to use it in the R data frame, we need to use the `header` argument.

```
ID num  char
1   6   a
2   7   b
3   8   c
4   9   d
5   10  e
```

We can use the following code to load the data correctly in R.

```
myFakeData <- read.table("C:\\peng\\eBooks\\STA321\\fakeDat.txt", header = TRUE)
```

or

```
myFakeData <- read.table("http://www.someserver.com/STA321/fakeDat.txt", header = TRUE)
```

- **Data With Bad Column Names**

Sometimes, we may have a data file with messy or unwanted column names, some comments about the data set, we can use `skip` argument to skip certain number of rows when read the data file to R then rename the columns with appropriate variable names. For example, if the data file has comments like

```
This data set was collected by someone in the cloud.
the column names also violates the naming convention.
ID num@3  char#
1   6   a
2   7   b
3   8   c
4   9   d
5   10  e
```

We can read the above data in R and rename the columns using the following code

```
myFakeData <- read.table("C:\\peng\\eBooks\\STA321\\fakeDat.txt", skip = 3)
names(myFakeData) = c("var1", "var2", "var3")
```

2.4 Data Values Separated By Special Symbol

If the data file has a special separator, we need to use argument `sep=` to specify the separator.

```
This data set was collected by someone in the cloud.
the column names also violates the naming convention.
ID @ num @  char
1  @  6  @  a
2  @  7  @  b
3  @  8  @  c
4  @  9  @  d
5  @ 10  @  e
```

The following code will accomplish the task.

```
myFakeData <- read.table("C:\\peng\\eBooks\\STA321\\fakeDat.txt", header = TRUE, skip =
```

2.4.1 Importing a CSV into R

If the values were separated with a `,` or `;`, we usually are working with a `.csv` file. Its contents will look similar to this:

```
Col1,Col2,Col3
1,    2,   a
4,    5,   b
7,    8,   d
```

To successfully load this file into R, we can also use the `read.table()` function in which we specify the separator character, or we can use the `read.csv()` or `read.csv2()` functions. The former function is used if the separator is a `,`, the latter if `;` is used to separate the values in your data file.

Remember that the `read.csv()` as well as the `read.csv2()` function are almost identical to the `read.table()` function, with the sole difference that they have the header and fill arguments set as TRUE by default.

2.5 Import SAS, SPSS, and Other Data Sets into R

R is a programming language and software environment for statistical computing. sometimes we need to import data from advanced statistical software programs such as SAS and SPSS. We need to install specialized packages to achieve this task. The well-known package `foreign` has several R function to read different data sets generated from different software programs.

2.5.1 Import SPSS Files into R

The R package `foreign` has a function `read.spss()` to read SPSS data set (with extension `.sav`). The code is something like

```
# Activate the `foreign` library
library(foreign)

# Read the SPSS data
mySPSSData <- read.spss("example.sav")
```

If we want the result to be displayed in a data frame, we can set the `to.data.frame` argument of the `read.spss()` function to `TRUE`. Furthermore, if we **do NOT** want the variables with value labels to be converted into R factors with corresponding levels, we should set the `use.value.labels` argument to `FALSE`:

```
# Activate the `foreign` library
library(foreign)

# Read the SPSS data
mySPSSData <- read.spss("example.sav",
                        to.data.frame=TRUE,
                        use.value.labels=FALSE)
```

Remember that factors are variables that can only contain a limited number of different values. As such, they are often called "categorical variables". The different values of factors can be labeled and are therefore often called "value labels"

2.5.2 mport Stata Files into R

To import Stata files, we use the `read.dta()` function in package `foreign` to read data into R:

```
# Activate the `foreign` library
library(foreign)

# Read Stata data into R
mydata <- read.dta("<Path to file>")
```

2.5.3 Import Systat Files into R

If you want to get Systat files into R, we also want to use the foreign package, just like shown below:

```
# Activate the `foreign` library
library(foreign)

# Read Systat data
mydata <- read.systat("<Path to file>")
```

2.5.4 Import SAS Files into R

We need to install the `sas7bdat` package and load it, and then invoke the `read.sas7bdat()` function contained within the package.

```
# Activate the `sas7bdat` library
library(sas7bdat)

# Read in the SAS data
mySASData <- read.sas7bdat("example.sas7bdat")
```

Note that we can also use the foreign library to load in SAS data in R. In such cases, we'll start from a SAS Permanent Dataset or a `SAS XPORT` Format Library with the `read.ssd()` and `read.xport()` functions, respectively. But this is relatively inconvenient.

Chapter 3

Base R Graphical Functions

This note introduces the graphical capabilities of base R graphical functions. Base R graphical system contains a set of **high-level plotting functions** such as `plot()`, `hist()`, `barplot()`, etc., and also a set of **low-level functions** that are used jointly with the high-level plotting functions such as `points()`, `lines()`, `text()`, `segments()`, etc. to make a flexible graphical system.

3.1 R Graphic Devices

R is able to output graphics to the screen or save them directly to a file (e.g. postscript, pdf, svg, png, jpeg, etc.). The different functions for producing graphical output are known as **Graphic Devices**. For example, `pdf()` would invoke the `pdf device`, while `png()` would invoke the `png device`. Type ?Devices into the R console to see a list of graphical devices that are available to R on your system.

By default, graphical output is sent to the screen. As R is cross-platform, the graphics device for producing **screen*“** graphics differs by the system. The available fonts may also differ by the system and graphical device.

3.2 Plotting with Base Graphics: `plot()`

`plot()` is the most important high-level graphic function. When using `plot()`, the output is sent to the **graphics device**. This creates a “plot region” on the graphics device. The visible area of the graphics device is also known as the “device region”. The plot region on the other hand is the area of the plot (usually bounded by a box) and does not include the plot axes, labels, or margins.

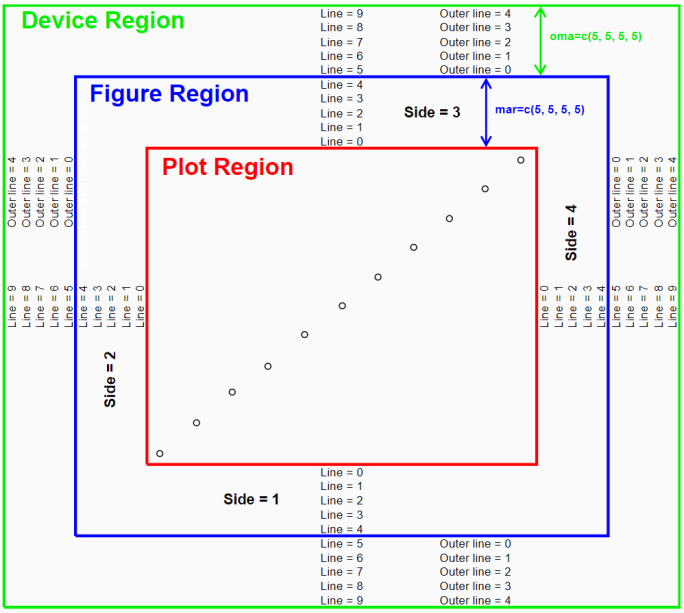
The plot region, plus the axes, labels, and margins are known as the “figure region”. Often, the device region and figure region can be the same size - but

they are not the same thing.

3.2.1 Outer and Inner Margins

`par(mar=c(x, x, x, x), oma = c(x, x, x, x))`

Almost all kinds of plots, charts, and graphs can be produced using base graphics, These plots can be fully customized using `par()` (graphical parameter). The following figure explains the graphical parameters we can use to customize the graphical layout.



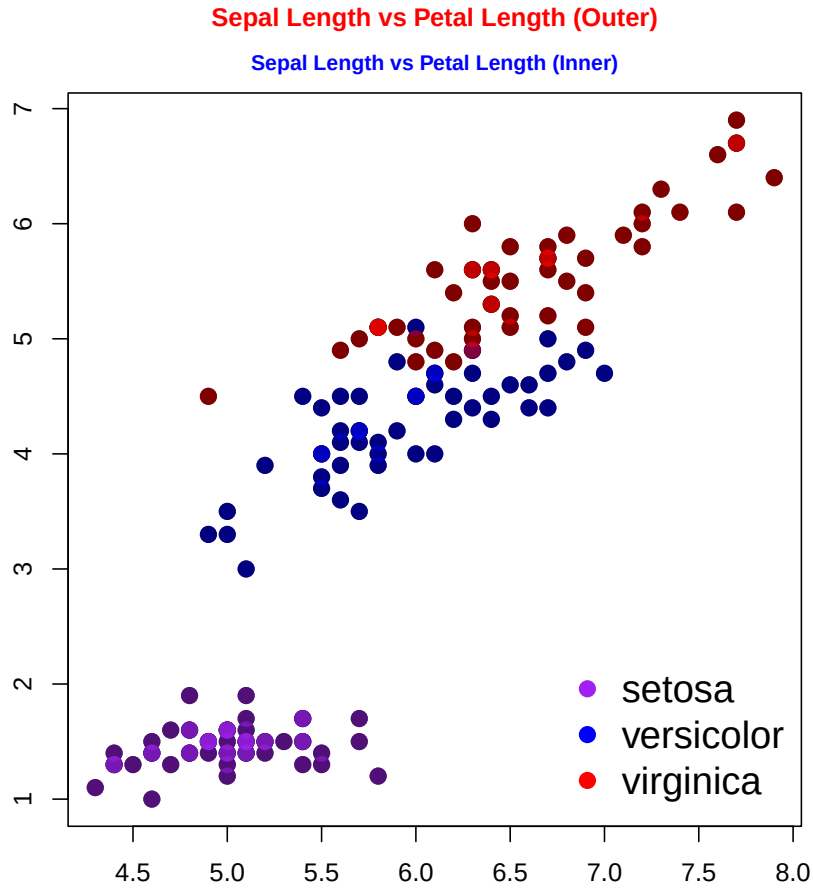
Example 1: We make a scatter plot of the two numerical variable in the well-known data set `iris`. To make nearly overlapped data points distinguishable, we are going to use R function `alpha()` in the library `scales`.

```
if (!require("scales")) {
   install.packages("scales")
   library(scales)
}
```

We read the data from the GitHub repository.

```
library(knitr)
library(scales)
iris = read.table("https://raw.githubusercontent.com/pengdsci/sta321/main/ww02/w02-iris
par(mfrow = c(1,1), oma=c(1, 1, 1, 1), mar=c(2, 2, 2, 2))
plot(iris$SepalLength, iris$PetalLength, pch = 16, cex = 1.5)
title(main="Sepal Length vs Petal Length (Outer)",
```

```
      outer=TRUE,
      col.main = "red",
      cex.main=1)
title(main="Sepal Length vs Petal Length (Inner)",
      outer=FALSE,
      col.main = "blue",
      cex.main=0.8)
## Coloring points with the transparency level
## species ID
setosa = which(iris$Classification == "Iris-setosa")
versicolor = which(iris$Classification == "Iris-versicolor")
virginica = which(iris$Classification == "Iris-virginica")
## adding points
points(iris$SepalLength[setosa], iris$PetalLength[setosa],
       pch=16,
       col = alpha("purple", 0.5),    # add a transparency level
       cex = 1.5)
points(iris$SepalLength[versicolor], iris$PetalLength[versicolor],
       pch=16,
       col = alpha("blue", 0.5),
       cex = 1.5)
points(iris$SepalLength[virginica], iris$PetalLength[virginica],
       pch=16,
       col = alpha("red", 0.5),
       cex = 1.5)
legend("bottomright", c("setosa", "versicolor", "virginica"),
       col=c("purple", "blue", "red"),
       pch=rep(16,3), cex=rep(1.5,3), bty="n")
```



3.2.2 Simple multi-panel Plots

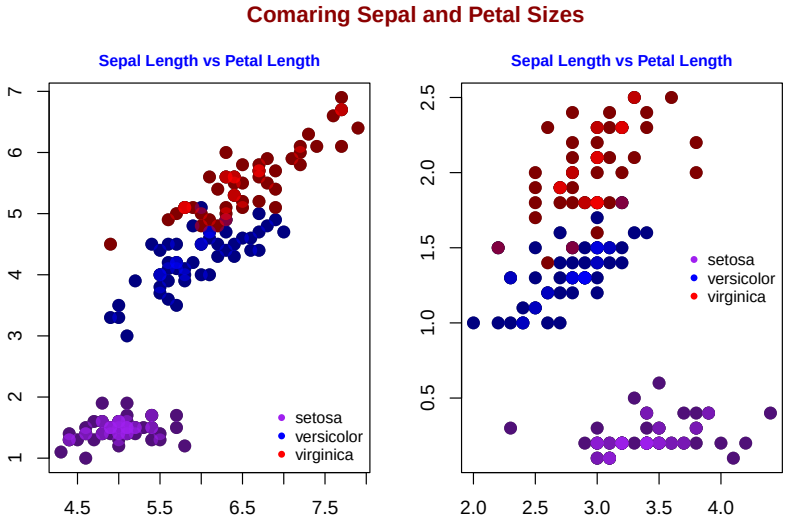
`layout()` or `par(mfrow = c())`

Example 2

```
iris = read.table("https://raw.githubusercontent.com/pengdsci/sta321/main/ww02/w02-iris
## species ID
setosa = which(iris$Classification == "Iris-setosa")
versicolor = which(iris$Classification == "Iris-versicolor")
virginica = which(iris$Classification == "Iris-virginica")
###
par(mfrow = c(1,2), oma=c(2, 2, 2, 2), mar=c(2, 2, 2, 2))
plot(iris$SepalLength, iris$PetalLength, pch = 16, cex = 1.5,
            xlab = "Sepal Width",
            ylab = "Petal Width")
title(main="Sepal Length vs Petal Length",
```

```
      outer=FALSE,
      col.main = "blue",
      cex.main=0.8)
## adding points
points(iris$SepalLength[setosa], iris$PetalLength[setosa],
       pch=16,
       col = alpha("purple", 0.5),
       cex = 1.5)
points(iris$SepalLength[versicolor], iris$PetalLength[versicolor],
       pch=16,
       col = alpha("blue", 0.5),
       cex = 1.5)
points(iris$SepalLength[virginica], iris$PetalLength[virginica],
       pch=16,
       col = alpha("red", 0.5),
       cex = 1.5)
legend("bottomright", c("setosa", "versicolor", "virginica"),
       col=c("purple", "blue", "red"),
       pch=rep(16,3), cex=0.8, bty="n")
##
plot(iris$SepalWidth, iris$PetalWidth, pch = 16, cex = 1.5,
            xlab = "Sepal Width",
            ylab = "Petal Width")
title(main="Sepal Length vs Petal Length",
      outer=FALSE,
      col.main = "blue",
      cex.main=0.8)
## adding points
points(iris$SepalWidth[setosa], iris$PetalWidth[setosa],
       pch=16,
       col = alpha("purple", 0.5),
       cex = 1.5)
points(iris$SepalWidth[versicolor], iris$PetalWidth[versicolor],
       pch=16,
       col = alpha("blue", 0.5),
       cex = 1.5)
points(iris$SepalWidth[virginica], iris$PetalWidth[virginica],
       pch=16,
       col = alpha("red", 0.5),
       cex = 1.5)
legend("right", c("setosa", "versicolor", "virginica"),
       col=c("purple", "blue", "red"),
       pch=rep(16,3), cex=0.8, bty="n")
## Overall Title
title(main="Comaring Sepal and Petal Sizes",
```

```
      outer=TRUE,
      col.main = "darkred",
      cex.main=1.2)
```



3.3 Making Our Own Colors

We can make transparent colors using R and the `rgb() command`. These colors can be useful for charts and graphics with overlapping elements.

The `rgb() command` defines a new color using numerical values (0–255) for red, green and blue. In addition, we also set an alpha value (also 0–255), which sets the transparency (0 being fully transparent and 255 being "solid").

The following example takes the standard blue and makes it transparent (~50%):

```
mycol <- rgb(0, 0, 255, max = 255, alpha = 125, names = "blue50")
```

You can also use `col2rgb()` to check the composition of an existing named R color and modify it to define your own favorite colors.

Example 3: We check the RGB code from several named R colors in the following

```
col2rgb("darkred")
```

```
##       [,1]
## red    139
## green    0
## blue     0
```

```
col2rgb("skyblue")
```

```
##       [,1]
## red    135
## green  206
## blue   235
```

```
col2rgb("gold")
```

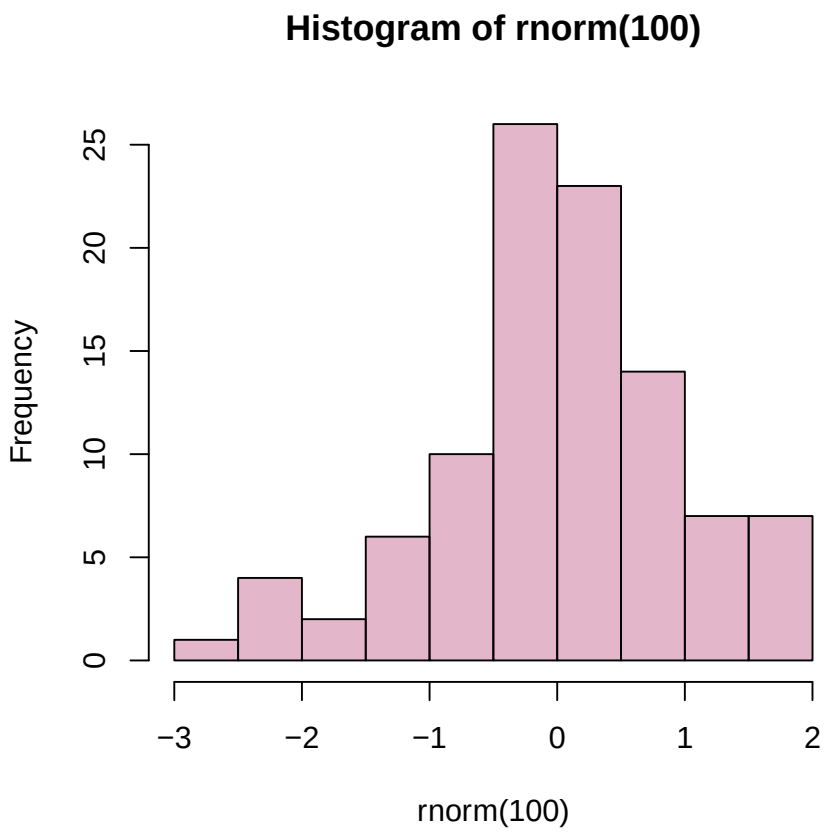
```
##       [,1]
## red    255
## green  215
## blue     0
```

```
col2rgb("purple")
```

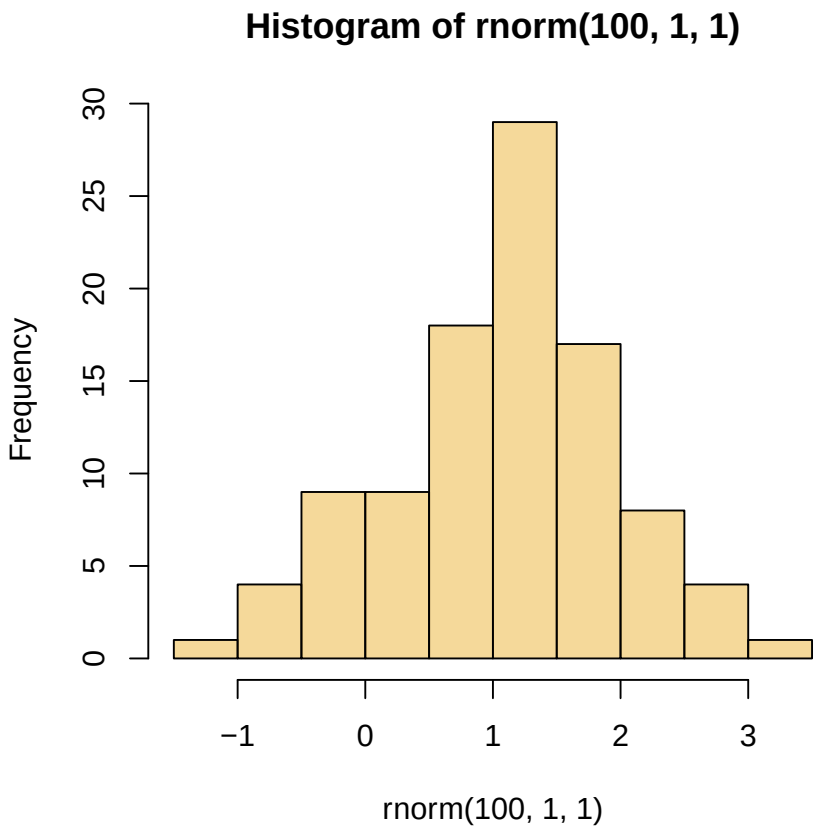
```
##       [,1]
## red    160
## green   32
## blue   240
```

Example 3 We define a color by modifying "gold" and "purple". I take the average of the RGB codes of "gold" and "purple" to see what the color looks like.

```
ratio = 0.4
avggoldpurple = round(ratio*col2rgb("gold") + (1-ratio)*col2rgb("purple"))
##
mycol.1 = rgb(avggoldpurple[1,1], avggoldpurple[2,1], avggoldpurple[3,1], max = 255, alpha = 125,
###
hist(rnorm(100),  col = mycol.1)
```




```
ratio = 0.8
avggoldpurple = round(ratio*col2rgb("gold") + (1-ratio)*col2rgb("purple"))
##
mycol.2 = rgb(avggoldpurple[1,1], avggoldpurple[2,1], avggoldpurple[3,1], max = 255, al
###
hist(rnorm(100,1,1),  col = mycol.2)
```


Histogram of rnorm(100, 1, 1)
Frequency
0
5
10
15
20
25
30
−1
0
1
2
3
rnorm(100, 1, 1)

Chapter 4

Effective Data Preparation

In this note, we use a health registry data set as an example to illustrate the use of some of the base R commands in creating an analytic data set analysis and modeling.

4.1 Data Set Description

The Current Population Survey (CPS, http://www.bls.census.gov/cps/overmain.htm) is a monthly survey of about 50,000 households conducted by the Bureau of the Census for the Bureau of Labor Statistics. The survey has been conducted for more than 50 years. The CPS is the primary source of information on the labor force characteristics of the U.S. population. The sample is scientifically selected to represent the civilian noninstitutional population. Respondents are interviewed to obtain information about the employment status of each member of the household 15 years of age and older. However, published data focus on those ages 16 and over. The sample provides estimates for the nation as a whole and serves as part of model-based estimates for individual states and other geographic areas.

Estimates obtained from the CPS include

- employment,
- unemployment,
- earnings,
- hours of work, and
- other indicators.

They are available by a variety of demographic characteristics including * age, * sex, * race, * marital status, and * educational attainment.

They are also available by

- occupation,
- industry, and
- class of worker.

Supplemental questions to produce estimates on a variety of topics including

- School enrollment,
- income,

- previous work experience,
- health,
- employee benefits, and
- work schedules

are also often added to the regular CPS questionnaire.

CPS data are used by government policymakers and legislators as important indicators of our nation's economic situation and for planning and evaluating many government programs. They are also used by the press, students, academics, and the general public.

In this note, we use a very small portion of the sample (https://raw.githubusercontent.com/pengdsci/sta321/main/ww04/cps_00003.csv) for illustrative purposes. The definitions of some of the variables can be found at (https://www.bls.gov/cps/definitions.htm). **We will not use this data to perform any meaningful analysis.**

The first few columns are what could be called administrative. They're unique identifiers for the different observations, the timing of the survey they've taken, and a bit of other information. So for now we don't need to pay much attention to MONTH, HWTFINL, CPSID, PERNUM, WTFINL, or CPSIDP. We will drop these variables.

The next few columns are concerned with the different geographies we have for the observations. This data is for individuals, but we also know the individual region (REGION), state (STATEFIP and STATECENSUS), and metropolitan area (METRO and METAREA). We can define a separate data set to store this geoinformation.

4.2 Base R Commands for Data Management

This note introduces several most commonly used R functions in data management.

```
library(knitr)
dat = read.csv("https://raw.githubusercontent.com/pengdsci/sta321/main/ww04/cps_00003.c
kable(head(dat))
```

YEAR	SERIAL	MONTH	HWTFINL	CPSID	REGION	STATEFIP	METRO	METAREA
2018	1	11	1703.832	20170800000000	32	1	2	344
2018	1	11	1703.832	20170800000000	32	1	2	344
2018	3	11	1957.313	20180900000000	32	1	2	524
2018	4	11	1687.784	20171000000000	32	1	2	524
2018	4	11	1687.784	20171000000000	32	1	2	524
2018	4	11	1687.784	20171000000000	32	1	2	524

The unique identifiers **CPSID** and **CPSIDP** are in the form of scientific notation, we need to convert them to a normal string version of the ID.

4.3 Working with Scientific Notations

Two global options we can use to print out the actual ID. See the self-explained options in the following code.

```
options(digits = 15, scipen=999)
dat = read.csv("https://raw.githubusercontent.com/pengdsci/sta321/main/ww04/cps_00003.csv")
kable(head(dat))
```

YEAR	SERIAL	MONTH	HWTFINL	CPSID	REGION	STATEFIP	METRO	METAREA
2018	1	11	1703.8321	20170800000000	32	1	2	344
2018	1	11	1703.8321	20170800000000	32	1	2	344
2018	3	11	1957.3134	20180900000000	32	1	2	524
2018	4	11	1687.7836	20171000000000	32	1	2	524
2018	4	11	1687.7836	20171000000000	32	1	2	524
2018	4	11	1687.7836	20171000000000	32	1	2	524

A new R function **pander()** in the `pander{}` library was used in the above code to produce an R markdown table. We add more features to the output table (check the help document for more information and examples).

If the ID variable was truncated before saving to CSV format, then the truncated digits will not be recovered.

```
options(digits = 7)  # change the default number digits
```

4.4 The ifelse() Function

ifelse statement is also called a vectorized conditional statement. It is commonly used in defining new variables.

```
ifelse (condition, TrueVector, FalseVector)
```

Condition	True branch	False branch
Condition is checked for every element of a vector	Select element from this if the condition is true	Select element from this if the condition is false

For example, we can define a categorical variable, denoted by **groupAge**, based on the **AGE** variable in the original data frame. If `AGE > 50, then groupAge = "(50, 150)"` otherwise `groupAge = "[16, 50]"` The following code defines this new variable.

```
dat$groupAge = ifelse(dat$AGE > 50, "(50, 150)", "[16, 50]")
kable(head(dat[, c("AGE", "groupAge")]))
```

AGE	groupAge
26	[16, 50]
26	[16, 50]
48	[16, 50]
53	(50, 150)
16	[16, 50]
16	[16, 50]

If we define another **groupAge** with more than two categories, we can still call **ifelse** multiple times. For example, we define **groupAge02** as: `is AGE > 50, then groupAge02 = "(50, 150)", if 30 <= AGE < 50, groupAge02 = [30, 50),otherwise, groupAge02 = "[16, 30)"`

```
dat$groupAge02 = ifelse(dat$AGE > 50, "(50, 150)", ifelse(dat$AGE < 30, "[16, 30)", "[
kable(head(dat[, c("AGE", "groupAge", "groupAge02")]))
```

AGE	groupAge	groupAge02
26	[16, 50]	[16, 30)
26	[16, 50]	[16, 30)
48	[16, 50]	[30, 50)
53	(50, 150)	(50, 150)
16	[16, 50]	[16, 30)
16	[16, 50]	[16, 30)

Remark: **ifelse()** is particularly useful when you want to combine categories of existing categorical variables.

4.5 The cut() Function

The **cut()** function is more flexible than **ifelse()**.

`Syntax`

```
cut(num_vector,              # Numeric input vector
    breaks,                  # Number or vector of breaks
    labels = NULL,           # Labels for each group
    include.lowest = FALSE,  # Whether to include the lowest 'break' or not
    right = TRUE,            # Whether the right interval is closed (and the left open) or vice v
    dig.lab = 3,             # Number of digits of the groups if labels = NULL
    ordered_result = FALSE,  # Whether to order the factor result or not
    …)                       # Additional arguments
```

We still use the above example to discretize the age variable using **cut()** function.

```
dat$cutAge01 = cut(dat$AGE, breaks =c(16, 30, 50, 150), labels=c( "[16, 30)", "[30, 50)", "(50, 1
kable(head(dat[, c("AGE", "groupAge", "groupAge02", "cutAge01")]))
```

AGE	groupAge	groupAge02	cutAge01
26	[16, 50]	[16, 30)	[16, 30)
26	[16, 50]	[16, 30)	[16, 30)
48	[16, 50]	[30, 50)	[30, 50)
53	(50, 150)	(50, 150)	(50, 150)
16	[16, 50]	[16, 30)	[16, 30)
16	[16, 50]	[16, 30)	[16, 30)

```
dat$cutAge02 = cut(dat$AGE, breaks =c(16, 30, 50, 150), include.lowest = TRUE)
kable(head(dat[, c("AGE", "groupAge", "groupAge02", "cutAge01", "cutAge02")]))
```

AGE	groupAge	groupAge02	cutAge01	cutAge02
26	[16, 50]	[16, 30)	[16, 30)	[16,30]
26	[16, 50]	[16, 30)	[16, 30)	[16,30]
48	[16, 50]	[30, 50)	[30, 50)	(30,50]
53	(50, 150)	(50, 150)	(50, 150)	(50,150]
16	[16, 50]	[16, 30)	[16, 30)	[16,30]
16	[16, 50]	[16, 30)	[16, 30)	[16,30]

4.6 with() and within() Functions

with() and **within()** are two closely related yet different base R functions that are useful in data management.

4.6.1 The with() Function

`with()` function enables us to define a new variable based on the variables in a **data frame** using basic **R expressions** that include mathematical and logical operations. We can add the newly defined variables to the existing data frame as usual.

with() Syntax

```
with(data-frame, R-expression)
```

Example 1

```
Num <- c(1400,1200,1100,1700,1500)
Cost <- c(1200,1300,1400,1500,1600)
##
dataA <- data.frame(Num,Cost,stringsAsFactors = FALSE)
##
product <- with(dataA, Num*Cost)
quotient <- with(dataA, Cost/Num)
logical <- with(dataA, Num > Cost)
kable(cbind(product = product, quotient = quotient, logical = logical))
```

product	quotient	logical
1680000	0.8571429	1
1560000	1.0833333	0
1540000	1.2727273	0
2550000	0.8823529	1
2400000	1.0666667	0

```
## add the new variables to data frame dataA
dataA$product = product
dataA$quotient = quotient
dataA$logical = logical
##
kable(dataA)
```

Num	Cost	product	quotient	logical
1400	1200	1680000	0.8571429	TRUE
1200	1300	1560000	1.0833333	FALSE
1100	1400	1540000	1.2727273	FALSE
1700	1500	2550000	0.8823529	TRUE
1500	1600	2400000	1.0666667	FALSE

The **within()** Function

within() function allows us to create a copy of the data frame and add a column that would eventually store the result of the R expression.

```
Num <- c(1400,1200,1100,1700,1500)
Cost <- c(1200,1300,1400,1500,1600)
##
dataA <- data.frame(Num,Cost,stringsAsFactors = FALSE)
##
dataB <- within(dataA, Product <- Num*Cost)    # defined Product and added to dataA sim
dataC <- within(dataB, Quotient <- Cost/Num)
dataD <- within(dataC, Logical <- Num > Cost)
kable(dataD)
```

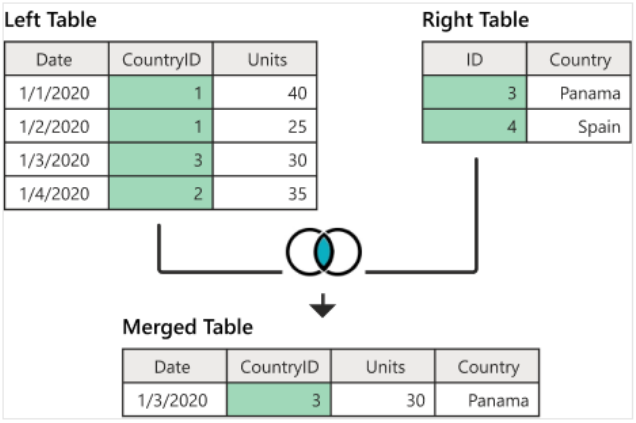
Num	Cost	Product	Quotient	Logical
1400	1200	1680000	0.8571429	TRUE
1200	1300	1560000	1.0833333	FALSE
1100	1400	1540000	1.2727273	FALSE
1700	1500	2550000	0.8823529	TRUE
1500	1600	2400000	1.0666667	FALSE

4.7 The merge() Function - Table Joins

The R **merge()** function allows merging two data frames by **row names** (common key). This function allows us to perform different database (SQL) joins, like left join, inner join, right join, or full join, among others. In this note, we only introduce four different ways of merging datasets in base R with examples. We will introduce the SQL clause in R later.

4.7.1 Inner Join

The following figure illustrates how A **left joins** B and the resulting merged data set.



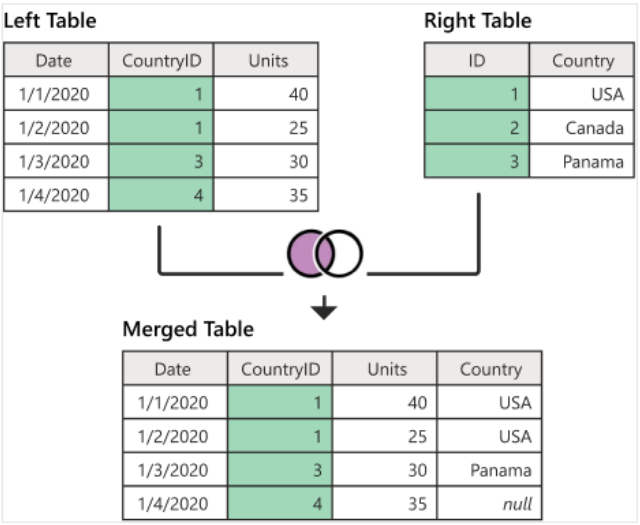
The following code implements the above left-join.

```
A = data.frame(Date = c("1/1/2020", "1/2/2020", "1/3/2020", "1/4/2020"),
               CountryID = c(1,1,3,2),
               Units = c(40, 25, 30, 35))
B = data.frame(ID=c(3,4),
               Country=c( "Panama", "Spain"))
AinnerB = merge(A, B, by.x = "CountryID", by.y = "ID")
kable(AinnerB)
```

CountryID	Date	Units	Country
3	1/3/2020	30	Panama

4.7.2 Left Join

The following figure illustrates how A **left joins** B and the resulting merged data set.



The following code implements the above left-join.

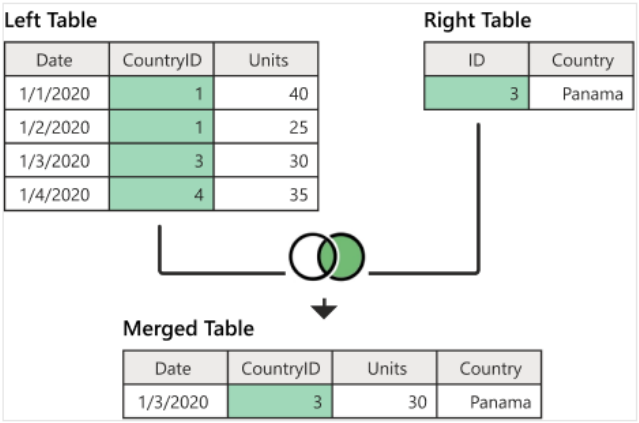
```
A = data.frame(Date = c("1/1/2020", "1/2/2020", "1/3/2020", "1/4/2020"),
               CountryID = c(1,1,3,4),
               Units = c(40, 25, 30, 35))
B = data.frame(ID=c(1,2,3),
               Country=c("USA", "Canada", "Panama"))
AleftB = merge(A, B, by.x = "CountryID", by.y = "ID", all.x = TRUE)
kable(AleftB)
```

CountryID	Date	Units	Country
1	1/1/2020	40	USA
1	1/2/2020	25	USA
3	1/3/2020	30	Panama
4	1/4/2020	35	NA

Note that, left-join produces missing values of the record in A and does not have any information in B.

4.7.3 Right Join

The following figure illustrates how A **right joins** B and the resulting merged data set.



The following code implements the above left-join.

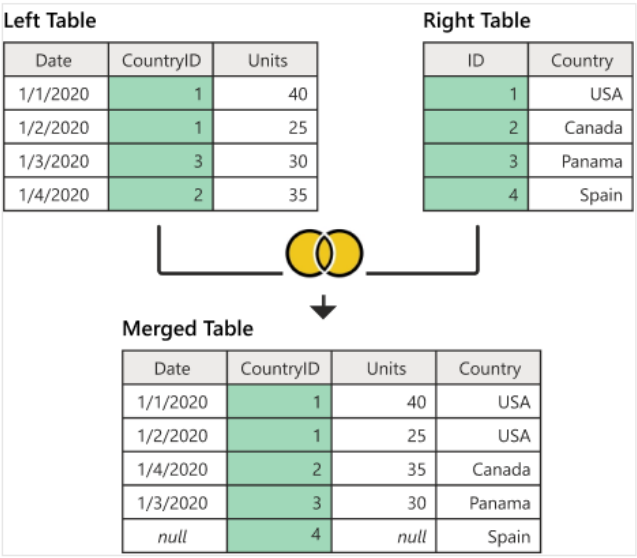
```
A = data.frame(Date = c("1/1/2020", "1/2/2020", "1/3/2020", "1/4/2020"),
               CountryID = c(1,1,3,4),
               Units = c(40, 25, 30, 35))
B = data.frame(ID=c(3),
               Country=c("Panama"))
ArightB = merge(A, B, by.x = "CountryID", by.y = "ID", all.y = TRUE)
kable(ArightB)
```

CountryID	Date	Units	Country
3	1/3/2020	30	Panama

Note also that right-join could also produce missing values.

4.7.4 Full (outer) Join

The following figure illustrates how A **Full outer joins** B and the resulting merged data set.



Date	CountryID	Units
1/1/2020	1	40
1/2/2020	1	25
1/3/2020	3	30
1/4/2020	2	35

ID	Country
1	USA
2	Canada
3	Panama
4	Spain

Date	CountryID	Units	Country
1/1/2020	1	40	USA
1/2/2020	1	25	USA
1/4/2020	2	35	Canada
1/3/2020	3	30	Panama
null	4	*null*	Spain

The following code implements the above left-join.

```
A = data.frame(Date = c("1/1/2020", "1/2/2020", "1/3/2020", "1/4/2020"),
               CountryID = c(1,1,3,2),
               Units = c(40, 25, 30, 35))
B = data.frame(ID=c(1,2,3,4),
               Country=c("USA", "Canada", "Panama", "Spain"))
AfullB = merge(A, B, by.x = "CountryID", by.y = "ID", all = TRUE)
kable(AfullB)
```

CountryID	Date	Units	Country
1	1/1/2020	40	USA
1	1/2/2020	25	USA
2	1/4/2020	35	Canada
3	1/3/2020	30	Panama
4	NA	NA	Spain

4.8 Subsetting Data Frame

There are two different ways for subsetting a data frame: subsetting by rows and by columns.

We first define the following working data set.

```
working.data <- data.frame(
  id = c(10,11,12,13,14,15,16,17),
```

```
  name = c('sai','ram','deepika','sahithi','kumar','scott','Don','Lin'),
  gender = c('M','M',NA,'F','M','M','M','F'),
  dob = as.Date(c('1990-10-02','1981-3-24','1987-6-14','1985-8-16',
                  '1995-03-02','1991-6-21','1986-3-24','1990-8-26')),
  state = c('CA','NY',NA,NA,'DC','DW','AZ','PH'),
  row.names=c('r1','r2','r3','r4','r5','r6','r7','r8')
)
kable(working.data)
```

	id	name	gender	dob	state
r1	10	sai	M	1990-10-02	CA
r2	11	ram	M	1981-03-24	NY
r3	12	deepika	NA	1987-06-14	NA
r4	13	sahithi	F	1985-08-16	NA
r5	14	kumar	M	1995-03-02	DC
r6	15	scott	M	1991-06-21	DW
r7	16	Don	M	1986-03-24	AZ
r8	17	Lin	F	1990-08-26	PH

4.8.1 Subsetting by Columns

This is a relatively easy job - we can simply select or drop variables to make a subset. The following is just an example.

```
# only keep id, name, dob
subset01 = working.data[, c("id", "name", "dob")]
subset01
```

```
##    id    name        dob
## r1 10     sai 1990-10-02
## r2 11     ram 1981-03-24
## r3 12 deepika 1987-06-14
## r4 13 sahithi 1985-08-16
## r5 14   kumar 1995-03-02
## r6 15   scott 1991-06-21
## r7 16     Don 1986-03-24
## r8 17     Lin 1990-08-26
```

We could also create the above subset by dropping `gender` and `state`.

```
# drop gender, state
subset02 = working.data[, -c(3,5)]
subset02
```

```
##    id    name        dob
## r1 10     sai 1990-10-02
## r2 11     ram 1981-03-24
## r3 12 deepika 1987-06-14
```

```
## r4 13 sahithi 1985-08-16
## r5 14   kumar 1995-03-02
## r6 15   scott 1991-06-21
## r7 16     Don 1986-03-24
## r8 17     Lin 1990-08-26
```

4.8.2 Subsetting by Rows

```
# subset by row name
kable(subset(working.data, subset=rownames(df) == 'r1'))
```

id	name	gender	dob	state

```
# subset row by the vector of row names
kable(subset(working.data, rownames(df) %in% c('r1','r2','r3')))
```

id	name	gender	dob	state

```
# subset by condition
kable(subset(working.data, gender == 'M'))
```

	id	name	gender	dob	state
r1	10	sai	M	1990-10-02	CA
r2	11	ram	M	1981-03-24	NY
r5	14	kumar	M	1995-03-02	DC
r6	15	scott	M	1991-06-21	DW
r7	16	Don	M	1986-03-24	AZ

```
# subset by condition with %in%
kable(subset(working.data, state %in% c('CA','DC')))
```

	id	name	gender	dob	state
r1	10	sai	M	1990-10-02	CA
r5	14	kumar	M	1995-03-02	DC

```
# subset by multiple conditions using |
kable(subset(working.data, gender == 'M' | state == 'PH'))
```

	id	name	gender	dob	state
r1	10	sai	M	1990-10-02	CA
r2	11	ram	M	1981-03-24	NY
r5	14	kumar	M	1995-03-02	DC
r6	15	scott	M	1991-06-21	DW
r7	16	Don	M	1986-03-24	AZ
r8	17	Lin	F	1990-08-26	PH

Chapter 5

Bootstrap Confidence Intervals

The bootstrap method is a data-based simulation method for statistical inference. The method assumes that

- The sample is a random sample that represents the population.
- The sample size is large enough such that the empirical distribution is close to the true distribution.

5.1 Basic Idea of Bootstrap Method.

The objective is to estimate a population parameter such as mean, variance, correlation coefficient, regression coefficients, etc. from a random sample without assuming any probability distribution of the underlying distribution of the population.

For convenience, we assume that the population of interest has a cumulative distribution function $F(x : \theta)$, where θ is a vector of the population. For example, You can think about the following distributions

- **Normal distribution**: $N(\mu, \sigma^2)$, the distribution function is given by

$$f(x : \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

where $\theta = (\mu, \sigma)$. Since the normal distribution is so fundamental in statistics, we use the special notation for the cumulative distribution $\phi_{\mu,\sigma^2}(x)$ or simply $\phi(x)$. The corresponding probability function

- **Binomial distribution**: $Binom(n, p)$, the probability distribution is given by

$$P(x) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}, x = 0, 1, 2, \cdots, n-1, n.$$

where $\theta = p$. *Caution*: n is NOT a parameter!

We have already learned how to make inferences about population means and variances under various assumptions in elementary statistics. In this note, we introduce a **new approach** to making inferences only based on a given random sample taken from the underlying population.

As an example, we focus on the population mean. For other parameters, we can follow the same idea to make bootstrap inferences.

5.1.1 Random Sample from Population

We have introduced various study designs and sampling plans to obtain random samples from a given population with the distribution function $F(x : \theta)$. Let μ be the population means.

- **Random Sample**. Let

$$\{x_1, x_2, \cdots, x_n\} \to F(x : \theta)$$

  be a random sample from population $F(x : \theta)$.

- **Sample Mean**. The point estimate is given by

$$\widehat{\mu} = \frac{\sum_{i=1}^{n} x_i}{n}$$

- **Sampling Distribution of $\widehat{\mu}$**. In order to construct the confidence interval of μ or make hypothesis testing about μ, we need to know the sampling distribution of $\widehat{\mu}$. From elementary statistics, we have the following results.
  - $\widehat{\mu}$ is normally distributed if (1). n is large; or (2). the population is normal and population variance is known.
  - the standardized $\widehat{\mu}$ follows a t-distribution if the population is normal and population variance is unknown.
  - $\widehat{\mu}$ is **unknown** of the population is not normal and the sample size is not large enough.

- In the last case of the previous bullet point, we don't have the theory to derive the sampling distribution based on a **single** sample. However, if the sampling is not too expensive and time-consuming, we take following the sample study design and sampling plan to repeatedly take a large number, 1000, samples of the same size from the population. We calculate

the mean of each of the 1000 samples and obtain 1000 sample means $\{\hat{\mu}_1, \hat{\mu}_2, \cdots, \hat{\mu}_{1000}\}$. Then the empirical distribution of $\hat{\mu}$.

The following figure depicts how to approximate the sampling distribution of the point estimator of the population parameter.

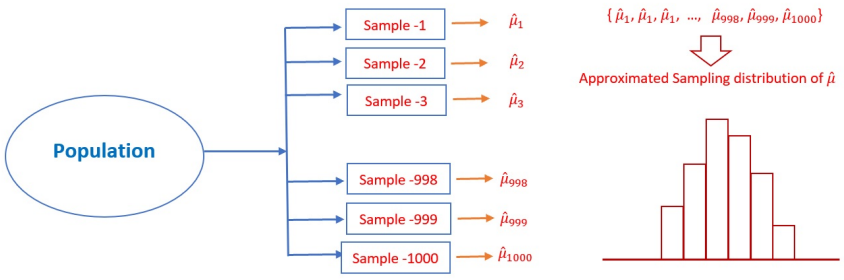


Figure 5.1: Steps for estimating the sampling distribution of a point estimator of the population parameter

Example 1: [**Simulated data**] Assume that the particular numeric characteristics of the WCU student population are the heights of all students.

- We don't know the distribution of the heights.
- We also don't know *whether a specific sample size is large enough to use the central limit theorem.* This means we don't know whether it is appropriate to use the central limit theorem to characterize the sampling distribution of the mean height.

Due to the above constraints, we cannot find the sampling distribution of the sample means using only the knowledge of elementary statistics. However, if sampling is not expensive, we take repeated samples with the same sample size. The resulting sample means can be used to approximate the sampling distribution of the sample mean.

Next, we use R and the simulated data set https://raw.githubusercontent.com/pengdsci/sta321/main/ww02/w02-wcuheights.txt to implement the above idea. I will use simple code with comments to explain the task of each line of code so you can easily understand the coding logic.

```
library(knitr)
# read the delimited data from URL
heightURL = "https://raw.githubusercontent.com/pengdsci/sta321/main/ww02/w02-wcuheights.txt"
wcu.height = read.table(heightURL, header = TRUE)
# define an empty vector to hold sample means of repeated samples.
sample.mean.vec = NULL
for(i in 1:1000){   # taking repeated random samples with n = 81
  ith.sample = sample( wcu.height$Height,       # population s
```

```
                        81,                         # boot sample size = 81
                        replace = FALSE             # without replacement
                    )
# calculate the mean of i-th sample and save it into sample.mean.vec
   sample.mean.vec[i] = mean(ith.sample)
 }
```

Next, we make a histogram of the sample means saved in **sample.mean.vec**.

```
hist(sample.mean.vec,    # data used for histogram
     breaks = 14,        # specify number of vertical bars
     xlab = "sample means of repeated samples", # change the label of x-axis
     # add multi-line title to the histogram
     main="Approximated Sampling Distribution  \n  of Sample Means")
```

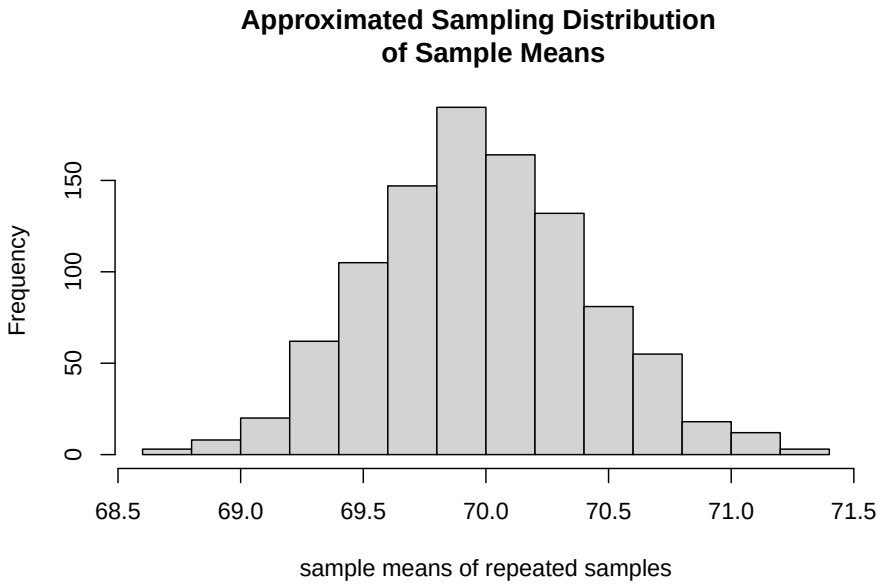


Figure 5.2: Approximated sampling distribution of sample mean used the repeated samples.

5.1.2 Bootstrap Sampling and Bootstrap Sampling Distribution

Recall the situation in **Example 1** in which we were not able to use the normality assumption of the population and the central limit theorem (CLT) but were allowed to take repeated samples from the population. In practice, taking samples from the population can be very expensive. Is there any way to estimate

the sampling distribution of the **sample means**? The answer is YES under the assumption the sample yields a valid estimation of the original population distribution.

- **Bootstrap Sampling** with the assumption that the sample yields a good approximation of the population distribution, we can take bootstrap samples from the **actual** sample. Let

$$\{x_1, x_2, \cdots, x_n\} \to F(x : \theta)$$

  be the actual random sample taken from the population. A **bootstrap sample** is obtained by taking a sample **with replacement** from the original data set (not the population!) with the same size as the original sample. Because **with replacement** was used, some values in the bootstrap sample appear once, some twice, and so on, and some do not appear at all.

- **Notation of Bootstrap Sample**. We use $\{x_1^{(i*)}, x_2^{(i*)}, \cdots, x_n^{(i*)}\}$ to denote the i^{th} bootstrap sample. Then the corresponding mean is called bootstrap sample mean and denoted by $\hat{\mu}_i^*$, for $i = 1, 2, ..., n$.

- **Bootstrap sampling distribution** of the sample mean can be estimated by taking a large number, say B, of bootstrap samples. The resulting B bootstrap sample means are used to estimate the sampling distribution. Note that, in practice, B is bigger than 1000.

The above Bootstrap sampling process is illustrated in the following figure.

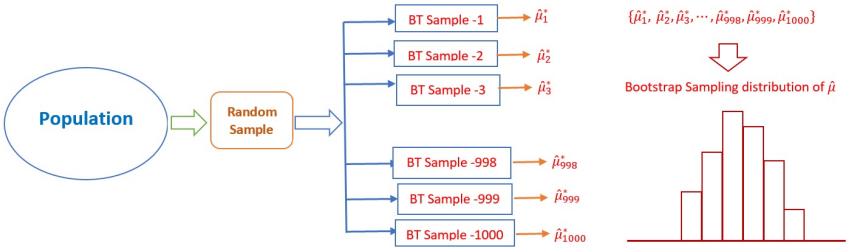


Figure 5.3: Steps for the Bootstrap sampling distribution of a point estimator of the population parameter

- **Example 2:** [**continue to use WCU Heights**]. We use the Bootstrap method to estimate the sampling distribution of the sample means.

```
### read the delimited data from URL
heightURL = "https://raw.githubusercontent.com/pengdsci/sta321/main/ww02/w02-wcuheights.txt"
wcu.height = read.table(heightURL, header = TRUE)
# taking the original random sample from the population
original.sample = sample( wcu.height$Height,    # population
```

```
                        81,                          # boot sample size = 81
                        replace = FALSE              # without replacement
                    )
### Bootstrap sampling begins
bt.sample.mean.vec = NULL       # empty vector to store boot sample means.
for(i in 1:1000){
  ith.bt.sample = sample( original.sample,     # Original sample
                        81,                    # boot sample size = 81!!
                        replace = TRUE         # MUST use WITH REPLACEMENT!!
                    )
  bt.sample.mean.vec[i] = mean(ith.bt.sample)
  }
```

The following histogram shows the bootstrap sampling distribution of sample means with size $n = 81$.

```
hist(bt.sample.mean.vec,  # data used for histogram
     breaks = 14,         # specify number of vertical bars
     xlab = "Bootstrap sample means",  # change the label of x-axis
     main="Bootstrap Sampling Distribution \n of Sample Means")
```

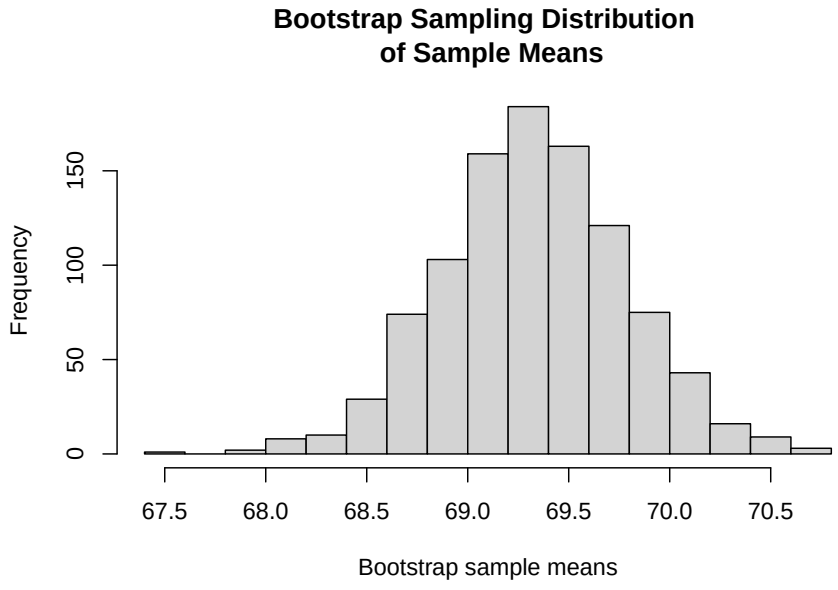


Figure 5.4: Bootstrap sampling distribution of sample means

5.1.3 Relationship between Two Estimated Sampling Distributions

We can see that the two sampling distributions are slightly different. If we are allowed to take repeated samples from the population, we should always use the repeated sample approach since it yields a better estimate of the true sampling distribution.

The bootstrap estimate of the sampling distribution is used when no theoretical confidence intervals are available and the repeated sample is not possible due to certain constraints. This does not mean that the bootstrap methods do not have limitations. In fact, the implicit assumption of the bootstrap method is that **the original sample has enough information to estimate the true population distribution**.

5.2 Bootstrap Confidence Intervals

First of all, all bootstrap confidence intervals are constructed based on the bootstrap sampling distribution of the underlying point estimator of the parameter of interest.

There are at least five different bootstrap confidence intervals. You can find these definitions from Chapter 4 of Roff's eBook https://ebookcentral.proquest.com/lib/wcupa/reader.action?docID=261114&ppg=7 (need WCU login credential to access the book). We only focus on the percentile method in which we simply define the confidence limit(s) by using the corresponding percentile(s) of the bootstrap estimates of the parameter of interest. R has a built-in function, **quantile()**, to find percentiles.

- **Example 3**: We construct a 95% two-sided bootstrap percentile confidence interval of the mean height of WCU students. This is equivalent to finding the 2.5% and 97.5% percentiles. We use the following one-line code.

```
CI = quantile(bt.sample.mean.vec, c(0.025, 0.975))
CI
```

```
##     2.5%    97.5%
## 68.43179 70.22253
```

```
#kable(CI, caption = "95% bootstrap percentile confidence interval of the mean height")
```

Various bootstrap methods were implemented in the R library **{boot}**. UCLA Statistical Consulting https://stats.idre.ucla.edu/r/faq/how-can-i-generate-bootstrap-statistics-in-r/ has a nice tutorial on bootstrap confidence intervals. You can use the built-in function **boot.ci()** to find all 5 bootstrap confidence intervals after you create the boot object. I will leave it to you if you want to explore more about the library.

5.3 Bootstrap Confidence Interval of Correlation Coefficient

As a case study, we will illustrate one bootstrap method to sample a random sample with multiple variables and use the bootstrap samples to calculate the corresponding bootstrap correlation coefficient. The bootstrap percentile confidence interval of the correlation coefficient.

5.3.1 Bootstrapping Data Set

There are different ways to take bootstrap samples. **The key point is that we cannot sample individual variables in the data frame separately to avoid mismatching!** The method we introduce here is also called bootstrap sampling cases. Here are the basic steps:

- Assume the data frame haven rows. We define the vector of row ID. That is, $ID = \{1, 2, 3, ..., n\}$.
- Take a bootstrap sample from ID (i.e., sampling with replacement) with same size = n, denoted by ID^*. As commented earlier, there will be replicates of ID^* and some values in ID are not in ID^*.
- Use ID^* to select the corresponding rows to form a bootstrap sample and then perform bootstrap analysis.

Here is an example of taking the bootstrap sample from the original sample with multiple variables. The data set we use here is well-known and is available at https://raw.githubusercontent.com/pengdsci/sta321/main/ww02/w02-iris.txt

```
# read data into R from the URL
irisURL = "https://raw.githubusercontent.com/pengdsci/sta321/main/ww02/w02-iris.txt"
iris = read.table(irisURL, header = TRUE)
n = dim(iris)[1]   # returns the dimension of the data frame: [row, col]
bt.ID = sample(1:n, replace = TRUE)   # bootstrap IDs, MUST use replacement method!
sort(bt.ID)      # sort bt.ID. to see replicated ID
```

```
##   [1]   2   5   5   5   6   6   6   7   7   8   8   8   9  10  11  11  12  12
##  [19]  13  14  16  19  20  20  20  22  23  23  23  23  23  24  24  24  25  25
##  [37]  25  27  27  28  30  31  32  32  36  40  40  44  47  48  49  53  54  55
##  [55]  55  56  58  58  59  60  60  63  63  64  65  67  68  68  68  69  71  71
##  [73]  72  73  74  74  79  80  81  82  82  82  82  82  83  83  86  87  89  91
##  [91]  91  94  97  98  98  99 100 100 102 103 103 103 104 104 104 105 106 107
## [109] 107 108 109 110 112 112 113 114 114 118 119 119 119 120 121 122 123 123
## [127] 124 125 126 126 127 127 128 128 131 131 132 133 134 134 134 137 138 140
## [145] 141 143 144 146 147 148
```

Next, we use the above bt.ID to take the bootstrap sample from the original data set **iris**.

```
bt.iris = iris[bt.ID,]    # taking bootstrap cases (or rows, records) using the bt.ID
bt.iris                   # display the bootstrap sample
```

5.3.2 Confidence Interval of Coefficient Correlation

In this section, we construct a 95% bootstrap percentile confidence interval for the coefficient correlation between the SepalLength and SepalWidth given in **iris**. Note that R built-in function **cor(x,y)** can be used to calculate the bootstrap correlation coefficient directly. The R code for constructing a bootstrap confidence interval for the coefficient correlation is given below.

```
irisURL = "https://raw.githubusercontent.com/pengdsci/sta321/main/ww02/w02-iris.txt"
iris = read.table(irisURL, header = TRUE)
n = dim(iris)[1]    # 1st component is the number of rows
##
bt.cor.vec = NULL    # empty vector for boot correlation coefficients
for (i in 1:5000){   # taking 5000 bootstrap samples for this example.
  bt.ID.i = sample(1:n, replace = TRUE)  # MUST use replacement method!
  bt.iris.i = iris[bt.ID.i, ]            # i-th bootstrap ID
  # i-th bootstrap correlation coefficient
  bt.cor.vec[i] = cor(bt.iris.i$SepalLength, bt.iris.i$SepalWidth)
}
bt.CI = quantile(bt.cor.vec, c(0.025, 0.975) )
bt.CI
```

```
##        2.5%       97.5%
## -0.25370320  0.03578238
```

Interpretation: we are 95% confident that there is no statistically significant correlation between sepal length and sepal width based on the given sample. This may be because the data set contains three different types of iris.

Next, we make two plots to visualize the relationship between the two variables.

```
par(mfrow=c(1,2))   # layout a plot sheet: 1 row and 2 columns
## histogram
hist(bt.cor.vec, breaks = 14,
       main="Bootstrap Sampling \n  Distribution of Correlation",
      xlab = "Bootstrap Correlation Coefficient")
## scatter plot
plot(iris$SepalLength, iris$SepalWidth,
     main = "Sepal Length vs Width",
     xlab = "Sepal Length",
     ylab = "Sepal Width")
```

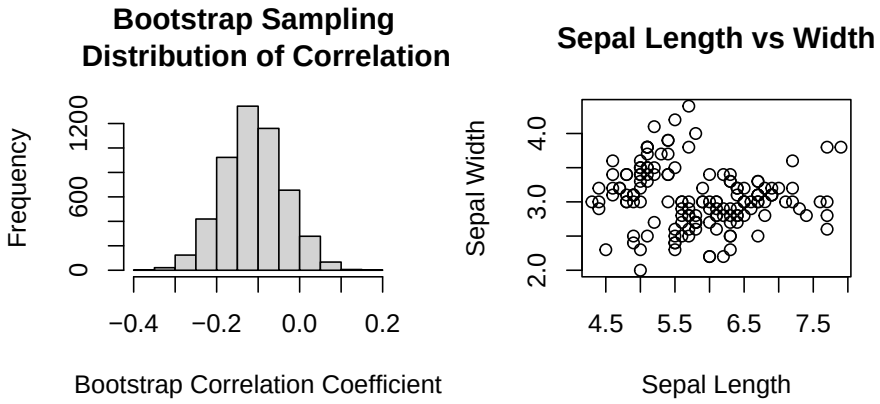


Figure 5.5: Left panel: histogram of the bootstrap coefficient of correlation. Right panel: the scatter plot of the sepal length and width.

5.4 Problem Set

5.4.1 Data Set Description

This data set includes the **percentage of protein intake** from different types of food in countries around the world. The last couple of columns also includes counts of obesity and COVID-19 cases as percentages of the total population for comparison purposes.

Data can be found at kaggle.com. I also upload this data set on the course web page STA321 Course Web Page.

```
proteinURL = "https://raw.githubusercontent.com/pengdsci/sta321/main/ww02/w02-Protein_S
protein = read.csv(proteinURL, header = TRUE)
var.name = names(protein)
kable(data.frame(var.name))
```

var.name
Country
AlcoholicBeverages
AnimalProducts
Animalfats
CerealsExcludingBeer
Eggs
FishSeafood
FruitsExcludingWine
Meat
MilkExcludingButter
Offals
Oilcrops
Pulses
Spices
StarchyRoots
Stimulants
Treenuts
VegetalProducts
VegetableOils
Vegetables
Miscellaneous
Obesity
Confirmed
Deaths
Recovered
Active
Population

5.4.2 Problems

Choose a numerical variable from the variable list and do the following problems.

1. Construct a confidence interval of the mean of the variable using methods in elementary statistics.
2. Use the Bootstrap method to construct the bootstrap confidence interval for the mean of the selected variable.
3. Plot the bootstrap sampling distribution of the sample mean.
4. Compare the two confidence intervals and interpret your findings and interpret the confidnce intervals.

Chapter 6

Bootstrapping SLR

In this note, we introduce how to use the bootstrap method to make inferences about the regression coefficients. Parametric inference of the regression models heavily depends on the assumptions of the underlying model. If there are violations of the model assumptions the resulting p-values produced in the outputs of software programs could be valid. In this case, if the sample size is large enough, we can use the Bootstrap method to make the inferences without imposing the distributional assumption of the response variable (hence the residuals).

6.1 Data Set and Practical Questions

The data set we use in this note is taken from the well-known Kaggle, a web-based online community of data scientists and machine learning practitioners, that allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges (– Wikipedia).

6.1.1 Data Description

The data in this note was found from Kaggle. I renamed the original variables and modified the sales dates to define the sales year indicator. The modified data set was uploaded to the course web page at https://raw.githubuserconten t.com/pengdsci/sta321/main/ww03/w03-Realestate.csv.

- ObsID
- TransactionYear(X_1): transaction date

- HouseAge(X_2): house age

- Distance2MRT(X_3): distance to the nearest MRT station
- NumConvenStores(X_4): number of convenience stores
- Latitude(X_5): latitude
- Longitude(X_6): longitude
- PriceUnitArea(Y): house price of unit area

6.1.2 Practical Question

The primary practical question is to see how the variables in the data set or derived variables from the data set affect the price of the unit area. Since we focus on the simple linear regression model in this module. We will pick one of the variables to perform the simple linear regression model.

6.1.3 Exploratory Data Analysis

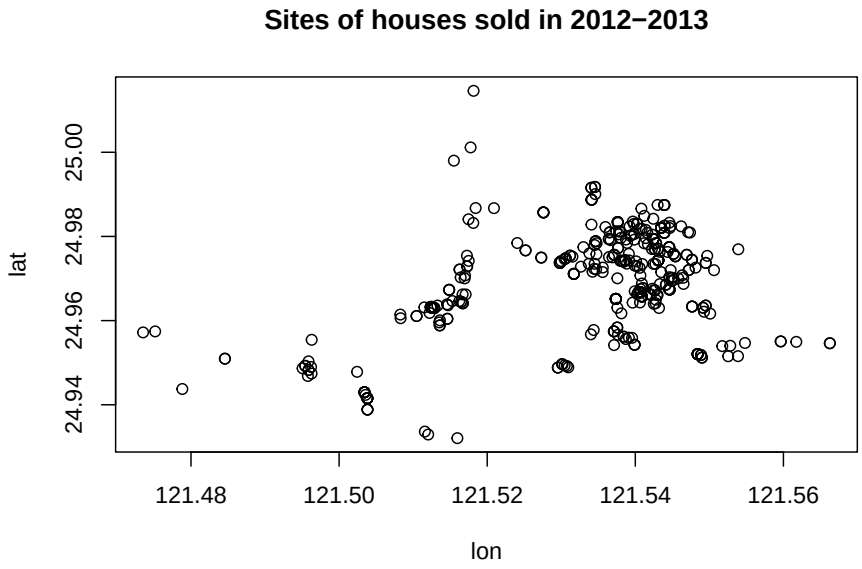
We first explore the pairwise association between the variables in the data set. Since longitude and latitude are included in the data set, we first make a map to see if we can define a variable according to the sales based on the geographic regions.

To start, we load the data to R.

```
library(knitr)
library(scales)
houseURL = "https://raw.githubusercontent.com/pengdsci/sta321/main/ww03/w03-Realestate
realestate <- read.csv(houseURL, header = TRUE)
realestate <- realestate[, -1]
# longitude and latitude will be used to make a map in the upcoming analysis.
lon <- realestate$Longitude
lat <- realestate$Latitude
```

Next, we make a scatter plot of longitude and latitude to see the potential to create a geographic variable. The source does not mention the geographical locations where this data was collected. In the upcoming analysis, we will draw maps and will see the site of the houses sold.

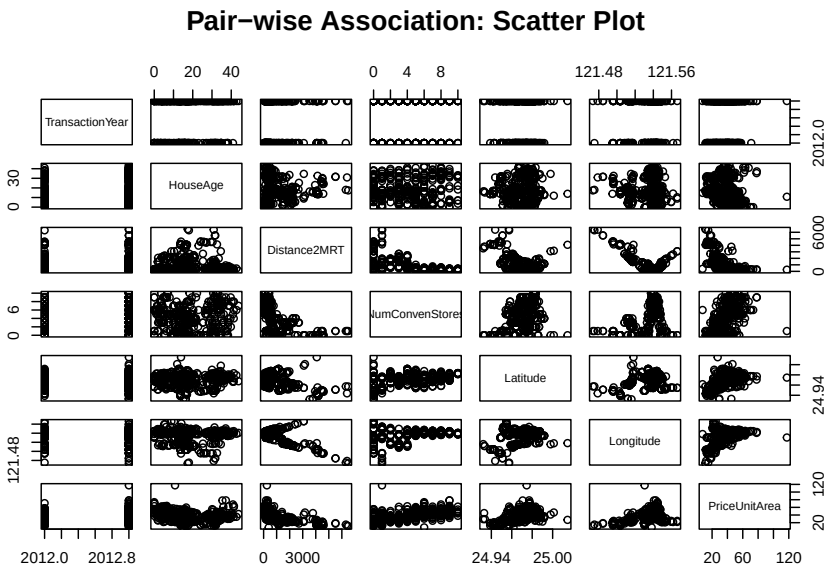
```
plot(lon, lat, main = "Sites of houses sold in 2012-2013")
```



We can see that there are clusters in the plot. We can use this information to define a cluster variable to capture the potential differences between the geographic clusters in terms of the sale prices.

We now make a simple pairwise plot to show the association between variables and pick one as an example for the simple linear regression model.

```
pairs(realestate, main ="Pair-wise Association: Scatter Plot")
```



The above pair-wise plot shows that some of the pair-wise associations are stronger than others. We will revisit this data set and use a multiple linear regression model to see which set of variables has a statistically significant impact on the sale prices.

In this module, We choose the distance to the nearest MRT station and the explanatory variable in the analysis.

6.2 Simple Linear Regression: Review

In this section, I list the basis of the simple linear regression model. I will use some mathematical equations to explain some key points. You can find the basic commands to create mathematical equations in the R Markdown on the following web page: https://rpruim.github.io/s341/S19/from-class/MathinRmd.html.

6.2.1 Structure and Assumptions of Simple Linear Regression Model

Let Y be the response variable (in our case, Y = Price of unit area and is assumed to be random) and X be the explanatory variable (in our case, X = distance to the nearest MRT station, X is assumed to be non-random). The mathematics expression of a typical simple linear regression is given by

$$y = \beta_0 + \beta_1 x + \epsilon$$

6.2.1.1 Assumptions

The basic assumptions of a linear regression model are listed below

- The responsible variable (y) and the explanatory variable (x) have a linear trend,
- The residual $\epsilon \sim N(0, \sigma^2)$. Equivalently, $y \sim N(\beta_0 + \beta_1 x)$.

6.2.1.2 Interpretation of Regression Coefficients

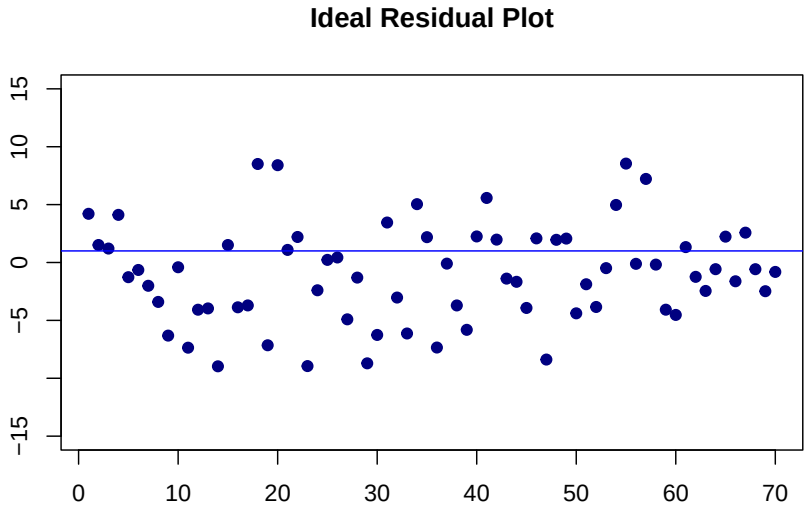
The simple linear regression model has three unknown parameters: intercept parameters (β_0), slope parameter (β_1), and the variance of the response variable (σ^2). The key parameter of interest is the slope parameter since it captures the information on whether the response variable and the explanatory variable are (linearly) associated.

- If y and x are not linearly associated, that is, $\beta_1 = 0$, then β_0 is the mean of y.
- If $\beta_1 > 0$, then y and x are positively linearly correlated. Furthermore, β_1 is the increment of the response when the explanatory variable increases by one unit.
- We can similarly interpret β_1 when it is negative.

6.2.1.3 Potential Violations to Model Assumptions

There are potential violations of model assumptions. These violations are usually reflected in the residual plot in data analysis. You can find different residual plots representing different violations from any linear regression model. The residual plot that has no model violation should be similar to the following figure.

```
# I arbitrarily choose n = 70, mu =0 and constant variance 25
residual<- rnorm(70, 0, 5)
plot(1:70, residual, pch = 19, col = "navy",
     xlab = "", ylab = "",
     ylim=c(-15, 15),
     main = "Ideal Residual Plot")
abline(h=1, col= "blue")
```



6.2.1.4 Estimation of Parameters

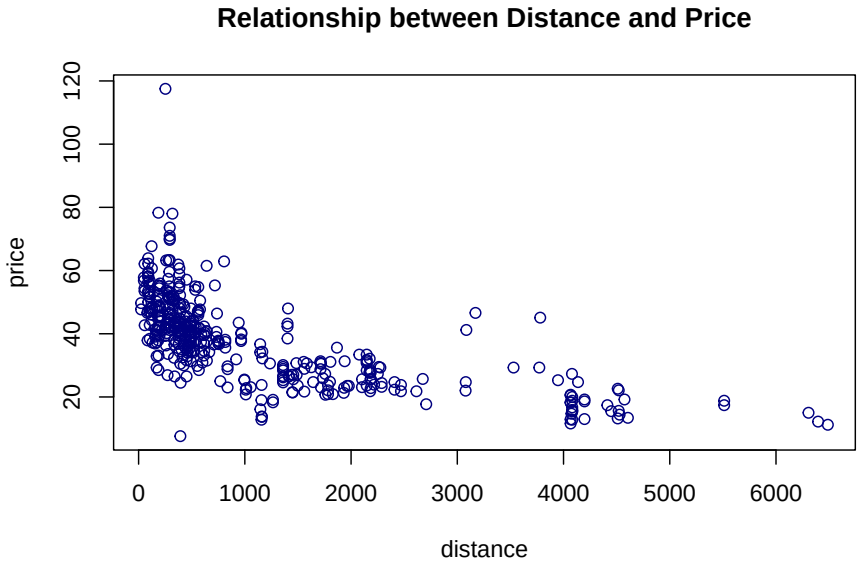
Two methods: least square estimation (LSE) and maximum likelihood estimation (MLE) yield the estimate. LSE does not use the distributional information of the response variable. The MLE uses the assumption of the normal distribution of the response variable.

When making inferences on the regression coefficients, we need to use the assumption that the response variable is normally distributed.

6.3 Fitting SLR to Data

We use both parametric and bootstrap regression models to assess the association between the sale price and the distance to the nearest MRT station. As usual, we make a scatter plot

```
distance <- realestate$Distance2MRT
price <- realestate$PriceUnitArea
plot(distance, price, pch = 21, col ="navy",
     main = "Relationship between Distance and Price")
```

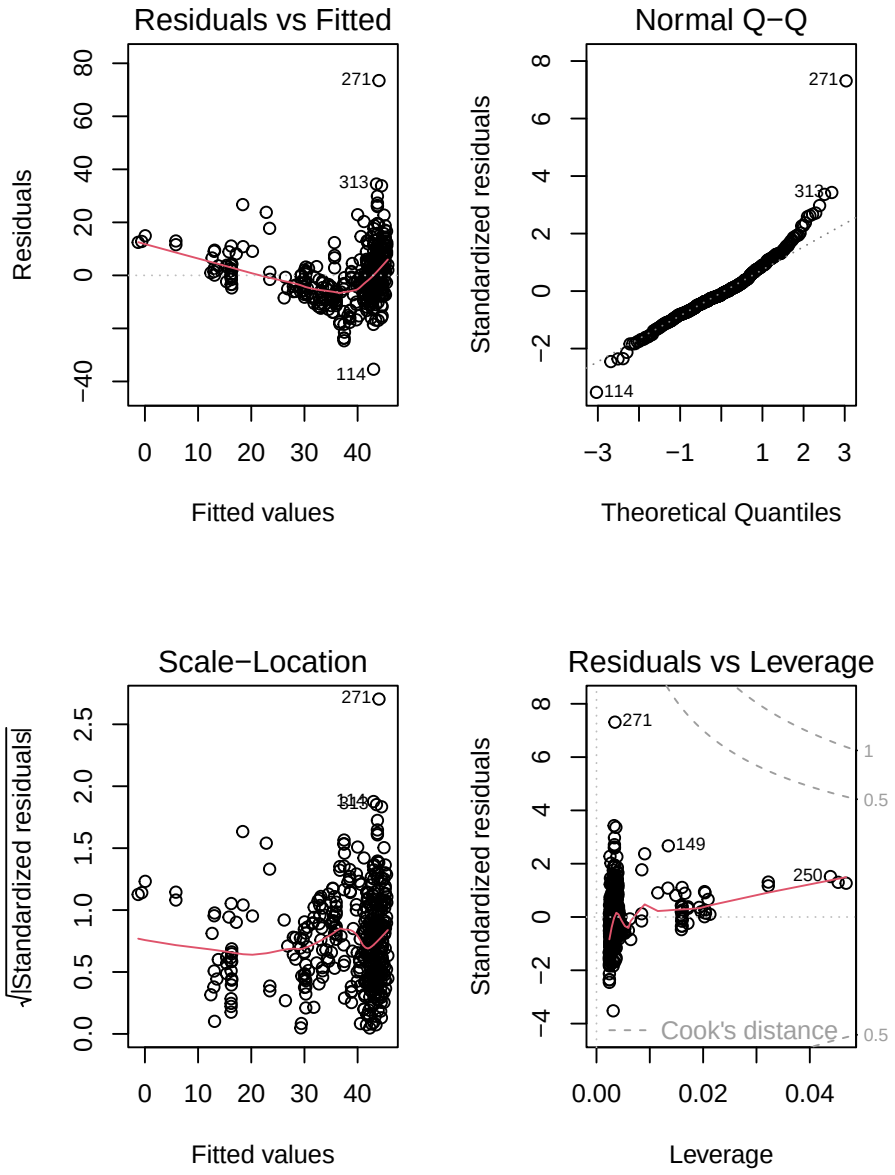


The above scatter plot indicates a negative linear association between the house price and distance to the nearest MRT station.

6.3.1 Parametric SLR

We use the built-in **lm()** to fit the SLR model.

```
distance <- realestate$Distance2MRT
price <- realestate$PriceUnitArea
parametric.model <- lm(price ~ distance)
par(mfrow = c(2,2))
plot(parametric.model)
```



6.3.1.1 Residual Plots

We can see from the residual plots that

- The top-left residual plot has three clusters indicating that some group

Table 6.1: Inferential statistics for the parametric linear regression model: house sale price and distance to the nearest MRT station

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	45.8514271	0.6526105	70.25849	0
distance	-0.0072621	0.0003925	-18.49971	0

variable is missing.

- The bottom-left plot also reveals the same pattern.
- The top-right plot reals the violation of the normality assumption.
- The bottom-right plot indicates that there are no serious outliers,
- In addition, the top-left residual plot also has a non-linear trend. We will not do the variable transformation to fix the problem,

6.3.1.2 Inferential Statistics

The inferential statistics based on the above model are summarized in the following table.

```
reg.table <- coef(summary(parametric.model))
kable(reg.table, caption = "Inferential statistics for the parametric linear
      regression model: house sale price and distance to the nearest MRT station")
```

We will not discuss the p-value since the residual plots indicate potential violations of the model assumption. In other words, the p-value may be wrong. We will wait for the bootstrap regression in the next sub-section.

A descriptive interpretation of the slope parameter is that, as the distance increases by 1000 feet, the corresponding price of unit area decreases by roughly $7.3.

6.3.2 Bootstrap Regression

There are two different non-parametric bootstrap methods for bootstrap regression modeling. We use the intuitive approach: sampling cases method.

6.3.2.1 Bootstrapping Cases

The idea is to take a bootstrap sample of the observation ID and then use the observation ID to take the corresponding records to form a bootstrap sample.

```
vec.id <- 1:length(price)   # vector of observation ID
boot.id <- sample(vec.id, length(price), replace = TRUE)   # bootstrap obs ID.
boot.price <- price[boot.id]           # bootstrap price
boot.distance <- distance[boot.id]     # corresponding bootstrap distance
```

Table 6.2: Bootstrap confidence intervals of regression coefficients.

	2.5%	97.5%
boot.beta0.ci	44.4583385	47.2190732
boot.beta1.ci	-0.0080914	-0.0065909

With bootstrap price and bootstrap distance, we fit a bootstrap linear regression.

6.3.2.2 Bootstrap Regression

If we repeat the bootstrap sampling and regression modeling many times, we will have many bootstrap regression coefficients. These bootstrap coefficients can be used to construct the bootstrap confidence interval of the regression coefficient. Since the sample size is 414. The bootstrap regression method will produce a robust confidence interval of the slope of the distance. If 0 is not in the confidence interval, then the slope is significant.

The following steps construct bootstrap confidence intervals of regression coefficients.

```
B <- 1000    # number of bootstrap replicates
# define empty vectors to store bootstrap regression coefficients
boot.beta0 <- NULL
boot.beta1 <- NULL
## bootstrap regression models using for-loop
vec.id <- 1:length(price)   # vector of observation ID
for(i in 1:B){
  boot.id <- sample(vec.id, length(price), replace = TRUE)   # bootstrap obs ID.
  boot.price <- price[boot.id]            # bootstrap price
  boot.distance <- distance[boot.id]      # corresponding bootstrap distance
  ## regression
  boot.reg <-lm(price[boot.id] ~ distance[boot.id])
  boot.beta0[i] <- coef(boot.reg)[1]   # bootstrap intercept
  boot.beta1[i] <- coef(boot.reg)[2]   # bootstrap slope
}
##  95% bootstrap confidence intervals
boot.beta0.ci <- quantile(boot.beta0, c(0.025, 0.975), type = 2)
boot.beta1.ci <- quantile(boot.beta1, c(0.025, 0.975), type = 2)
boot.coef <- data.frame(rbind(boot.beta0.ci, boot.beta1.ci))
names(boot.coef) <- c("2.5%", "97.5%")
kable(boot.coef, caption="Bootstrap confidence intervals of regression coefficients.")
```

The 95% bootstrap confidence interval of the slope is $(-0.0079753, -0.0065808)$. Since both limits are negative, the price of the unit area and the distance to the nearest MRT station are negatively associated. Note that zero is NOT in the

confidence interval. Both parametric and bootstrap regression models indicate that the slope coefficient is significantly different from zero. This means the sale price and the distance to the nearest MRT station are statistically correlated.

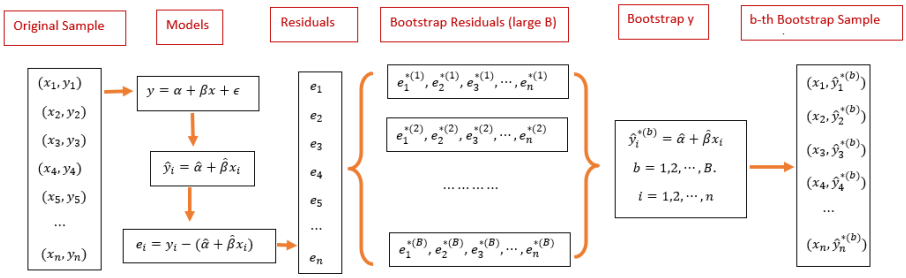
6.3.3 Concluding Remarks

Here are several remarks about the parametric and bootstrap regression models.

- If there are serious violations to the model assumptions and the sample size is not too small, bootstrap confidence intervals of regression coefficients are more reliable than the parametric p-values since the bootstrap method is non-parametric inference.
- If the form of the regression function is misspecified, the bootstrap confidence intervals are valid based on the misspecified form of the relationship between the response and the explanatory variable. However, the p-values could be wrong if the residual is not normally distributed.
- If the sample size is significantly large, both Bootstrap and the parametric methods yield the same results.
- If the sample size is too small, the bootstrap confidence interval could still be correct (depending on whether the sample empirical distribution is close to the true joint distribution of variables in the data set). The parametric regression is very sensitive to the normal assumption of the residual!
- General Recommendations
  - If there is no violation of the model assumption, always use the parametric regression. The residual plots reveal potential violations of the model assumption.
  - If there are potential violations to the model assumptions and the violations cannot be fixed by remedy methods such as variable transformation, the bootstrap method is more reliable.
  - if the same size is significantly large, bootstrap and parametric methods yield similar results.

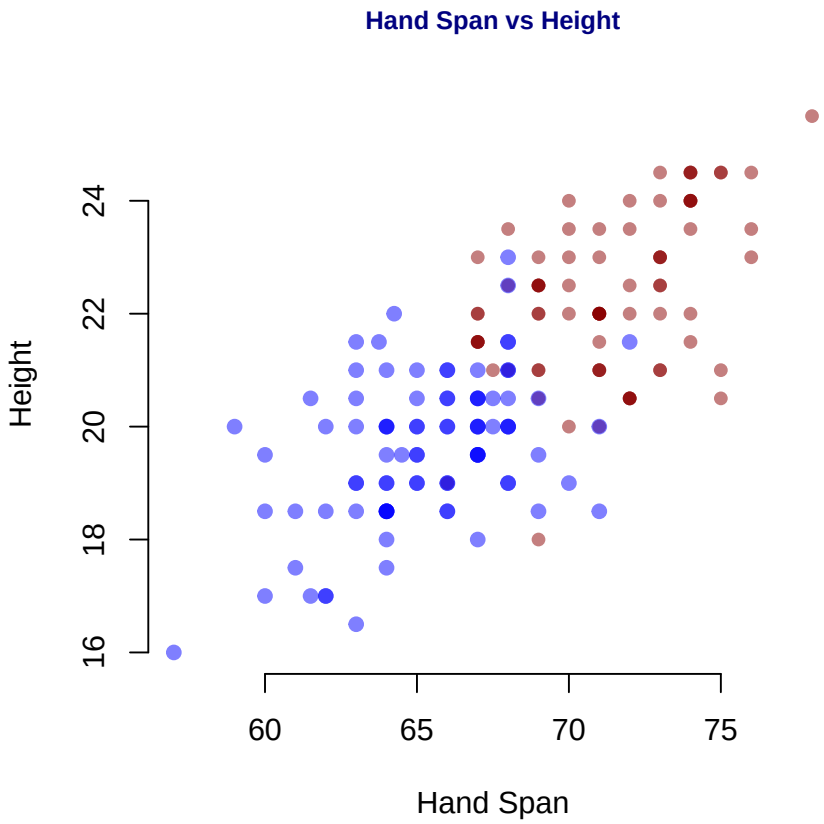
6.3.4 Bootstrap Residual Method

Bootstrapping residuals is another way to generate bootstrap random samples that are supposed to have the same distribution as that Y in the original random sample. The following flow chart explains the process of how to generate bootstrap random samples.



Next, we use a simple to demonstrate the steps to generate bootstrap samples based on sampling bootstrap residuals. The data set at https://raw.github usercontent.com/pengdsci/sta321/main/ww03/handheight.txt has three variables: sex, height, and hand span. We will this data set to assess the linear correlation between height and hand span. We first do some exploratory analysis to visualize the potential association between height and hand size.

```
myurl = "https://online.stat.psu.edu/stat501/sites/stat501/files/data/handheight.txt"
handheight = read.table(myurl, header = TRUE)
MID = which(handheight$Sex=="Male")
MaleData = handheight[MID,]
FealeData = handheight[-MID,]
plot(handheight$Height, handheight$HandSpan, pch=16, col="white",
                        xlab = "Hand Span",
                        ylab = "Height",
                        main = "Hand Span vs Height",
                        col.main = "navy",
                        cex.main = 0.8,
                        bty="n")
points(handheight$Height[MID], handheight$HandSpan[MID], pch=16, col=alpha("darkred", (
points(handheight$Height[-MID], handheight$HandSpan[-MID], pch=19, col=alpha("blue", 0
```



6.3.4.1 Implementing Bootstrapping Residuals

The following code reflects the steps in the above flow-chart

```
height = handheight$Height
handspan = handheight$HandSpan
m0 = lm(height~handspan)
E = resid(m0)              # Original residuals
a.hat = coef(m0)[1]
b.hat = coef(m0)[2]
##
B = 1000                   # generating 1000 bootstrap samples
bt.alpha = rep(0, B)
bt.beta = bt.alpha
for(i in 1:B){
  bt.e = sample(E, replace = TRUE)          # bootstrap residuals
```

```
  y.hat = a.hat + b.hat*handspan + bt.e     # bootstrap heights
  ## bootstrap SLR
  bt.m = lm(y.hat ~ handspan)
  bt.alpha[i] = coef(bt.m)[1]
  bt.beta[i] = coef(bt.m)[2]
 }
alpha.CI = quantile(bt.alpha, c(0.025, 0.975))
beta.CI = quantile(bt.beta, c(0.025, 0.975))
##
per.025 = c(alpha.CI[1],beta.CI[1])     # lower CI for alpha and beta
per.975 = c(alpha.CI[2],beta.CI[2])     # upper CI for alpha and beta
```

Next, we add the confidence limits to the output inferential table from the SLR based on the original sample.

```
lm.inference = as.data.frame((summary(m0))$coef)
lm.inference$per.025 = per.025
lm.inference$per.975 = per.975
kable(as.matrix(lm.inference))
```

	Estimate	Std. Error	t value	Pr(>|t|)	per.025	per.975
(Intercept)	35.52504	2.3159512	15.33929	0	31.012926	39.97761
handspan	1.56008	0.1105437	14.11278	0	1.344807	1.77226

6.4 Data Set Selection for Project One

It is time to find a data set for project #1 focusing on the parametric and non-parametric linear regression model. As I did in this note, you will choose one of the variables in this data set to complete this week's assignment.

6.4.1 Data Set Requirements

The basic requirements of the data set are:

The desired data set must have

- the response variable **must** be continuous random variables.
- at least two categorical explanatory variables.
- at least one of the categorical variables has more than two categories.
- at least two numerical explanatory variables.
- at least 15 observations are required for estimating each regression coefficient. For example, if your final linear model has 11 variables (including dummy variables), you need $12 \times 15 = 180$ observations.

6.4.2 Web Sites with Data Sets

The following sites have some data sets. You can also choose data from other sites.

- 10 open data sets for linear regression: https://lionbridge.ai/datasets/10-open-datasets-for-linear-regression/
- UFL Larry Winner's Teaching Data Sets: http://users.stat.ufl.edu/~winner/datasets.html
- Kaggle site: https://www.kaggle.com/datasets?tags=13405-Linear+Regression

6.4.3 Analysis and Writing Components

6.4.3.1 Description of the Data Set

Write an essay to describe the data set. The following information is expected to be included in this description.

- How the data was collected?
- List of all variables: names and their variable types.
- What are your practical and analytic questions
- Does the data set have enough information to answer the questions

6.4.3.2 Simple Linear Regression

Make a pair-wise scatter plot of all variables in your selected data set and choose an explanatory variable that is linearly correlated to the response variable.

- Make a pairwise scatter plot and comment on the relationship between the response and explanatory variables.
  - If there is a non-linear pattern, can you perform a transformation of one of the variables so that the transformed variable and the other original variable have a linear pattern?
  - If you have a choice to transform either the response variable or the explanatory variable, what is your choice and why?
- Fit an ordinary least square regression (SLR) to capture the linear relationship between the two variables. If you transformed one of the variables to achieve the linear relationship, then use the transformed variable in the model. and then perform the model diagnostics. Comment on the residual plots and point out the violations to the model assumptions.
- Using the bootstrap algorithm on the **previous final linear regression model** to estimate the bootstrap confidence intervals of regression coefficients (using 95% confidence level).

- compare the p-values and bootstrap confidence intervals of corresponding regression coefficients of the final linear regression model, make a recommendation on which inferential result to be reported, and justify.

Chapter 7

Multiple Linear Regression Model

The general purpose of multiple linear regression (MLR) is to identify a relationship in an explicit functional form between explanatory variables or predictors and the dependent response variable. This relationship will be used to achieve two primary tasks:

- **Association analysis** - understanding the association between predictors and the response variable. As a special association, the causal relationship can be assessed under certain conditions.
- **Prediction** - the relationship between predictors and the response can be used to predict the response with new out-of-sample values of predictors.

7.1 The structure of MLR

Let $\{x_1, x_2, \cdots, x_k\}$ be k explanatory variables and y be the response variables. The general form of the multiple linear regression model is defined as

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \epsilon.$$

This is a very special form in that y is linear in both parameters and predictors. The actual linear regression only assumes that y is linear only in parameters but not predictors since the value of predictors will be observed in data.

7.1.1 Assumptions based on the above special form.

- The function form between y and $\{x_1, x_2, \cdots, x_k\}$ must be correctly specified.

- The residual ϵ must be normally distributed with $\mu = 0$ and a constant variance σ^2.
- An implicit assumption is that predictor variables are non-random.

7.1.2 Potential Violations

There are various potential violations of the model assumptions. The following is a shortlist of potential violations of the model assumptions.

- The potential incorrect functional relationship between the response and the predictor variables.
  - the correct form may have power terms.
  - the correct form may have a cross-product form.
  - the correct form may need important variables that are missing.
- The residual term does not follow the normal distribution $N(0, \sigma^2)$. That means that
  - ϵ is not normally distributed at all.
  - ϵ is normally distributed but the variance is not a constant.

7.1.3 Variable types

Since there will be multiple variables involved in the multiple regression model.

- All explanatory variables are continuous variables - classical linear regression models.
- All explanatory variables are categorical variables - analysis of variance (ANOVA) models.
- The model contains both continuous and categorical variables - analysis of covariance (ANCOVA) model.

7.1.4 Dummy and discrete numerical explanatory variables

- Categorical variables with n ($n > 2$) categories MUST be dichotomized into $n - 1$ dummy variables (binary indicator variables).
  - **Caution**: categorical variable with a numerical coding - need to use R function **factor()** to automatically define a sequence of dummy variables for the category variable.
- **Discrete (numerical) variable** - If we want to use discrete numerical as a categorical variable, we must dichotomize it and define a sequence of dummy variables since interpretations of the two types of variables are different. In a categorical variable case, the coefficient of a dummy variable is the relative contribution to the response compared with the baseline category. In the discrete case, the regression coefficient is the relative contribution to the response variable compared with adjacent values and the

relative contribution is constant across all adjacent values of the discrete predictor variable.

7.1.5 Interpretation of Regression Coefficients

A _multiple linear regression model with k predictor variables has k+1 unknown parameters: intercept parameters (β_0), slope parameters ($\beta_i, i = 1, 2, \cdots, k$), and the variance of the response variable (σ^2). The key parameters of interest are the slope parameters since they capture the information on whether the response variable and the corresponding explanatory variables are (linearly) associated.

- If y and x_i are not linearly associated, that is, $\beta_i = 0, i = 1, 2, \cdots, k$, then β_0 is the mean of y.
- If $\beta_i > 0$, then y and x_i are positively linearly correlated. Furthermore, β_i is the increment of the response when the explanatory variable increases by one unit.
- We can similarly interpret β_i when it is negative.

7.2 Model building

Modeling building is an iterative process for searching for the best model to fit the data. An implicit assumption is that the underlying data is statistically valid.

7.2.1 Data structure, sample size, and preparation for MLR

In the model-building phase, we assume data is valid and has sufficient information to address the research hypothesis.

- Data records are independent - collected based on a cross-sectional design.
- The sample size should be large enough such that each regression coefficient should have 14 distinct data points to warrant reliable and robust estimates of regression coefficients.
- Imbalanced categorical variables and extremely distributed continuous explanatory variables need to be treated to a warrant valid estimate of regression coefficients. This includes combining categories in meaningful and practically interpretable ways and discretizing extremely skewed continuous variables.
- New variable definition - sometimes we can extract information from several variables to define new variables to build a better model. This is an active area in machine learning fields and data science. There are many different methods and algorithms in literature and practice for creating new variables based on existing ones.

  - Empirical approach - based on experience and numerical pattern.
  - Model-based approach - this may require a highly technical understanding of algorithms and modeling ideas. This is not the main consideration in this course.

7.2.2 Candidate models and residual diagnostics

- Consider only the multiple linear regression models that have a linear relationship between response and predictor variables.
- Perform residual analysis
  - if a curve pattern appears in residual plots, identify a curve linear relationship between the response and the individual predictor variable
  - if non-constant variance appears in the residual plots, then perform an appropriate transformation to stabilize the constant variance - for example, Box-cox transformation.
  - if the QQ plot indicates non-normal residuals, try transformations to convert it to a normal variable.
  - if there are serial patterns in the residual plot, we need to remove the serial pattern with an appropriate method.
  - if some clusters appear in the residual plot, create a group variable to capture the clustering information.

7.2.3 Significant test, goodness-of-fit, and Variable selection

Significant tests and goodness-of-fit measures are used to identify the final model. Please keep in mind that a good statistical model must have the following properties;

- Interpretability
- parsimony
- Accuracy
- Scalability

7.2.3.1 Significant Tests

Significant tests are used for selecting statistically significant variables to include in the model. However, in practical applications, some practically important variables should always be included in the model regardless of their statistical significance. The t-test is used for selecting (or dropping) individual statistically significant variables.

7.2.3.2 Variable (model) selection criteria

There are many different methods for model selection.

- R^2 - coefficient of determination. It explains the variation explained by the underlying regression model. R^2 is used to compare two candidate models. Adjusted R^2 is used when there are many predictor variables in the model.
- Likelihood ratio χ^2 test - comparing two candidate models with a hierarchical relationship.
- Information criteria - likelihood-based measures: AIC and SBC.
- Mallow's Cp - a residual-based measure that is used for comparing two models that do not necessarily have a hierarchical structure.
- F tests - testing the overall significance of a group of regression coefficients.

7.2.3.3 Variable selection methods

- Step-wise Procedures
- Criterion-based procedures

This short note summarized the above two methods for Variable Selection(click the link to view the text).

7.3 Case Study -Factors That Impact the House Sale Prices

We present a case study to implement various model-building techniques.

7.3.1 Data Description

The data in this note was found from Kaggle. I renamed the original variables and modified the sales dates to define the sales year indicator. The modified data set was uploaded to the course web page at https://raw.githubuserconten t.com/pengdsci/sta321/main/ww03/w03-Realestate.csv.

- ObsID
- TransactionYear(X1): transaction date
- HouseAge(X2): house age

- Distance2MRT(X3): distance to the nearest MRT station
- NumConvenStores(X4): number of convenience stores
- Latitude(X5): latitude

- Longitude(X6): longitude
- PriceUnitArea(Y): house price of unit area

7.3.2 Practical Question

The primary question is to identify the association between the house sale price and relevant predictor variables available in the data set.

7.3.3 Exploratory Data Analysis

We first explore the pairwise association between the variables in the data set. Since longitude and latitude are included in the data set, we first make a map to see if we can define a variable according to the sales based on the geographic regions.

To start, we load the data to R.

```
library(knitr)
realestate0 <- read.csv("https://raw.githubusercontent.com/pengdsci/sta321/main/ww03/w
realestate <- realestate0[, -1]
# longitude and latitude will be used to make a map in the upcoming analysis.
lon <- realestate$Longitude
lat <- realestate$Latitude
plot(lon, lat, main = "Sites of houses sold in 2012-2013")
abline(v=121.529, h=24.96, col="red", lty=2)
```

We use longitude and latitude to define a group variable, **geo.group**, in the following.

geo.group = TRUE if longitude > 121.529 AND latitude > 24.96; **geo.group = FALSE** otherwise.

From the map representation of the locations of these houses given below (generated by Tableau Public), we can see that **geo. group** is an indicator

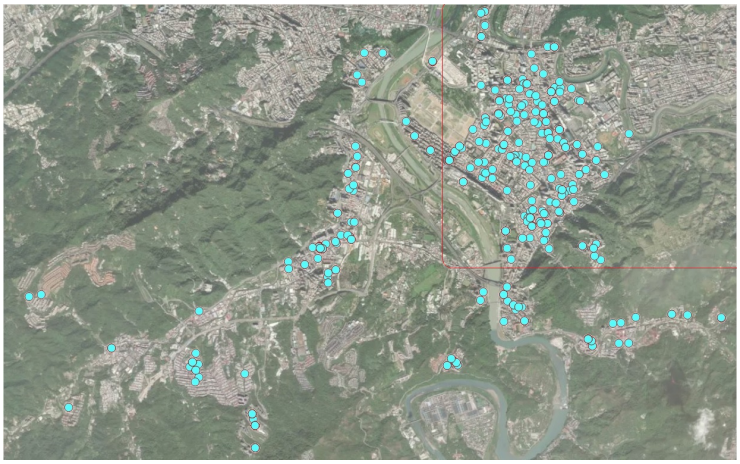


Figure 7.1: Locations of houses for sale

We also turn the variable **TransactionYear** into an indicator variable. At the same time, we scale the distance from the house to the nearest MRT by defining **Dist2MRT = Distance2MRT/1000**.

```
geo.group = (lon > 121.529) & (lat > 24.96)      # define the geo.group variable
                                                 # top-right region = TRUE, other region = FALSE
realestate$geo.group = as.character(geo.group)   # convert the logical values to character values.
realestate$sale.year = as.character(realestate$TransactionYear) # convert transaction year to dum
realestate$Dist2MRT.kilo = (realestate$Distance2MRT)/1000    # re-scale distance: foot -> kilo fee
final.data = realestate[, -c(1,3,5,6)]           # keep only variables to be used in the candidate
                                                 # models
kable(head(final.data))
```

HouseAge	NumConvenStores	PriceUnitArea	geo.group	sale.year	Dist2MRT.kilo
32.0	10	37.9	TRUE	2012	0.0848788
19.5	9	42.2	TRUE	2012	0.3065947
13.3	5	47.3	TRUE	2013	0.5619845
13.3	5	54.8	TRUE	2013	0.5619845
5.0	5	43.1	TRUE	2012	0.3905684
7.1	3	32.1	FALSE	2012	2.1750300

Table 7.1: Statistics of Regression Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	35.9559432	1.7134680	20.984310	0.0000000
HouseAge	-0.3000749	0.0390329	-7.687735	0.0000000
NumConvenStores	1.0707846	0.1897962	5.641761	0.0000000
geo.groupTRUE	7.5447005	1.3420266	5.621871	0.0000000
sale.year2013	3.0332784	0.9447021	3.210831	0.0014283
Dist2MRT.kilo	-3.6589504	0.5317107	-6.881469	0.0000000

7.3.4 Fitting MLR to Data

We start the search process for the final model.

7.3.4.1 Full model and diagnostics

We start with a linear model that includes all predictor variables.

```
full.model = lm(PriceUnitArea ~ ., data = final.data)
kable(summary(full.model)$coef, caption ="Statistics of Regression Coefficients")
```

Next, we conduct residual diagnostic analysis to check the validity of the model before making an inference about the model.

```
par(mfrow=c(2,2))
plot(full.model)
```

We can see from the residual plots that there are some minor violations:

- the variance of the residuals is not constant.
- the QQ plot indicates the distribution of residuals is slightly off the normal distribution.
- The residual plot seems to have a weak curve pattern.

We first perform Box-Cox transformation to correct the non-constant variance and correct the non-normality of the QQ plot.

7.3.4.2 Models Based on Box-Cox transformation

We first perform Box-Cox transformation and then choose appropriate transformations for both response and predictor variables to build candidate regression models.

7.3.4.2.1 Box-Cox Transformations Since non-constant variance, we perform the Box-Cox procedure to search for a transformation of the response variable. We perform several tried Box-Cox transformations with different transformed

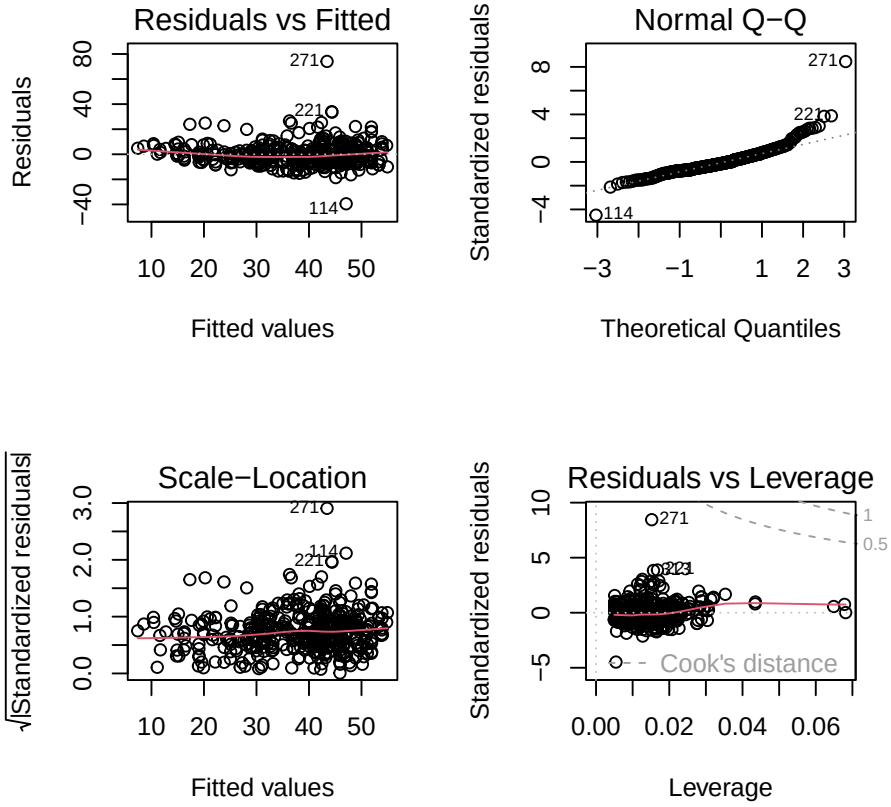


Figure 7.2: Residual plots of the full model

```
library(MASS)
par(pty = "s", mfrow = c(2, 2), oma=c(.1,.1,.1,.1), mar=c(4, 0, 2, 0))
##
boxcox(PriceUnitArea ~ HouseAge + NumConvenStores + sale.year +  log(Dist2MRT.kilo)
       + geo.group, data = final.data, lambda = seq(0, 1, length = 10),
       xlab=expression(paste(lambda, ": log dist2MRT")))
##
boxcox(PriceUnitArea ~ HouseAge + NumConvenStores + sale.year +  Dist2MRT.kilo  +
       geo.group, data = final.data, lambda = seq(-0.5, 1, length = 10),
       xlab=expression(paste(lambda, ": dist2MRT")))
##
boxcox(PriceUnitArea ~ log(1+HouseAge) + NumConvenStores + sale.year +  Dist2MRT.kilo
      geo.group, data = final.data, lambda = seq(-0.5, 1, length = 10),
       xlab=expression(paste(lambda, ": log-age")))
##
boxcox(PriceUnitArea ~ log(1+HouseAge) + NumConvenStores + sale.year +  log(Dist2MRT.ki
      geo.group, data = final.data, lambda = seq(-0.5, 1, length = 10),
      xlab=expression(paste(lambda, ": log-age, log.dist2MRT")))
```

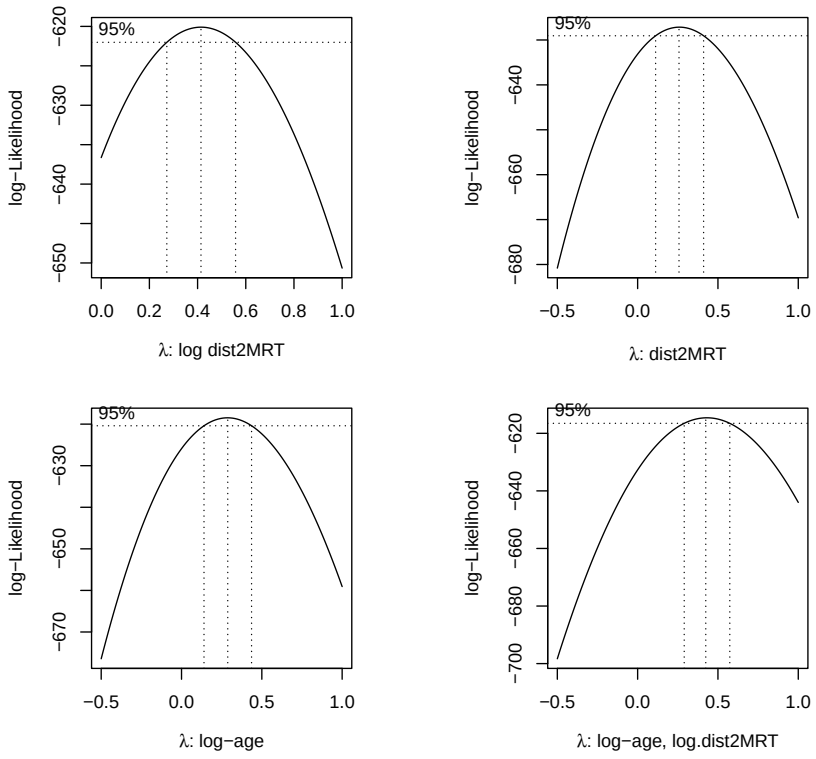


Table 7.2: log-transformed model

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	5.3978808	0.0923831	58.429342	0.0000000
HouseAge	-0.0209198	0.0029561	-7.076954	0.0000000
NumConvenStores	0.0513220	0.0153946	3.333767	0.0009351
sale.year2013	0.2951406	0.0707905	4.169214	0.0000374
log(Dist2MRT.kilo)	-0.4905637	0.0481611	-10.185897	0.0000000
geo.groupTRUE	0.5709120	0.0973966	5.861723	0.0000000

The above Box-cox transformation plots indicate the optimal λ under different transformed predictor variables. log-Transformed distance from MRT impacts the coefficient of the power transformation: λ.

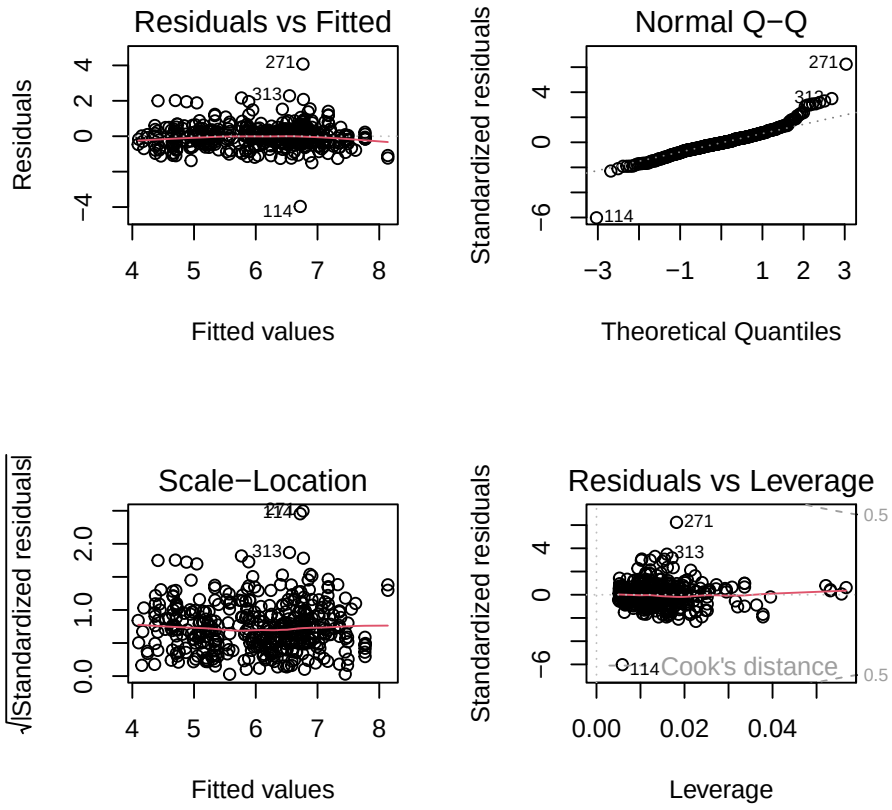
As a special power transformation, if $\lambda = 0$, the transformation degenerates to log transformation.

7.3.4.2.2 Square-root Transformation We perform Box-Cox transformation with log-transformed distance to the nearest MRT in the following.

```
sqrt.price.log.dist = lm((PriceUnitArea)^0.5 ~ HouseAge + NumConvenStores + sale.year +  log(Dist
kable(summary(sqrt.price.log.dist)$coef, caption = "log-transformed model")
```

Residual plots are given below.

```
par(mfrow = c(2,2))
plot(sqrt.price.log.dist)
```



There are two improvements in the above residual diagnostic plots: (1) the weak curve pattern has been removed from the residual plot; (2) the non-constant variance has also been corrected. However, the violation of the normality assumption is still an issue.

7.3.4.2.3 Log-Transformation We take the log transformation of the sale price according to the Box-Cox transformation and then build a linear regression based on the log price.

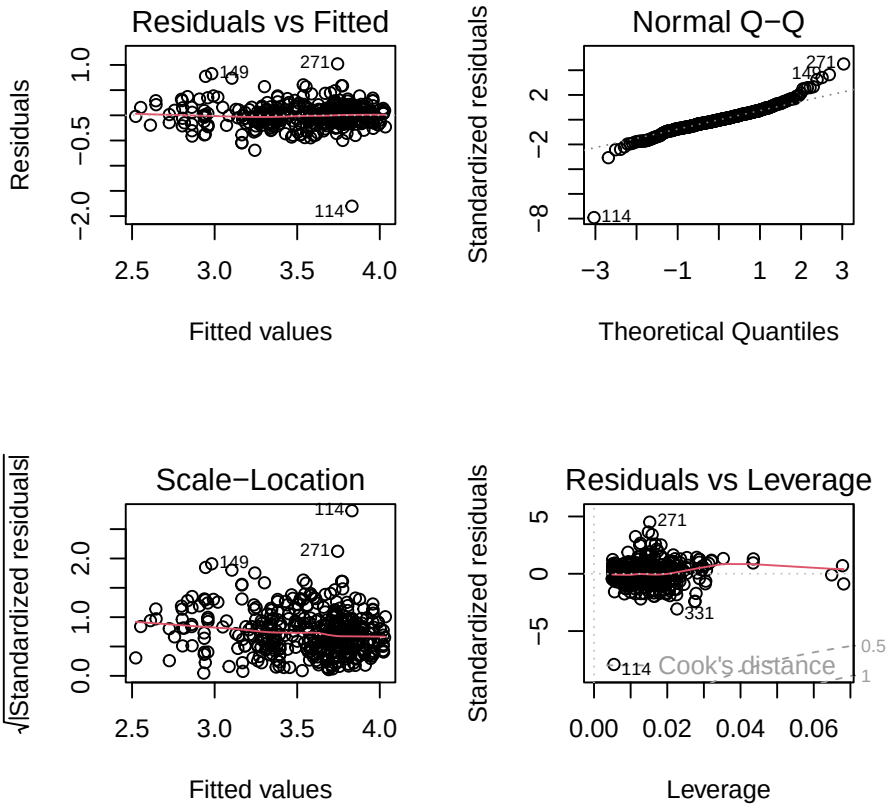
```
log.price = lm(log(PriceUnitArea) ~ HouseAge + NumConvenStores + sale.year +  Dist2MRT
kable(summary(log.price)$coef, caption = "log-transformed model")
```

Residual plots are given below.

```
par(mfrow = c(2,2))
plot(log.price)
```

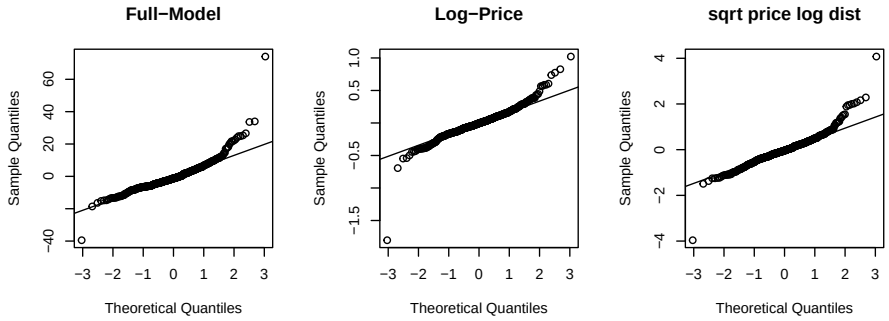
Table 7.3: log-transformed model

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	3.5724282	0.0443235	80.599030	0.0000000
HouseAge	-0.0075712	0.0010097	-7.498507	0.0000000
NumConvenStores	0.0274872	0.0049096	5.598667	0.0000000
sale.year2013	0.0805519	0.0244373	3.296272	0.0010655
Dist2MRT.kilo	-0.1445122	0.0137541	-10.506820	0.0000000
geo.groupTRUE	0.1825871	0.0347151	5.259583	0.0000002



The above residual diagnostic plots are similar to that of the previous model. The Q-Q plots of all three models are similar to each other, this means that the assumption of normal residuals is not satisfied for all three models.

```
#define plotting area
par(pty = "s", mfrow = c(1, 3))
#Q-Q plot for original model
qqnorm(full.model$residuals, main = "Full-Model")
qqline(full.model$residuals)
#Q-Q plot for Box-Cox transformed model
qqnorm(log.price$residuals, main = "Log-Price")
qqline(log.price$residuals)
#display both Q-Q plots
qqnorm(sqrt.price.log.dist$residuals, main = "sqrt price log dist")
qqline(sqrt.price.log.dist$residuals)
```



7.3.4.2.4 Goodness-of-fit Measures Next, we extract several other goodness-of-fit from each of the three candidate models and summarize them in the following table.

```
select=function(m){ # m is an object: model
 e = m$resid                              # residuals
 n0 = length(e)                           # sample size
 SSE=(m$df)*(summary(m)$sigma)^2          # sum of squared error
 R.sq=summary(m)$r.squared                # Coefficient of determination: R square!
 R.adj=summary(m)$adj.r                   # Adjusted R square
 MSE=(summary(m)$sigma)^2                 # square error
 Cp=(SSE/MSE)-(n0-2*(n0-m$df))            # Mellow's p
 AIC=n0*log(SSE)-n0*log(n0)+2*(n0-m$df)          # Akaike information criterion
 SBC=n0*log(SSE)-n0*log(n0)+(log(n0))*(n0-m$df)  # Schwarz Bayesian Information criter
 X=model.matrix(m)                        # design matrix of the model
```

Table 7.4: Goodness-of-fit Measures of Candidate Models

	SSE	R.sq	R.adj	Cp	AIC	SBC	PRESS
full.model	31831.19255	0.5836958	0.5785940	6	1809.7271	1833.8823	32792.58816
sqrt.price.log.dist	177.56943	0.6587132	0.6545308	6	-338.4528	-314.2976	182.95839
log.price	21.29945	0.6651882	0.6610851	6	-1216.4146	-1192.2594	21.94809

Table 7.5: Inferential Statistics of Final Model

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	3.5724282	0.0443235	80.599030	0.0000000
HouseAge	-0.0075712	0.0010097	-7.498507	0.0000000
NumConvenStores	0.0274872	0.0049096	5.598667	0.0000000
sale.year2013	0.0805519	0.0244373	3.296272	0.0010655
Dist2MRT.kilo	-0.1445122	0.0137541	-10.506820	0.0000000
geo.groupTRUE	0.1825871	0.0347151	5.259583	0.0000002

```
H=X%*%solve(t(X)%*%X)%*%t(X)          # hat matrix
d=e/(1-diag(H))
PRESS=t(d)%*%d    # predicted residual error sum of squares (PRESS)- a cross-validation measure
tbl = as.data.frame(cbind(SSE=SSE, R.sq=R.sq, R.adj = R.adj, Cp = Cp, AIC = AIC, SBC = SBC, PRD
names(tbl)=c("SSE", "R.sq", "R.adj", "Cp", "AIC", "SBC", "PRESS")
tbl
}
```

```
output.sum = rbind(select(full.model), select(sqrt.price.log.dist), select(log.price))
row.names(output.sum) = c("full.model", "sqrt.price.log.dist", "log.price")
kable(output.sum, caption = "Goodness-of-fit Measures of Candidate Models")
```

We can see from the above table that the goodness-of-fit measures of the third model are unanimously better than the other two models. Considering the interpretability, goodness-of-fit, and simplicity, we choose the last model as the final model.

7.3.4.3 Final Model

The inferential statistics of the final working model are summarized in the following table.

```
kable(summary(log.price)$coef, caption = "Inferential Statistics of Final Model")
```

Since the sample size (414) is large, the argument for validating p-values is the Central Limit Theorem (CLT). all p-values are close to 0 meaning that all coefficients are significantly different from 0.

In this specific case study, there is no need to perform variable selection to determine the final model.

7.3.5 Summary of the model

We can explicitly write the final model in the following

$$\log(price) = 3.5723 - 0.0076 \times HouseAge + 0.0275 \times NumConvenStores +$$

$$0.0805 \times Sale.year2013 - 0.1445 \times Dist2MRT.kilo + 0.1826 \times geo.groupTRUE$$

Note that the estimated regression coefficients are based on log price. we now consider **the set of** all houses with ages x_0 and $x_0 + 1$ that are in the same conditions except for the sale prices. The exact practical interpretation is given below. Let p_{x_0} and p_{x_0+1} be the mean prices of houses with ages x_0 and $x_0 + 1$, respectively. Since the two types of houses are in the same conditions except for the age and the prices. Then

$$\log(p_{x_0+1}) - \log(p_{x_0}) = -0.0076 \rightarrow \log(p_{x_0+1}/p_{x_0}) = -0.0076 \rightarrow p_{x_0+1} = 0.9924 p_{x_0}$$

We re-express the above equation can be re-written as

$$p_{x_0+1} - p_{x_0} = -0.0076 p_{x_0} \rightarrow \frac{p_{x_0+1} - p_{x_0}}{p_{x_0}} = -0.076 = -0.76\%$$

That is, as the house age increases by one year, the house price **decreases** by 0.76%. We can similarly interpret other regression coefficients.

The distance to the nearest MRT is also negatively associated with the sale price. The rest of the factors are positively associated with house prices.

7.3.6 Discussions

We use various regression techniques such as Box-Cox transformation for response variables and other transformations of the explanatory variables to search for the final model in the case study. Since there are five variables in the data set and all are significant, we did not perform any variable selection procedure.

All candidate models have the same number of variables. We use commonly-used global goodness-of-fit measures as model selection criteria.

The interpretation of the regression coefficients is not straightforward since the response variable was transformed into a log scale. We used some algebra to derive the practical interpretation of the regression coefficients associated with the variables at their original scales.

The violation of the normal assumption of the residuals remains uncorrected. The inference on the regression coefficients is based on the central limit theorem. We will introduce bootstrap methods to construct bootstrap confidence intervals of the regression coefficients of the **final model**.

7.4 Written Assignment

This assignment focuses on the multiple regression model using various techniques you learned from your previous courses. You will use the data set you selected last week. That data set will also be used for the next assignment that is based on this week's report.

Please study the first three sections of the class note. Your data analysis and write-up should be similar to what did in the case study in section 4 of the note. In fact, the case study of this chapter can be considered a standalone statistical report.

Your analysis and write-up should have all components in the case study of this chapter.

- Description of your data set
- What are the research questions
- Exploratory analysis of the data set and prepare the analytic data for the regression
  - create new variables based on existing ones?
  - drop some irrelevant variables based on your judgment.
- Initial full model with all relevant variables and conduct residual diagnostic
  - special patterns in residual plots?
  - violation of model assumptions?
- Need to transform the response variable with Box-Cox?
- Want to transform explanatory variables to improve goodness-of-fit measures?
  - Please feel free to use my code to extract the goodness-of-fit measures. If you forgot the meaning of the goodness-of-fit measures, please check your old textbook or the eBook that I suggested on the course web page.
  - Several packages have a Box-Cox transformation function. The one that I used in the case study is from the library {MASS}. You can check the help document if you are sure how to use it.
- Build several candidate models and then select the best one as your model.
- Summarize the output including residual diagnostic plots
- Interpret the regression coefficient as I did in the case study.
- Conclusions and discussion.

Chapter 8

Bootstrapping MLR Model

In this note, we introduce two versions of bootstrap procedures to generate bootstrap samples to estimate the confidence intervals of the coefficients of the regression model identified in the previous note. We first fit the linear model to the data and then use bootstrap methods to construct the confidence intervals of regression coefficients.

8.1 Parametric Linear Regression Model

For reference, we copy the code from the last note that creates the analytic data set for the final model and the summarized statistics of the model as well.

```
library(knitr)
housingURL = "https://raw.githubusercontent.com/pengdsci/sta321/main/ww03/w03-Realestate.csv"
realestate0 <- read.csv(housingURL , header = TRUE)
realestate <- realestate0[, -1]
# longitude and latitude will be used to make a map in the upcoming analysis.
lat <- realestate$Latitude
lon <- realestate$Longitude
## define the geo.group variable
## top-right region = TRUE, other region = FALSE
geo.group <- (lon > 121.529) & (lat > 24.96)
# convert the logical values to character values.
realestate$geo.group <- as.character(geo.group)
# convert transaction year to dummy.
realestate$sale.year <- as.character(realestate$TransactionYear)
# re-scale distance: foot -> kilo feet
realestate$Dist2MRT.kilo <- (realestate$Distance2MRT)/1000
# keep only variables to be used in the candidate models
final.data = realestate[, -c(1,3,5,6)]
```

Table 8.1: Inferential Statistics of Final Model

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	3.5724	0.0443	80.5990	0.0000
HouseAge	-0.0076	0.0010	-7.4985	0.0000
NumConvenStores	0.0275	0.0049	5.5987	0.0000
sale.year2013	0.0806	0.0244	3.2963	0.0011
Dist2MRT.kilo	-0.1445	0.0138	-10.5068	0.0000
geo.groupTRUE	0.1826	0.0347	5.2596	0.0000

```
final.data$logAreaPrice = log(final.data$PriceUnitArea)  #
## the final model
log.price <- lm(log(PriceUnitArea) ~ HouseAge + NumConvenStores + sale.year +
                  Dist2MRT.kilo  + geo.group, data = final.data)
log.price02 <- lm(logAreaPrice ~ HouseAge + NumConvenStores + sale.year +
                  Dist2MRT.kilo  + geo.group, data = final.data)
cmtrx <- round(summary(log.price)$coef,4)
cmtrx02 <- round(summary(log.price02)$coef)
kable(cmtrx, caption = "Inferential Statistics of Final Model")
```

The explicit expression of the final model is given by

$$\log(price) = 3.5723 - 0.0076 \times HouseAge + 0.0275 \times NumConvenStores +$$

$$0.0805 \times Sale.year2013 - 0.1445 \times Dist2MRT.kilo + 0.1826 \times geo.groupTRUE$$

As another example of the interpretation of the regression coefficient, we choose the coefficient associated with **geo.group**. In the output, you see the name of the dummy variable with the suffix `TRUE`, **geo.groupTRUE**. The suffix **TRUE** indicates that the dummy variable represents the category 'TRUE' of the category variable **geo.group**. The associated coefficient reflects the **mean** difference between the category `TRUE` and the baseline category `FALSE`. In R, the default baseline category is the lowest value of the categorical variable (in alphabetical order).

Let's consider **the set of all houses** that are in the same conditions except the regions (region `TRUE` and region `FALSE`) and the sale prices.

Next, we explain the estimated regression coefficient of 0.1826. Let p_{TRUE} be the mean price of a house in the region `TRUE` and p_{FALSE} be the mean price of houses in the region `FALSE`. Then

$$\log(p_{TRUE}) - \log(p_{FALSE}) = 0.1826 \rightarrow \log(p_{TRUE}/p_{FALSE}) = 0.1826 \rightarrow p_{TRUE} = 1.20 p_{FALSE}$$

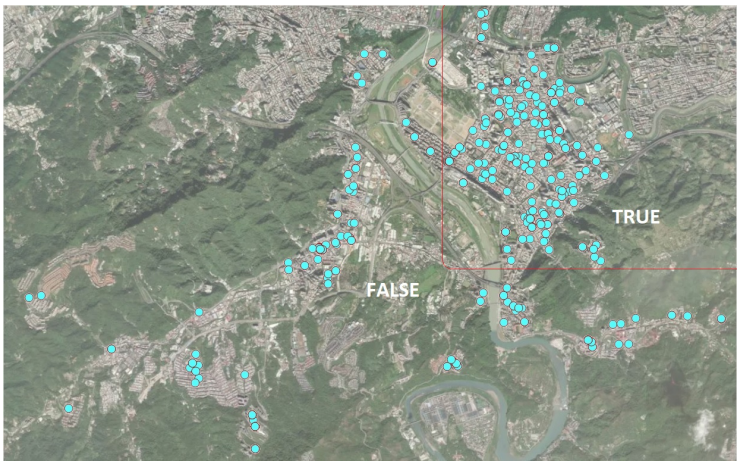


Figure 8.1: Location of Houses for Sale

We re-express the above equation can be re-written as

$$p_{TRUE} - p_{FALSE} = 0.206 p_{FALSE} \rightarrow \frac{p_{TRUE} - p_{FALSE}}{p_{FALSE}} = 0.20 = 20\%.$$

That is, the average house sales price in the `TRUE` region (top right corner on the map) is about 20% higher than that in the `FALSE` region. We can similarly interpret other regression coefficients.

8.2 Bootstrap Cases (BOOT.C)

In this section, we use bootstrapping cases to find the confidence intervals for the coefficients in the final regression model. The method was used in bootstrap simple linear regression (SLR) in week #3. The following code finds the confidence interval.

```
log.price = lm(log(PriceUnitArea) ~ HouseAge + NumConvenStores + sale.year +
                 Dist2MRT.kilo  + geo.group, data = final.data)
##
B = 1000    # choose the number of bootstrap replicates.
##
num.p = dim(model.frame(log.price))[2]  # returns number of parameters
smpl.n = dim(model.frame(log.price))[1] # sample size
## zero matrix to store bootstrap coefficients
coef.mtrx = matrix(rep(0, B*num.p), ncol = num.p)
##
for (i in 1:B){
```

```
  bootc.id = sample(1:smpl.n, smpl.n, replace = TRUE)
  # fit final model to the bootstrap sample
  log.price.btc = lm(log(PriceUnitArea) ~ HouseAge + NumConvenStores + sale.year +
                       Dist2MRT.kilo  + geo.group, data = final.data[bootc.id,])
  # extract coefs from bootstrap regression model
  coef.mtrx[i,] = coef(log.price.btc)
}
```

We define an R function to make histograms of the bootstrap regression coefficients in the following. I will also use this function to make histograms for the residual bootstrap estimated regression coefficients as well.

```
boot.hist = function(cmtrx, bt.coef.mtrx, var.id, var.nm){
  ## bt.coef.mtrx = matrix for storing bootstrap estimates of coefficients
  ## var.id = variable ID (1, 2, ..., k+1)
  ## var.nm = variable name on the hist title,
  ## must be the string in the double quotes
  ## coefficient matrix of the final model
  ## Bootstrap sampling distribution of the estimated coefficients
  x1.1 <- seq(min(bt.coef.mtrx[,var.id]), max(bt.coef.mtrx[,var.id]), length=300 )
  y1.1 <- dnorm(x1.1, mean(bt.coef.mtrx[,var.id]), sd(bt.coef.mtrx[,var.id]))
  # height of the histogram - use it to make a nice-looking histogram.
  highestbar = max(hist(bt.coef.mtrx[,var.id], plot = FALSE)$density)
  ylimit <- max(c(y1.1,highestbar))
  hist(bt.coef.mtrx[,var.id], probability = TRUE, main = var.nm, xlab="",
       col = "azure1",ylim=c(0,ylimit), border="lightseagreen")
  lines(x = x1.1, y = y1.1, col = "red3")
  lines(density(bt.coef.mtrx[,var.id], adjust=2), col="blue")
  #legend("topright", c(""))
}
```

The following histograms of the bootstrap estimates of regression coefficients represent the sampling distributions of the corresponding estimates in the final model.

```
par(mfrow=c(2,3))  # histograms of bootstrap coefs
boot.hist(bt.coef.mtrx=coef.mtrx, var.id=1, var.nm ="Intercept" )
boot.hist(bt.coef.mtrx=coef.mtrx, var.id=2, var.nm ="House Age" )
boot.hist(bt.coef.mtrx=coef.mtrx, var.id=3, var.nm ="Num Conven Stores" )
boot.hist(bt.coef.mtrx=coef.mtrx, var.id=4, var.nm ="Year Sold" )
boot.hist(bt.coef.mtrx=coef.mtrx, var.id=5, var.nm ="Dist to MRT" )
boot.hist(bt.coef.mtrx=coef.mtrx, var.id=6, var.nm ="Region" )
```

Two normal-density curves were placed on each of the histograms.

- The red **density curve** uses the estimated regression coefficients and their corresponding standard error in the output of the regression procedure.

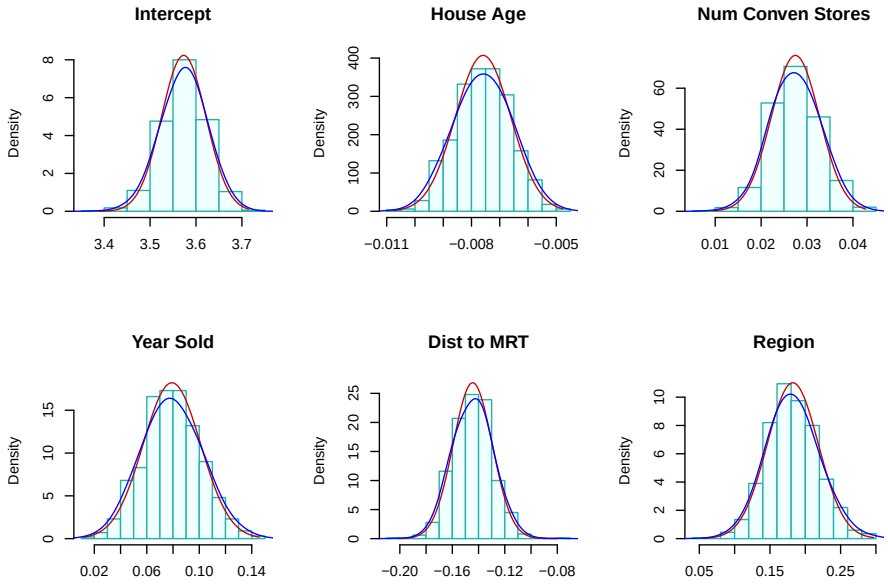


Figure 8.2: (Case) Bootstrap confidence intervals of regression coefficients

The p-values reported in the output are based on the red curve.

- The **blue curve** is a non-parametric data-driven estimate of the density of bootstrap sampling distribution. The bootstrap confidence intervals of the regressions are based on these non-parametric bootstrap sampling distributions.

We can see from the above histograms that the two density curves in all histograms are close to each other. we would expect that significance test results and the corresponding bootstrap confidence intervals are consistent. Next, we find 95% bootstrap confidence intervals of each regression coefficient and combined them with the output of the final model.

```
num.p = dim(coef.mtrx)[2]  # number of parameters
btc.ci = NULL
btc.wd = NULL
for (i in 1:num.p){
  lci.025 = round(quantile(coef.mtrx[, i], 0.025, type = 2),8)
  uci.975 = round(quantile(coef.mtrx[, i],0.975, type = 2 ),8)
  btc.wd[i] =  uci.975 - lci.025
  btc.ci[i] = paste("     [", round(lci.025,4),", ", round(uci.975,4),"]")
 }
#btci.95 = as.data.frame(btc.ci)
```

Table 8.2: Regression Coefficient Matrix

	Estimate	Std. Error	t value	Pr(>|t|)	btc.ci
(Intercept)	3.5724	0.0443	80.599	0	[3.4742 , 3.6675]
HouseAge	-0.0076	0.001	-7.4985	0	[-0.0094 , -0.0057]
NumConvenStores	0.0275	0.0049	5.5987	0	[0.0172 , 0.0374]
sale.year2013	0.0806	0.0244	3.2963	0.0011	[0.0374 , 0.1242]
Dist2MRT.kilo	-0.1445	0.0138	-10.5068	0	[-0.1733 , -0.1162]
geo.groupTRUE	0.1826	0.0347	5.2596	0	[0.1135 , 0.2574]

```
kable(cbind(cmtrx, btc.ci),
      caption = "Regression Coefficient Matrix")
```

We can see from the above table of summarized statistics, the significance tests of regression coefficients based on the p-values and the corresponding 95% confidence intervals are consistent.

8.3 Bootstrap Residuals (BOOT.R)

In this section, we introduce bootstrap residual methods to estimate bootstrap confidence intervals. The idea is straightforward and is summarized in the following.

8.3.1 Fitted Model

Assume that the fitted regression model is given by

$$\begin{array}{rcl} y_1 & = & \hat{\beta}_0 + \hat{\beta}_1 x_{11} + \hat{\beta}_2 x_{12} + \cdots + \hat{\beta}_k x_{1k} + e_1 \\ y_2 & = & \hat{\beta}_0 + \hat{\beta}_1 x_{21} + \hat{\beta}_2 x_{22} + \cdots + \hat{\beta}_k x_{2k} + e_2 \\ y_3 & = & \hat{\beta}_0 + \hat{\beta}_1 x_{31} + \hat{\beta}_2 x_{32} + \cdots + \hat{\beta}_k x_{3k} + e_3 \\ \vdots & \vdots & \vdots \\ y_n & = & \hat{\beta}_0 + \hat{\beta}_1 x_{n1} + \hat{\beta}_2 x_{n2} + \cdots + \hat{\beta}_k x_{nk} + e_n \end{array}$$

where $\{e_1, e_2, \cdots, e_n\}$ is the set of residuals obtained from the final model. $\{x_{i1}, x_{i2}, \cdots, x_{ik}\}$ is the i-th record from the data, and $\{\hat{\beta}_0, \hat{\beta}_1, \cdots, \hat{\beta}_k\}$

The distribution of the residuals is depicted in the following histogram.

```
hist(sort(log.price$residuals),n=40,
     xlab="Residuals",
     col = "lightblue",
     border="navy",
     main = "Histogram of Residuals")
```

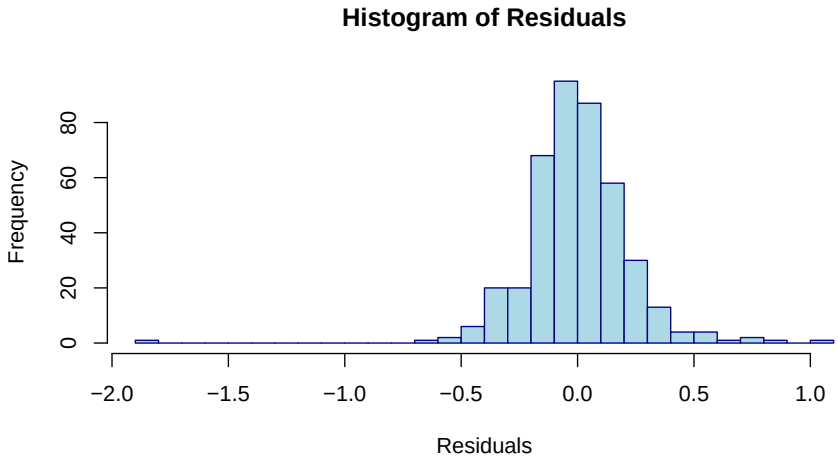


Figure 8.3: Distribution of residuals of the linear regression model

The above histogram reveals the same information as we saw in the residual plot in the last note: (1) one out-lier; (2). The distribution is skewed to the right.

8.3.2 Residual Bootstrap Samples

The residual bootstrap sample of y is defined in the following:

- Take a **bootstrap sample** from the set of residuals $\{e_1, e_2, \cdots, e_n\}$, denoted by $\{e_1^*, e_2^*, \cdots, e_n^*\}$.
- The residual bootstrap sample of $\{y_1^*, y_2^*, \cdots, y_n^*\}$ is defined by

$$\begin{array}{rcl} y_1^* & = & \hat{\beta}_0 + \hat{\beta}_1 x_{11} + \hat{\beta}_2 x_{12} + \cdots + \hat{\beta}_k x_{1k} + e_1^* \\ y_2^* & = & \hat{\beta}_0 + \hat{\beta}_1 x_{21} + \hat{\beta}_2 x_{22} + \cdots + \hat{\beta}_k x_{2k} + e_2^* \\ y_3^* & = & \hat{\beta}_0 + \hat{\beta}_1 x_{31} + \hat{\beta}_2 x_{32} + \cdots + \hat{\beta}_k x_{3k} + e_3^* \\ \vdots & \vdots & \vdots \\ y_n^* & = & \hat{\beta}_0 + \hat{\beta}_1 x_{n1} + \hat{\beta}_2 x_{n2} + \cdots + \hat{\beta}_k x_{nk} + e_n^* \end{array}$$

The above definition implies that the residual bootstrap is equal to the **fitted value + bootstrap residuals**.

- The resulting **residual bootstrap sample** is given by

$$\begin{array}{ccccc} y_1^* & x_{11} & x_{12} & \cdots & x_{1k} \\ y_2^* & x_{21} & x_{22} & \cdots & x_{2k} \\ y_3^* & x_{31} & x_{32} & \cdots & x_{3k} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_n^* & x_{n1} & x_{n2} & \cdots & x_{nk} \end{array}$$

- We fit the final model to the **residual bootstrap sample** and denote the bootstrap estimates of regression coefficients in the following

$$\{\hat{\beta}_0^*, \hat{\beta}_1^*, \cdots, \hat{\beta}_k^*\}$$

- Repeat the above steps B times, we obtain the following bootstrap estimates

$$\begin{array}{cccc} \hat{\beta}_0^{1*} & \hat{\beta}_1^{1*} & \cdots & \hat{\beta}_k^{1*} \\ \hat{\beta}_0^{2*} & \hat{\beta}_1^{2*} & \cdots & \hat{\beta}_k^{2*} \\ \hat{\beta}_0^{3*} & \hat{\beta}_1^{3*} & \cdots & \hat{\beta}_k^{3*} \\ \vdots & \vdots & \vdots & \vdots \\ \hat{\beta}_0^{b*} & \hat{\beta}_1^{b*} & \cdots & \hat{\beta}_k^{b*} \\ \vdots & \vdots & \vdots & \vdots \\ \hat{\beta}_0^{B*} & \hat{\beta}_1^{B*} & \cdots & \hat{\beta}_k^{B*} \end{array}$$

The residual bootstrap confidence intervals of regression coefficients can be estimated from the above bootstrap coefficients.

8.3.3 Implementation of Residual Bootstrap Regression

The following code generates bootstrap confidence intervals of regression coefficients.

```
## Final model
log.price <- lm(log(PriceUnitArea) ~ HouseAge + NumConvenStores + sale.year +
                  Dist2MRT.kilo  + geo.group, data = final.data)
model.resid = log.price$residuals
##
B=1000
num.p = dim(model.matrix(log.price))[2]   # number of parameters
samp.n = dim(model.matrix(log.price))[1]  # sample size
btr.mtrx = matrix(rep(0,6*B), ncol=num.p) # zero matrix to store boot coefs
for (i in 1:B){
  ## Bootstrap response values
  bt.lg.price = log.price$fitted.values +
        sample(log.price$residuals, samp.n, replace = TRUE)  # bootstrap residuals
  # replace PriceUnitArea with bootstrap log price
  final.data$bt.lg.price =  bt.lg.price   #  send the boot response to the data
```

```
  btr.model = lm(bt.lg.price ~ HouseAge + NumConvenStores + sale.year +
                 Dist2MRT.kilo  + geo.group, data = final.data)   # b
  btr.mtrx[i,]=btr.model$coefficients
}
```

Next, we make histograms of the residual bootstrap estimates of the regression coefficients.

```
boot.hist = function(bt.coef.mtrx, var.id, var.nm){
  ## bt.coef.mtrx = matrix for storing bootstrap estimates of coefficients
  ## var.id = variable ID (1, 2, ..., k+1)
  ## var.nm = variable name on the hist title,
  ## must be the string in the double quotes
  ## Bootstrap sampling distribution of the estimated coefficients
  x1.1 <- seq(min(bt.coef.mtrx[,var.id]), max(bt.coef.mtrx[,var.id]), length=300 )
  y1.1 <- dnorm(x1.1, mean(bt.coef.mtrx[,var.id]), sd(bt.coef.mtrx[,var.id]))
  # height of the histogram - use it to make a nice-looking histogram.
  highestbar = max(hist(bt.coef.mtrx[,var.id], plot = FALSE)$density)
  ylimit <- max(c(y1.1,highestbar))
  hist(bt.coef.mtrx[,var.id], probability = TRUE, main = var.nm, xlab="",
       col = "azure1",ylim=c(0,ylimit), border="lightseagreen")
  lines(x = x1.1, y = y1.1, col = "red3")       # normal density curve
  lines(density(bt.coef.mtrx[,var.id], adjust=2), col="blue")    # loess curve
}
```

```
par(mfrow=c(2,3))  # histograms of bootstrap coefs
boot.hist(bt.coef.mtrx=btr.mtrx, var.id=1, var.nm ="Intercept" )
boot.hist(bt.coef.mtrx=btr.mtrx, var.id=2, var.nm ="House Age" )
boot.hist(bt.coef.mtrx=btr.mtrx, var.id=3, var.nm ="Num Conven Stores" )
boot.hist(bt.coef.mtrx=btr.mtrx, var.id=4, var.nm ="Year Sold" )
boot.hist(bt.coef.mtrx=btr.mtrx, var.id=5, var.nm ="Dist to MRT" )
boot.hist(bt.coef.mtrx=btr.mtrx, var.id=6, var.nm ="Region" )
```

The residual bootstrap sampling distributions of each estimated regression coefficient. The normal and LOESS curves are close to each other. This also indicated that the inference of the significance of variables based on p-values and residual bootstrap will yield the same results.

The 95% residual bootstrap confidence intervals are given in the following

```
#
num.p = dim(coef.mtrx)[2]  # number of parameters
btr.ci = NULL
btr.wd = NULL
for (i in 1:num.p){
  lci.025 = round(quantile(btr.mtrx[, i], 0.025, type = 2),8)
  uci.975 = round(quantile(btr.mtrx[, i],0.975, type = 2 ),8)
```

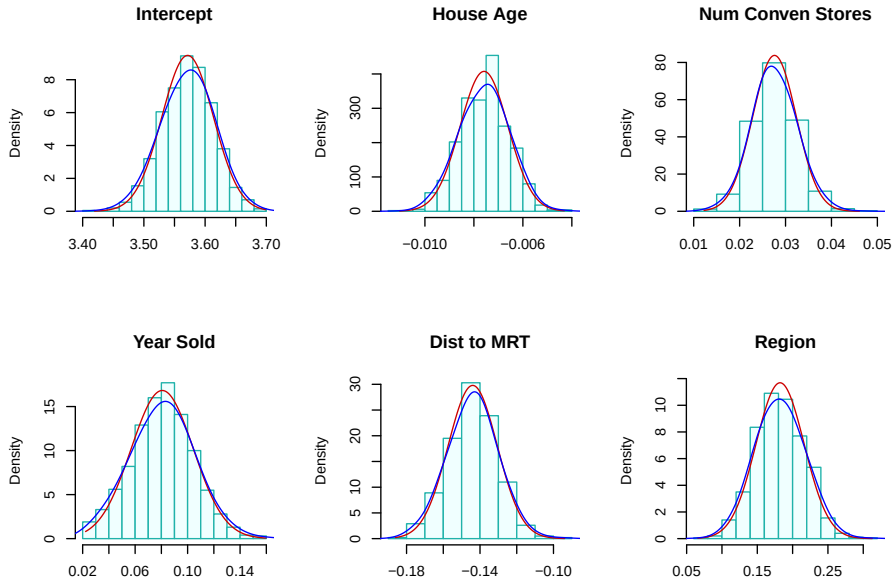


Figure 8.4: (Residual) Bootstrap confidence intervals

```
  btr.wd[i] = uci.975 - lci.025
  btr.ci[i] = paste("[", round(lci.025,4),", ", round(uci.975,4),"]")
}
#as.data.frame(btc.ci)
cbind(cmtrx, btr.ci.95=btr.ci)
```

```
##                 Estimate  Std. Error t value    Pr(>|t|)
## (Intercept)     "3.5724"  "0.0443"   "80.599"   "0"
## HouseAge        "-0.0076" "0.001"    "-7.4985"  "0"
## NumConvenStores "0.0275"  "0.0049"   "5.5987"   "0"
## sale.year2013   "0.0806"  "0.0244"   "3.2963"   "0.0011"
## Dist2MRT.kilo   "-0.1445" "0.0138"   "-10.5068" "0"
## geo.groupTRUE   "0.1826"  "0.0347"   "5.2596"   "0"
##                 btr.ci.95
## (Intercept)     "[ 3.4847 ,  3.6516 ]"
## HouseAge        "[ -0.0096 ,  -0.0058 ]"
## NumConvenStores "[ 0.018 ,  0.0375 ]"
## sale.year2013   "[ 0.033 ,  0.1268 ]"
## Dist2MRT.kilo   "[ -0.1727 ,  -0.1182 ]"
## geo.groupTRUE   "[ 0.1182 ,  0.2487 ]"
```

Table 8.3: Final Combined Inferential Statistics: p-values and Bootstrap CIs

	Estimate	Std. Error	Pr(>|t|)	btc.ci.95	btr.ci.95
(Intercept)	3.5724	0.0443	0	[3.4742 , 3.6675]	[3.4847 , 3.6516]
HouseAge	-0.0076	0.001	0	[-0.0094 , -0.0057]	[-0.0096 , -0.0058]
NumConvenStores	0.0275	0.0049	0	[0.0172 , 0.0374]	[0.018 , 0.0375]
sale.year2013	0.0806	0.0244	0.0011	[0.0374 , 0.1242]	[0.033 , 0.1268]
Dist2MRT.kilo	-0.1445	0.0138	0	[-0.1733 , -0.1162]	[-0.1727 , -0.1182]
geo.groupTRUE	0.1826	0.0347	0	[0.1135 , 0.2574]	[0.1182 , 0.2487]

Table 8.4: width of the two bootstrap confidence intervals

btc.wd	btr.wd
0.1932667	0.1668870
0.0037541	0.0037804
0.0202037	0.0195303
0.0868612	0.0937271
0.0571331	0.0544304
0.1439029	0.1304962

```
#kable(cbind(cmtrx, btr.ci.95=btr.ci),
#      caption = "Regression Coefficient Matrix with 95% Residual Bootstrap CI")
```

As expected, the residual bootstrap confidence intervals yield the same results as p-values do. This is because the sample size is large enough so that the sampling distributions of estimated coefficients have sufficiently good approximations of normal distributions.

8.3.4 Combining All Inferential Statistics

Finally, we put all inferential statistics in a single table so we can compare these results.

```
kable(cbind(cmtrx[,-3], btc.ci.95=btc.ci,btr.ci.95=btr.ci),
      caption="Final Combined Inferential Statistics: p-values and Bootstrap CIs")
```

The above table shows that

- All three methods yield the same results in terms of the significance of individual explanatory variables. The reason is that the final working model does not have a serious violation of the model assumption.

```
kable(cbind(btc.wd, btr.wd), caption="width of the two bootstrap confidence intervals")
```

- The widths of residual bootstrap and case-bootstrap confidence intervals

are similar to each other. See the above table for the widths of each confidence interval.

8.4 Modeling Assignment

This assignment focuses on bootstrapping the multiple regression model you identified in your last assignment. You are expected to use both bootstrap sampling methods to find the confidence intervals for regression coefficients.

Please follow what did in the lecture note to do a similar analysis using your data and summarize the bootstrap analysis results. The mini-project report combines the results you obtained last week with this week's bootstrap results.

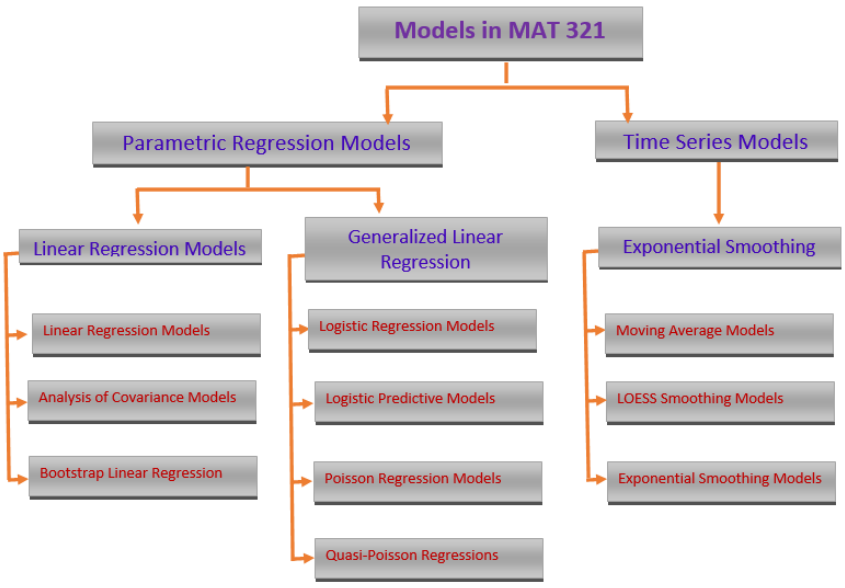
The format of the report should have the following components using last and this week's work as building blocks.

- Introduction
  - rationale of the project - research questions
  - description of data - sampling?
  - sufficient information for addressing the research questions
- Data preparation and exploratory analysis
  - create new variables based on existing variables?
  - discretizing continuous variable? combining categories of categorical variables?
- Model building
  - starting with a full model and performing residual analysis
  - identifying violations and finding remedies
  - need model transformations like the Cox-Cox procedure?
  - variable selection? -backward and forward selection, stepwise selection.
  - model selection using goodness-of-fit measures.
  - selection of the final model.
- Bootstrapping the final model
  - bootstrapping records
  - bootstrapping residuals of the final model obtained in the previous model
- Combining the results of the regular model and bootstrap results
  - Summarizing the results
  - Correct interpretations of the coefficients of the regression model.
  - Interpretation with caution if your response variable was transformed on a different scale.
- Summary and discussion
  - outline main findings
  - drawbacks and future improvements
  - recommendations on the applications of the model.

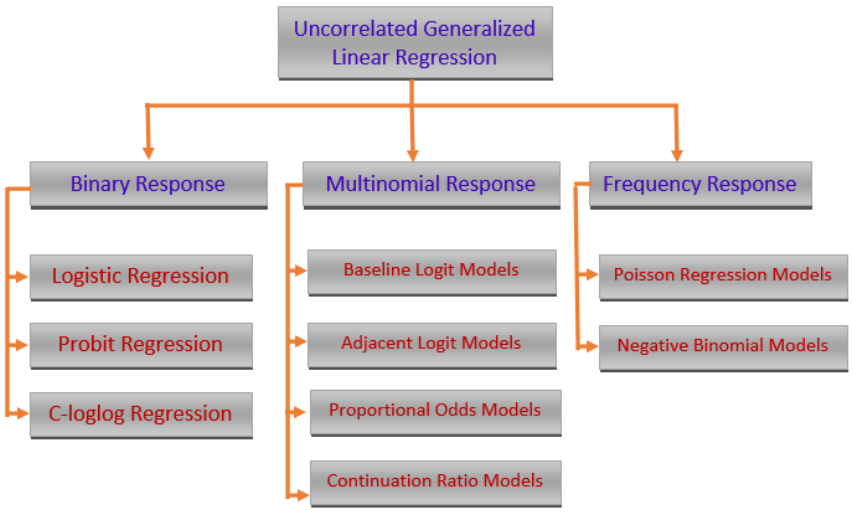
Chapter 9

Simple Logistic Regression Model

We have systematically reviewed linear regression models and used parametric and bootstrap methods to build the linear regression models.



Starting from this note, we will study several models in the family of generalized linear models (GLM). We first overview the major members of this family.



Three models will be introduced in the subsequent notes. We primarily focus on the commonly used models that are used in statistics and machine learning fields. The **binary logistic regression models** and their variants are probably the most popular models that have been used in almost all areas of quantitative decision.

Before taking presenting the technical introduction of the binary logistic regression model, we look at the major structural differences between the regular linear regression model and logistic regression models.

	Linear Regression	Logistic Regression
Response Variable	• Continuous variable	• Categorical variable
Predictor Variable	• Continuous variables • Discrete variables • Categorical variables	• Continuous variables • Discrete variables • Categorical variables
Fitted and Predict Values	• Mean of the response • Predicted value and response variable value have the same scale.	• Log odds of being in the non-baseline category • Predicted odds of being in the non-baseline category
Model Assumptions	• Correct specification of response function. • Random errors are normally distributed with a constant variances.	• Correct specification of response function. • There is no random error specified in the model.
Diagnostics	• Residual plots • Global goodness-of-fit	• No residual analysis • Global goodness-of-fit
Estimation Methods	• Least Square Estimation (LSE) • Maximum Likelihood Estimation (MLE)	• Maximum Likelihood Estimation (MLE)

9.1 Simple Binary Logistic Regression

In this module, we study a new regression model with a binary response variable that takes on exactly one of two possible values such as **success** v.s. **failure**, **diseased** v.s. **disease-free**, etc.

For a binary population, we are interested in the proportion of one of the two values. For example, if we study a disease of a certain population, the primary interest is the prevalence of a disease - the proportion of subjects in the population who had the disease. We also know that the relative frequency (proportion) of the disease can be used to estimate the disease probability of the population. A natural and practical question is whether the disease probability is impacted by some factors - this is a repression problem. In the subsequent sections, we present a brief but little technical introduction to **the binary logistic regression model with a single predictor variable**.

9.1.1 The Model Structure

Let Y be a binary variable that takes on exactly one of the two possible values, say **success** or **failure**. Assume the proportion of **success** to be p. That is, $P(Y = \text{success}) = p$. Let x be a factor that may impact the success probability.

Clearly, Y is a Bernoulli random variable. Since we are interested in the success probability, we encode the values of variable Y by setting **success** to 1 and **failure** to 0.

Recall that the linear regression model is defined by $E[Y] = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k$ and

$$\epsilon = Y - E[Y] \to N(0, \sigma^2).$$

where Y is a continuous random variable and X's are assumed to be non-random.

When we have a binary population (i.e., Y takes only two possible distinct values: **success** or **failure**), we are primarily interested in the proportion of **success** or **failure**. Since the Y is a Bernoulli (random) variable, the expected value of the binary response variable Y is given by.

$$E[Y] = 1 \times p + 0 \times (1 - p) = p.$$

That is, the success probability p is the expected value of the binary random variable. The regression problem is to look at how the **success** probability is affected by potential factors. One initial guess is to use the formulation of the linear regression model by setting

$$p = E[Y] = \beta_0 + \beta_1 x.$$

The above formulation is inappropriate since the right-hand side of the equation can take on any real value while the left-hand side equation can take on only values in the interval $[0, 1]$.

One of the most important criteria for a good statistical model is its interpretability. We cannot link the success probability with the linear function of the predictor. Next, we link the **odds of success** $(p/(1-p))$ with the linear function of the predictor variable in the following.

$$\frac{p}{1-p} = \beta_0 + \beta_1 x.$$

The above expression shows the association between the **odds of success**, $p/(1-p)$, and the predictor variable x. It is easy to interpret. But the **odds of success** take on the value in $[0, \infty)$. The above formulation is still inappropriate.

However, $-\infty < \log \frac{p}{1-p} < \infty$, it is reasonable to explore the association between the logarithm of odds of success and the predictor variable x. That is,

$$\log \frac{p}{1-p} = \beta_0 + \beta_1 x.$$

The above model is called the **simple logistic regression model**.

Remarks on the model formulation

- The above logistic regression model was formulated based on the interpretability of the transformed mean response, $E[Y]$, and its linear association with the predictor variable.
- Mathematically, there are different transformations of the mean response to define a family of regression models $g(p) = \beta_0 + \beta_1 x$. Clearly, the above simple logistic regression model is a special member of this family - generalized linear regression models. There are two other less famous but commonly used in some specific fields: the complementary log-log model and the probit model.

9.1.2 Interpretation of Regression Coefficients

If we use numerical coding 1 = "success" and 0 = "failure", the simple logistic regression model can be explicitly expressed in the following form

$$\log \frac{P(Y=1|x)}{1-P(Y=1|x)} = \beta_0 + \beta_1 x.$$

The expression $P(Y = 1|x)$ highlights that the success probability is dependent on the predictor variable x.

The regression coefficients of the simple logistic regression model have a meaningful interpretation.

- Intercept β_0 is the baseline log odds of success. In other words, if the success probability is not impacted by any factors, β_0 is the log odds of success of the homogeneous population.
- the slope parameter β_1 is called log odds ratio of two categories corresponding to x and $x + 1$. To see this, denote $p_x = P(Y = 1|x)$ and $p_{x+1} = P(Y = 1|x + 1)$, then

$$\log \frac{p_x}{1-p_x} = \beta_0 + \beta_1 x \quad \text{and} \quad \log \frac{p_{x+1}}{1-p_{x+1}} = \beta_0 + \beta_1 (x+1).$$

Taking the difference between the above two equations, we have

$$\log \frac{p_{x+1}}{1-p_{x+1}} - \log \frac{p_x}{1-p_x} = \beta_1$$

Therefore,

$$\log \frac{p_{x+1}/(1-p_{x+1})}{p_x/(1-p_x)} = \beta_1,$$

that is, β_1 is the ratio of log odds of success in two sub-populations.

In general, we don't make inferences on the intercept parameter! This does not mean that it is unimportant. It is as important as other slope parameters when the model is used for prediction!

9.1.3 Use of Simple Logistic Regression Model

We first express the success probability with respect to the predictor variable x in the following

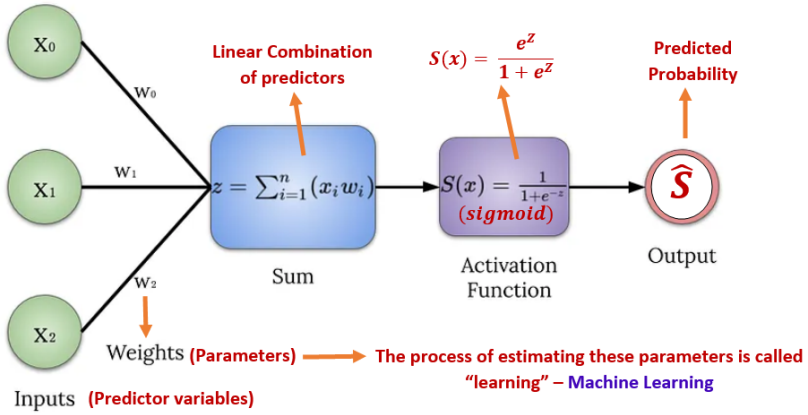
$$P(Y=1|x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}.$$

Using the above expression, we can do one of the following analyses.

- **Association Analysis** - if $\beta_1 \neq 0$, then the success probability is impacted by the predictor variable x. Note that, this is a non-linear association.
- **Predictive Analysis** - predicting the success probability for a given new value of the predictor variable. That is,

$$\widehat{P(Y=1|x_{\text{new}})} = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 x_{\text{new}}}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 x_{\text{new}}}},$$

where $\hat{\beta}_0$ and $\hat{\beta}_1$ are estimated from the data.



- **Classification Analysis** - predicting the status of success. That is, for a given new value of the predictor variable, we predict the value of Y through $P(Y = 1|x_{new})$.
  - To predict whether the value of Y is **success** or **failure**, we need to identify the cut-off probability to determine the value of Y.
  - The predicted model can be used in an **intervention analysis** - this means values of X can alter the value of Y. This is commonly used in clinical studies. For example, an effective treatment (X) can permanently cure a disease (Y).
  - The predicted model can be used for membership classification. For example, the response value is the gender (Y) of a car buyer at a car dealer, the predictor variable is the purchase status (X). If a customer bought a car from the dealer, the fitted model can identify whether the customer is a man or woman. This is apparently different from the intervention analysis since purchase status cannot change the gender (Y) of the customer.

9.1.4 Parameter Estimation

In linear regression models, both likelihood and least square methods can be used for estimating the coefficients of linear regression models. Both methods yield the same estimates. However, in the logistic regression model, we can only use the likelihood methods to estimate the regression coefficients.

Let $\{(y_1, x_1), (y_2, x_2), \cdots, (y_n, x_n)\}$ be a random sample taken from a binary population associated with Y. x is a nonrandom predictor variable associated with Y. The logistic model is defined to be

$$p(x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}.$$

Since Y_i is a Bernoulli random variable with success probability p_x. We use numerical coding: 1 = "success" and 0 = "failure". The likelihood function of (β_0, β_1) is given by

$$L(\beta_0, \beta_1) = \prod_{i=1}^{n} p(x_i)^{y_i} \left[1 - p(x_i)\right]^{1-y_i} = \prod_{i=1}^{n} \left[\frac{e^{\beta_0+\beta_1 x_i}}{1 + e^{\beta_0+\beta_1 x_i}}\right]^{y_i} \times \left[\frac{1}{1 + e^{\beta_0+\beta_1 x_i}}\right]^{1-y_i}$$

The maximum likelihood estimate (MLE) of β_0 and β_1, denoted by $\hat{\beta}_0$ and $\hat{\beta}_1$, maximizes the above likelihood. We will use the R build-in function **glm()** to find the MLE of the parameters and related statistics.

9.1.5 Model Assumptions and Diagnostics

In linear regression, we assume the response variable follows a normal distribution with a constant variance. With this assumption, several effective diagnostic methods were developed based on the residual analysis. In logistic regression, we don't have many diagnostic methods. However, several likelihood-based goodness of fit metrics such as AIC and deviance can be used for comparing the performance of candidate models.

More technical discussion of diagnostics with the left to the future specialized courses in generalized linear regression models.

9.1.6 Concluding Remarks

We only introduced the basic logistic regression modeling in this note. some important topics you may want to study but not mentioned in this note are

- logistic regression as a machine learning algorithm for predictive modeling.
- logistic regression model with a large number of predictor variables - regularized logistic regression.
- performance metrics based on prediction errors.

9.2 A Case Study

The diabetes data set in this case study contains 768 observations on 9 variables. The data set is available in the UCI machine learning data repository. R library {mlbench} has two versions of this data. The data set contains a significant number of missing values.

9.2.1 Data and Variable Descriptions

There are 9 variables in the data set.

1. **pregnant**: Number of times pregnant
2. **glucose**: Plasma glucose concentration (glucose tolerance test)
3. **pressure**: Diastolic blood pressure (mm Hg)
4. **triceps**: Triceps skin fold thickness (mm)
5. **insulin**: 2-Hour serum insulin (mu U/ml)
6. **mass**: Body mass index (weight in kg/(height in m)^2)
7. **pedigree**: Diabetes pedigree function
8. **age**: Age (years)
9. **diabetes**: Class variable (test for diabetes)

I load the data from R **library{mlbench}** in the following code.

```
library(knitr)
library(mlbench)
data(PimaIndiansDiabetes2)           # load the data to R work-space
diabetes.0 = PimaIndiansDiabetes2    # make a copy of the data for data cleansing
diabetes = na.omit(diabetes.0)       # Delete all records with missing components
y0=diabetes$diabetes
diabete.01 = rep(0, length(y0))      # define a 0-1 to test which probability is used in glm()
diabete.01[which(y0=="pos")] = 1
diabetes$diabetes.01 = diabete.01
# head(diabetes)
```

For convenience, I delete all records with missing values and keep only the records with complete records in this case study. The final analytic data set has 392 records.

9.2.2 Clinical Question

Many studies indicated that body mass index (BMI) is a more powerful risk factor for diabetes than genetics. The objective of this case study is to explore the **association** between BMI and diabetes.

The general interpretation of BMI for adults is given below:

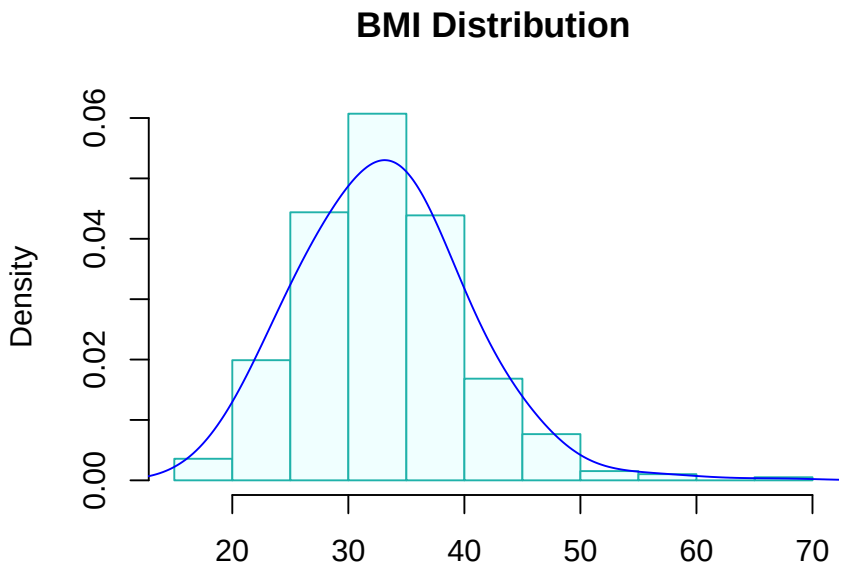
- **underweight**: < 18.5
- **Normal and Healthy Weight**: [18.5, 24.9]
- **Overweight**: [25.0, 29.9]
- **Obese**: > 30.0

9.2.3 Building the Simple Logistic Regression

Since we only study the simple logistic regression model, only one predictor variable is included in the model. We first perform exploratory data analysis on

the predictor variable to make sure the variable is not extremely skewed.

```
ylimit = max(density(diabetes$mass)$y)
hist(diabetes$mass, probability = TRUE, main = "BMI Distribution", xlab="",
       col = "azure1", border="lightseagreen")
  lines(density(diabetes$mass, adjust=2), col="blue")
```



Since the simple logistic regression contains only one continuous variable of a binary categorical variable as the predictor variable, no there is no issue of potential imbalance. We will not transform BMI and fit a logistic regression directly to the data.

```
s.logit = glm(diabetes ~ mass,
          family = binomial(link = "logit"),  # family is the binomial, logit(p) = log
          data = diabetes)                    # the data frame is a subset of the orig
summary(s.logit)
```

```
##
## Call:
## glm(formula = diabetes ~ mass, family = binomial(link = "logit"),
##     data = diabetes)
##
```

Table 9.1: The summary stats of regression coefficients

	Estimate	Std. Error	z value	Pr(>|z|)	2.5 %	97.5 %
(Intercept)	-3.6061432	0.5917329	-6.094208	0.0000000	-4.8064211	-2.4817472
mass	0.0863295	0.0170535	5.062276	0.0000004	0.0538266	0.1208267

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7737  -0.9032  -0.6794   1.3004   1.9017
##
## Coefficients:
##             Estimate Std. Error z value     Pr(>|z|)
## (Intercept) -3.60614    0.59173  -6.094 0.0000000011 ***
## mass         0.08633    0.01705   5.062 0.0000004143 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 498.10  on 391  degrees of freedom
## Residual deviance: 469.03  on 390  degrees of freedom
## AIC: 473.03
##
## Number of Fisher Scoring iterations: 4
```

Note that the response variable is a binary factor variable, R uses alphabetical order to define the level of the factor variable. In our case, "neg" = 0 and "pos" = 1. The "success" probability is defined to be P(diabetes = "pos"). The simple logistic regression is fitted in the following.

The summary of major statistics is given below.

```
model.coef.stats = summary(s.logit)$coef       # output stats of coefficients
conf.ci = confint(s.logit)                     # confidence intervals of betas
```

```
## Waiting for profiling to be done...
```

```
sum.stats = cbind(model.coef.stats, conf.ci.95=conf.ci)   # rounding off decimals
kable(sum.stats,caption = "The summary stats of regression coefficients")
```

From the above table, we can see that BMI is positively associated with the status of diabetes since $\beta_1 = 0.0863$ with a p-value close to 0. The 95% confidence interval [0.0538, 0.1208]. This also supports the results of the research in the literature.

It is more common to interpret the association results from a practical perspective using the odds ratio. Next, we convert the estimated regression coefficients

Table 9.2: Summary Stats with Odds Ratios

	Estimate	Std. Error	z value	Pr(>|z|)	odds.ratio
(Intercept)	-3.6061432	0.5917329	-6.094208	0.0000000	0.0271564
mass	0.0863295	0.0170535	5.062276	0.0000004	1.0901655

to the odds ratio.

```
# Odds ratio
model.coef.stats = summary(s.logit)$coef
odds.ratio = exp(coef(s.logit))
out.stats = cbind(model.coef.stats, odds.ratio = odds.ratio)
kable(out.stats,caption = "Summary Stats with Odds Ratios")
```

The odds ratio associated with BMI is 1.09 meaning that as the BMI increases by one unit, the odds of being tested positive for diabetes increase by about 9%. This is a practically significant risk factor for diabetes.

Some global goodness-of-fit measures are summarized in the following table.

```
## Other global goodness-of-fit
dev.resid = s.logit$deviance
dev.0.resid = s.logit$null.deviance
aic = s.logit$aic
goodness = cbind(Deviance.residual =dev.resid, Null.Deviance.Residual = dev.0.resid,
      AIC = aic)
kable(goodness)
```

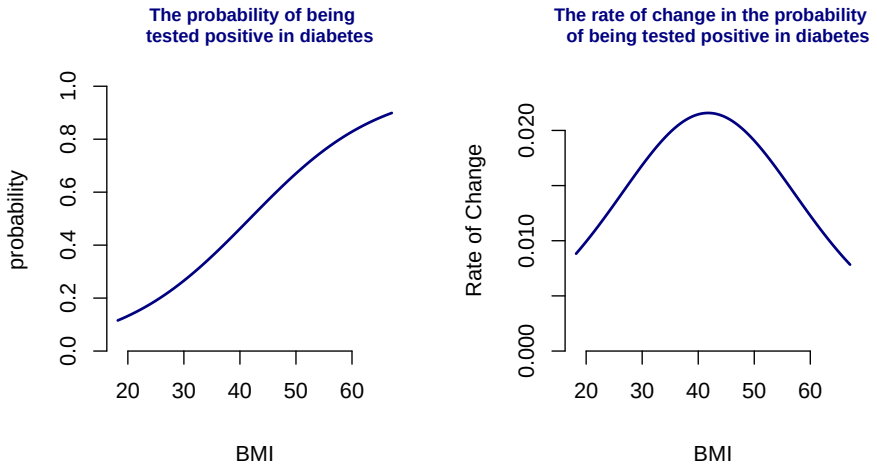
Deviance.residual	Null.Deviance.Residual	AIC
469.031	498.0978	473.031

Since the above global goodness-of-fit is based on the likelihood function, we don't have other candidate models with corresponding likelihood at the same scale to compare in this simple logistic regression model, we will not interpret these goodness-of-fit measures.

The success probability curve (so-called S curve) is given below.

```
bmi.range = range(diabetes$mass)
x = seq(bmi.range[1], bmi.range[2], length = 200)
beta.x = coef(s.logit)[1] + coef(s.logit)[2]*x
success.prob = exp(beta.x)/(1+exp(beta.x))
failure.prob = 1/(1+exp(beta.x))
ylimit = max(success.prob, failure.prob)
##
beta1 = coef(s.logit)[2]
success.prob.rate = beta1*exp(beta.x)/(1+exp(beta.x))^2
##
```

```
##
par(mfrow = c(1,2))
plot(x, success.prob, type = "l", lwd = 2, col = "navy",
     main = "The probability of being \n  tested positive in diabetes",
     ylim=c(0, 1.1*ylimit),
     xlab = "BMI",
     ylab = "probability",
     axes = FALSE,
     col.main = "navy",
     cex.main = 0.8)
# lines(x, failure.prob,lwd = 2, col = "darkred")
axis(1, pos = 0)
axis(2)
# legend(30, 1, c("Success Probability", "Failure Probability"), lwd = rep(2,2),
#        col = c("navy", "darkred"), cex = 0.7, bty = "n")
##
y.rate = max(success.prob.rate)
plot(x, success.prob.rate, type = "l", lwd = 2, col = "navy",
     main = "The rate of change in the probability \n  of being tested positive in diabetes",
     xlab = "BMI",
     ylab = "Rate of Change",
     ylim=c(0,1.1*y.rate),
     axes = FALSE,
     col.main = "navy",
     cex.main = 0.8
     )
axis(1, pos = 0)
axis(2)
```



The left-hand side plot in the above figure is the standard **S curve** representing how the probability of a positive test increases as the BMI increases. After diving deeper to see the rate of change in the probability of a positive test, we obtain the curve on the right-hand side that indicates that the rate of change in the probability of positive test increases when BMI is less than 40 and decreases when BMI is greater than 40. The turning point is about 40.

9.3 Conclusion

This note focuses on the structure and association analysis of the simple logistic regression model. The case study uses a real-world diabetes data set to illustrate the steps for carrying out the simple logistic regression model.

In the following modules, we will discuss multiple logistic regression models that will include more modeling techniques.

Since logistic regression has been used as a standard machine-learning algorithm for classification, I will use a standalone module to discuss this topic.

9.4 Analysis Assignment

This assignment focuses on the simple linear regression model. Please follow the instructions to pick an appropriate data set for the next three assignments that will be ensembled for project #2.

1. Find a data set that contains

- at least **two** categorical variables,
- **three** numerical variables, and

- a **binary response variable**.

2. The number of observations is at least 150. This number of observations is slightly larger than the one required in project #1 since we will introduce a machine-learning algorithm using the logistic regression model.

3. If you have a data set with no binary response variable but you really want to use it for your assignments, you can dichotomize the response variable in a meaningful way. For example, you have a data set with several predictor variables that are potentially associated with the response variable GPA. You can dichotomize the continuous GPA in the following

   bin.gpa = 1 if GPA $<$ 2.75

   bin.gpa = 0 if GPA $>=$ 2.75

   This dichotomization is meaningful since most graduate schools use 2.75 as the admission cut-off GPA.

4. Choose one numerical predictor variable or a **binary** predictor variable to fit a simple logistic regression model in this assignment as I did in the case study. Other variables will be used in subsequent assignments.

To be more specific, you need to provide the following key components in your analysis report.

```
(a) Describe your data set and the variables.

(b) Formulate a practically meaningful analytic question.

(c) Perform exploratory data analysis using a graphical or numerical approach.

(d) Build a simple logistic regression model.

(e) Interpret the regression coefficients from the practical perspective (odds ratio).

(f) study the behavior of the success probability (probability curve and the rate of change in su
```

Chapter 10

Multiple Logistic Regression Model

The general multivariable linear regression model is given below.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \epsilon,$$

where y is the response variable that is assumed to be a random variable and $\epsilon \to N(0, \sigma^2)$. This also implies that

$$E[y] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$$

For a population with binary data, the underlying random variable can only take exactly two values, say $Y = 1$ or $Y = 0$, and $P(Y = 1) = p$, then $E[Y] = 1 \times p + 0 \times (1 - p) = p$.

That is, the success probability is the expected value of the binary random variable. If we mimic the formulation of the linear regression model by setting

The simple linear regression model (also called the log-odds regression model) is also formulated with the mean response $E[Y]$

$$\frac{E[Y]}{1 - E[Y]} = \beta_0 + \beta_1 x.$$

Let $g(t) = t/(1 - t)$ (also called logit function), the simple logistic regression is re-expressed as $g(E[Y]) = \beta_0 + \beta_1 x$.

10.1 Multiple Logistic Regression Model

Let Y be the binary response variable and $\{x_1, x_2, \cdots, x_n\}$ be the set of predictor variables. If Y takes on either 1 or 0, the multiple logistic regression model is then defined as

$$\frac{E[Y]}{1-E[Y]} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$$

The success probability function

$$p(x_1, x_2, \cdots, x_k) = P(Y = 1 | x_1, x_2, \cdots, x_k) = \frac{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k)}{1 + \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k)}$$

If Y takes on character values, R uses chooses the alphabetically higher value to model the above probability. For example, if $Y =$ "disease" or "no.disease", by default, the logistic regression will be defined as

$$p(x_1, x_2, \cdots, x_k) = P(Y = \text{"}no.disease\text{"} | x_1, x_2, \cdots, x_k) = \frac{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k)}{1 + \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k)}$$

Of cause, you can also redefine the factor level of the response variable to model the probability of the desired category.

10.1.1 Data Requirements: Sources, Layout, and Cleansing

The logistic regression models we are discussing require an I.I.D. sample collected from a cross-sectional study design. Auto-correlation between observations is **not** allowed. For longitudinal data that involves auto-correlation, different models can be used to handle the correlation between observations taken from the same subject in the study.

The general data layout for fitting a logistic regression in R has the following form.

10.1.2 Issues of Predictor Variables and Variable Inspection-Transformation

All models have some explicit and implicit assumptions about the predictor variables and structure of the models. Unlike multiple linear regression models in which the diagnostic residual plots reveal some special patterns of potential violations of the model assumptions, in logistic regression modeling, we don't have many diagnostic tools to use. Some pre-processing procedures should be

Table 10.1: Data set layout for multiple logistic regression model

Y	X1	X2		Xk
Y1	X11	X21	...	Xk1
Y2	X12	X22	...	Xk2
...	...	...	...	...
Yn	X1n	X2n	...	Xkn

performed on predictor variables before a logistic regression model is fit to the data.

The variable inspection-transformation is an iterative process, some of the following potential issues of predictor variables may be considered in the inspection-transformation-inspection workflow.

- **Variable Types**: Predictor variables could be numeric, categorical, or a mixture of numeric and categorical.
- **Collinearity**: Predictor variables are assumed to be non-linearly correlated since the multicollinearity causes unstable estimates of the regression coefficients, hence, fails to obtain a valid model. Remedy for collinearity
  - Remove some of the highly correlated independent variables - Variable selection.
  - Perform an analysis designed for highly correlated variables such as principal components analysis or partial least squares regression - variable extraction.
  - Variable centralization.
  - Non-probabilistic variable selection - Regularization.
- **Dummy Variables**: If categorical predictor variables were numerically coded, we have to turn these numerically coded variables into factor variables. For example, the status of a disease could be "severe", "mild", and "disease-free", if a numerical coding: 2 = "severe", 1 = "mild" and 0 = "disease-free", then you need to R function **factor()** to convert the numerically coded disease status to a factor variable.
- **"Fake Variable"**: The observation ID is **NOT** a variable, you should **never** include the observation ID in any of your regression models.
- **Sparse Category Variables**: Group the categories in a meaningful way if necessary. For example, Assume that you have a data set of information about cars of different models from various manufacturers. If you want to build regression on a data set with a relatively small sample size, the use of the car-model as a categorical variable is not appropriate since too many different car models will result in too many dummy variables. From

a mathematical point of view, the number of parameters should be less than the number of data points. However, from the statistical point of view, the desired sample size is 15 times the number of parameters to ensure stable estimates of model parameters.

- **Variable transformation**: - In logistic regression models, the response variable has already been transformed in the form of log odds of "success". The predictor variables could be transformed in different ways for different purposes.
  - **Association Analysis** - a transformation of predictor variables makes the interpretation of the coefficient much more difficult.
  - **Predictive Analysis** - transforming all *numerical variables* to the same scale may improve the performance of predictive models. One of the benefits of standardizing predictor variables is to make variable selection (model regularization) straightforward and interpretable.
- **Variable Discretization** - several methods can be used for the discretization: (1) **empirical approaches** include equally spaced and equal frequency, (2) **model-assisted approaches** include decision tree and k-mean. Discretization is commonly used for different purposes.
  - Model interpretability and understandability - it is easier to understand continuous data (such as age) when divided and stored into meaningful categories or groups. It is commonly used in association analysis.
  - Fixing the potential imbalance issues that could potentially lead to an unstable estimate of the coefficients.

10.1.3 Estimation and Interpretation of Regression Coefficients

As mentioned in the simple logistic regression model, regression coefficients are estimated by using the maximum likelihood approach.

Let $\{(y_1, x_{11}, x_{21}, \cdots, x_{k1}), (y_2, x_{12}, x_{22}, \cdots, x_{k2}), \cdots, (y_n, x_{1n}, x_{2n}, \cdots, x_{kn})\}$ be a random sample taken from a binary population associated with Y. x is a nonrandom predictor variable associated with Y. The logistic model is defined to be

$$p(x) = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_k x_k}}{1+e^{\beta_0+\beta_1 x_1+\cdots+\beta_k x_k}}.$$

The likelihood function of $(\beta_0, \beta_1, \cdots, \cdots, \beta_k)$ is given by

$$L(\beta_0, \beta_1, \cdots, \beta_k) = \prod_{i=1}^{n} \left[\frac{e^{\beta_0+\beta_1 x_{i1}+\cdots+\beta_k x_{ik}}}{1+e^{\beta_0+\beta_1 x_{i1}+\cdots+\beta_k x_{ik}}}\right]^{y_i} \times \left[\frac{1}{1+e^{\beta_0+\beta_1 x_{i1}+\cdots+\beta_k x_{ik}}}\right]^{1-y_i}$$

The maximum likelihood estimate (MLE) of $\beta_0, \beta_1, \cdots, \beta_k$, denoted by $\hat{\beta}_0, \hat{\beta}_1, \cdots, \hat{\beta}_k$, maximizes the above likelihood. The R build-in function **glm()** uses the MLE method to estimate parameters and reports related to MLE-based statistics.

The coefficients are interpreted similarly as used in the simple logistic regression model. To interpret β_j in the multiple logistic regression model,

$$\log\left(\frac{P[Y=1|\cdots,x_j,\cdots]}{1-P[Y=1|\cdots,x_j,\cdots]}\right) = \beta_0+\beta_1x_1+\cdots+\beta_{j-1}x_{j-1}+\beta_jx_j+\beta_{j+1}+\cdots+\beta_kx_k$$

If we **fix the values of all X_i except for increasing x_j by one unit**, then

$$\log\left(\frac{P[Y=1|\cdots,(x_j+1),\cdots]}{1-P[Y=1|\cdots,(x_j+1),\cdots]}\right) = \beta_0+\beta_1x_1+\cdots+\beta_{j-1}x_{j-1}+\beta_j(x_j+1)+\beta_{j+1}x_{j+1}+\cdots+\beta_kx_k$$

Then

$$\beta_j = \log\left(\frac{P[Y=1|\cdots,(x_j+1),\cdots]}{1-P[Y=1|\cdots,(x_j+1),\cdots]}\right)-\log\left(\frac{P[Y=1|\cdots,x_j,\cdots]}{1-P[Y=1|\cdots,x_j,\cdots]}\right) = \log\left(\frac{\frac{P[Y=1|\cdots,(x_j+1),\cdots]}{1-P[Y=1|\cdots,(x_j+1),\cdots]}}{\frac{P[Y=1|\cdots,x_j,\cdots]}{1-P[Y=1|\cdots,x_j,\cdots]}}\right)$$

Therefore, β_j (for $j = 1, 2, \cdots, k$) is the log odds ratio as explained in the simple logistic regression model.

If $\beta_j = 0$, then x_j is insignificant meaning that the odds of "success" in a subset of subjects with predictor values $\{x_1, \cdots, x_{j-1}, x_{j+1}, x_{j+1}, \cdots, x_k\}$ is equal to the odds of "success" in other subsets with predictor values $\{x_1, \cdots, x_{j-1}, x_j, x_{j+1}, \cdots, x_k\}$.

10.2 Building Blocks for Predictive Performance

Prediction in the logistic regression is not straightforward. The logistic regression function

$$p(x) = \frac{e^{\beta_0+\beta_1x_1+\cdots+\beta_kx_k}}{1+e^{\beta_0+\beta_1x_1+\cdots+\beta_kx_k}}.$$

predicts the probability of "success" but not the status of "success". In order to predict the value of Y, we still need to have a cut-off probability to define the

predicted "success" and "failure". How to find the optimal cut-off probability will be addressed later module.

Next, we assume there is a cut-off probability for predicting the original value of the response variable. Most software programs as R and SAS use 0.5 as the cut-off for predicting Y.

10.2.1 Understanding the Performance of Medical Diagnostics

In this section, we define some performance metrics of the logistic regression as a predictive model based on the predictive error. For ease of illustration, we consider a simple model using a diagnostic test result (X = T+ or T-) to predict a disease (Y = D+ or D-) to define these performance metrics where

T+ = positive test result: numerical coding 1

T- = negative test result: numerical coding 0

D+ = diseased: numerical coding 1

D- = disease-free: numerical coding 0

The following metrics measure the predictive performance of the logistic regression model. The first two measures reflect the correct decision of the model and the last two error rates of the logistic regression model.

- **Positive Predictive Value**: P(Y = D+ | X = T+)

  $PPV = P(Y = 1|X = 1) = \frac{e^{\beta_0+\beta_1}}{1+e^{\beta_0+\beta_1}}$

- **Negative Predictive Value**: P(Y = D- | X = T-)

  $NPV = P(Y = 1|X = 0) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$

- **False Positive Predictive Rate**: P(Y = D- | X = T+)

  $FPPV = P(Y = 0|X = 1) = \frac{1}{1+e^{\beta_0+\beta_1}}$

- **False Positive Predictive Rate**: P(Y = D+ | X = T-)

  $FNPV = P(Y = 0|X = 0) = \frac{1}{1+e^{\beta_0}}$

The above four conditional probabilities can also be estimated by calculating the corresponding relative frequencies from the following two-way contingency table - also called **confusion matrix**. For convenience, we call the above four metrics **prediction performance metrics**.

```
D1 = c("n11", "n12")
D0 = c("n21", "n22")
M=as.data.frame(rbind(D1, D0))
names(M)=c("T+", "T-")
```

```
row.names(M) = c("D+", "D-")
kable(M)
```

	T+	T-
D+	n11	n12
D-	n21	n22

The above four metrics are used by clinical diagnosis after the test was approved by the FDA since the diagnostic decision is based on the test result.

10.2.2 Performance Metrics Used in Clinical Trials:

Now, let's consider the case that a manufacturer conducting a clinical phase II trial and submitting the results for FDA approval. The FDA uses the following metrics in the approval process.

- **Sensitivity**: P(T+ | D+)
- **Specificity**: P(T- | D-)
- **False Negative Rate**: P(T- | D+)
- **False Positive Rate**: P(T+ | D-)

The above metrics are well-defined since the disease status of subjects is known in the clinical trial. The estimated values of these metrics can be found in the clinical data. For convenience, we call the above four metrics **Validation Performance Metrics**.

10.2.3 Remarks

Here are several remarks on the above two sets of performance metrics.

- The **prediction performance metrics** are dependent on the choice of the cut-off "success" probability. They can be estimated from a fitted logistic regression model.
- The **validation performance metrics** are defined based on the data with known disease status. A proposed diagnostic test is good if both sensitivity and specificity are high.
- Thinking about the logistic regression model you developed as "a diagnostic test" (since it can predict the status of a disease), which sets of metrics you should use to show the goodness of your model? The answer is the set of **validation performance metrics**.
- **Sensitivity and Specificity** are the basic building blocks used to define various performance metrics to assess the goodness of the predictive model using the **testing data** with known response values. This will be one of the major topics in the next module.

10.3 Case Study

In this case study, we still use the diabetes data that was used in the last module.

10.4 Data and Variable Descriptions

There are 9 variables in the data set.

1. **pregnant**: Number of times pregnant
2. **glucose**: Plasma glucose concentration (glucose tolerance test)
3. **pressure**: Diastolic blood pressure (mm Hg)
4. **triceps**: Triceps skin fold thickness (mm)
5. **insulin**: 2-Hour serum insulin (mu U/ml)
6. **mass**: Body mass index (weight in kg/(height in m)^2)
7. **pedigree**: Diabetes pedigree function
8. **age**: Age (years)
9. **diabetes**: Class variable (test for diabetes)

I load the data from R **library{mlbench}** in the following code. For convenience, I delete all records with missing values and keep only the records with complete records in this case study. The final analytic data set has 392 records. This

```
library(mlbench)
data(PimaIndiansDiabetes2)           # load the data to R work-space
diabetes.0 = PimaIndiansDiabetes2    # make a copy of the data for data cleansing
diabetes = na.omit(diabetes.0)       # Delete all records with missing components
#head(diabetes)
```

10.4.1 Research Question

The objective of this case study is to identify the risk factors for diabetes.

10.4.2 Exploratory Analysis

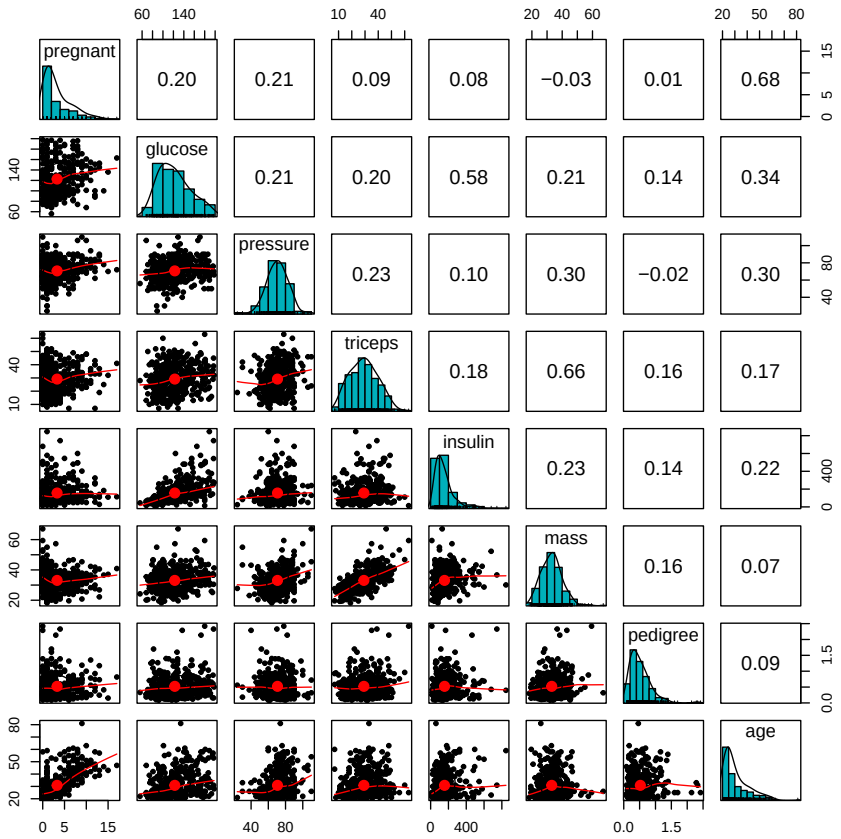
We first make the following pairwise scatter plots to inspect the potential issues with predictor variables.

```
library(psych)
```

```
##
##    'psych'
```

```
## The following objects are masked from 'package:scales':
##
##     alpha, rescale
```

```
pairs.panels(diabetes[,-9],
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = TRUE,  # show density plots
             ellipses = TRUE # show correlation ellipses
             )
```



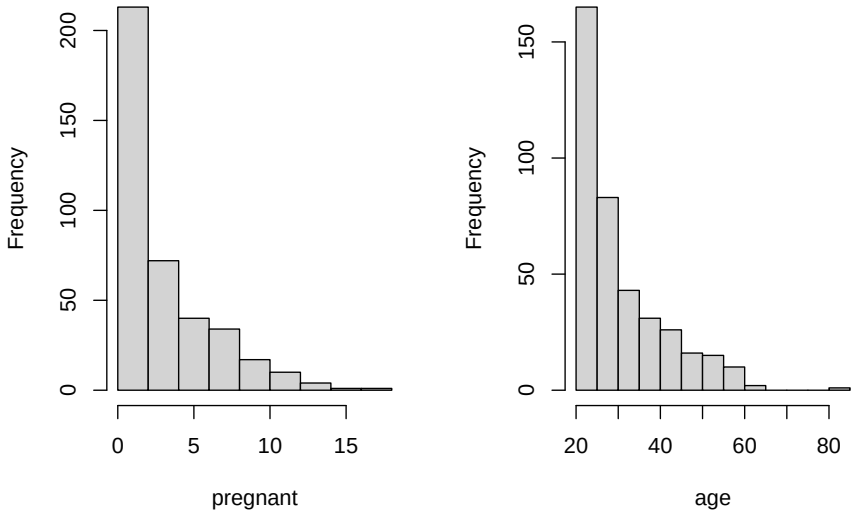

```
detach("package:psych", unload = TRUE)
```

From the correlation matrix plot, we can see several patterns in the predictor variables.

- All predictor variables are unimodal. But **pregnant** and **age** are signifi-

cantly skewed. We next take a close look at the frequency distribution of these two variables.

```
par(mfrow=c(1,2))
hist(diabetes$pregnant, xlab="pregnant", main = "")
hist(diabetes$age, xlab = "age", main = "")
```



Based on the above histogram, we discretize **pregnant** and **age** in the following.

```
preg = diabetes$pregnant
grp.preg = preg
grp.preg[preg %in% c(4:6)] = "4-6"
grp.preg[preg %in% c(7:9)] = "7-9"
grp.preg[preg %in% c(10:17)] = "10+"
##
age = diabetes$age
##
grp.age = age
grp.age[age %in% c(21:24)] = "21-25"
grp.age[age %in% c(25:30)] = "25-30"
grp.age[age %in% c(31:40)] = "31-40"
grp.age[age %in% c(41:50)] = "41-50"
grp.age[age %in% c(51:99)] = "50 +"
## added to the diabetes data set
diabetes$grp.age = grp.age
```

```
diabetes$grp.preg = grp.preg
```

- A moderate correlation is observed in several pairs of variables: **age** v.s. **pregnant**, **glucose** v.s. **insulin**, and **triceps** v.s. **mass**. We will not drop any of these variables for the moment but will perform an automatic variable selection process to remove potential redundant variables since a few of them will be forced to be included in the final model.
- Since our goal is association analysis, we will not perform variable transformations for the time being.
- It is very common in real-world applications that some of the practically important variables are always included in the final model regardless of their statistical significance. In the diabetes study, three insulin, MBI, and pedigree are considered significant risk factors. We will include these three variables in the final model. This means the smallest model must have these three variables.

10.4.3 Building the Multiple Logistic Regression Model

Based on the above exploratory analysis, we first build the full model and the smallest model.

```
full.model = glm(diabetes ~grp.preg+glucose+pressure+triceps+insulin+mass+pedigree+grp.age,
          family = binomial(link = "logit"),  #  logit(p) = log(p/(1-p))!
          data = diabetes)
kable(summary(full.model)$coef,
      caption="Summary of inferential statistics of the full model")
```

```
reduced.model = glm(diabetes ~ insulin + mass + pedigree,
          family = binomial(link = "logit"),  # logit(p) = log(p/(1-p))!
          data = diabetes)
kable(summary(reduced.model)$coef,
      caption="Summary of inferential statistics of the reduced model")
```

```
## automatic variable selection
library(MASS)
final.model.forward = stepAIC(reduced.model,
                      scope = list(lower=formula(reduced.model),upper=formula(full.model)),
                      direction = "forward",   # forward selection
                      trace = 0   # do not show the details
                      )
kable(summary(final.model.forward)$coef,
      caption="Summary of inferential statistics of the final model")
```

```
## Other global goodness-of-fit
global.measure=function(s.logit){
dev.resid = s.logit$deviance
```

Table 10.2: Summary of inferential statistics of the full model

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-8.7503322	1.3331796	-6.5635058	0.0000000
grp.preg1	-0.3977538	0.5093087	-0.7809679	0.4348214
grp.preg10+	0.3935217	0.7902561	0.4979673	0.6185071
grp.preg2	-0.2958769	0.5571478	-0.5310564	0.5953797
grp.preg3	0.3842810	0.5700718	0.6740923	0.5002526
grp.preg4-6	-0.9140847	0.5668123	-1.6126761	0.1068149
grp.preg7-9	-0.2284477	0.6514907	-0.3506538	0.7258481
glucose	0.0382521	0.0060502	6.3224381	0.0000000
pressure	-0.0072937	0.0121999	-0.5978495	0.5499403
triceps	0.0107744	0.0180674	0.5963470	0.5509435
insulin	-0.0002279	0.0013854	-0.1645109	0.8693290
mass	0.0583354	0.0284914	2.0474744	0.0406115
pedigree	1.0412541	0.4461747	2.3337362	0.0196095
grp.age25-30	1.0829606	0.4193260	2.5826224	0.0098053
grp.age31-40	1.5405843	0.4909205	3.1381545	0.0017002
grp.age41-50	2.2601500	0.6144945	3.6780639	0.0002350
grp.age50 +	2.0014940	0.7174832	2.7896040	0.0052773

Table 10.3: Summary of inferential statistics of the reduced model

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-4.3288188	0.6463940	-6.696874	0.0000000
insulin	0.0047984	0.0010835	4.428585	0.0000095
mass	0.0673268	0.0178494	3.771943	0.0001620
pedigree	1.0779952	0.3601870	2.992876	0.0027636

Table 10.4: Summary of inferential statistics of the final model

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-9.5340546	1.0941037	-8.7140321	0.0000000
insulin	-0.0005617	0.0013568	-0.4140043	0.6788710
mass	0.0703082	0.0212351	3.3109430	0.0009298
pedigree	1.0567094	0.4305469	2.4543423	0.0141143
glucose	0.0387018	0.0058814	6.5804090	0.0000000
grp.age25-30	1.0012238	0.3918271	2.5552695	0.0106106
grp.age31-40	1.3679308	0.4163693	3.2853782	0.0010185
grp.age41-50	2.1962925	0.4712387	4.6606796	0.0000032
grp.age50 +	1.9277491	0.5802575	3.3222305	0.0008930

Table 10.5: Comparison of global goodness-of-fit statistics

	Deviance.residual	Null.Deviance.Residual	AIC
full.model	324.9106	498.0978	358.9106
reduced.model	434.7276	498.0978	442.7276
final.model	334.2005	498.0978	352.2005

```
dev.0.resid = s.logit$null.deviance
aic = s.logit$aic
goodness = cbind(Deviance.residual =dev.resid, Null.Deviance.Residual = dev.0.resid,
      AIC = aic)
goodness
}
goodness=rbind(full.model = global.measure(full.model),
      reduced.model=global.measure(reduced.model),
      final.model=global.measure(final.model.forward))
row.names(goodness) = c("full.model", "reduced.model", "final.model")
kable(goodness, caption ="Comparison of global goodness-of-fit statistics")
```

10.4.4 Final Model

In the exploratory analysis, we observed three pairs of variables are linearly correlated. After automatic variable selection, triceps and age were dropped out from the final model. Both insulin and glucose are still in the model. Although insulin is statistically insignificant, we still include it in the model since it is clinically important.

```
# Odds ratio
model.coef.stats = summary(final.model.forward)$coef
odds.ratio = exp(coef(final.model.forward))
out.stats = cbind(model.coef.stats, odds.ratio = odds.ratio)
kable(out.stats,caption = "Summary Stats with Odds Ratios")
```

The interpretation of the odds ratios is similar to the case of simple logistic regression. The group-age variable **grp.age** has five categories. The baseline category is aged 21-24. We can see from the above table inferential table that the odds of getting diabetes increase as age increases. For example, the odds ratio associated with the age group 31-39 is 3.927 meaning that, given the same level of insulin, BMI, pedigree, and glucose, the odds of being diabetic in the age group of 31-40 is almost 4 times of that in the baseline group aged 21-24. But the same ratio becomes nine times when comparing the age group 41-50 with the baseline group of age 21-24.

Table 10.6: Summary Stats with Odds Ratios

	Estimate	Std. Error	z value	Pr(>|z|)	odds.ratio
(Intercept)	-9.5340546	1.0941037	-8.7140321	0.0000000	0.0000723
insulin	-0.0005617	0.0013568	-0.4140043	0.6788710	0.9994384
mass	0.0703082	0.0212351	3.3109430	0.0009298	1.0728388
pedigree	1.0567094	0.4305469	2.4543423	0.0141143	2.8768887
glucose	0.0387018	0.0058814	6.5804090	0.0000000	1.0394605
grp.age25-30	1.0012238	0.3918271	2.5552695	0.0106106	2.7216105
grp.age31-40	1.3679308	0.4163693	3.2853782	0.0010185	3.9272159
grp.age41-50	2.1962925	0.4712387	4.6606796	0.0000032	8.9916151
grp.age50 +	1.9277491	0.5802575	3.3222305	0.0008930	6.8740202

10.4.5 Summary and Conclusion

The case study focused on the association analysis between a set of potential risk factors for diabetes. The initial data set has 8 numerical and categorical variables.

After exploratory analysis, we decide to re-group two sparse discrete variables **pregnant** and **age**, and then define dummy variables for the associated variables. These new group variables were used in the model search process.

Since **insulin**, **BMI**, and **pedigree** are considered to be major contributors to the development of diabetes, we include three risk factors in the final model regardless of the statistical significance.

After automatic variable selection, we obtain the final model with 4 factors, BMI, pedigree, glucose, age (with 4 dummy variables), and insulin (that is not statistically significant but clinically important).

Diabetes prediction or classification is another important practical issue. We will address this practical question in the next module.

10.5 Analysis Assignment

This assignment focuses on multiple logistic regression modeling using the same data set you used in the previous week. To be more specific, the data set has to meet the following requirements:

- The response variable must be binary. It could be made by dichotomizing a numerical variable or regrouping a categorical variable.
- At least two continuous predictor variables
- At least two categorical predictor variables

- The sample size should be at least 15 times the total number of numerical variables and *dummy* variables.

Components of the analysis report

The report should contain the same components as I included in the case study in this week's class note. Please keep in mind that the interpretation of results is VERY important.

- Description of your data set and variables
- Research questions
- Data management and variable inspection
  - variable creation based on existing variables
  - variable transformation
  - variable discretization
  - handling sparse categorical variables
- model building process
  - candidate models
  - manual variable selection
  - automatic variable selection
  - final model identification
  - summarize the inferential statistics in the final model.
- Conclusion and discussion

Remarks:

1. This assignment focuses only on the association analysis.
2. Convert the regression coefficients in the final to odds ratio and then provide practical interpretation.
3. The global goodness-of-fit measures (deviance, AIC, etc) in all candidate models should be reported during model selection.

Chapter 11

Predictive Modeling with Logistic Regression

The logistic regression model as a member of the family of the generalized linear regression model has the following form.

Let Y be the binary response variable and $\{x_1, x_2, \cdots, x_n\}$ be the set of predictor variables. If Y takes on either 1 or 0, the multiple logistic regression model is then defined as

$$\frac{E[Y]}{1 - E[Y]} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$$

The success probability function

$$p(x_1, x_2, \cdots, x_k) = P(Y = 1 | x_1, x_2, \cdots, x_k) = \frac{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k)}{1 + \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k)}$$

If Y takes on character values, R uses chooses the alphabetically higher value to model the above probability. For example, if $Y =$ "diseased" or "disease-free", by default, the logistic regression will be defined as

$$p(x_1, x_2, \cdots, x_k) = P(Y = \text{disease-free} | x_1, x_2, \cdots, x_k) = \frac{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k)}{1 + \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k)}$$

We can also redefine the factor level of the response variable to model the probability of the desired category.

In the previous module, we introduced the strategies for searching the final model using both manual and automatic variable selection approaches using likelihood-based performance metrics. However, in predictive modeling, the classical model search strategies might not work satisfactorily. It is a common practice to use data-driven and algorithm-based approaches to assess the predictive performance of a predictive model or an algorithm.

In the following sections, we assume that a set of several candidate models/algorithms have already been developed. The candidate models and algorithms include logistic regression models and other algorithms such as tree-based algorithms, neural nets, support vector machines, etc. The objective is to use various methods to choose the one with the best predictive power to implement in real-world applications.

11.1 Cross-Validation in Predictive Modeling

In classical statistical modeling, we evaluate the predictive performance of a model using the large sample theory developed based on the likelihood. However, in algorithm-based predictive modeling, we don't assume the distribution of the population or the underlying population is uncertain. In other words, there is no general theory that can derive the predictive performance. Therefore, data-driven methods are used to define the goodness of the model. The key is to hold up a portion of the sample as "unseen" observations to test the actual performance of the predictive models and algorithms.

11.1.1 The Logic of the Three-way Split

The general idea is to randomly partition the data into several subsets for different purposes. One of them will be used to build the model, one is used to validate the model, and one is used as "unseen real-world" data to report the "actual" performance of the final model.

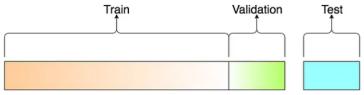


Figure 11.1: Three-way splitting mechanism

With this three-way split, the **model selection** and the **true error rate computation** can be carried out simultaneously. One important observation is that the error rate estimate of the final model on validation data is, in general, underestimated since the validation set is used to select the final model. Therefore, a third independent part of the data, the **test data**, is needed to report the actual prediction performance.

After the performance of the final model was evaluated on the test

set, all estimated parameters of the model based on the training data *must not* be changed any further.

The three-way splitting method may face practical challenges such as the insufficiency of the data and variation of the error estimate based on a single-step process of model development. To overcome these potential challenges, researchers and practitioners developed resampling-based cross-validation (CV) methods to combine training, validating, and testing in an iterative process. The following subsection outlines this

11.1.2 Cross-validation (CV)

Among the methods available for estimating prediction error, the most widely used is cross-validation. Contrary to most people who thought that CV was developed by machine learning researchers, it was developed by statisticians. The idea was initially formulated in the 1930s and formally published in statistics literature in the 1970s. Maybe due to computational constraints, the cross-validation methods have not been used by the statistics community. Recently, machine learning researchers dug this old but gold algorithm out for assessing the performance of predictive modeling and algorithms.

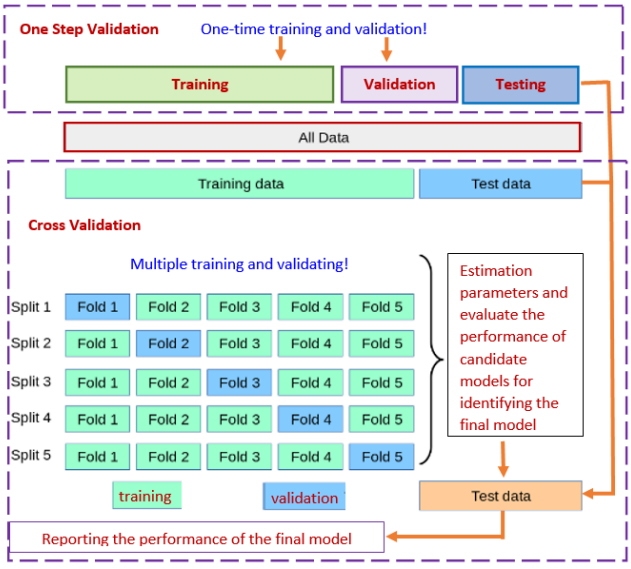


Figure 11.2: Three-way data splitting versus cross-validation

Essentially cross-validation includes techniques to split the sample into multiple training and test data sets. To obtain multiple training and testing subsets, we use various random sub-sampling methods to perform K data splits of the entire sample. Different random splits constitute different cross-validation methods. The most commonly used cross-validation method is k-fold cross-validation.

The following are steps for choosing the final working model from a set of candidate models using the cross-validation method.

- **Step 1**: Split the entire data into a training set and a testing set.
- **Step 2**: Perform the cross-validation in the training set and don't touch the test data.
  - *step 2.1.* Randomly split the training set into k subsets with equal size, say,F_1, F_2, $\cdots$, F_k.
  - *step 2.2.* hold up F_i and combine $T_i = \{\cdots, F_{i-1}, F_{i+1}, \cdots\}$. Fit all candidate models to T_i and then use the fitted model to predict the response using the testing set F_i, for $i = 1, 2, \cdots, k$.
  - *step 2.3.* Since the actual response values are available in the test set F_i, we can compare the predicted response value with the true response values to calculate the predictive error for each candidate model in each of the k-rounds.
- **Step 3**: Calculate the average of the predictive errors for each candidate model. The one with the smallest average predictive error is the winner.
- **Step 4**: Fit the winner to the entire training set $\{F_1, F_2, \cdots, F_k\}$ to obtain the final working model.
- **Step 5**: Report the *actual* predictive error using the testing data set that has not been used in the above cross-validation process.

The above process is summarized in the following figure.

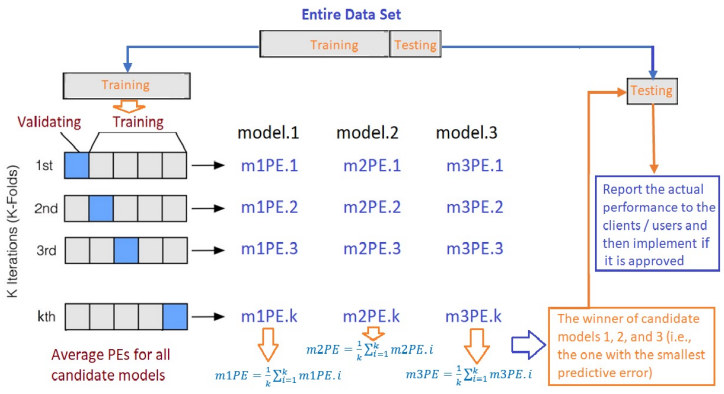


Figure 11.3: The workflow of the cross-validation process

Remarks on the cross-validation.

1. The proportions of the training and testing sets are determined by modelers. 75%:25% is a commonly used ratio.

2. The number of folds used in the cross-validation is the **hyper-parameter** or **tuning-parameter**. The 10-fold cross-validation is commonly used. The reason 10-fold is commonly used is that when $k \geq 10$, the predictive errors in the training and testing become stable.

11.2 Predictive Performance Measures

Most of the predictive performance measures are defined based on the predictive errors in logistic modeling.

11.2.1 Error-based Measures

We introduced two sets of conditional probabilities using clinical language in the previous module. As an analogy, we can think about the logistic predictive model to be a **diagnostics test** for predicting a specific disease. Note that what the logistic model predicted is the probability of having a **disease**. In order to predict the value (T+ = "predicted disease" or T- = "predicted no-disease") of the binary response, we need to use a cut-off probability to define "T+" and "T-". In most software programs, 0.5 was used. If the success probability is greater than 0.5, the predicted value of the response will be "T+", otherwise, the predicted response value will be "T-".

Since each of the hold-up validating set has the true values of the response, we then can use corresponding predicted values based on a selected cut-off probability to make a two-way table in the following form

	Actual pos	Actual neg
Predict pos	a (TP)	b (FP)
Predict neg	c (FN)	d (TN)

The above two-way contingency table is called the **confusion matrix** which can be used to estimate the following four performance measures introduced in the previous module.

- **Sensitivity**:

$$TPR = P(\widehat{T+|}D+) = TP/(TP + FN)$$

- **Specificity**:

$$TNR = P(\widehat{T-|}D-) = TN/(TN + FP)$$

- **False Negative Rate**:

$$FNR = P(\widehat{T-|}D+) = FN/(TP + FN)$$

- **False Positive Rate**:

$$\widehat{P(T+|D-)} = FP/(FP+TN)$$

Ideally, a good binary predictive model such as the logistic regression model should have a high sensitivity and specificity and a low false negative rate and false-positive rate (FPR, also called false alarm rate).

The overall accuracy is defined by

$$\text{accuracy} = (TP+TN)/(TP+TN+FP+FN)$$

and the predictive error (PE) is defined by

$$PE = (FP+FN)/(TP+FP+TN+FN).$$

The above PE was used in the iterations of the process of cross-validation for a binary predictive model. In the logistic regression model, the confusion matrix is dependent on the choice of the cut-off probability. In other words, the values of the above 6 measures are dependent on the choice of the cut-off probability. Therefore, the above performance metrics are called **local performance** measures.

Remarks

1. The default choice of 0.5 in most software programs may not be the optimal choice.
2. The "disease" probability is a continuous variable on [0,1], we can search the optimal cut-off probability by plotting the cut-off probability against the accuracy and then find the optimal cut-off from the plot.

11.3 Measuring Global Performance - ROC Curve

In many practical applications, we may have different candidate models that behaved differently from different perspectives. For example, (1). how to report the performance of a medical diagnostic test if it is dependent on the age of the patient? (2). If there are two tests and both are dependent on the age of patients, how to compare the performance of the two tests? (3). sometimes the costs of false positive and false negative rates are very different in a specific application, how to minimize the cost due to errors? These questions can be addressed using the Receiver Operating Characteristics (ROC) analysis.

11.3.1 ROC Curve

An ROC curve is the plot of sensitivity (i.e., TPR = True Positive Rate) against (1-specificity) = False Positive Rate (FPR). Drawing the ROC curve is straightforward. We only need to choose a sequence of cut-off probability and then calculate the corresponding TPR and FPR. Then we can plot these points to get the ROC curve which is similar to the following figure.

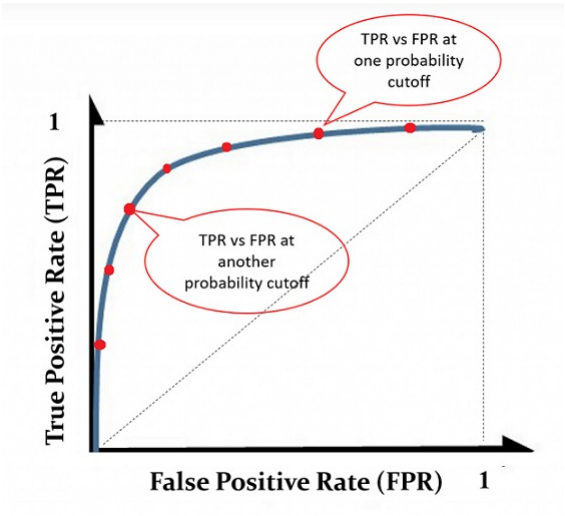


Figure 11.4: An illustrative ROC curve

Using the same set of cut-off probabilities, different candidate models have different ROC curves. The area under the curve (AUC) reflects the **global goodness of the model**. See the following illustrative curve.

With the ROC curve, we can answer the questions in the opening paragraph of this section.

11.3.2 Area Under The Curve (AUC)

If two or more ROC curves intersect at least one point, we may want to report the area under the curves (AUC) to compare the global performance between the two corresponding models. See the illustrative example below.

There are several R packages that have functions to calculate the AUC. We are going use R functions `roc()` and `auc()` in package `pROC` to calculate the AUC in the case study.

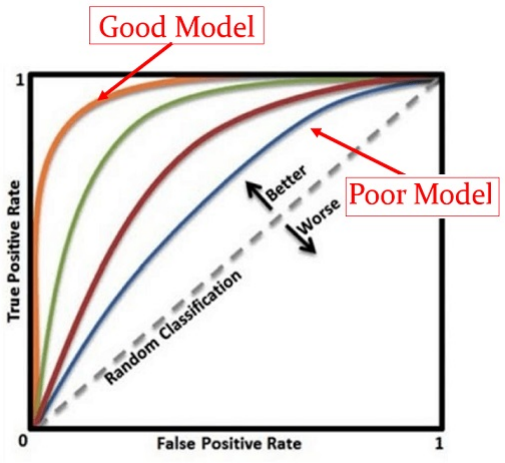


Figure 11.5: Illustrative comparison of multiple ROC curves associated with the corresponding candidate models

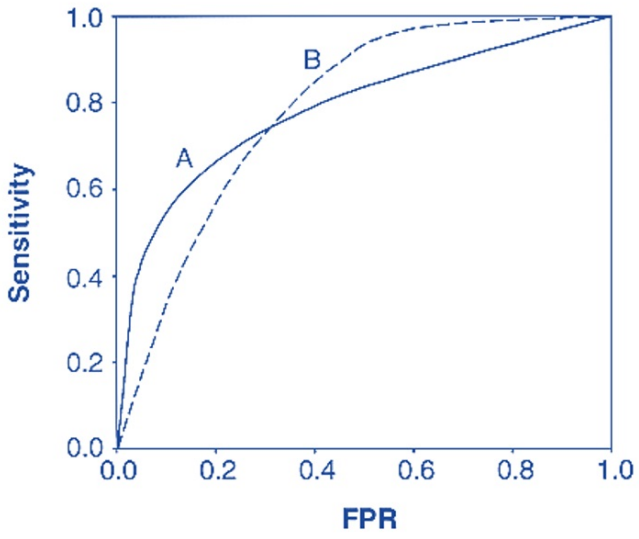


Figure 11.6: Using ROC for model selection.

11.4 Case Study - Diabetes Prediction with Logistic Regression

In this case study, we still use the diabetes data that was used in the last module.

11.4.1 Data and Variable Descriptions

There are 9 variables in the data set.

1. **pregnant**: Number of times pregnant
2. **glucose**: Plasma glucose concentration (glucose tolerance test)
3. **pressure**: Diastolic blood pressure (mm Hg)
4. **triceps**: Triceps skinfold thickness (mm)
5. **insulin**: 2-Hour serum insulin (mu U/ml)
6. **mass**: Body mass index (weight in kg/(height in m)^2)
7. **pedigree**: Diabetes pedigree function
8. **age**: Age (years)
9. **diabetes**: Class variable (test for diabetes)

I load the data from R **library{mlbench}** in the following code. For convenience, I delete all records with missing values and keep only the records with complete records in this case study. The final analytic data set has 392 records. This

```
library(mlbench)
data(PimaIndiansDiabetes2)           # load the data to R work-space
diabetes.0 = PimaIndiansDiabetes2    # make a copy of the data for data cleansing
diabetes = na.omit(diabetes.0)       # Delete all records with missing components
#head(diabetes)
```

11.4.2 Research Question

The objective of this case study is to build a logistic regression model to predict diabetes using various risk factors associated with the individual patient.

11.4.3 Exploratory Analysis

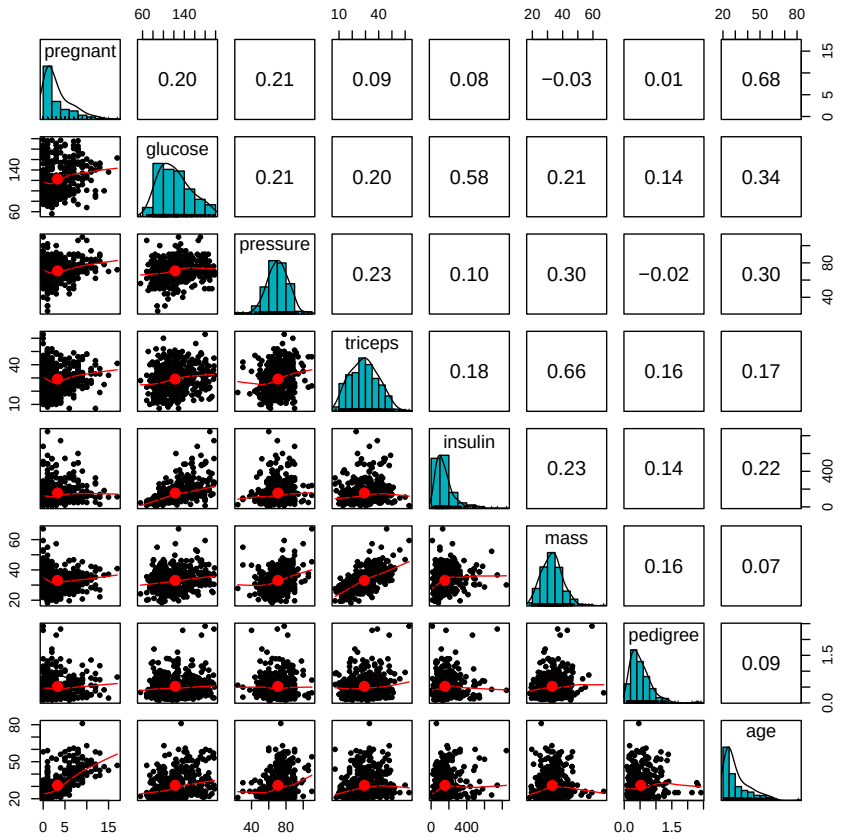
We first make the following pairwise scatter plots to inspect the potential issues with predictor variables.

```
library(psych)
```

```
##
##     'psych'
```

```
## The following objects are masked from 'package:scales':
##
##     alpha, rescale
```

```
pairs.panels(diabetes[,-9],
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = TRUE,  # show density plots
             ellipses = TRUE # show correlation ellipses
             )
```




```
detach("package:psych", unload = TRUE)
```

From the correlation matrix plot, we can see several patterns in the predictor variables.

- All predictor variables are unimodal. But **pregnant** and **age** are signifi-

cantly skewed. we discretize **pregnant** and **age** in the following.

```
preg = diabetes$pregnant
grp.preg = preg
grp.preg[preg %in% c(4:6)] = "4-6"
grp.preg[preg %in% c(7:9)] = "7-9"
grp.preg[preg %in% c(10:17)] = "10+"
##
age = diabetes$age
##
grp.age = age
grp.age[age %in% c(21:24)] = "21-25"
grp.age[age %in% c(25:30)] = "25-30"
grp.age[age %in% c(31:40)] = "31-40"
grp.age[age %in% c(41:50)] = "41-50"
grp.age[age %in% c(51:99)] = "50 +"
## added to the diabetes data set
diabetes$grp.age = grp.age
diabetes$grp.preg = grp.preg
```

A moderate correlation is observed in several pairs of variables: **age** v.s. **pregnant**, **glucose** v.s. **insulin**, and **triceps** v.s. **mass**. We will not drop any of these variables. We will standardize these variables. Part of the correlation may be removed after they are standardized.

11.4.4 Standizing Numerical Predictor Variables

Since this is a predictive model, we don't worry about the interpretation of the coefficients. The objective is to identify a model that has the best predictive performance.

```
## standardizing numerical variables
diabetes$sd.glucose = (diabetes$glucose-mean(diabetes$glucose))/sd(diabetes$glucose)
diabetes$sd.pressure = (diabetes$pressure-mean(diabetes$pressure))/sd(diabetes$pressure)
diabetes$sd.triceps = (diabetes$triceps-mean(diabetes$triceps))/sd(diabetes$triceps)
diabetes$sd.insulin = (diabetes$insulin-mean(diabetes$insulin))/sd(diabetes$insulin)
diabetes$sd.mass = (diabetes$mass-mean(diabetes$mass))/sd(diabetes$mass)
diabetes$sd.pedigree = (diabetes$pedigree-mean(diabetes$pedigree))/sd(diabetes$pedigree)
## drop the original variables except for the response variable
sd.diabetes = diabetes[, -(1:8)]
```

Remark: If the final model is used for real-time prediction, we need to write a separate function to perform all variable transformations and modifications before feeding it to the model for prediction.

11.4.5 Data Split - Training and Testing Data

We **randomly** split the data into two subsets. 70% of the data will be used as training data. We will use the training data to search the candidate models, validate them and identify the final model using the cross-validation method. The 30% of the hold-up sample will be used for assessing the performance of the final model.

```
## splitting data: 80% training and 20% testing
n <- dim(sd.diabetes)[1]
train.n <- round(0.8*n)
train.id <- sample(1:n, train.n, replace = FALSE)
## training and testing data sets
train <- sd.diabetes[train.id, ]
test <- sd.diabetes[-train.id, ]
```

11.4.6 Best Model Identification

In the previous module, we introduced full and reduced models to set up the scope for searching for the final model. In this case study, we use the full, reduced, and final models obtained based on the step-wise variable selection as the three candidate models.

For illustration, we use 0.5 as the common cut-off for all three models to define the predicted. In a real application, **we may want to find the optimal cut-off for each candidate model in the cross-validation process**.

11.4.6.1 Cross-Validation for Model Identification

Since our training data is relatively small, I will use 5-fold cross-validation to ensure the validation data set has enough diabetes cases.

```
library(MASS)
library(knitr)
## 5-fold cross-validation
k=5
## floor() function must be used to avoid producing NA in the subsequent results
fold.size = floor(dim(train)[1]/k)
## PE vectors for candidate models
PE1 = rep(0,5)
PE2 = rep(0,5)
PE3 = rep(0,5)
for(i in 1:k){
  ## Training and testing folds
  valid.id = (fold.size*(i-1)+1):(fold.size*i)
  valid = train[valid.id, ]
  train.dat = train[-valid.id,]
  ##  full model
```

Table 11.2: Average of prediction errors of candidate models

PE1	PE2	PE3
0.7870968	0.7870968	0.7870968

```
  candidate01 = glm(diabetes ~grp.preg + sd.glucose +sd.pressure+ sd.triceps + sd.insulin +
                    sd.mass + sd.pedigree + grp.age, family = binomial(link = "logit"),
                    data = train.dat)
## reduced model
  candidate03 = glm(diabetes ~ sd.insulin + sd.mass + sd.pedigree,
                    family = binomial(link = "logit"),
                    data = train.dat)
##
   candidate02 = stepAIC(candidate01,
                      scope = list(lower=formula(candidate03),upper=formula(candidate01)),
                      direction = "forward",    # forward selection
                      trace = 0                 # do not show the details
                      )
  ##  predicted probabilities of each candidate model
   pred01 = predict(candidate01, newdata = valid, type="response")
   pred02 = predict(candidate02, newdata = valid, type="response")
   pred03 = predict(candidate03, newdata = valid, type="response")

   pre.outcome01 = ifelse(as.vector(pred01) > 0.5, "pos", "neg")
   pre.outcome02 = ifelse(as.vector(pred02) > 0.5, "pos", "neg")
   pre.outcome03 = ifelse(as.vector(pred03) > 0.5, "pos", "neg")

   PE1[i] = sum(pre.outcome01 == valid$diabetes )/length(pred01)
   PE2[i] = sum(pre.outcome02 == valid$diabetes )/length(pred02)
   PE3[i] = sum(pre.outcome02 == valid$diabetes )/length(pred03)
}
avg.pe = cbind(PE1 = mean(PE1), PE2 = mean(PE2), PE3 = mean(PE3))
kable(avg.pe, caption = "Average of prediction errors of candidate models")
```

The average predictive errors show that candidate models 1 and 2 have the same predictive error. Since model 2 is simpler than model 1, we choose model 2 as the final predictive model. **This selection of the final model is based on the cut-off probability 0.5**.

11.4.6.2 Final Model Reporting

The previous cross-validation procedure identified the best model with pre-selected cut-off 0.5. The actual accuracy to be report to the client MUST be based on the withheld **test data**. Therefore, the actual accuracy of the final

Table 11.3: The actual accuracy of the final model

x
0.7435897

model is given by

```
pred02 = predict(candidate02, newdata = test, type="response")
pred02.outcome = ifelse(as.vector(pred02)>0.5, "pos", "neg")

accuracy = sum(pred02.outcome == test$diabetes)/length(pred02)
kable(accuracy, caption="The actual accuracy of the final model")
```

Therefore, the final model has an accuracy rate given in the above table.

Remark: Since we used a random split method to define the training and testing data when re-running the code, the performance metrics will be slightly different.

11.4.7 ROC Analysis - Global Performance

The ROC curve as a visual tool is used to compare the **global performance** of binary predictive models. Since it is used for the purpose of model selection, we need to construct ROC curves based on the training data. Next, we will construct ROC curves and calculate the corresponding AUCs using package `pROC`.

We first estimate the TPR (true positive rate, sensitivity) and FPR (false positive rate, 1 - specificity) at each cut-off probability for each of the three candidate models using the following R function.

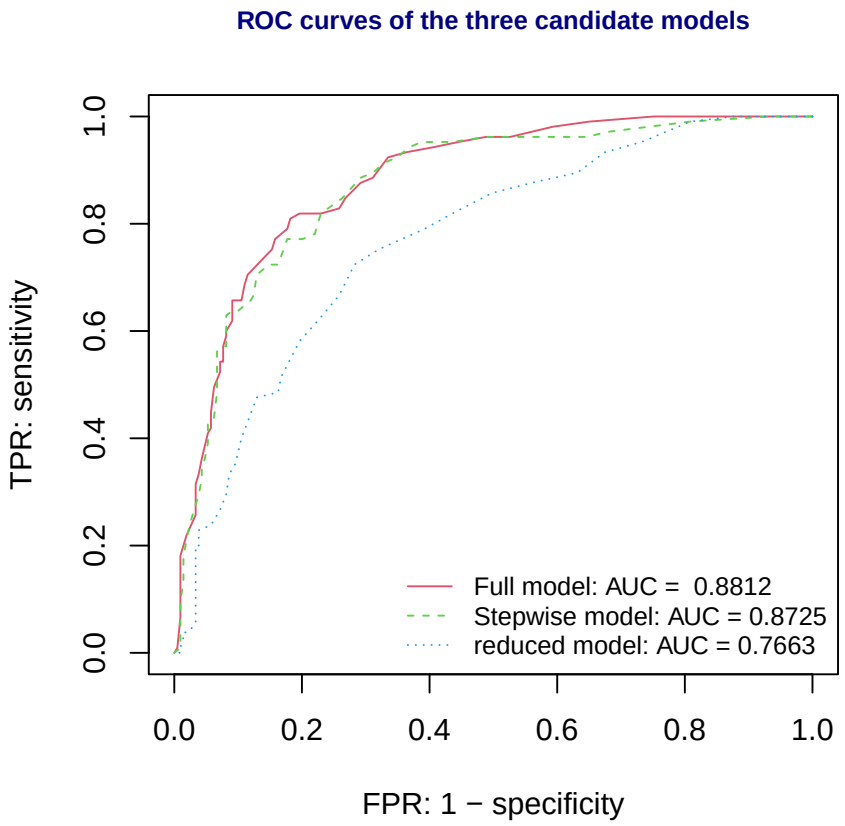
```
## A function to extract false positive and false negative rates
TPR.FPR=function(pred){
  prob.seq = seq(0,1, length=50)  # 50 equally spaced cut-off probabilities
  pn=length(prob.seq)
  true.lab=as.vector(train$diabetes)
  TPR = NULL
  FPR = NULL
  ##
  for (i in 1:pn){
   pred.lab = as.vector(ifelse(pred >prob.seq[i],"pos", "neg"))
   TPR[i] = length(which(true.lab=="pos" & pred.lab=="pos"))/length(which(true.lab=="po
   FPR[i] = length(which(true.lab=="neg" & pred.lab=="pos"))/length(which(true.lab=="ne
  }
 cbind(FPR = FPR, TPR = TPR)
}
```

The ROC curves of the three candidate models are given below.

```
## we need a function to calculate the AUC.
if (!require("pROC")) {
   install.packages("pROC")
   library(pROC)
}
##  candidate models
##  full model
candidate01 = glm(diabetes ~grp.preg + sd.glucose +sd.pressure+ sd.triceps + sd.insulin +
                    sd.mass + sd.pedigree + grp.age, family = binomial(link = "logit"),
                    data = train)
## reduced model
candidate03 = glm(diabetes ~ sd.insulin + sd.mass + sd.pedigree,
                    family = binomial(link = "logit"),
                    data = train)
##
candidate02 = stepAIC(candidate03,
                      scope = list(lower=formula(candidate03),upper=formula(candidate01)),
                      direction = "forward",    # forward selection
                      trace = 0                 # do not show the details
                      )
##  predicted probabilities
pred01 = predict.glm(candidate01, newdata = train, type="response")
pred02 = predict.glm(candidate02, newdata = train, type="response")
pred03 = predict.glm(candidate03, newdata = train, type="response")
####
## ROC curve
 plot(TPR.FPR(pred01)[,1], TPR.FPR(pred01)[,2],
      type="l", col=2, lty=1, xlim=c(0,1), ylim=c(0,1),
      xlab = "FPR: 1 - specificity",
      ylab ="TPR: sensitivity",
      main = "ROC curves of the three candidate models",
      cex.main = 0.8,
      col.main = "navy")
 lines(TPR.FPR(pred02)[,1], TPR.FPR(pred02)[,2],  col=3, lty=2)
 lines(TPR.FPR(pred03)[,1], TPR.FPR(pred03)[,2],  col=4, lty=3)

  ##
  category = train$diabetes == "pos"
  ROCobj01 <- roc(category, as.vector(pred01))
  ROCobj02 <- roc(category, as.vector(pred02))
  ROCobj03 <- roc(category, as.vector(pred03))
  AUC01 = round(auc(ROCobj01),4)
  AUC02 = round(auc(ROCobj02),4)
  AUC03 = round(auc(ROCobj03),4)
```

```
##
legend("bottomright", c(paste("Full model: AUC = ",AUC01),
                        paste("Stepwise model: AUC =",AUC02),
                        paste("reduced model: AUC =", AUC03)),
       col=2:4, lty=1:3, cex = 0.8, bty="n")
```



We can see from the ROC curve that full model and the stepwise model are better than the reduced model. However, the full model and the stepwise model have similar ROC curve and AUCs. Since stepwise model is simpler than the full model, therefore, the final model to report to the client is the stepwise model.

The accuracy measure of the stepwise model based one the test data has been reported in the earlier section.

11.4.8 Summary and Conclusion

The case study focused on predicting diabetes. For illustrative purposes, we used three models as candidates and use both cross-validation and ROC curve to select the final working model. Both cross-validation and ROC curve yielded the same result.

11.5 Analysis Assignment

This assignment focuses on binary predictive modeling using the logistic regression model. I used the same data set that was used in the previous few weeks to build a logistic predictive model for predicting the occurrence of diabetes.

You are expected to use the same data set you used in the three of your assignment to build a predictive logistic regression model.

The write-up of your assignment should be the same as my case study. To be more specific, you are expected to use my case study as a template to complete this assignment. The following are the major components I expected you to include in your report.

- Introduction - description of what you plan to do in the analysis
- Description of data and variables
  - information on the data collection process.
  - list of variable names and **definitions**.
- Research question(s) - what is the objective of the analysis
- Variable transformation and discretization
  - list the numerical variables you standardize
  - list of the variable you discretize
- Data split - the proportions of data for training and testing sets
- Candidate models - you can the candidate model you used in the previous assignment on the multiple logistic regression model.
- The final model selection
  - Cross-validation method
  - ROC approach (**This is optional**)

Chapter 12

Poison Regression Modeling

We have studied the normal-based linear and binary logistic regression models dependent on the types of random response variables.

12.1 Linear Regression Models

The primary regression models are normal linear models. The basic distributional assumption is that the residuals follow a normal distribution with mean zero and constant variance. The explanatory variables (also called predictor variables) are assumed to be uncorrelated with the response variable. Of course, the functional form of the explanatory variables must be correctly specified. Furthermore, the predictor variables are assumed to be non-random. This means that the response variable is a normal random variable - a special continuous random variable.

The regression coefficients are estimated by the least square method - also the least square estimation (LSE). When making inferences about the LSE, we still assume the residuals are normally distributed in order to construct confidence intervals of the regression coefficients and test the significance of the regression coefficient as well.

However, many continuous variables in the real world are not normally distributed, for example, a system's lifetime in reliability engineering, toxic concentrations in underground water, survival times of cancer patients who received surgery, the waiting time of a customer at a service desk, etc. These random variables are not normal.

12.2 Binary Logistic Regression Model

Contrary to the linear regression model that requires the response variable to be a continuous normal random variable, in the logistic regression model, the response variable is assumed to be a Bernoulli random variable that takes on only two distinct values such as "diseased" vs "disease-free", "success" vs "failure", etc.

The actual regression function in the logistic regression is the probability of "success" not the value of the response variable "success". The model was constructed with a special structure. The estimation of the regression coefficients is based on the likelihood theory.

The interpretation of the logistic regression model is also different from that of the linear regression model due to the special structure of the logistic regression. The regression coefficients measure how the corresponding explanatory variable impacts the log odds of success.

The resulting logistic regression model can be used for association analysis and prediction as well. The use of predictive modeling is one of the most important classification algorithms in data science. This module will focus on the discrete response variable which represents the number of occurrences of some event. Here are some examples.

- The number of sunspots over the years.
- the number of positive COVID-19 cases in a period of time.
- the number of the COVID-19 death counts.
- the number of people walking into an Emergency Room per hour.

12.3 Poisson Regression Models

The Poisson regression model assumes the random response variable to be a frequency count or a rate of a specific event such as COVID-19 positivity rates, COVID-19 death mortality, etc. As in the linear and logistic regression models, we also assume that predictor variables are non-random.

The family of logistic regression models assumes that the response variable follows a binomial distribution while Poisson regression models assume that the response variable has a Poisson distribution.

12.3.1 Assumptions of the Poisson Regression Model

The basic assumptions of Poisson regression are

- **Poisson Response**: The response variable is a count per unit of time or space, described by a Poisson distribution.
- **Independence**: The observations must be independent of one another.

- **Mean is equal to variance**: By definition, the mean of a Poisson random variable must be equal to its variance.
- **Linearity**: The log of the mean rate, $\log(\lambda)$, must be a linear function of x.

12.3.2 Structure of Poisson Regression Model for Counts

Let Y be the response variable that takes on frequency counts as values and X be the set of predictor variables such as demographics and social determinants. Further, let $\mu = E[Y]$ be the mean of the response variable. The Poisson regression model is defined in the following analytic expression.

$$\log(\mu) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p,$$

where $\beta_0, \beta_1, \cdots, \beta_p$ are coefficients of the Poisson regression model. The interpretation of the regression coefficient β_i is as follows

- β_0 = the baseline logarithm of the mean of Y, $\log(\mu)$, when all predictor variables $x_i = 0$, for $i = 1, 2, \cdots, p$. As usual, we are not interested in the inference of the intercept parameter.
- β_i = is the change of the log mean due to one unit increases in x_i with all other x_j being fixed, for $j \neq i$.

To be more specific, let $\mu(x_i) = E[Y|(\cdots, x_i, \cdots)]$ be the mean counts with variables $\cdots, x_{i-1}, x_{i+1}, \cdots$ being fixed except for x_i. To look at how x_i impacts the value of $E[Y]$, we increase x_i by one unit and fix all other predictor variables.

$$\log \mu(x_i) = \beta_0 + \cdots + \beta_{i-1} x_{i-1} + \beta_i x_i + \beta_{i+1} x_{i+1} + \cdots + \beta_p x_p$$

After increasing x_i by one unit, the corresponding log mean is given by

$$\log \mu(x_i + 1), = \beta_0 + \cdots + \beta_{i-1} x_{i-1} + \beta_i (x_i + 1) + \beta_{i+1} x_{i+1} + \cdots + \beta_p x_p$$

Therefore,

$$\beta_i = \log \mu(x_i + 1) - \log \mu(x_i)$$

- If $\beta_i = 0$, then $\log \mu(x_i + 1) = \log \mu(x_i)$. This implies that x_i does not impact the mean of Y, equivalently, Y and X_i are not associated with each other.
- If $\beta_i > 0$, then $\log \mu(x_i+1) > \log \mu(x_i)$. This implies that $\mu(x_i+1) > \mu(x_i)$, equivalently, Y and X_i are positively associated with each other.
- Similarly, if $\beta_i < 0$, then Y and X_i are negatively associated with each other.

Because the Poisson distribution is usually used to model rare events such as diseases and anomalies and the regression coefficients β_i can be expressed as $\beta_i = \log(\mu(x_i+1)/\mu(x_i))$, β_i is called **relative risk**, sometimes also called **risk ratio** or **log risk ratio**.

12.3.3 Poisson Models for Rates

The Poisson log-linear regression model for the expected rate of the occurrence of the event is defined by

$$\log(\mu/t) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

This can be re-expressed as

$$\log(\mu) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \log(t)$$

The term $\log(t)$ is referred to as an offset. It is an adjustment term and a group of observations may have the same offset, or each individual may have a different value of t. $\log(t)$ is an observation and it will change the value of estimated counts:

$$\mu = \exp[\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p+\log(t)] = t\exp(\beta_0)\exp(\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p)$$

This means that the mean count is proportional to t.

Note that the interpretation of parameter estimates β_0 and $\beta_1, \beta_2, \cdots, \beta_p$ will stay the same as for the model of counts; you just need to multiply the expected counts by t.

12.3.4 Estimation of Regression Coefficients and Goodness-of-fit

Unlike linear regression in which we have assumptions about the residuals. The estimated residuals can be used to test the assumptions about the distribution. In GLM, the goodness-of-fit is much more complex than the normal-based regression modeling. But we can mimic residuals in the linear regression modeling to define a similar quantity called **deviance residual** based on the likelihood of the model. We will give this definition in the next module.

The estimation of the regression coefficients is based on the maximum likelihood estimation (MLE) which requires numerical solutions. In the Poisson distribution, the mean and variance are equal ($E[Y] = var[Y]$). Failing to meet this assumption results in the issue of **dispersion**, a common violation of the Poisson regression. We will discuss this issue and the relevant remedies in the next module.

We will not go into detail about how to estimate the regression coefficients and perform model diagnostics in this module. Instead, we will focus on data analysis, in particular, the interpretation of regression coefficients.

12.3.5 Data Set Layout

The data set required for the Poisson regression model in R should have the following layout.

ID(optional)	x_1	x_2	...	x_k	y (counts)	total (counts, optional)
1	x_{11}	x_{21}	...	x_{k1}	y_1	t_1
2	x_{12}	x_{22}	...	x_{k2}	y_2	t_2
...	...	...	...	...	...	...
n	x_{1n}	x_{2n}	...	x_{kn}	y_n	t_n

As usual, if there are categorical variables (with numerical coding), we need to introduce dummy variables to capture the unequal effects on the response across the categories of the variables.

12.4 Case Study: Modeling Lung Cancer Rates in Four Cities of Denmark - Part I

The World Health Organisation (WHO) statistics suggests that Denmark has the highest cancer rates in the world, with about 326 people out of every 100,000 developing cancer each year. The country is known to have a good record of diagnosing cancer, but also has high rates of smoking among women and high levels of alcohol consumption.

```
knitr::include_graphics("img09/DenmarkCitiesMap.png")
```

Table 12.2: First few records in the data set

city	age	pop	cases
Fredericia	40-54	3059	11
Horsens	40-54	2879	13
Kolding	40-54	3142	4
Vejle	40-54	2520	5
Fredericia	55-59	800	11
Horsens	55-59	1083	6

In this case study, we use a data set that summarized the lung cancer incident counts (cases) per age group for four Danish cities from 1968 to 1971. The primary random response variable is lung cancer cases. The predictor variables are the age group and the total population size of the neighboring cities.

The data set was built in the R library {ISwR}.

```
library(knitr)
library(ISwR)
#eba1977 = read.csv("eba1977.csv")
data(eba1977)
kable(head(eba1977), caption = "First few records in the data set")
```

```
# check the values of the variables in the data set
```

12.4.1 Poisson Regression on Cancer Counts

We first build a Poisson frequency regression model and ignore the population size of each city in the data.

Table 12.3: The Poisson regression model for the counts of lung cancer cases versus the geographical locations and the age group.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	2.2437446	0.2036265	11.0189233	0.0000000
cityHorsens	-0.0984401	0.1812909	-0.5429952	0.5871331
cityKolding	-0.2270575	0.1877041	-1.2096561	0.2264109
cityVejle	-0.2270575	0.1877041	-1.2096561	0.2264109
age55-59	-0.0307717	0.2480988	-0.1240298	0.9012916
age60-64	0.2646926	0.2314278	1.1437369	0.2527328
age65-69	0.3101549	0.2291839	1.3533017	0.1759593
age70-74	0.1923719	0.2351660	0.8180261	0.4133423
age75+	-0.0625204	0.2501222	-0.2499593	0.8026188

```
model.freq <- glm(cases ~ city + age, family = poisson, data = eba1977)
##
pois.count.coef = summary(model.freq)$coef
kable(pois.count.coef, caption = "The Poisson regression model for the counts of lung cancer case
```

The above inferential table about the regression coefficients indicates both city and age are insignificant. This means, if we look at cancer count across the age group and city, there is no statistical evidence to support the potential discrepancy across the age groups and cities. However, this does not imply that the model is meaningless from the practical perspective since statistical significance is not equivalent the clinical importance. Moreover, the sample size could impact the statistical significance of some of the variables.

The other way to look at the model is the appropriateness model. The cancer counts are dependent on the population size. Ignoring the population size implies the information in the sample was not effectively used. In the next subsection, we model the cancer rates that involve the population size.

The other way to look at the model is goodness of the model. The cancer counts are dependent on the population size. Ignoring the population size implies the information in the sample was not effectively used. In the next subsection, we model the cancer rates that involve the population size.

Since it's reasonable to assume that the expected count of lung cancer incidents is proportional to the population size, we would prefer to model the rate of incidents per capita. However, for the purposed of illustration, we will fit the Poisson regression model with both counts and rate of cancer rates.

12.4.2 Poisson Regression on Rates

The following model assesses the potential relationship between cancer death rates and age. This is the primary interest of the model. We also want to

Table 12.4: Poisson regression on the rate of the the cancer rate in the four Danish cities adjusted by age.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-5.6320645	0.2002545	-28.124529	0.0000000
cityHorsens	-0.3300600	0.1815033	-1.818479	0.0689909
cityKolding	-0.3715462	0.1878063	-1.978348	0.0478895
cityVejle	-0.2723177	0.1878534	-1.449629	0.1471620
age55-59	1.1010140	0.2482858	4.434463	0.0000092
age60-64	1.5186123	0.2316376	6.555985	0.0000000
age65-69	1.7677062	0.2294395	7.704455	0.0000000
age70-74	1.8568633	0.2353230	7.890701	0.0000000
age75+	1.4196534	0.2502707	5.672472	0.0000000

adjust the relationship be the potential neighboring cities.

```
model.rates <- glm(cases ~ city + age, offset = log(pop),
                   family = poisson, data = eba1977)
kable(summary(model.rates)$coef, caption = "Poisson regression on the rate of the the c
```

The above table indicates that the log of cancer rate is not identical across the age groups and among the four cities. To be more specific, the log rates of Fredericia (baseline city) were higher than in the other three cities. The youngest age group (45-55) has the lowest log rate. The regression coefficients represent the change of log rate between the associate age group and the reference age group. The same interpretation applies to the change in log rate among the cities.

```
model.rates <- glm(cases ~ city + age, offset = log(pop),
                   family = quasipoisson, data = eba1977)
summary(model.rates)
```

```
##
## Call:
## glm(formula = cases ~ city + age, family = quasipoisson, data = eba1977,
##     offset = log(pop))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.63573  -0.67296  -0.03436   0.37258   1.85267
##
## Coefficients:
##             Estimate Std. Error t value          Pr(>|t|)
## (Intercept)  -5.6321     0.2456 -22.932 0.000000000000431 ***
## cityHorsens  -0.3301     0.2226  -1.483           0.15884
## cityKolding  -0.3715     0.2303  -1.613           0.12756
```

Table 12.5: Poisson regression on the rate of the the cancer rate in the four Danish cities adjusted by age.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-5.6320645	0.2455964	-22.932196	0.0000000
cityHorsens	-0.3300600	0.2225995	-1.482753	0.1588444
cityKolding	-0.3715462	0.2303296	-1.613107	0.1275571
cityVejle	-0.2723177	0.2303873	-1.181999	0.2556057
age55-59	1.1010140	0.3045029	3.615775	0.0025421
age60-64	1.5186123	0.2840852	5.345623	0.0000817
age65-69	1.7677062	0.2813894	6.282063	0.0000147
age70-74	1.8568633	0.2886051	6.433924	0.0000113
age75+	1.4196534	0.3069372	4.625224	0.0003301

```
## cityVejle    -0.2723     0.2304  -1.182           0.25561
## age55-59      1.1010     0.3045   3.616           0.00254 **
## age60-64      1.5186     0.2841   5.346 0.000081665456528 ***
## age65-69      1.7677     0.2814   6.282 0.000014695563904 ***
## age70-74      1.8569     0.2886   6.434 0.000011253999281 ***
## age75+        1.4197     0.3069   4.625           0.00033 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 1.504109)
##
##     Null deviance: 129.908  on 23  degrees of freedom
## Residual deviance:  23.447  on 15  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

```
kable(summary(model.rates)$coef, caption = "Poisson regression on the rate of the  the cancer rat
```

The intercept represents the **baseline log-cancer rate** (of baseline *age group 44-55* in the baseline city *Fredericia*). The actual rate is $\exp(-5.6321) \approx 0.36\%$ which is close the recently reported rate of the country by WHO. The intercept -0.3301 is the difference of the log-rates between baseline city Fredericia and the city of Horsens at any given age group, to be more specific, $\log(R_{\text{Horsen}}) - \log(R_{\text{Fredericia}}) = -0.3301$ which is equivalent to

$$\log\left(\frac{R_{\text{Horsen}}}{R_{\text{Fredericia}}}\right) = -0.3301 \quad \Rightarrow \quad \frac{R_{\text{Horsen}}}{R_{\text{Fredericia}}} = e^{-0.3301} \approx 0.7188518.$$

This means, with fixed age groups, the cancer rate in Horsens is about 28% lower than that in Fredericia. Next, we look at the coefficient 1.4197 associated with age group 75+. For any given city,

$$\log\left(\frac{R_{\text{age75+}}}{R_{\text{age45-54}}}\right) = 1.4197 \quad \Rightarrow \quad \frac{R_{\text{age75+}}}{R_{\text{age45-54}}} = e^{1.41971} \approx 4.135921.$$

This implies that the cancer rate in age group 75+ is 4.14 times that of the baseline age group of 45-54.

12.4.3 Some Graphical Comparison

The inferential tables of the Poisson regression models in the previous sections give numerical information about the potential discrepancy across the age group and among the cities. But it is not intuitive. Next, we create a graphic to visualize the relationship between cancer rate and age across cities.

First of all, every city has a trend line that reflects the relationship between the cancer rate and the age. We next find the rates of combinations of city and age-group based on the following working rate model.

$$\text{log-rate} = -5.6321 - 0.3301 \times \text{cityHorsens} - 0.3715 \times \text{cityKolding} - 0.2723 \times \text{cityVejle} + 1.1010 \times \text{age55-59}$$

$$+1.5186 \times \text{age60-64} + 1.7677 \times \text{age65-69} + 1.8569 \times \text{age70-74} + 1.4197 \times \text{age75+}$$

Or equivalently, we can write rate model as

$$\textit{rate} = \exp(-5.6321 - 0.3301 \times \text{cityHorsens} - 0.3715 \times \text{cityKolding} - 0.2723 \times \text{cityVejle} + 1.1010 \times \text{age55-59})$$

$$\times \exp(1.5186 \times \text{age60-64} + 1.7677 \times \text{age65-69} + 1.8569 \times \text{age70-74} + 1.4197 \times \text{age75+})$$

Next, we made a table cancer rates of combinations of city and age group.

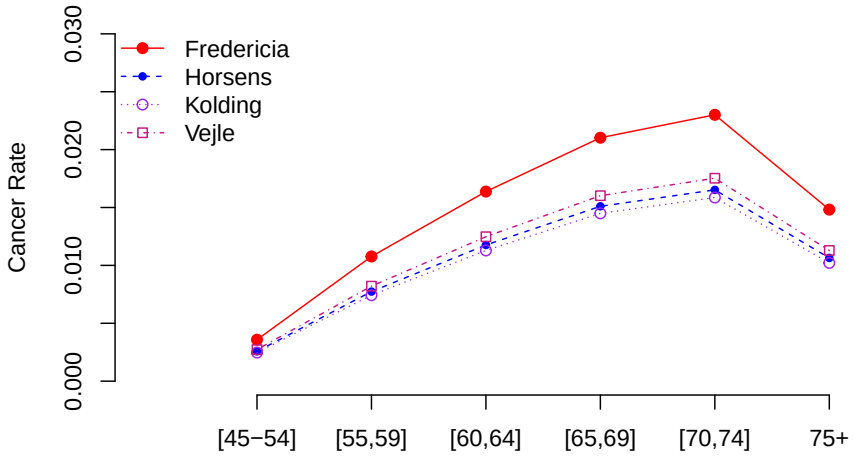
The following calculation is based on the regression equation with coefficients given in above table 3. Note that all variables in the model are indicator variables. Each of these indicator variables takes only two possible values: 0 and 1.

For example, $\exp(-5.632)$ gives the cancer rate of the baseline city, Fredericia, and the baseline age group [45-54]. $\exp(-5.632 + 1.101)$ gives the cancer rate of baseline city, Fredericia, and age group [55-59]. Following the same pattern, you can find the cancer rate for each combination of the city and age group.

```
# Fredericia
Fredericia = c(exp(-5.632), exp(-5.632+1.101),
               exp(-5.632+1.52),exp(-5.632+1.77),
               exp(-5.632+1.86),exp(-5.632+1.42))
```

```
# Horsens
Horsens = c(exp(-5.632-0.331), exp(-5.632-0.331+1.101),
            exp(-5.632-0.331+1.52),exp(-5.632-0.331+1.77),
            exp(-5.632-0.331+1.86),
            exp(-5.632-0.331+1.42))
# Kolding
Kolding= c(exp(-5.632-0.372), exp(-5.632-0.372+1.101),
           exp(-5.632-0.372+1.52),exp(-5.632-0.372+1.77),
           exp(-5.632-0.372+1.86), exp(-5.632-0.372+1.42))
# Vejle
Vejle = c(exp(-5.632-0.272), exp(-5.632-0.272+1.101),
          exp(-5.632-0.272+1.52),exp(-5.632-0.272+1.77),
          exp(-5.632-0.272+1.86), exp(-5.632-0.272+1.42))
minmax = range(c(Fredericia,Horsens,Kolding,Vejle))
####
```

```
plot(1:6,Fredericia, type="l", lty =1, col="red", xlab="",
               ylab="Cancer Rate", xlim=c(0,6), ylim=c(0, 0.03), axes=FALSE )
axis(2)
axis(1, labels=c("[45-54]","[55,59]","[60,64]","[65,69]","[70,74]","75+"),
            at = 1:6)
points(1:6,Fredericia, pch=19, col="red")
##
lines(1:6, Horsens, lty =2, col="blue")
points(1:6, Horsens, pch=20, col="blue")
##
lines(1:6, Kolding, lty =3, col="purple")
points(1:6, Kolding, pch=21, col="purple")
###
lines(1:6, Vejle, lty =4, col="mediumvioletred")
points(1:6, Vejle, pch=22, col="mediumvioletred")
##
legend("topleft", c("Fredericia","Horsens", "Kolding", "Vejle" ),
                  pch=19:22, lty=1:4,  bty="n",
        col=c("red", "blue", "purple", "mediumvioletred"))
```



12.4.4 Discussions and Conclusions

Several conclusions we can draw from the output of the regression models.

The regression model based on the cancer count is not appropriate since the information on the population size is a key variable in study the cancer distribution. Simply including the population size in the regression model will relduce the significance of age. See the following output of the fitted Poisson regression model of count adjusted by population size.

```
model.freq.pop <- glm(cases ~ city + age + log(pop), family = poisson,
                      data = eba1977)
##
pois.count.coef.pop = summary(model.freq.pop)$coef
kable(pois.count.coef.pop, caption = "The Poisson regression model for the counts of lu
```

We can see from the above output the adding population size to the model

The cancer rate in Fredericia is significantly higher than in the other three cities. It seems that there is no significant difference between Horsens, Kolding, and Vejle. The reason why Fredericia has a higher cancer rate needs further investigation with additional information.

There is a curve linear relationship between age and the cancer rate. The cancer rate increases as age increase. However, the rate starts decreasing after 75. This pattern is consistent with the clinical studies since lung cancer patients were mostly diagnosed between 65-70. It is rare to see lung cancer patients aged

Table 12.6: The Poisson regression model for the counts of lung cancer cases versus the geographical locations, population size, and age group.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	11.7495934	8.8151328	1.3328890	0.1825682
cityHorsens	0.1832573	0.3192679	0.5739922	0.5659731
cityKolding	-0.0483001	0.2519622	-0.1916957	0.8479806
cityVejle	-0.1679335	0.1964757	-0.8547289	0.3927012
age55-59	-1.3842350	1.2728775	-1.0874849	0.2768226
age60-64	-1.2366489	1.4049520	-0.8802073	0.3787470
age65-69	-1.4377681	1.6310051	-0.8815228	0.3780349
age70-74	-1.8048920	1.8607922	-0.9699589	0.3320670
age75+	-1.8383162	1.6587773	-1.1082357	0.2677600
log(pop)	-1.2095837	1.1227191	-1.0773698	0.2813151

under 45.

The last statistical observation is that there is no interaction effect between the age groups and the geographic locations. The rate curves are "parallel".

This is only a small data set with limited information. All conclusions in this report are only based on the given data set.

12.5 Concluding Remarks

This note briefly outlines the regular Poisson regression model for fitting frequency data. The Poisson regression model has a simple structure and is easy to interpret but has a relatively strong assumption - variance is equal to the mean.

If this assumption is violated, we can use negative binomial regression as an alternative. The other potential issue is the data has excess zeros, then we can consider zero-inflated Poisson or zero-inflated negative binomial regression models.

For this week's assignment, you will model the daily counts (and proportion) of cyclists who entered and left the four bridges in New York City. The data set will be provided in the document of the assignment instruction.

12.6 Analysis Assignment

12.6.1 Data Description

The daily total of bike counts was conducted monthly on the Brooklyn Bridge, Manhattan Bridge, Williamsburg Bridge, and Queensboro Bridge. To keep

count of cyclists entering and leaving Queens, Manhattan, and Brooklyn via the East River Bridges. The Traffic Information Management System (TIMS) collects the count data. Each record represents the total number of cyclists per 24 hours at Brooklyn Bridge, Manhattan Bridge, Williamsburg Bridge, and Queensboro Bridge.

12.6.2 Data Formats and Loading

To save you time in finding a data set for Poisson regression, I created several subsets that contain the relevant information you need for this week's assignment. The data was saved in Excel format with multiple tabs. Please find the tab with your last name and then copy-and-paste your data file to a new Excel sheet and save it as a CSV format file or simply copy-and-paste to Notepad to create a TXT format file so you can read the file to R. You can also read the Excel file directly to R using appropriate R functions in relevant R libraries.

Here is the link to the Excel data set:NYC cyclist data set (AssignDataSet.xlsx).

12.6.3 Assignment Instructions

Your analysis and report should be similar to section 3 of my class note. PLEASE TELL STORIES BEHIND ALL R OUTPUTS (tables and figures) YOU GENERATED IN THE ANALYSIS. You can earn half of the credit if you only generate relevant outputs correctly but with no good storytelling. *The model diagnostic is not required in this assignment but will be required in the next assignment.*

The following components must be included in your analysis report.

- Your description of the data and the variables
  - data collection
  - variable names and definitions. Keep in mind that the variable **Date** is the observation ID.
- What is the research/practical question?
  - Explicit definition of the questions
  - what is the response variable that captures the information to address the research/practical questions
- What statistical model will be used in the analysis
  - Assumptions and conditions - give a brief description of the models
  - Build a Poison regression model on the counts only.
    - Use the p-values to perform variable selection - keep only significant variables in the final model
    - Interpret the regression coefficients of the Poison regression model
    - Explain/describe the steps of your analysis, motivation, and findings.

  - Build a Poison regression on the proportions (rates) of cyclists entering and leaving the bridge in your data.
    * Use the p-values to select the significant variables
    * Interpret the regression coefficients of the Poison rate model
    * Explain/describe the steps of your analysis, motivation, and findings.
- Summarize the findings of the above two models.

Chapter 13

Dispersed Poisson Regression

In the last module, we introduced the basics of Poisson regression on counts and rates. Sometimes Poisson regression may not work well since the variance of the response may not be equal to the mean. In this module, we will look into the issue of potential **dispersion** and other relevant issues and find an alternative count and rate regression model.

The general structure of the Poisson regression model is given by

$$\log \mu(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

There are several assumptions for the Poisson regression model. The most important one is that the mean and variance of the response variable are equal. The other assumption is the linear relationship between the explanatory variables and the log mean of the response. It is not straightforward to detect the potential violation of these assumptions. In the next section, we define some metrics based on *residuals* based on the hypothetical model and the observed data.

The learning objectives of this module are (1) to develop measures to detect the violation of the assumptions of the Poisson regression, (2) to define a measure to estimate the dispersion, (3) to introduce the quasi-likelihood Poisson regression model to make robust inference of the regression coefficients.

13.1 Residuals of Poisson Regression

In linear regression, we can use the residual plots to check the potential violation of the model assumption since the residuals are normally distributed with

zero mean and constant standard deviation if the model is appropriate. In Poisson regression, we can mimic the way of defining the *kind of residuals* as we did in linear regression. Under the large sample assumptions, these residuals are approximately normally distributed if the underlying hypothetical model is appropriate. Using this large sample property, we can define some metrics to detect the potential violation of the model assumptions.

Recall that the residual of i-th observation under a model defined by

$$e_i = y_i - \hat{\mu}_i.$$

where $\hat{\mu}_i$ is the fitted value based on the hypothetical model $\log(\mu) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p$. The regression coefficients can be estimated using least squares and likelihood methods.

Next, I am going to use a portion of NYC cyclist data (you will use similar data for this week's assignment). I will use this data set as an example to explain the concepts and models discussed in this module.

```
cyclist = read.csv("dat/w10-NYCcyclistData.csv")
cyclist$log.Brooklynbridge = log(cyclist$BrooklynBridge)
m0.loglinear = lm(log.Brooklynbridge ~ Day + HighTemp + LowTemp + Precipitation,
                  data = cyclist)
m1.Poisson = glm(BrooklynBridge ~ Day + HighTemp + LowTemp + Precipitation,
                 family = poisson(link = "log"), offset = log(Total),  data = cyclist)
```

- **Least Square Estimate (LSE) of** β's

In this method, we need to take the logarithm of the observed count as in the data table as shown in the following table.

ID	x_1	x_2	...	x_k	y (counts)	log-count [$\log(y)$]
1	x_{11}	x_{21}	...	x_{k1}	y_1	$\log(y_1)$
2	x_{12}	x_{22}	...	x_{k2}	y_2	$\log(y_2)$
...	...	...	...	...	...	...
n	x_{1n}	x_{2n}	...	x_{kn}	y_n	$\log(y_n)$

We can use the log of the observed count and values of the explanatory variables to find the least square estimates (LSE) of the regression coefficients, denoted by $\tilde{\beta}_0, \tilde{\beta}_1, \cdots, \tilde{\beta}_p$, of the Poisson regression model. Then the i-th fitted value $\hat{\mu}_i = \exp(\tilde{\beta}_0 + \tilde{\beta}_1 x_1 + \cdots + \tilde{\beta}_p x_p)$.

- **Maximum Likelihood Estimate (MLE) of** β's

Since the response variable is assumed to have a Poisson with its mean $\mu_j = \exp(\beta_0 + \beta_1 x_{1j} + \cdots + \beta_p x_{pj})$. The kernel of the log-likelihood of observing the data set is defined in the following

$$l(\beta_0, \beta_1, \cdots, \beta_p) \propto \sum_{j=1}^{n} \left[y_j(\beta_0 + \beta_1 x_{1j} + \cdots + \beta_p x_{pj}) - \exp(\beta_0 + \beta_1 x_{1j} + \cdots + \beta_p x_{pj}) \right]$$

The MLE of the β's, denoted by $\hat{\beta}_0, \hat{\beta}_1, \cdots, \hat{\beta}_p$, maximizes the above log-likelihood through solving the following **score equations**,

$$\begin{cases} \frac{\partial l(\beta_0,\beta_1,\cdots,\beta_p)}{\partial \alpha_0} = \frac{\partial}{\partial \alpha_0} \sum_{j=1}^{n} \left[y_j(\beta_0 + \beta_1 x_{1j} + \cdots + \beta_p x_{pj}) - \exp(\beta_0 + \beta_1 x_{1j} + \cdots + \beta_p x_{pj}) \right] = 0 \\ \frac{\partial l(\beta_0,\beta_1,\cdots,\beta_p)}{\partial \alpha_1} = \frac{\partial}{\partial \alpha_1} \sum_{j=1}^{n} \left[y_j(\beta_0 + \beta_1 x_{1j} + \cdots + \beta_p x_{pj}) - \exp(\beta_0 + \beta_1 x_{1j} + \cdots + \beta_p x_{pj}) \right] = 0 \\ \cdots \quad \cdots \quad \cdots \quad \cdots \\ \frac{\partial l(\beta_0,\beta_1,\cdots,\beta_p)}{\partial \alpha_p} = \frac{\partial}{\partial \alpha_p} \sum_{j=1}^{n} \left[y_j(\beta_0 + \beta_1 x_{1j} + \cdots + \beta_p x_{pj}) - \exp(\beta_0 + \beta_1 x_{1j} + \cdots + \beta_p x_{pj}) \right] = 0 \end{cases}$$

With the MLE, we can find the fitted value by $\hat{\mu}_i = \exp(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p)$.

13.1.1 Pearson Residuals

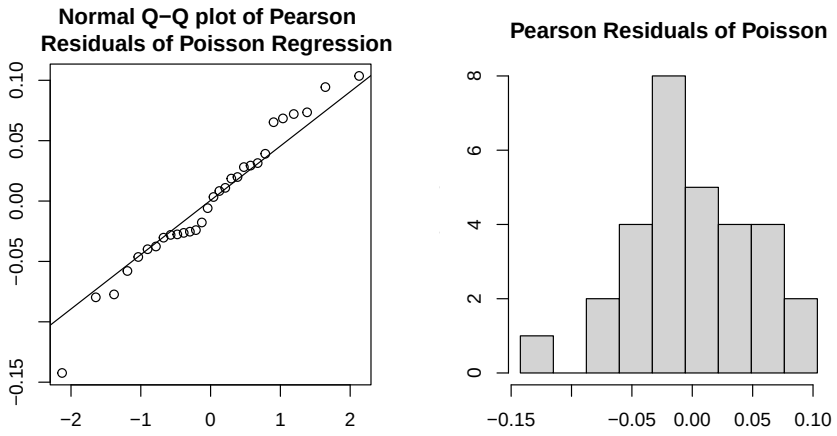
The Pearson residual of i-th observation is defined to be

$$\text{Pearson.Residual}_i = \frac{y_i - \hat{\mu}_i}{\sqrt{\hat{\mu}_i}}$$

Pearson residuals are the standardized value of the observed y_i with the assumption that Y_i is a Poisson normal random variable. Under a large sample assumption, we would expect that residuals are approximately normally distributed. We can use this property to assess the appropriateness of the Poisson regression model. Equivalently, the square of the Pearson residual is a chi-square distribution with one degree of freedom.

Next, we extract the residuals $(y_i - \hat{\mu}_i)$ direct from the linear regression model and then divided by the square root of $\hat{\mu}_i$, fitted values of y_i, to find the Pearson residuals.

```
par(mfrow=c(1,2), mar=c(3,3,3,3))
resid.loglin = m0.loglinear$residuals
fitted.loglin = m0.loglinear$fitted.values
pearson.resid = resid.loglin/sqrt(fitted.loglin)
qqnorm(pearson.resid, main = "Normal Q-Q plot of Pearson \n  Residuals of Poisson Regression")
qqline(pearson.resid)
##
seq.bound=seq(range(pearson.resid)[1], range(pearson.resid)[2], length=10)
hist(pearson.resid, breaks = seq.bound,
     main = "Pearson Residuals of Poisson")
```



Both the Q-Q plot and histogram indicate that the distribution of Pearson residuals is skewed. There is a discrepancy between frequency distribution and normal distribution. Since the Pearson residuals are derived based on the least square algorithm, they don't have good distributional properties to develop a test.

13.1.2 Deviance Residuals

Deviance residuals of Poisson regression are defined based on the likelihood method in the following

$$\text{Deviance.Residual}_i = \text{sign}(y_i - \hat{\mu}_i)\sqrt{2\left[y_i \log(y_i/\hat{\mu}_i) - (y_i - \hat{\mu}_i)\right]}$$

where

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \end{cases}$$

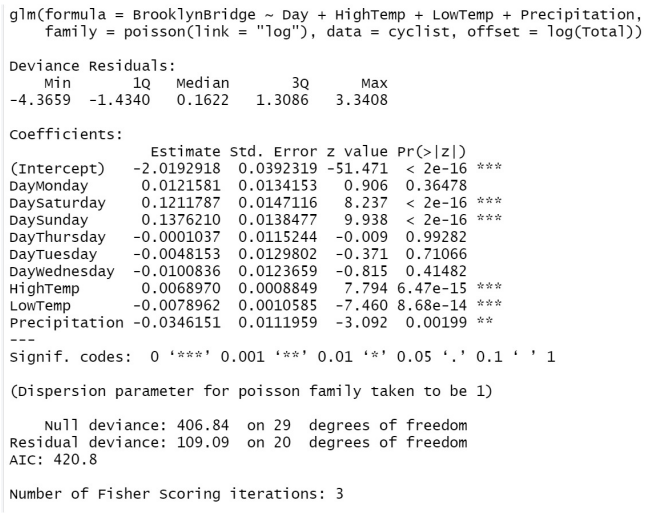
Since the deviance residuals based on Poisson regression are defined based on the likelihood, there are asymptotically normally distributed. We can use this asymptotic property of normal distribution to assess the appropriateness of the Poisson regression.

13.1.3 Numerical Example

The residual deviance of Poisson (or other generalized linear models) is defined to be the sum of all deviance residuals: deviance $= \sum_i (\text{deviance.residual}_i)^2$.

In the output of **glm()** in R, the **null deviance residual** (corresponding to the model with no explanatory variable in it) and **deviance residual** (corresponding to the fitted model) are reported with the corresponding degrees of freedom.

```
knitr::include_graphics("img10/w10-glmPoisOutput.jpg")
```



```
glm(formula = BrooklynBridge ~ Day + HighTemp + LowTemp + Precipitation,
    family = poisson(link = "log"), data = cyclist, offset = log(Total))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.3659  -1.4340   0.1622   1.3086   3.3408

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)   -2.0192918  0.0392319 -51.471  < 2e-16 ***
DayMonday      0.0121581  0.0134153   0.906  0.36478
DaySaturday    0.1211787  0.0147116   8.237  < 2e-16 ***
DaySunday      0.1376210  0.0138477   9.938  < 2e-16 ***
DayThursday   -0.0001037  0.0115244  -0.009  0.99282
DayTuesday    -0.0048153  0.0129802  -0.371  0.71066
DayWednesday  -0.0100836  0.0123659  -0.815  0.41482
HighTemp       0.0068970  0.0008849   7.794 6.47e-15 ***
LowTemp       -0.0078962  0.0010585  -7.460 8.68e-14 ***
Precipitation -0.0346151  0.0111959  -3.092  0.00199 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 406.84  on 29  degrees of freedom
Residual deviance: 109.09  on 20  degrees of freedom
AIC: 420.8

Number of Fisher Scoring iterations: 3
```

Figure 13.1: R glm() output of Poisson regression model

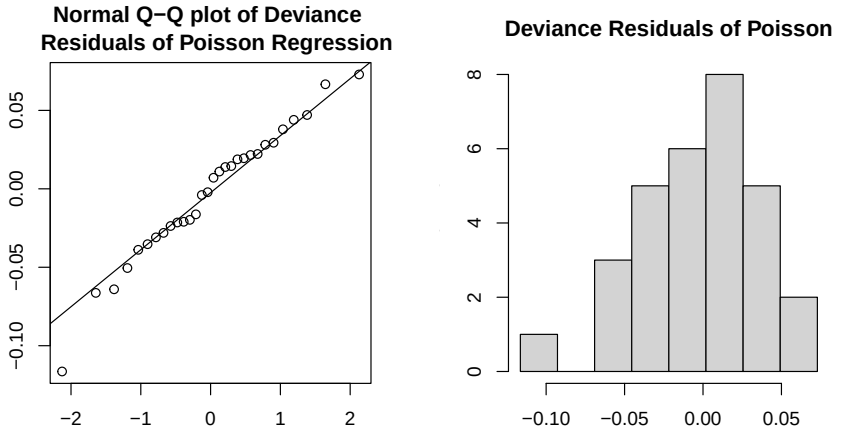
We can see from the Poisson regression output in R that only deviance residuals and the corresponding descriptive statistics are reported (the five-number-summary of deviance residuals, null deviance, and deviance with corresponding degrees of freedoms). This is because the inference of Poisson regression in **glm()** is based on the likelihood theory.

We can also extract the deviance residuals from **glm()** object and make a Q-Q plot in the following

```
par(mfrow=c(1,2), mar=c(3,3,3,3))
qqnorm(m1.Poisson$residuals,
       main = "Normal Q-Q plot of Deviance \n  Residuals of Poisson Regression")
qqline(m1.Poisson$residuals)
resid.m1 = m1.Poisson$residuals
seq.bound=seq(range(resid.m1)[1], range(resid.m1)[2], length=9)
hist(m1.Poisson$residuals, breaks = seq.bound,
     main = "Deviance Residuals of Poisson")
```

Table 13.2: The p-value of deviance chi-square test

p.value
0



Both Q-Q plot and histogram indicate that the distribution of the deviance residuals is slightly different from a normal distribution. If the model is correctly specified, the sum of squared residuals $\sum_i (\text{Deviance.Residual}_i)^2$ is distributed as χ^2_{n-p}. The deviance and the degrees of freedom of the deviance are given in the output of the **glm()** (see the output given in the above figure).

For example, we next extract the deviance and degrees of freedom from the output and perform a chi-square test.

```
deviance.resid = m1.Poisson$deviance
deviance.df = m1.Poisson$df.residual
# p-value of chi-square test
p.value = 1-pchisq(deviance.resid, deviance.df)
pval = cbind(p.value = p.value)
kable(pval, caption="The p-value of deviance chi-square test")
```

The p-value is almost equal to zero. The assumption of the Poisson regression model was violated.

13.1.4 Goodness-of-fit

The deviance has an asymptotic χ^2_{n-p} distribution if the model is correct. If the p-value calculated based on the deviance from χ^2_{n-p} is less than the significance level, we claim the model has a poor fit (also lack-of-fit, badness-of-fit). There could be different reasons that cause the poor fit. For example, (1) data issues such as outliers, (2) functional form of the explanatory variables (non-linear

relationship between the log of the mean of the response), (3) missing some important explanatory variable in the data set, (4) dispersion issue, etc.

The deviance residual can be used naturally to **compare hierarchical models** by defining the likelihood ratio chi-square tests.

The dispersion issue will be detailed in the next section.

13.2 Dispersion and Dispersed Poisson Regression Model

The issue of **Over-dispersion** in Poisson regression is common. It indicates that the variance is bigger than the mean.

13.2.1 Definition of Dispersion

To detect over-dispersion (i.e., the violation of the assumption in Poisson regression), we define the following dispersion parameter

$$\hat{\phi} = \frac{\sum_i (\text{Pearson.Residual}_i)^2}{n-p},$$

where p is the number of regression coefficients. Note that $\sum_i (\text{Pearson.Residual}_i)^2$ has a χ^2_{n-1} if the Poisson assumption is correct. Since the expectation of a chi-square distribution is equal to the degrees of freedom, this means that the **estimated dispersion parameter**, $\hat{\phi}$, should be around 1 if the Poisson assumption is correct. Therefore, the estimated dispersion parameter can be used to detect potential dispersion issues.

- **Impact of Dispersion**

Over-dispersion means the assumptions of the Poisson model (or other models in the exponential family) are not met, therefore, the p-values in the output of **glm()** with the regular *log link* in the *poisson family* are not reliable. We should use p-values in the output to perform significant tests and use them for variable selection.

13.2.2 Quasi-Poisson Regression Model

We can make an adjustment to the Poisson variance by adding a dispersion parameter. In other words, while for Poisson data $\bar{Y} = s^2_Y$, the quasi-Poisson

allows for $\bar{Y} = \phi \cdot s^2_Y$, and estimates the over-dispersion parameter ϕ (or under-dispersion, if $\phi < 1$). The estimated ϕ is given earlier.

The parameters of the Poisson regression model are estimated based on the following **adjusted score** equations.

$$\begin{cases} \frac{\partial}{\partial \alpha_0} \frac{\sum_{j=1}^{n} [y_j(\beta_0+\beta_1 x_{1j}+\cdots+\beta_p x_{pj}) - \exp(\beta_0+\beta_1 x_{1j}+\cdots+\beta_p x_{pj})]}{\phi} & = & 0 \\ \frac{\partial}{\partial \alpha_1} \frac{\sum_{j=1}^{n} [y_j(\beta_0+\beta_1 x_{1j}+\cdots+\beta_p x_{pj}) - \exp(\beta_0+\beta_1 x_{1j}+\cdots+\beta_p x_{pj})]}{\phi} & = & 0 \\ \cdots & \cdots & \cdots \\ \frac{\partial}{\partial \alpha_p} \frac{\sum_{j=1}^{n} [y_j(\beta_0+\beta_1 x_{1j}+\cdots+\beta_p x_{pj}) - \exp(\beta_0+\beta_1 x_{1j}+\cdots+\beta_p x_{pj})]}{\phi} & = & 0 \end{cases}$$

Note that the above **score equations** are **not** derived from the loglikelihood function of the regular Poisson regression model. Unlike the likelihood-based goodness of fit measures such as *deviance*, *AIC*, and *SBC* measures should not be used in inference.

Apparently, the estimated regression coefficients are identical since the scale parameter ϕ will not impact the solution to the linear system. However, the variance $\mathrm{V}(\mathbf{Y}) = \phi \mathrm{V}(\hat{\mu})$. Thus, we now have a parameter that allows the variance to be larger or smaller than the mean by a multiplicative factor ϕ. Hence, it will affect the inference of QMLE of the regression coefficients

$$\hat{} \rightarrow \mathbf{N}[\ ,\ \phi(\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}]$$

That means the standard errors of $\hat{\beta}_i$ $(i = 0, 1, 2, \cdots, k)$ are different from the MLE in the regular Poisson regression. Because of this, the reported p-values are also different from those of the output of the regular Poisson regression model.

13.2.3 Numerical Example

Next, we use **glm()** to fit the quasi-Poisson model and compare its output with that of the regular Poisson regression.

```
m2.quasi.pois = glm(BrooklynBridge ~ Day + HighTemp + LowTemp + Precipitation,
                    family = quasipoisson, offset = log(Total),  data = cyclist)
```

```
knitr::include_graphics("img10/w10-QuasiPoisOutput.jpg")
```

We can see from the output of the quasi-likelihood-based Poisson regression that the dispersion parameter is $\hat{\phi} = 5.420292$. Since the dispersion parameter is significantly different from 1, the p-values in the output of the Poisson regression model are not reliable. The main effect is the substantially larger errors for the

```
glm(formula = BrooklynBridge ~ Day + HighTemp + LowTemp + Precipitation,
    family = quasipoisson, data = cyclist, offset = log(Total))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.3659  -1.4340   0.1622   1.3086   3.3408

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   -2.0192918  0.0913379 -22.108 1.58e-15 ***
DayMonday      0.0121581  0.0312328   0.389 0.701188
DaySaturday    0.1211787  0.0342508   3.538 0.002065 **
DaySunday      0.1376210  0.0322396   4.269 0.000375 ***
DayThursday   -0.0001037  0.0268305  -0.004 0.996953
DayTuesday    -0.0048153  0.0302199  -0.159 0.874999
DayWednesday  -0.0100836  0.0287896  -0.350 0.729813
HighTemp       0.0068970  0.0020601   3.348 0.003204 **
LowTemp       -0.0078962  0.0024644  -3.204 0.004455 **
Precipitation -0.0346151  0.0260658  -1.328 0.199138
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for quasipoisson family taken to be 5.420292)

    Null deviance: 406.84  on 29  degrees of freedom
Residual deviance: 109.09  on 20  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 3
```

Figure 13.2: R glm() output of quasi-Poisson regression model

estimates (the point estimates do not change), and hence potentially changed the significance of explanatory variables.

We can manually compute the corrected standard errors in the quasi-Poisson model by adjusting the standard error from the Poisson standard errors using relation $SE_Q(\hat{\beta}) = SE(\hat{\beta}) \times \sqrt{\hat{\phi}}$. For example, considering the standard error of $\hat{\beta}_1$ (associated with dummy variable **DayMonday**), $SE(\hat{\beta}_1) = 0.0134153$, in the output of the regular Poisson regression model. The corresponding corrected standard error in the quasi-Poisson model is given by $\sqrt{5.420292} \times 0.0134153 = 0.03123286$, which is the same as the one reported in the quasi-Poisson model.

13.2.4 Summary and Concluding Remarks

We have introduced regular Poisson and quasi-Poisson regression modes in this and previous notes. The two models use the same formulation, but different estimations are used. The regular Poisson model is based on the likelihood estimation, all statistics reported in the out are valid. However, the quasi-Poisson model is **not** based on the standard maximum likelihood estimation, and not all reported statistics can be used for inference.

- Regular Poisson Model
  1. Assume Y to be a Poisson random variable.
  2. link function is $\log(\cdot)$, the model is defined to be $\log(\mu) = \beta_0 + \sum_{i=1}^{k} \beta_i x_i$
  3. Score equation: first-order partial derivative of the log-likelihood function
  4. Parameter estimation - MLE via Fisher scoring algorithm
  5. Regression coefficient is log risk ratio.
  6. Pearson and Deviance residuals are defined for model diagnosis.
  7. All likelihood-based statistics such as *R-square, AIC, and SBC* are valid and can be used as usual.
- Quasi-Poisson Model:
  1. Assume Y to be a Poisson random variable.
  2. link function is $\log(\cdot)$, the model is defined to be $\log(\mu) = \beta_0 + \sum_{i=1}^{k} \beta_i x_i$
  3. Score equation: first order partial derivative of the scaled log-likelihood function (*quasi-likelihood*).
  4. Parameter estimation - MLE via Fisher scoring algorithm
  5. Regression coefficient is log risk ratio.
  6. Pearson and Deviance residuals are defined for model diagnosis.
  7. All likelihood based statistics such as *R-square, AIC, SBC* are **theoretically invalid**.
- Which Model Should Be Used?
  1. In predictive modeling, both models will yield the same results.
  2. In the association analysis, the regular Poisson model should be used when dispersion is not an issue (i.e., ϕ is close to 1.). However, the

quasi-Poisson should be used when ϕ is significantly different from 1.

3. When there is no dispersion, could we simply use the quasi-Poisson in the association analysis? The answer is No. The reason is that the additional approximation was used to adjust the estimation of the standard error and the approximation also add rounding errors to the result. From the computational perspective, it uses more system resources.

13.3 Case Study I: Denmark Cities Lung Cancer Rates

This is a complete analysis of the case study of Denmark Cities Ling Cancer rate we started in the previous module.

13.3.1 Introduction

The World Health Organisation (WHO) statistics suggest that Denmark has the highest cancer rates in the world, with about 326 people out of every 100,000 developing cancer each year. The country is known to have a good record of diagnosing cancer but also has high rates of smoking among women and high levels of alcohol consumption.

```
knitr::include_graphics("img10/DenmarkCitiesMap.png")
```

In this case study, we use a data set that summarized the lung cancer incident counts (cases) per age group for four Danish cities from 1968 to 1971. The primary random response variable is lung cancer cases. The predictor variables are the age group and the total population size of the neighboring cities.

Table 13.3: First few records in the data set

city	age	pop	cases
Fredericia	40-54	3059	11
Horsens	40-54	2879	13
Kolding	40-54	3142	4
Vejle	40-54	2520	5
Fredericia	55-59	800	11
Horsens	55-59	1083	6

The data set was built in the R library {ISwR}.

```
library(knitr)
data(eba1977)
kable(head(eba1977), caption = "First few records in the data set")
```

```
# check the values of the variables in the data set
```

Since it's reasonable to assume that the expected count of lung cancer incidents is proportional to the population size, we would prefer to model the rate of incidents per capita. However, for the purpose of illustration, we will fit the Poisson regression model with both counts and rate of cancer rates.

13.3.2 Poisson Regression on Cancer Counts

We first build a Poisson frequency regression model and ignore the population size of each city in the data.

```
model.freq <- glm(cases ~ city + age, family = poisson(link = "log"), data = eba1977)
##
pois.count.coef = summary(model.freq)$coef
kable(pois.count.coef, caption = "The Poisson regression model for the counts of lung
      cancer cases versus the geographical locations and the age group.")
```

The above inferential table about the regression coefficients indicates both city and age are insignificant. This means, if we look at cancer count across the age group and city, there is no statistical evidence to support the potential discrepancy across the age groups and cities. However, this does not imply that the model is meaningless from the practical perspective since statistical significance is not equivalent the clinical importance. Moreover, the sample size could impact the statistical significance of some of the variables.

The other way to look at the model is the appropriateness model. The cancer counts are dependent on the population size. Ignoring the population size implies the information in the sample was not effectively used. In the next subsection, we model the cancer rates that involve the population size.

Table 13.4: The Poisson regression model for the counts of lung cancer cases versus the geographical locations and the age group.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	2.2437446	0.2036265	11.0189233	0.0000000
cityHorsens	-0.0984401	0.1812909	-0.5429952	0.5871331
cityKolding	-0.2270575	0.1877041	-1.2096561	0.2264109
cityVejle	-0.2270575	0.1877041	-1.2096561	0.2264109
age55-59	-0.0307717	0.2480988	-0.1240298	0.9012916
age60-64	0.2646926	0.2314278	1.1437369	0.2527328
age65-69	0.3101549	0.2291839	1.3533017	0.1759593
age70-74	0.1923719	0.2351660	0.8180261	0.4133423
age75+	-0.0625204	0.2501222	-0.2499593	0.8026188

Table 13.5: Poisson regression on the rate of the the cancer rate in the four Danish cities adjusted by age.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-5.6320645	0.2002545	-28.124529	0.0000000
cityHorsens	-0.3300600	0.1815033	-1.818479	0.0689909
cityKolding	-0.3715462	0.1878063	-1.978348	0.0478895
cityVejle	-0.2723177	0.1878534	-1.449629	0.1471620
age55-59	1.1010140	0.2482858	4.434463	0.0000092
age60-64	1.5186123	0.2316376	6.555985	0.0000000
age65-69	1.7677062	0.2294395	7.704455	0.0000000
age70-74	1.8568633	0.2353230	7.890701	0.0000000
age75+	1.4196534	0.2502707	5.672472	0.0000000

The other way to look at the model is goodness of the model. The cancer counts are dependent on the population size. Ignoring the population size implies the information in the sample was not effectively used. In the next subsection, we model the cancer rates that involve the population size.

13.3.3 Poisson Regression on Rates

The following model assesses the potential relationship between cancer death rates and age. This is the primary interest of the model. We also want to adjust the relationship be the potential neighboring cities.

```
model.rates <- glm(cases ~ city + age, offset = log(pop),
                   family = poisson(link = "log"), data = eba1977)
kable(summary(model.rates)$coef, caption = "Poisson regression on the rate of the
      the cancer rate in the four Danish cities adjusted by age.")
```

Table 13.6: Quasi-Poisson regression on the rate of the cancer rate in the four Danish cities adjusted by age.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-5.6320645	0.2002545	-28.124529	0.0000000
cityHorsens	-0.3300600	0.1815033	-1.818479	0.0689909
cityKolding	-0.3715462	0.1878063	-1.978348	0.0478895
cityVejle	-0.2723177	0.1878534	-1.449629	0.1471620
age55-59	1.1010140	0.2482858	4.434463	0.0000092
age60-64	1.5186123	0.2316376	6.555985	0.0000000
age65-69	1.7677062	0.2294395	7.704455	0.0000000
age70-74	1.8568633	0.2353230	7.890701	0.0000000
age75+	1.4196534	0.2502707	5.672472	0.0000000

The above table indicates that the log of cancer rate is not identical across the age groups and among the four cities. To be more specific, the log rates of Fredericia (baseline city) were higher than in the other three cities. The youngest age group (45-55) has the lowest log rate. The regression coefficients represent the change of log rate between the associate age group and the reference age group. The same interpretation applies to the change in log rate among the cities.

13.3.4 Quasi-Poisson Rate Model

The above two Poison models assume that there is no dispersion issue in the model. The quasi-Poisson through `glm()` returns the dispersion coefficient.

```
quasimodel.rates <- glm(cases ~ city + age, offset = log(pop),
                        family = quasipoisson, data = eba1977)
kable(summary(model.rates)$coef, caption = "Quasi-Poisson regression on the rate of the
```

The dispersion index can be extracted from the quasi-Poisson object with the following code

```
ydif=eba1977$cases-exp(model.rates$linear.predictors)  # diff between y and yhat
prsd = ydif/sqrt(exp(model.rates$linear.predictors))   # Pearson residuals
phi = sum(prsd^2)/15          # Dispersion index: 24-9 = 15
kable(cbind(Dispersion = phi))
```

Dispersion
1.504109

13.3.5 Final Working Model

The dispersion index is 1.56. It is slightly dispersed. We stay with the regular Poisson regression model.

The intercept represents the **baseline log-cancer rate** (of baseline *age group 44-55* in the baseline city *Fredericia*). The actual rate is $\exp(-5.6321) \approx 0.36\%$ which is close to the recently reported rate of the country by WHO. The slope -0.3301 is the difference of the log-rates between baseline city Fredericia and the city of Horsens at any given age group, to be more specific, $\log(R_{\text{Horsen}}) - \log(R_{\text{Fredericia}}) = -0.3301$ which is equivalent to

$$\log\left(\frac{R_{\text{Horsen}}}{R_{\text{Fredericia}}}\right) = -0.3301 \quad \Rightarrow \quad \frac{R_{\text{Horsen}}}{R_{\text{Fredericia}}} = e^{-0.3301} \approx 0.7188518.$$

This means, with fixed age groups, the cancer rate in Horsens is about 28% lower than that in Fredericia. Next, we look at the coefficient 1.4197 associated with age group 75+. For any given city,

$$\log\left(\frac{R_{\text{age75+}}}{R_{\text{age45-54}}}\right) = 1.4197 \quad \Rightarrow \quad \frac{R_{\text{age75+}}}{R_{\text{age45-54}}} = e^{1.41971} \approx 4.135921.$$

This implies that the cancer rate in the age group 75+ is 4.14 times that of the baseline age group of 45-54.

13.3.6 Some Visual Comparisons

The inferential tables of the Poisson regression models in the previous sections give numerical information about the potential discrepancy across the age group and among the cities. Next, we create a graph to visualize the relationship between cancer rate and age across cities.

First of all, every city has a trend line that reflects the relationship between the cancer rate and age. We next find the rates of combinations of city and age group based on the following working rate model.

$$\text{log-rate} = -5.6321 - 0.3301 \times \text{cityHorsens} - 0.3715 \times \text{cityKolding} - 0.2723 \times \text{cityVejle} + 1.1010 \times \text{age55-59} + 1.5186 \times \text{age60-6}$$

Or equivalently, we can write the rate model as

$$\textit{rate} = \exp(-5.6321 - 0.3301 \times \text{cityHorsens} - 0.3715 \times \text{cityKolding} - 0.2723 \times \text{cityVejle} + 1.1010 \times \text{age55-59}) \times \exp(1.5186 \times \text{a}$$

To make the visual representation of the output, we tabulate cancer rates of the corresponding combinations of city and age group in the following calculation

based on the regression equation with coefficients given in above table 3. Note that all variables in the model are indicator variables. Each of these indicator variables takes only two possible values: 0 and 1.

For example, $\exp(-5.632)$ gives the cancer rate of the baseline city, Fredericia, and the baseline age group [45-54]. $\exp(-5.632 + 1.101)$ gives the cancer rate of baseline city, Fredericia, and age group [55-59]. Following the same pattern, we can find the cancer rate for each combination of the city and age group.

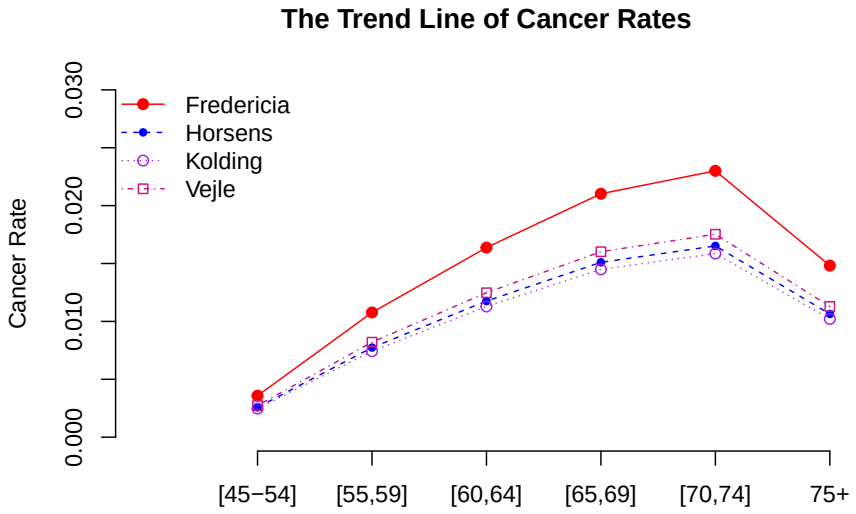
The following table calculates the **estimated** cancer rates of cities Fredericia and Horsens across age groups. The rates for other cities can be similarly calculated.

Age	Fredericia's Rate	Horsens' Rates
[40 - 49]	$\exp(-5.632)$	$\exp(-5.632 - 0.331)$
[55 - 59]	$\exp(-5.632 + 1.101)$	$\exp(-5.632 - 0.331 + 1.101)$
[60 - 64]	$\exp(-5.632 + 1.52)$	$\exp(-5.632 - 0.331 + 1.52)$
[65 - 69]	$\exp(-5.632 + 1.77)$	$\exp(-5.632 - 0.331 + 1.77)$
[70 - 74]	$\exp(-5.632 + 1.86)$	$\exp(-5.632 - 0.331 + 1.86)$
75+	$\exp(-5.632 + 1.42)$	$\exp(-5.632 - 0.331 + 1.42)$

We use age as the horizontal axis and the estimated cancer rates (in the above table) as the vertical axis to make the trend lines for each of the four cities using the following code.

```
# Fredericia
Fredericia = c(exp(-5.632), exp(-5.632+1.101),
               exp(-5.632+1.52),exp(-5.632+1.77),
               exp(-5.632+1.86),exp(-5.632+1.42))
# Horsens
Horsens = c(exp(-5.632-0.331), exp(-5.632-0.331+1.101),
            exp(-5.632-0.331+1.52),exp(-5.632-0.331+1.77),
            exp(-5.632-0.331+1.86),
            exp(-5.632-0.331+1.42))
# Kolding
Kolding= c(exp(-5.632-0.372), exp(-5.632-0.372+1.101),
           exp(-5.632-0.372+1.52),exp(-5.632-0.372+1.77),
           exp(-5.632-0.372+1.86), exp(-5.632-0.372+1.42))
# Vejle
Vejle = c(exp(-5.632-0.272), exp(-5.632-0.272+1.101),
          exp(-5.632-0.272+1.52),exp(-5.632-0.272+1.77),
          exp(-5.632-0.272+1.86), exp(-5.632-0.272+1.42))
minmax = range(c(Fredericia,Horsens,Kolding,Vejle))
####
```

```
plot(1:6,Fredericia, type="l", lty =1, col="red", xlab="",
               ylab="Cancer Rate", xlim=c(0,6), ylim=c(0, 0.03), axes=FALSE )
title("The Trend Line of Cancer Rates")
axis(2)
axis(1, labels=c("[45-54]","[55,59]","[60,64]","[65,69]","[70,74]","75+"),
            at = 1:6)
points(1:6,Fredericia, pch=19, col="red")
##
lines(1:6, Horsens, lty =2, col="blue")
points(1:6, Horsens, pch=20, col="blue")
##
lines(1:6, Kolding, lty =3, col="purple")
points(1:6, Kolding, pch=21, col="purple")
###
lines(1:6, Vejle, lty =4, col="mediumvioletred")
points(1:6, Vejle, pch=22, col="mediumvioletred")
##
legend("topleft", c("Fredericia","Horsens", "Kolding", "Vejle" ),
                  pch=19:22, lty=1:4,  bty="n",
        col=c("red", "blue", "purple", "mediumvioletred"))
```



13.3.7 Discussions and Conclusions

Several conclusions we can draw from the output of the regression models.

Table 13.8: The Poisson regression model for the counts of lung cancer cases versus the geographical locations, population size, and age group.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	11.7495934	8.8151328	1.3328890	0.1825682
cityHorsens	0.1832573	0.3192679	0.5739922	0.5659731
cityKolding	-0.0483001	0.2519622	-0.1916957	0.8479806
cityVejle	-0.1679335	0.1964757	-0.8547289	0.3927012
age55-59	-1.3842350	1.2728775	-1.0874849	0.2768226
age60-64	-1.2366489	1.4049520	-0.8802073	0.3787470
age65-69	-1.4377681	1.6310051	-0.8815228	0.3780349
age70-74	-1.8048920	1.8607922	-0.9699589	0.3320670
age75+	-1.8383162	1.6587773	-1.1082357	0.2677600
log(pop)	-1.2095837	1.1227191	-1.0773698	0.2813151

The regression model based on the cancer count is not appropriate since the information on the population size is a key variable in the study of cancer distribution. Simply including the population size in the regression model will reduce the statistical significance of age. See the following output of the fitted Poisson regression model of count adjusted by population size.

```
model.freq.pop <- glm(cases ~ city + age + log(pop), family = poisson(link = "log"),
                      data = eba1977)
##
pois.count.coef.pop = summary(model.freq.pop)$coef
kable(pois.count.coef.pop, caption = "The Poisson regression model for
         the counts of lung cancer cases versus the geographical locations,
         population size, and age group.")
```

We can see from the above output the adding population size to the model

The cancer rate in Fredericia is significantly higher than in the other three cities. It seems that there is no significant difference between Horsens, Kolding, and Vejle. The reason why Fredericia has a higher cancer rate needs further investigation with additional information.

There is a curve linear relationship between age and the cancer rate. The cancer rate increases as age increase. However, the rate starts decreasing after 75. This pattern is consistent with the clinical studies since lung cancer patients were mostly diagnosed between 65-70. It is rare to see lung cancer patients aged under 45.

The last statistical observation is that there is no interaction effect between the age groups and the geographic locations. The rate curves are “parallel”.

This is only a small data set with limited information. All conclusions in this report are only based on the given data set.

13.4 Case Study II - Ph.D. and Mentor's Productivity

In this case study, we use data from Long (1990) on the number of publications produced by Ph.D. biochemists to illustrate the application of Poisson models. The variables in the data set are listed below.

13.4.1 Variable Description

- articles: integer. articles in the last three years of Ph.D.
- gender: factor. coded one for females.
- married: factor. coded one if married.
- kids: integer. the number of children under age six.
- prestige: numeric.the prestige of Ph.D. program
- mentor: integer. articles by the mentor in last three years

13.4.2 Research Question

We want to assess how factors affect the number of articles published in the last three years in the Ph.D. programs.

13.4.3 Data Preparation and Variable Processing

Variable **kids** is a discrete variable. We create a frequency table of **kids** and found that 16 of 915 Ph.D. students had 3 kids. After additional exploratory analysis. We decide to dichotomize **kids** and redefine a new variable under the name **newkids**.

```
phd=read.table("img10/w10-ph-data.txt",skip=10, header=TRUE )[,-1] # drop the ID variable
id.3 = which(phd$kids > 0)
newkids = phd$kids
newkids[id.3] = 1
phd$newkids = newkids
```

13.4.4 Poison Regression Modeling

We build both the regular Poisson and Quasi-Poisson regression models and extract the dispersion parameter to decide which model should be used as working model.

Table 13.9: Dispersion parameter

dispersion
1.841565

Table 13.10: Summary statistics of quasi-poisson regression model

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.4579404	0.1290407	3.5488055	0.0004068
genderWomen	-0.2179247	0.0742536	-2.9348721	0.0034208
marriedSingle	-0.1516973	0.0855315	-1.7735846	0.0764666
factor(newkids)1	-0.2495633	0.0859576	-2.9033304	0.0037815
prestige	0.0102754	0.0359069	0.2861675	0.7748150
mentor	0.0258173	0.0027397	9.4233384	0.0000000

13.5

```
## phd=read.table("w10-ph-data.txt",skip=10, header=TRUE )[,-1] # drop the ID variable
## Regular Poison Model
pois.model = glm(article ~ gender + married + factor(newkids) + prestige + mentor,
                 family = poisson(link="log"), data =phd)
## Quasi Poisson or dispersed Poisson model
quasi.model = glm(article ~ gender + married + factor(newkids) + prestige + mentor,
                  family = quasipoisson, data =phd)
## Extracting dispersion parameter
SE.q = summary(quasi.model)$coef[2,2]
SE = summary(pois.model)$coef[2,2]
dispersion = (SE.q/SE)^2
disp = cbind(dispersion = dispersion)
kable(disp, caption="Dispersion parameter", align = 'c')
```

The dispersion parameter 1.829006 indicates that the Poisson model is inappropriate. We need to correct the dispersion issue. The quasi-likelihood-based Poisson model is one option.

Next, we summarize the inferential statistics about the regression coefficients in the following table

```
SE.quasi.pois = summary(quasi.model)$coef
kable(SE.quasi.pois, caption = "Summary statistics of quasi-poisson regression model")
```

In the above quasi-Poisson regression, variable prestige is insignificant (p-value = 0.72). The p-value for testing the significance of the variable married is 0.079. We refit the quasi-Poisson model by dropping **prestige** and **married**.

Table 13.11: Inferential statistics of the Poisson regression coefficients in the final working model.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.4238445	0.0645972	6.561345	0.0000000
genderWomen	-0.2332786	0.0738848	-3.157329	0.0016446
factor(newkids)1	-0.1796153	0.0767849	-2.339200	0.0195404
mentor	0.0257762	0.0026586	9.695432	0.0000000

```
quasi.model.02 = glm(article ~ gender + factor(newkids) + mentor,
                     family = quasipoisson, data =phd)
kable(summary(quasi.model.02)$coef, caption = "Inferential statistics of
the Poisson regression coefficients  in the final working model.")
```

The above model will be used as the final model. The interpretation of the regression coefficient of the Poisson model is not as straightforward as that in the linear regression models since the response variable in the model is at a log-scale.

For example, the coefficient associated with gender is -0.233. This is the estimated Poisson regression coefficient comparing females to males, given the other variables are held constant in the model. The difference in the logs of expected publications is expected to be 0.2332786 units lower for females compared to males while holding the other variables constant in the model. This is still not easy to understand for the general audience.

13.5.1 Some Visual Comparisons

Next, we make a visualization to show how the explanatory variables in the final working model affect the **actual** number of publications of doctoral students.

To this end, we classify all Ph.D. students in the following four groups defined by **gender** and **status** of having at least one child:

phd.m0 = male and had no child

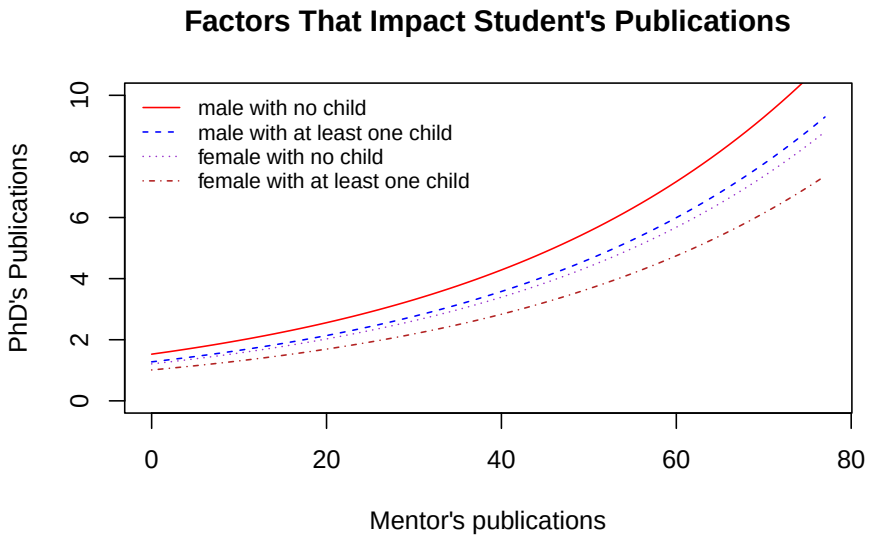
phd.m1 = male and had at least one child

phd.f0 = female and had no child

phd.f1 = female and had at least one child

Next, We exponentiate the log-count of publications of Ph.D. students to the actual number of publications and then make graphs to show the relationship between doctoral students and their mentors in terms of the number of publications in each of the groups defined above.

```
mentors =  range(phd$mentor)[1]:range(phd$mentor)[2]
phd.m0 = 0.42384447 + 0.02577624*mentors
phd.m1 = 0.42384447 - 0.17961531 + 0.02577624*mentors
phd.f0 = 0.42384447 -0.23327860 + 0.02577624*mentors
phd.f1 = 0.42384447 -0.23327860 -0.17961531 + 0.02577624*mentors
##
plot(mentors, exp(phd.m0), ylim=c(0,10),
     type = "l",
     col = "red",
     lty = 1,
     ylab = "PhD's Publications",
     xlab = "Mentor's publications",
     main = "Factors That Impact Student's Publications")
lines(mentors, exp(phd.m1), col = "blue", lty = 2)
lines(mentors, exp(phd.f0), col = "darkorchid", lty = 3)
lines(mentors, exp(phd.f1), col = "firebrick", lty = 4)
legend("topleft", c("male with no child", "male with at least one child",
                    "female with no child", "female with at least one child"),
       col=c("red", "blue", "darkorchid", "firebrick"),  lty=1:4, bty="n", cex=0.8)
```



We can see the relationship between the number of publications of doctoral students and other factors.

1. the number of publications of doctoral students is positively associated with their mentor publication.

2. Male doctoral students with no kid published more articles than those who had at least one kid. Female doctoral students also have the same pattern.
3. Overall, male doctoral students published more than female students.

13.5.2 Conclusions

The Poisson regression model is used for modeling counts/rates-based data sets. If the model is appropriate, its results are explainable and comparable and backed by statistical theory.

If Poisson regression is not appropriate, we can consider other models depending on the situation. The complex alternatives to the Poisson regression model that can be considered are negative binomial regression, zero-inflated regression models, random-forest-based regression models, and neural-network-based regression models. The last two models are "black-box" models because of the lack of interpretability.

13.6 Analysis Assignment

This week's assignment is to revise your analysis in Week #9 by adding a new section to include a quasi-Poisson model to the report. In this new analysis, we modify the predictor variables in the following ways.

1. Instead of using two variables **HightTemp** and **LowTemp** in the model, we will use the new variable **AvgTemp = (HighTemp + LowTemp)/2**.
2. Discretize **Precipitation** using the following definition: if **Precipitation** = 0, then NewPrecip = 0; if **Precipitation** > 0, then NewPrecip = 1.

The dispersed Poisson regression model will have three predictor variables: **Day**, **AvgTemp**, and **NewPrecip**.

Here are the steps for building Poisson (quasi-Poisson) model (similar to the two case studies):

1. Fit a quasi-Poisson regression model on the counts of cyclists who entered and left the Bridge in your data set.
2. Report the value of the estimated dispersion parameter and based on the value determine whether the regular Poisson model or the quasi-Poisson should be used as the final model. The two models have the same estimated coefficients by different p-values.
3. Make a visualization to show the relationship between the number of cyclists who entered and left the bridge and the related predictor variables.

Chapter 14

Concepts of Time Series Forecasting

A univariate time series is a sequence of data values with certain random behaviors that occur in successive order over some period of time. A time series can be constructed by any data that is measured over time at evenly-spaced intervals. Some examples are historical stock prices, earnings, GDP, rainfalls, university enrollments, daily counts of ER patients, annual death counts due to traffic accidents in a country, inflation rates, or other sequences of financial or economic data that can be analyzed as a time series.

A forecast is a prediction of some future events. Forecasting problems can be found in many applied areas such as business and industry, economics, finance, environmental sciences, social sciences, political sciences, etc., and can be categorized into

1. Short-term forecasting is when a prediction is only made for a few periods ahead (hours, days, weeks).
2. Medium-term in which one or two years of forecasting will be made.
3. Long-term forecasting in which predictions for several years ahead will be made.

We have discussed different regression models such as linear models and logistic regression models. We also discussed predictions using the fitted regression models. In practice, if the data are collected over time, then the aforementioned regression models are not valid for this type of data since the data values are serially correlated. New models that address time series are needed.

For time-series data, the typical scatter plot of the data values versus the time point at which the data values are observed is shown below.

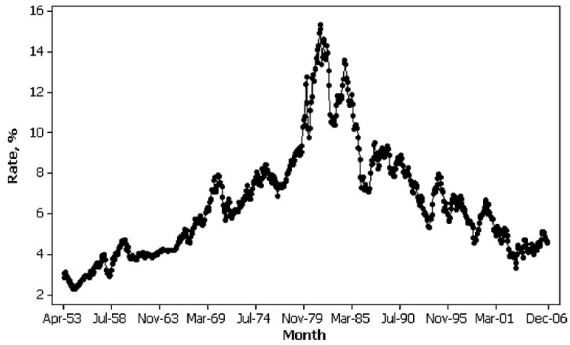


FIGURE 1.1 Time series plot of the market yield on U.S. Treasury Securities at 10-year constant maturity. (Source: U.S. Treasury.)

Many business applications of forecasting utilize daily, weekly, monthly, quarterly, or annual data, but any reporting interval may be used.

The data may be instantaneous, such as the viscosity of a chemical product at the point in time where it is measured; it may be cumulative, such as the total sales of a product during the month; or it may be a statistic that in some way reflects the activity of the variable during the period, such as the daily closing price of a specific stock on the New York Stock Exchange.

- **Methods of forecasting**: There are two major methods for forecasting:
  - Quantitative forecasting methods that make formal use of historical data to build a mathematical/statistical model to project into the future.
  - Qualitative forecasting methods are used when little data are available (new product introduction). In this case, expert opinion is often used.

There are two modes for analyzing time series: frequency domain and time domain approaches. We restrict our discussion within the time domain to avoid more advanced mathematical tools in this class.

- **Statistical forecasting models**:
  - Regression methods.
  - Smoothing methods.
  - Formal time series analysis methods - ARIMA Box-Jenkins methodology.
  - Dynamic regression methods.
- **Some terminology**

- Point forecast or point estimate - a single predicted value (no information on the variation of the predicted value)
- Prediction interval - interval prediction, information on accuracy is available.
- Forecast horizon or lead time - based on the fitted model, the number of periods specified to be predicted.

The forecast horizon is the length of time into the future for which forecasts are to be prepared.

14.1 Types of Time Series Data

14.1.1 Uncorrelated data, constant process model: random time series

Random time series is the result of many influences that act independently to yield nonsystematic and non-repeating patterns about some average value.

A purely random series has a constant mean and no systematic patterns. Simply averaging the models is often the best and the most accurate way to forecast them.

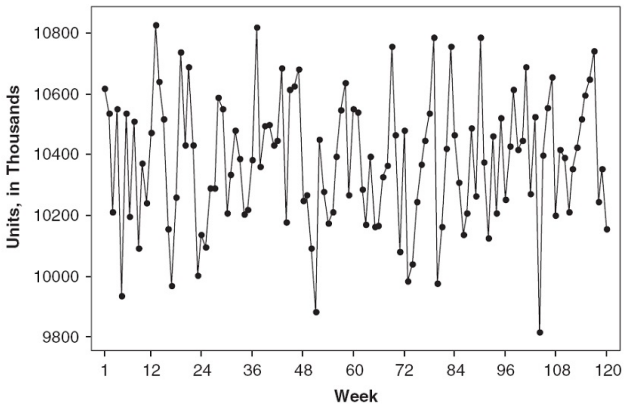


FIGURE 1.2 Pharmaceutical product sales.

A special uncorrelated process: **white noise process**, $x_t \rightarrow_{i.i.d} N(0, \sigma^2)$. That is, a sequence $\{x_t\}$, $(i = 1, 2, \cdots,)$ is a white noise process if each value in the sequence has a) zero-mean; b) constant conditional variance; and c) is uncorrelated with all other realizations.

14.1.2 Autocorrelated data

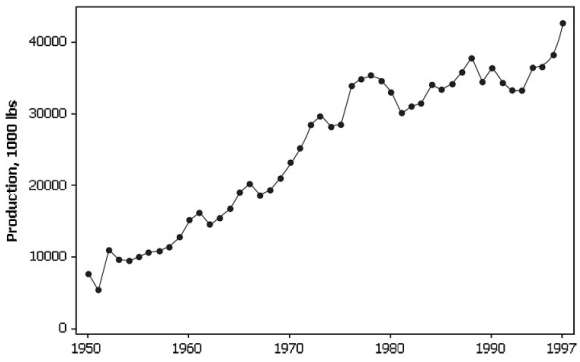


FIGURE 1.4 The U.S. annual production of blue and gorgonzola cheeses. (Source: USDA–NASS.)

The correlation measures the degree of dependence or association between two variables.

The term auto-correlation means that the value of a series in one time period is related to the value itself in previous periods.

14.1.3 Trend Data

A trend is a general increase or decrease in a time series that lasts for approximately seven or more periods. it can be a period-to-period increase or decrease that follows a straight line - this kind of pattern is called a linear trend.

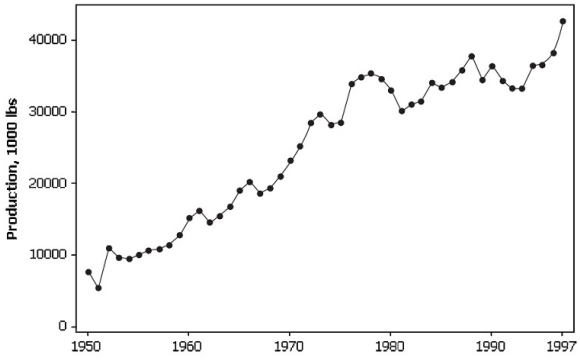


FIGURE 1.4 The U.S. annual production of blue and gorgonzola cheeses. (Source: USDA–NASS.)

Trends are not necessarily linear because there are a large number of nonlinear causal influences that yield nonlinear series.

14.1.4 Seasonal Data

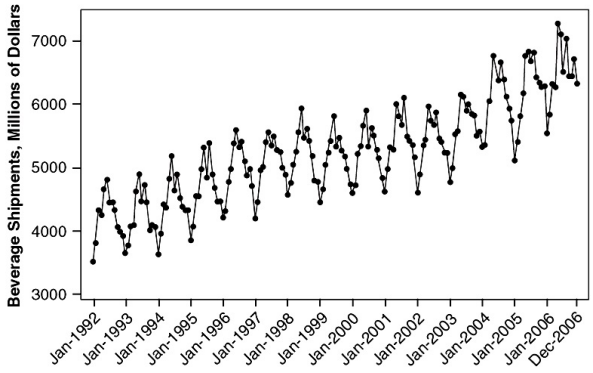


FIGURE 1.5 The U.S. beverage manufacturer monthly product shipments, unadjusted. (Source: U.S. Census Bureau.)

The seasonal series result from events that are periodic and recurrent (e.g., monthly, quarterly, changes recurring each year). The common seasonal influences are climate, human habits, holidays, repeating promotions, and so on.

14.1.5 Nonstationary Data

A stationary time series is one whose properties do not depend on the time at which the series is observed.

More formally, a strictly stationary stochastic process is one where given $t_1, \cdots, t_l$ the joint statistical distribution of $X_{t_1}, \cdots, X_{t_l}$ is NOT the same as the joint statistical distribution of $X_{t_1+\tau}, ..., X_{t_l+\tau}$ for all l and τ.

Thus, time series with trends, or with seasonality, are not stationary — the trend and seasonality will affect the value of the time series at different times.

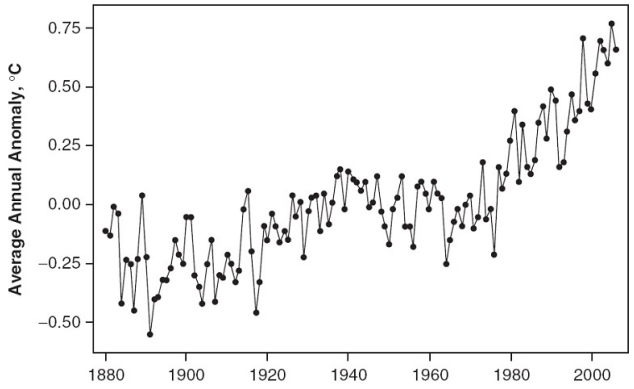


FIGURE 1.6 Global mean surface air temperature annual anomaly. (Source: NASA–GISS.)

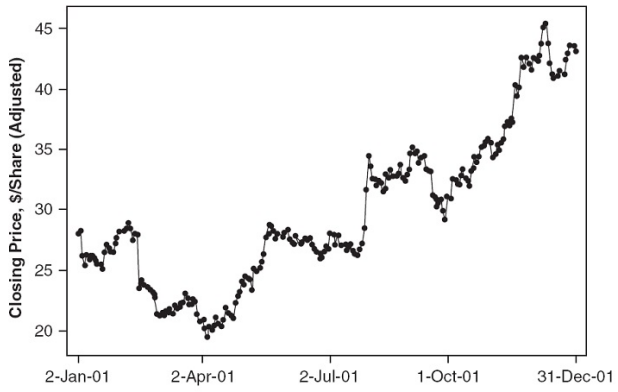


FIGURE 1.7 Whole Foods Market stock price, daily closing adjusted for splits.

However, a time series with cyclic behavior (but with no trend or seasonality) is stationary.

14.1.6 Atypical Event (Outliers)

The analysis of past data can be made very complex when the included values are not typical of the past or future.

These non-typical values are called outliers, which are very large or small observations that are not indicative of repeating past or future patterns. Outliers occur because of unusual events.

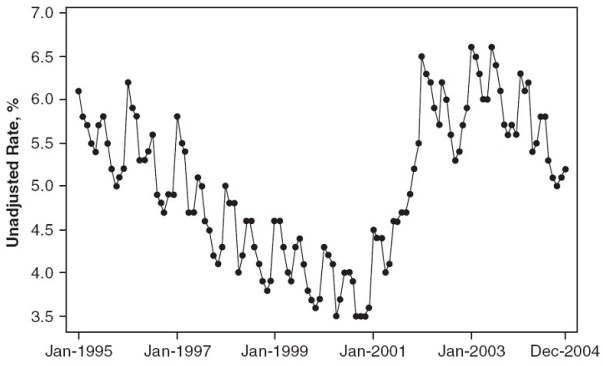


FIGURE 1.8 Monthly unemployment rate—full-time labor force, unadjusted. (Source: U.S. Department of Labor–BLS.)

14.2 Some Technical Terms and Baseline Forecasting Models

There are many forecasting models for different time series models. Some of them are extremely simple and surprisingly effective. In this module, we introduce some naive methods as baseline methods to understand some basic concepts and technical terms before introducing formal forecasting models.

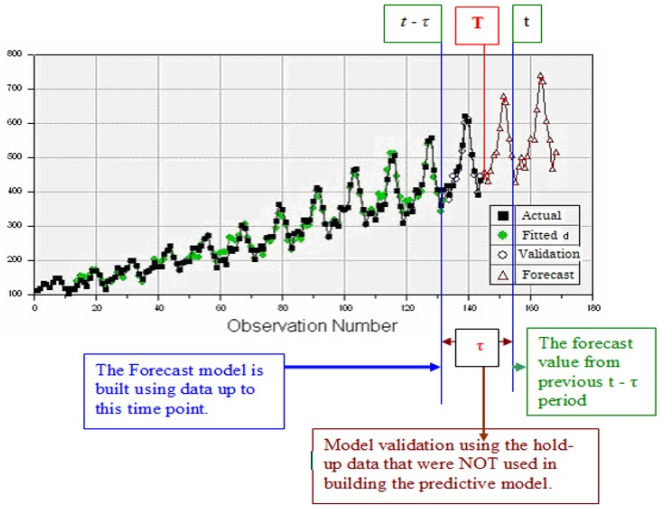
14.2.1 Some Definitions

Notation of a time series data: suppose there are T periods of data available with T the most recent period. The observed data values will be denoted by y_t for $t = 1, 2, \cdots, T$.

Forecast v.s. fitted values: in time series analysis, we distinguish between forecast (predicted) values from the fitted value:

- **Forecast value**: the value was produced at some previous period, say, $t - \tau$ (the predictive model on which the forecasting will be made should be built using the data up to $t - \tau$).

- **Fitted value**: The values produced by the fitted model at historical time points.



Seasonal/Cycle Frequency is used to define time series objects for forecasting models in R library **{forecast}**. In the R function **ts()**, the argument **frequency** needs to be specified.

The "**frequency**" is the number of observations **per seasonal cycle**. When using the ts() function in R, the following choices should be used.

Data	frequency
Annual	1
Quarterly	4
Monthly	12
Weekly	52

Actually, there are not 52 weeks in a year, but 365.25/7 = 52.18 on average. But most functions that use **ts** objects require integer frequency.

Once the frequency of observations is smaller than a week, then there is usually more than one way of handling the frequency. For example, data *observed every minute* might have an **hourly seasonality** (frequency=60), a **daily seasonality** (frequency=24x60=1440), a **weekly seasonality** (frequency=24x60x7=10080), and an **annual seasonality** (frequency=24x60x365.25=525960). If we want to use a **ts** object, then we need to decide which of these is the most important.

14.2.2 Baseline Methods

Working Data Set: The Nile dataset was selected to expand knowledge of time series analysis. The Nile dataset contains 100 annual readings of the Nile

River at Aswan from the years 1871 to 1970 (Anonymous, n.d.-a). Time series analysis and modeling are concerned with data elements that are ordered in a temporal fashion. The Nile data set is analyzed using various time series approaches and the results are discussed.

```
1120, 1160, 963, 1210, 1160, 1160, 813, 1230, 1370, 1140, 995, 935, 1110, 994, 1020, 960,
1180, 799, 958, 1140, 1100, 1210, 1150, 1250, 1260, 1220, 1030, 1100, 774, 840, 874, 694,
940, 833, 701, 916, 692, 1020, 1050, 969, 831, 726, 456, 824, 702, 1120, 1100, 832, 764,
821, 768, 845, 864, 862, 698, 845, 744, 796, 1040, 759, 781, 865, 845, 944, 984, 897, 822,
1010, 771, 676, 649, 846, 812, 742, 801, 1040, 860, 874, 848, 890, 744, 749, 838, 1050,
918, 986, 797, 923, 975, 815, 1020, 906, 901, 1170, 912, 746, 919, 718, 714, 740
```

When fitting time series models in R, most of the R functions for the time series require a time series object which has a special structure and other features. For a given numeric vector, we can use the R function **ts()** in the library **forecast**.

```
nile.vec = c(1120, 1160, 963, 1210, 1160, 1160, 813, 1230, 1370, 1140, 995, 935, 1110, 994,
1020, 960, 1180, 799, 958, 1140, 1100, 1210, 1150, 1250, 1260, 1220, 1030, 1100, 774, 840,
874, 694, 940, 833, 701, 916, 692, 1020, 1050, 969, 831, 726, 456, 824, 702, 1120, 1100, 832,
764, 821, 768, 845, 864, 862, 698, 845, 744, 796, 1040, 759, 781, 865, 845, 944, 984, 897,
822, 1010, 771, 676, 649, 846, 812, 742, 801, 1040, 860, 874, 848, 890, 744, 749, 838, 1050,
918, 986, 797, 923, 975, 815, 1020, 906, 901, 1170, 912, 746, 919, 718, 714, 740)
##
nile.ts = ts(nile.vec,
             start = 1871,
             end = 1970,
             frequency =1  # one observation per year.
                           # if there is a weekly seasonal pattern,
                           # then the frequency = 52
             )
 nile.ts
```

```
## Time Series:
## Start = 1871
## End = 1970
## Frequency = 1
##   [1] 1120 1160  963 1210 1160 1160  813 1230 1370 1140  995  935 1110  994 1020
##  [16]  960 1180  799  958 1140 1100 1210 1150 1250 1260 1220 1030 1100  774  840
##  [31]  874  694  940  833  701  916  692 1020 1050  969  831  726  456  824  702
##  [46] 1120 1100  832  764  821  768  845  864  862  698  845  744  796 1040  759
##  [61]  781  865  845  944  984  897  822 1010  771  676  649  846  812  742  801
##  [76] 1040  860  874  848  890  744  749  838 1050  918  986  797  923  975  815
##  [91] 1020  906  901 1170  912  746  919  718  714  740
```

This data set is included in library{forecast} as a time series object. We can view the data values in this time series by typing *Nile* to view the time series object.

Moving Average Method

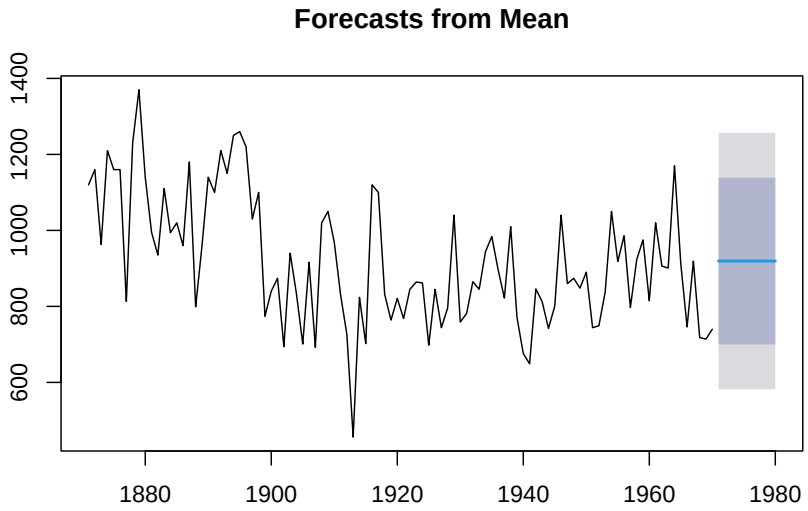
The moving average method forecasts all future values to be equal to the average (or "mean") of the historical data. If we let the historical data be denoted by $\{y_1, y_2, \cdots, y_T\}$, then we can write the forecasts as

$$\hat{y}_{T+h|T} = (y_1 + y_2 + \cdots + y_T)/T$$

$\hat{y}_{T+h|T}$ is the estimated value of $y_{T+h|T}$ based on $\{y_1, y_2, \cdots, y_T\}$.

Example 1. We only use the basic moving average on the Nile River data. The following plot shows the pattern of the time series.

```
nile.fcast <- meanf(Nile, h=10)
plot(nile.fcast)
```



The forecasted values for 10 future periods (h = 10 is called the forecast horizon) are given below

```
result = as.data.frame(nile.fcast)
result
```

```
##      Point Forecast    Lo 80   Hi 80    Lo 95    Hi 95
## 1971         919.35 699.9303 1138.77 581.8912 1256.809
## 1972         919.35 699.9303 1138.77 581.8912 1256.809
## 1973         919.35 699.9303 1138.77 581.8912 1256.809
## 1974         919.35 699.9303 1138.77 581.8912 1256.809
```

```
## 1975          919.35 699.9303 1138.77 581.8912 1256.809
## 1976          919.35 699.9303 1138.77 581.8912 1256.809
## 1977          919.35 699.9303 1138.77 581.8912 1256.809
## 1978          919.35 699.9303 1138.77 581.8912 1256.809
## 1979          919.35 699.9303 1138.77 581.8912 1256.809
## 1980          919.35 699.9303 1138.77 581.8912 1256.809
```

Naive method

For naive forecasts, we simply set all forecasts to be the value of the last observation. That is,

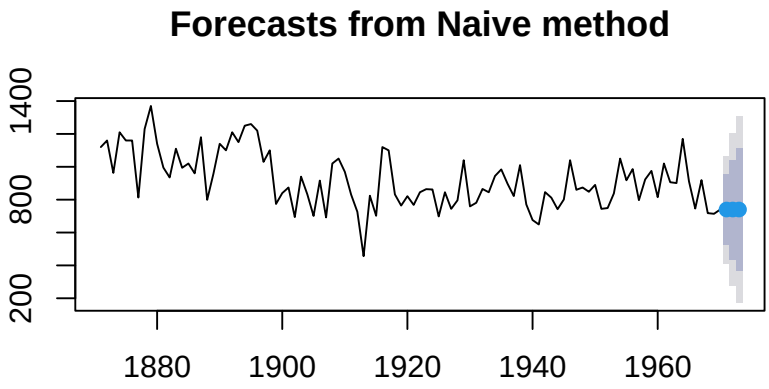
$$\hat{y}_{T+h|T} = y_T$$

The naive forecast method works really well on the time series data generated from a process in which the next value in the sequence is a modification of the previous value in the sequence.

Example 2. We will use the naive forecast method on the Nile River data.

The forecasted values for the three future periods and the forecasted intervals are plotted in the following figure.

```
naive.nile = naive(Nile, h=3)
plot(naive.nile)
```



The corresponding values and confidence intervals are given in the following table.

Table 14.2: Naive forecasting method on Nile River data

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
1971	740	525.5648	954.4352	412.0497	1067.950
1972	740	436.7429	1043.2571	276.2083	1203.792
1973	740	368.5874	1111.4126	171.9735	1308.027

```
kable(naive.nile, caption = "Naive forecasting method on Nile River data")
```

14.2.3 Seasonal Naive Method

The values of seasonal time series data are influenced by seasonal factors (e.g., the quarter of the year, the month, or the day of the week). seasonal time series are also called periodic time series.

If a data set has a seasonal pattern such as weekly and monthly patterns, the **moving average** and **naive** methods work poorly. We need a method to capture the seasonal pattern. The **seasonal naive method** is modified from the **naive method** and has the following explicit forecasting function

$$\hat{y}_{T+h|T} = y_{T+h-m(k+1)}$$

where m = the seasonal period and k is the integer part of $(h-1)/m$ (i.e., the number of complete years in the forecast period before time $T+h$). The forecasting formula looks complex, but it essentially says that, for example, with monthly data, the seasonal naive method forecasts for all future February values are equal to the last observed February value.

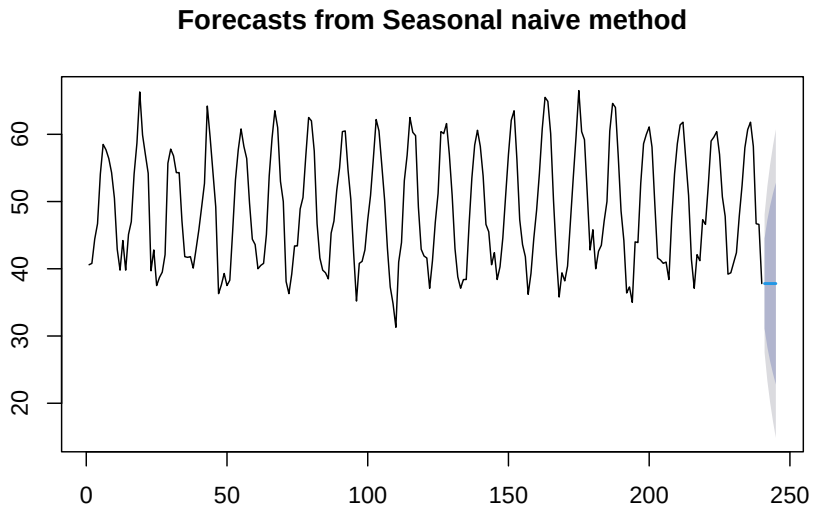
Example 3 Nottingham monthly average temperature - the following data set is a collection of average monthly temperatures in Nottingham from 1920-1939. Since this is monthly data, when we define the time series object with **ts()**, the option **frequency** should be set to 12.

```
nottem =c(40.6,40.8,44.4,46.7,54.1,58.5,57.7,56.4,54.3,50.5,42.9,39.8,44.2,39.8,45.1,47
          54.1,58.7,66.3,59.9,57.0,54.2,39.7,42.8,37.5,38.7,39.5,42.1,55.7,57.8,56.8,54
          54.3,47.1,41.8,41.7,41.8,40.1,42.9,45.8,49.2,52.7,64.2,59.6,54.4,49.2,36.3,37
          39.3,37.5,38.3,45.5,53.2,57.7,60.8,58.2,56.4,49.8,44.4,43.6,40.0,40.5,40.8,45
          53.8,59.4,63.5,61.0,53.0,50.0,38.1,36.3,39.2,43.4,43.4,48.9,50.6,56.8,62.5,62
          57.5,46.7,41.6,39.8,39.4,38.5,45.3,47.1,51.7,55.0,60.4,60.5,54.7,50.3,42.3,39
          40.8,41.1,42.8,47.3,50.9,56.4,62.2,60.5,55.4,50.2,43.0,37.3,34.8,31.3,41.0,43
          53.1,56.9,62.5,60.3,59.8,49.2,42.9,41.9,41.6,37.1,41.2,46.9,51.2,60.4,60.1,61
          57.0,50.9,43.0,38.8,37.1,38.4,38.4,46.5,53.5,58.4,60.6,58.2,53.8,46.6,45.5,40
          42.4,38.4,40.3,44.6,50.9,57.0,62.1,63.5,56.3,47.3,43.6,41.8,36.2,39.3,44.5,48
          54.2,60.8,65.5,64.9,60.1,50.2,42.1,35.8,39.4,38.2,40.4,46.9,53.4,59.6,66.5,60
          59.2,51.2,42.8,45.8,40.0,42.6,43.5,47.1,50.0,60.5,64.6,64.0,56.8,48.6,44.2,36
```

```
          37.3,35.0,44.0,43.9,52.7,58.6,60.0,61.1,58.1,49.6,41.6,41.3,40.8,41.0,38.4,47.4,
          54.1,58.6,61.4,61.8,56.3,50.9,41.4,37.1,42.1,41.2,47.3,46.6,52.4,59.0,59.6,60.4,
          57.0,50.7,47.8,39.2,39.4,40.9,42.4,47.8,52.4,58.0,60.7,61.8,58.2,46.7,46.6,37.8)
nottem.ts = ts(nottem, frequency = 12, start = c(1920, 1))
```

Next, we plot the original series and the forecasted future values as well. Assuming forecasting horizon h = 5.

```
seasonal.naive = snaive(nottem, h = 5)
plot(seasonal.naive)
```



The actual forecasted temperatures in the next 5 months and their predictive intervals are given by

```
kable(seasonal.naive, caption="Forecasted monthly average temperatures of Nottingham between Jan
```

14.2.4 Drift Method

The seasonal naive method allows non-constant predictive values by taking the same values from the time period in the previous season. The drift method is another variation of the naive method that allows the forecasts to increase or decrease over time. To be more specific, the next one-step forecast value is set to be the average change seen in the historical data. Thus the forecast for time $T + h$ is given by

$$\hat{y}_{T+h|T} = y_T + h(y_T - y_1)/(T - 1)$$

Table 14.3: Forecasted monthly average temperatures of Nottingham between Jan - May 1940

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
241	37.8	31.08754	44.51246	27.53418	48.06582
242	37.8	28.30715	47.29285	23.28193	52.31807
243	37.8	26.17368	49.42632	20.01907	55.58093
244	37.8	24.37508	51.22492	17.26835	58.33165
245	37.8	22.79048	52.80952	14.84492	60.75508

where the amount of change over time $y_T - y_1$ is called the drift. This is where the name of the method comes from.

Example 4 New York City Monthly Birth Counts - the dataset contains the number of births per month in New York City, from January 1946 to December 1958. The data set can be downloaded from https://raw.githubusercontent.com/pengdsci/sta321/main/datasets/w11-nycbirths.txt. We will use the seasonal naive method to forecast future values using function **naive()** in library{forecast}.

```
nycbirth = read.table("https://raw.githubusercontent.com/pengdsci/sta321/main/datasets/
births <- ts(nycbirth, frequency = 12, start = c(1946, 1))
drift.pred = rwf(births, h= 5, drift = TRUE)
plot(drift.pred)
```

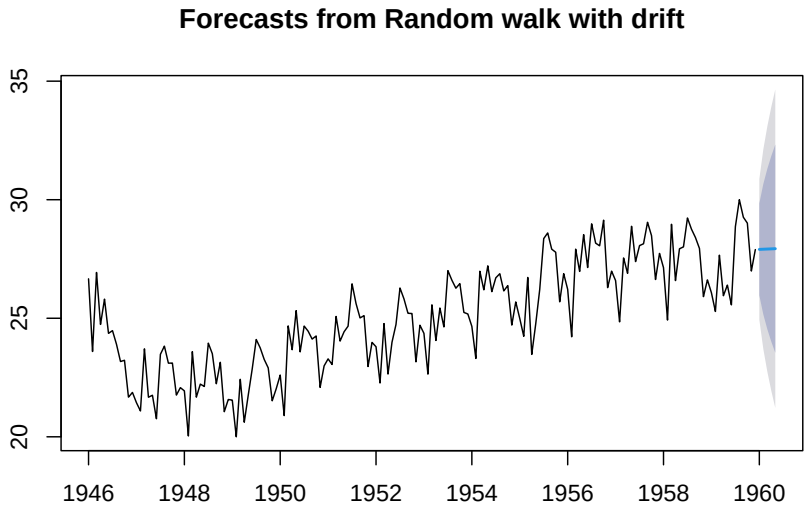


Table 14.4: Forecast birth counts using the drift method

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Jan 1960	27.90439	25.96383	29.84495	24.93656	30.87222
Feb 1960	27.91178	25.15926	30.66430	23.70216	32.12139
Mar 1960	27.91917	24.53807	31.30026	22.74823	33.09010
Apr 1960	27.92656	24.01094	31.84217	21.93814	33.91498
May 1960	27.93395	23.54337	32.32452	21.21914	34.64875

As usual, we can extract the predicted value from the above drift forecast procedure in the following

```
kable(drift.pred, caption = "Forecast birth counts using the drift method")
```

14.3 Accuracy Measures in Time Series Forecasting

14.3.1 Training and Testing Data

To evaluate forecast accuracy using genuine forecasts, we separate the available data into two portions, training and test data, where the training data is used to estimate any parameters of a forecasting method, and the test data is used to evaluate its accuracy. We have used this logic in logistic predictive modeling. The difference is that we cannot use the random split method as we did in the predictive modeling using logistic regression.

We have introduced several baseline forecasting methods in the previous section.

14.3.2 Forecasting Errors

In a given time series data with n observations, we hold up k observations for evaluating forecast errors and use the rest of the $n-k$ values to build time series models. We set the forecast horizon h = k. We can find the difference between the predicted and actual hold-up values to define various error metrics in the following table.

Absolute Error	*Relative Error*
$ME = \frac{\sum \hat{e}_t}{n}$	$PE_t = \frac{(Y_t - \hat{Y}_t)}{Y_t} \times 100$
$MAD = \frac{\sum \lvert\hat{e}_t\rvert}{n}$	$MPE = \frac{\sum PE_t}{n}$
$SSE = \sum \hat{e}_t^2$	$MAPE = \frac{\sum \lvert PE_t\rvert}{n}$
$MSE = \frac{\sum \hat{e}_t^2}{n}$	$RSE = \sqrt{\frac{\sum \hat{e}_t^2}{n-1}}$

The error term $\hat{e}_i = \hat{y}_i - y_i$. We will use the above error measures to define the accuracy metrics in a case study.

14.4 Case Study

Daily Female Births Dataset: This dataset describes the number of daily female births in California in 1959. The units are a count and there are 365 observations

A. Training and testing data

```
female.births = read.csv("https://raw.githubusercontent.com/pengdsci/sta321/main/datase
## training and testing data: hold-up last 6 months of data for calculating forecastin
training = female.births[1:350]
testing = female.births[351:365]
##
female.births.ts = ts(training, frequency = 12, start = c(1959, 1))
## shampoo.ts
```

B. Building 4 Forecasting Models

We next use the four baseline forecasting methods and the first 30 data values to forecast the next 6 month's data values.

```
pred.mv = meanf(female.births.ts, h=15)$mean
pred.naive = naive(female.births.ts, h=15)$mean
pred.snaive = snaive(female.births.ts, h=15)$mean
pred.rwf = rwf(female.births.ts, h=15, drift = TRUE)$mean
###
###
```

Table 14.5: Forecasting Table

pred.mv	pred.naive	pred.snaive	pred.rwf
41.91429	52	34	52.04871
41.91429	52	33	52.09742
41.91429	52	36	52.14613
41.91429	52	49	52.19484
41.91429	52	43	52.24355
41.91429	52	43	52.29226
41.91429	52	34	52.34097
41.91429	52	39	52.38968
41.91429	52	35	52.43840
41.91429	52	52	52.48711
41.91429	52	47	52.53582
41.91429	52	52	52.58453
41.91429	52	34	52.63324
41.91429	52	33	52.68195
41.91429	52	36	52.73066

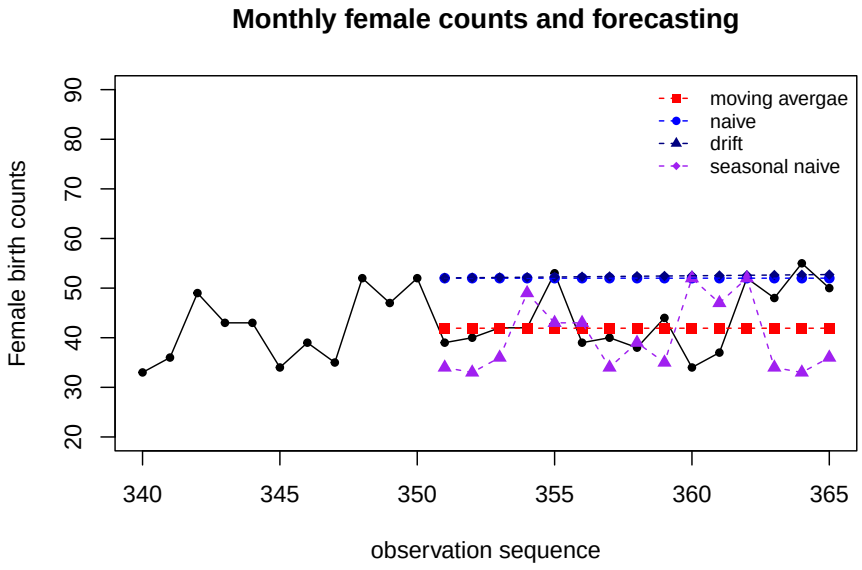
```
pred.table = cbind( pred.mv = pred.mv,
                    pred.naive = pred.naive,
                    pred.snaive = pred.snaive,
                    pred.rwf = pred.rwf)
kable(pred.table, caption = "Forecasting Table")
```

C. Visualization

We now make a time series plot and the predicted values. Note that, the forecast values were based on the model that uses 350 historical data in the time series. The following only show observations #320 -#365 and the 15 forecasted values.

```
plot(340:365, female.births[340:365], type="l", xlim=c(340,365), ylim=c(20, 90),
     xlab = "observation sequence",
     ylab = "Female birth counts",
     main = "Monthly female counts and forecasting")
points(340:365, female.births[340:365],pch=20)
##
points(351:365, pred.mv, pch=15, col = "red")
points(351:365, pred.naive, pch=16, col = "blue")
points(351:365, pred.rwf, pch=18, col = "navy")
points(351:365, pred.snaive, pch=17, col = "purple")
##
lines(351:365, pred.mv, lty=2, col = "red")
lines(351:365, pred.snaive, lty=2, col = "purple")
lines(351:365, pred.naive, lty=2, col = "blue")
```

```
lines(351:365, pred.rwf, lty=2, col = "navy")
##
legend("topright", c("moving avergae", "naive", "drift", "seasonal naive"),
       col=c("red", "blue", "navy", "purple"), pch=15:18, lty=rep(2,4),
       bty="n", cex = 0.8)
```



We can see that the **moving average** method worked pretty well. The performance of naive and drift methods in this seasonal time series are close to each other. But **naive**, **seasonal naive**, and **drift** methods worked poorly compared to the **moving average** method.

Intuitively, **moving average** should produce non-constant forecast values. What

D. Accuracy Metrics

We will use the mean absolute prediction error (MAPE) to compare the performance of the four forecasting methods.

```
true.value = female.births[351:365]
PE.mv =  100*(true.value - pred.mv)/true.value
PE.naive =  100*(true.value - pred.naive)/true.value
PE.snaive =  100*(true.value - pred.snaive)/true.value
PE.rwf =  100*(true.value - pred.rwf)/true.value
##
MAPE.mv = mean(abs(PE.mv))
MAPE.naive = mean(abs(PE.naive))
```

Table 14.6: Overall performance of the four forecasting methods

	MAPE	MAD	MSE
Moving Average	11.29861	77.08571	41.93687
Naive	22.83107	135.00000	111.00000
Seasonal Naive	20.37455	133.00000	112.86667
Drift	23.49040	138.99427	116.60627

```
MAPE.snaive = mean(abs(PE.snaive))
MAPE.rwf = mean(abs(PE.rwf))
##
MAPE = c(MAPE.mv, MAPE.naive, MAPE.snaive, MAPE.rwf)
## residual-based Error
e.mv = true.value - pred.mv
e.naive = true.value - pred.naive
e.snaive = true.value - pred.snaive
e.rwf = true.value - pred.rwf
## MAD
MAD.mv = sum(abs(e.mv))
MAD.naive = sum(abs(e.naive))
MAD.snaive = sum(abs(e.snaive))
MAD.rwf = sum(abs(e.rwf))
MAD = c(MAD.mv, MAD.naive, MAD.snaive, MAD.rwf)
## MSE
MSE.mv = mean((e.mv)^2)
MSE.naive = mean((e.naive)^2)
MSE.snaive = mean((e.snaive)^2)
MSE.rwf = mean((e.rwf)^2)
MSE = c(MSE.mv, MSE.naive, MSE.snaive, MSE.rwf)
##
accuracy.table = cbind(MAPE = MAPE, MAD = MAD, MSE = MSE)
row.names(accuracy.table) = c("Moving Average", "Naive", "Seasonal Naive", "Drift")
kable(accuracy.table, caption ="Overall performance of the four forecasting methods")
```

In summary, the **moving average method** has the best performance. Note that the methods introduced in this module are baseline forecasting. They are all descriptive since we did not use any statistical assumptions. In the next module, we will introduce a few formal non-parametric forecasting methods - exponential forecasting methods. We will use the same accuracy measures to compare different forecasting methods.

14.5 Analysis Assignment

Find a time series with at least 76 periods of values from the internet. Any type of time series (trend or non-trend, seasonal or non-seasonal, etc) will be fine for this assignment. The objective is to use the accuracy measures to compare the performance of the four baseline forecasting methods. To be more specific,

1. Make a time series plot based on the data.
2. Split the time series into training and testing sets. Please hold up the most recent 10 periods to calculate the forecasting error.
3. Define a time series object using the R function **ts()** using the training data.
4. Build the four baseline forecasting time series models.
5. Create a graph to visualize the forecasted values.
6. Calculate the accuracy measures for the four forecasting models and summarize the results.

Chapter 15

Time Series Decomposition

Time series decomposition is a process of splitting a time series into basic components: trend, seasonality random error. The method originated a century ago and new developments in the past few decades.

Seasonal: Patterns that repeat for a fixed period. For example, a website might receive more visits during weekends; this would produce data with seasonality of 7 days.

Trend: The underlying trend of the metrics. A website increasing in popularity should show a general trend that goes up.

Random Error: Also call "noise", "residual" or "remainder". These are the residuals of the original time series after the seasonal and trend series are removed.

The objective of time series decomposition is to model the trend and seasonality and estimate the overall time series as a combination of them. A seasonally adjusted value removes the seasonal effect from a value so that trends can be seen more clearly.

The following two working data sets were widely used in different textbooks. We will use them to illustrate some of the concepts.

Australian Beer Production Data

The following data gives quarterly beer production figures in Australia from 1956 through the 2nd quarter of 2010. The beer production figure is in megalitres.

```
ausbeer0=c(284, 213, 227, 308, 262, 228, 236, 320, 272, 233, 237, 313, 261, 227, 250, 314,
          286, 227, 260, 311, 295, 233, 257, 339, 279, 250, 270, 346, 294, 255, 278, 363,
          313, 273, 300, 370, 331, 288, 306, 386, 335, 288, 308, 402, 353, 316, 325, 405,
          393, 319, 327, 442, 383, 332, 361, 446, 387, 357, 374, 466, 410, 370, 379, 487,
          419, 378, 393, 506, 458, 387, 427, 565, 465, 445, 450, 556, 500, 452, 435, 554,
```

```
          510, 433, 453, 548, 486, 453, 457, 566, 515, 464, 431, 588, 503, 443, 448, 5
          513, 427, 473, 526, 548, 440, 469, 575, 493, 433, 480, 576, 475, 405, 435, 5
          453, 430, 417, 552, 464, 417, 423, 554, 459, 428, 429, 534, 481, 416, 440, 5
          474, 440, 447, 598, 467, 439, 446, 567, 485, 441, 429, 599, 464, 424, 436, 5
          443, 410, 420, 532, 433, 421, 410, 512, 449, 381, 423, 531, 426, 408, 416, 5
          409, 398, 398, 507, 432, 398, 406, 526, 428, 397, 403, 517, 435, 383, 424, 5
          421, 402, 414, 500, 451, 380, 416, 492, 428, 408, 406, 506, 435, 380, 421, 4
          435, 390, 412, 454, 416, 403, 408, 482, 438, 386, 405, 491, 427, 383, 394, 4
          420, 390, 410)
```

- **Airline Passengers Data**

This data set records monthly totals of international airline passengers (1949-1960).

```
AirPassengers0=c(112, 118, 132, 129, 121, 135, 148, 148, 136, 119, 104, 118, 115, 126,
                 135, 125, 149, 170, 170, 158, 133, 114, 140, 145, 150, 178, 163, 172,
                 199, 199, 184, 162, 146, 166, 171, 180, 193, 181, 183, 218, 230, 242,
                 191, 172, 194, 196, 196, 236, 235, 229, 243, 264, 272, 237, 211, 180,
                 204, 188, 235, 227, 234, 264, 302, 293, 259, 229, 203, 229, 242, 233,
                 269, 270, 315, 364, 347, 312, 274, 237, 278, 284, 277, 317, 313, 318,
                 413, 405, 355, 306, 271, 306, 315, 301, 356, 348, 355, 422, 465, 467,
                 347, 305, 336, 340, 318, 362, 348, 363, 435, 491, 505, 404, 359, 310,
                 360, 342, 406, 396, 420, 472, 548, 559, 463, 407, 362, 405, 417, 391,
                 461, 472, 535, 622, 606, 508, 461, 390, 432)
```

15.1 Classical Decompositions

Classical decomposition was developed about a century ago and is still widely used nowadays. Depending on the types of time series models, there are two basic methods of decomposition: additive and multiplicative.

The following two time series represent the above two basic types of times series models.

```
ausbeer.ts = ts(ausbeer0[9:72], frequency = 4, start = c(1958, 1))
AirPassengers.ts = ts(AirPassengers0, frequency = 4, start = c(1949, 1))
par(mfrow=c(1,2), mar=c(2,2,2,2))
plot(ausbeer.ts, xlab="", ylab="", main = "Additive Model")
plot(AirPassengers.ts, xlab="", ylab="", main = "Multiplicative Model")
```

Denote $T = trend$, $S = seasonality$, and $E = error$. With these notations, we can characterize the structure of **additive and multiplicative** time series.

In a **multiplicative time series**, the components multiply together to make the time series. As the time series increases in magnitude, the **seasonal variation** increases as well. The structure of a multiple time series has the following form.

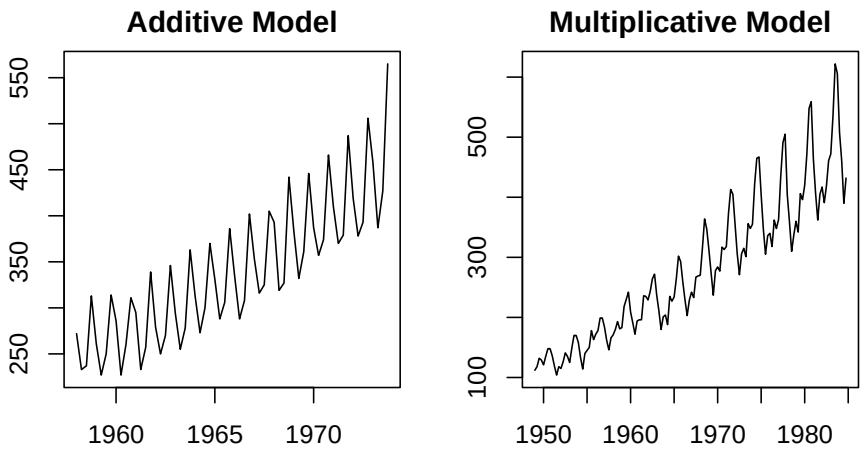


Figure 15.1: time series plots of additive and multiplicative series

$$y_t = T_t \times S_t \times E_t.$$

In an **additive time series**, the components add together to make the time series. If you have an increasing trend, you still see roughly the same size peaks and troughs throughout the time series. This is often seen in indexed time series where the absolute value is growing but changes stay relative. The structure of an additive time series has the following form

$$y_t = T_t + S_t + E_t.$$

For an **additive time series**, the detrended additive series has for $D_t = y_t - T_t$. For the multiplicative time series, the detrended time series is calculated by $D_t = y_t/T_t$

15.2 Understanding the Classical Decomposition of Time Series

To understand the structure of additive and multiplication ties series, we decompose these time series by calculating the trend, seasonality, and errors *manually* by writing a basic R script to gain a technical understanding of decomposing a time series. At the very end of this section, we introduce the R function **decompose()** to extract the three components of additive and multiplicative time series.

15.2.1 Detect Trend

To detect the underlying trend, we use a smoothing technique called **moving average** and it's variant **centered moving average**. For a seasonal time series, the width of the moving window must be the same as the seasonality. Therefore, to decompose a time series we need to know the seasonality period: weekly, monthly, etc.

Example 1: Australian beer production data has an annual seasonality. Since the data set is quarterly data, the moving average window should be 4.

```
trend.beer = ma(ausbeer.ts, order = 4, centre = T)  # centre = T => centered moving av
par(mar=c(2,2,2,2))
plot(as.ts(ausbeer.ts), xlab="", ylab="", col="darkred", lwd =2)
title(main = "Extract trend from Australia Beer Production")
lines(trend.beer, lwd =2, col = "blue")
legend("topleft", c("original series", "trend curve"), lwd=rep(2,2),
       col=c("darkred", "blue"), bty="n")
```

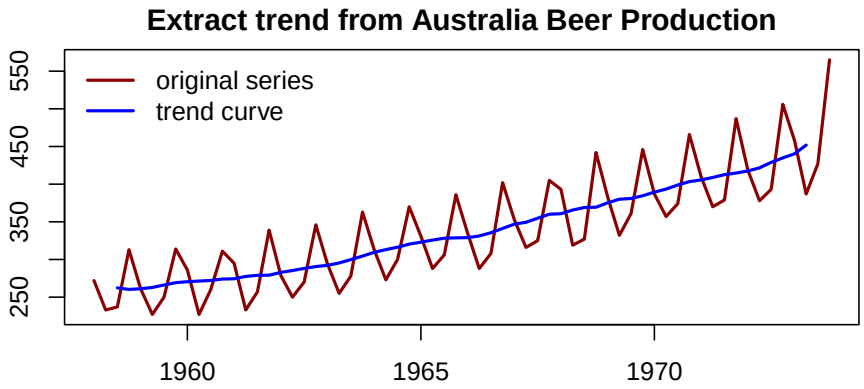


Figure 15.2: Serires plot with trend curve

Example 2: The airline passenger data were recorded monthly. It has an annual seasonal pattern. We choose a moving average window of 12 to extract the trend from this multiplicative time series.

```
trend.air = ma(AirPassengers.ts, order = 12, centre = T)  # centre = T => centered mov
par(mar=c(2,2,2,2))
plot(as.ts(AirPassengers.ts), xlab="", ylab="", col="darkred", lwd =2)
title(main = "Extract trend from Airline Passengers Monthly Data")
lines(trend.air, lwd =2, col = "blue")
legend("topleft", c("original series", "trend curve"), lwd=rep(2,2),
       col=c("darkred", "blue"), bty="n")
```

The **moving averages** of both time series are recorded in the above two code

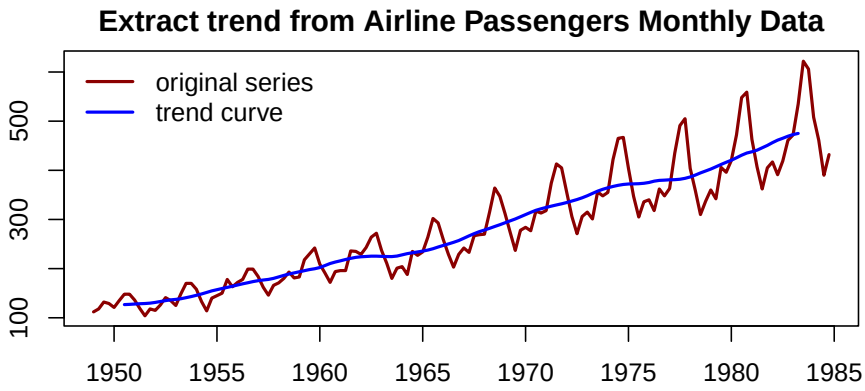


Figure 15.3: series plot with curve trend

chunks and will be used to restore the original series.

The process of removing the trend from a time series is called **detrending** time series.

The way of detrending a time series is dependent on the types of the time series. The following code shows how to calculate the detrended time series.

Example 3: We calculate the detrended series using the Australian Beer data and the Airline Passengers data as an example.

```
detrend.beer = ausbeer.ts - trend.beer
detrend.air = AirPassengers.ts/trend.air
par(mar=c(2,2,2,2))
par(mfrow=c(1,2))
# plot(ausbeer.ts, xlab="", main = "Australia Beer", col="darkred")
# plot(AirPassengers.ts, xlab="", main = "Air Passengers", col="blue")
plot(detrend.beer, xlab="", main = "Australia Beer", col="darkred")
plot(detrend.air, xlab="", main = "Air Passengers", col="blue")
```

The technique we used in removing the trend from a time series model is a non-parametric smoothing procedure. There are different techniques in statistics to estimate a curve for a given set. The **moving average** is one of the simplest ones and is widely used in time series.

15.2.2 Extracting the Seasonality

Similar to the trend in a time series, the seasonality of a time series is also a non-random structural pattern. We can extract the seasonality from the detrended time series.

The idea is to redefine a **seasonal series** based on the detrended series by

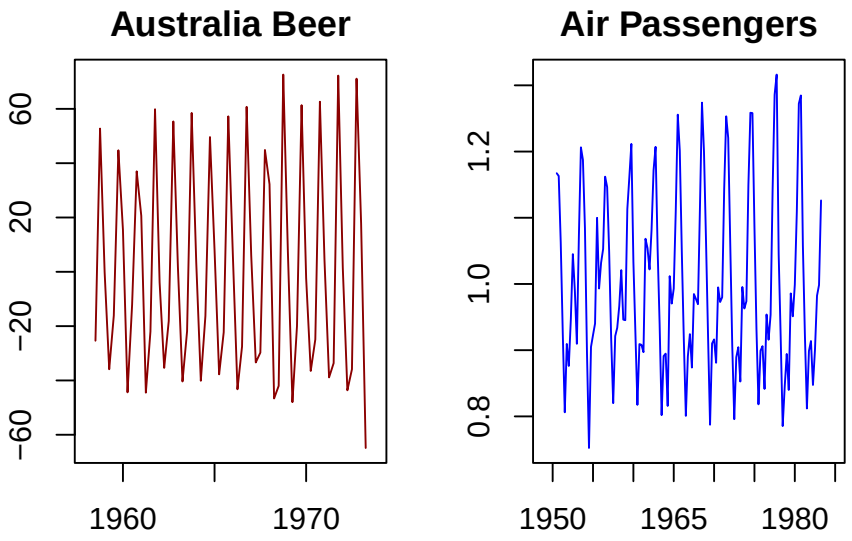


Figure 15.4: Detrended series

replacing all observations taken from the same seasonal period with the average of these observations. This process is called **averaging seasonality**. This idea is implemented in R. Here is how to do it in R.

Example 4: Use the **Australian Beer Production Data** and the **Airline Passenger Data** after their trends were removed. The following code illustrates how to calculate and graph the seasonality of both series.

```
par(mfrow=c(2,1), mar=c(3,2,3,2))
## Australia Beer
mtrx.beer = t(matrix(data = detrend.beer, nrow = 4))
seasonal.beer = colMeans(mtrx.beer, na.rm = T)
seasonal.beer.ts = as.ts(rep(seasonal.beer,16))
plot(seasonal.beer.ts, xlab = "", col="darkred", main="Seasonal series of Australia bee
##
mtrx.air = t(matrix(data = detrend.air, nrow = 12))
seasonal.air = colMeans(mtrx.air, na.rm = T)
seasonal.air.ts = as.ts(rep(seasonal.air,16))
plot(seasonal.air.ts, xlab = "", col = "blue", main="Seasonal series of air passengers"
```

15.2.3 Extracting Remainder Errors

The **error term** is the random component in the time series. We learned the of extracting the trend and seasonality from the original time series. How to extract the "random" noise from a given time series?

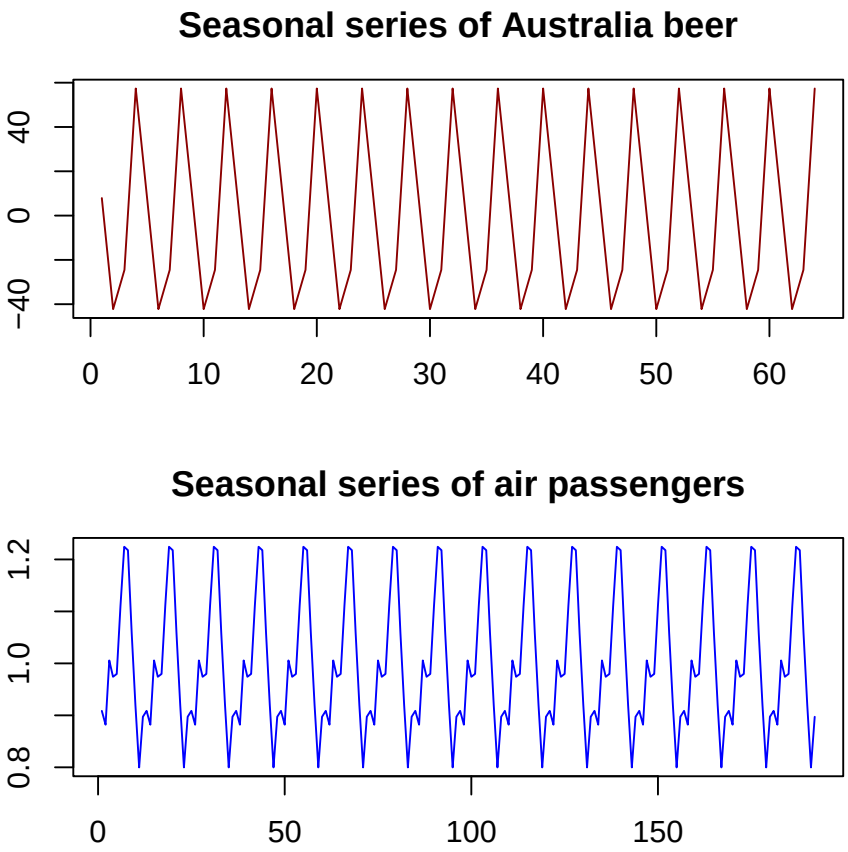


Figure 15.5: Seasonal series

In the additive model, the random **error term** is given by $E_t = y_t - T_t - S_t$. The **random error** for a multiplicative model is given by $E_t = y_t/(T_t \times S_t)$.

Example 5: Use the above formulas to separate the random error components in additive and multiplicative models using Australian beer production and Airline passenger series data.

```
random.beer = ausbeer.ts - trend.beer - seasonal.beer
random.air = AirPassengers.ts / (trend.air * seasonal.air)
##
par(mfrow=c(2,1), mar=c(3,2,3,2))
plot(random.beer , xlab = "", col="darkred", main="Random errors of Australia beer")
plot(random.air, xlab = "", col = "blue", main="Random errors of air passengers")
```

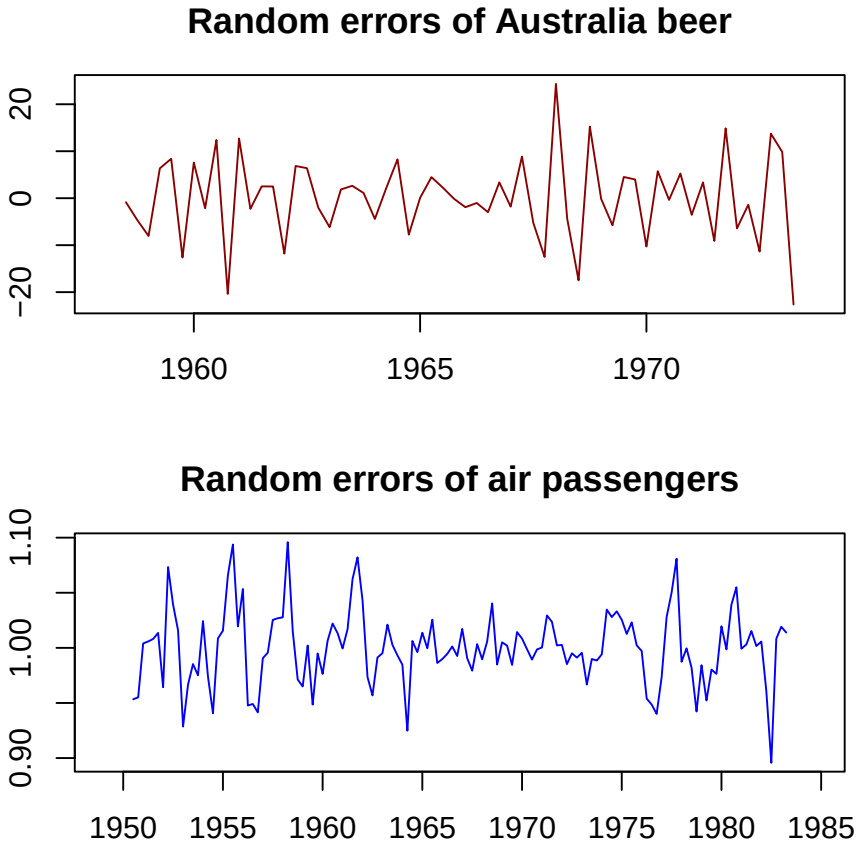


Figure 15.6: Random error components

15.2.4 Reconstruct the Original Series / Compose New Series

The original series can be **reconstructed** by using the decomposed components. Since the **moving average technique** was used in the detrending series, the resulting **reconstructed** series with T_t, S_t, and E_t will generate a few missing values in the beginning and the end depending on the width of the **moving average window**.

Example 6: **Reconstruct** the **original** series of Australian beer data and the airline passenger data.

```
recomposed.beer = trend.beer+seasonal.beer+random.beer
recomposed.air = trend.air*seasonal.air*random.air
par(mfrow=c(1,2), mar=c(3,2,3,2))
plot(ausbeer.ts, col="darkred", lty=1)
lines(recomposed.beer, col="blue", lty=2, lwd=2)
legend("topleft", c("original series", "reconstructed series"),
       col=c("darkred", "blue"), lty=1:2, lwd=1:2, cex=0.8, bty="n")
title(main="Australian Beer")
##
plot(AirPassengers.ts, col="darkred", lty=1)
lines(recomposed.air, col="blue", lty=2, lwd=2)
legend("topleft", c("original series", "reconstructed series"),
       col=c("darkred","blue"), lty=1:2, lwd=1:2, cex=0.8, bty="n")
title(main="Airline Passengers")
```

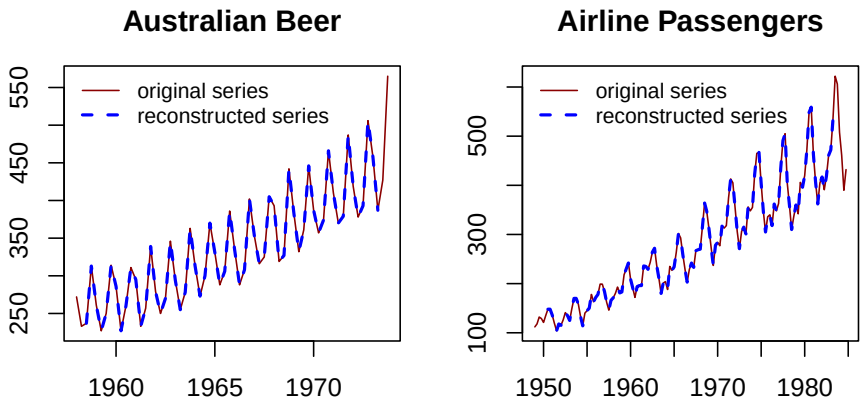


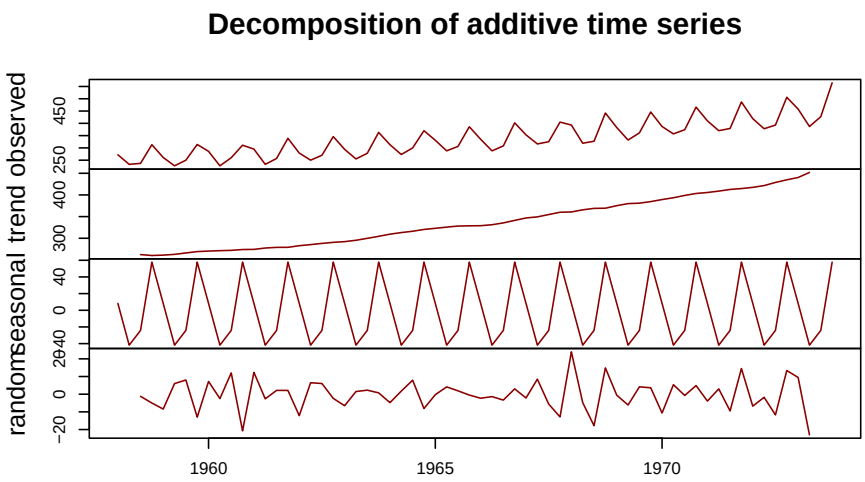
Figure 15.7: Adding trend series to the original series

15.2.5 Decomposing Time Series with decompose()

The R library **forecast** was created by a team led by a leading expert in the discipline. We'll use the R function **decompose( )** in library{forecast} as a decomposition function to decompose a series into seasonal, trend, and random components. The Australian beer production (additive) and airline passenger numbers (multiplicative) will still be used to illustrate the steps.

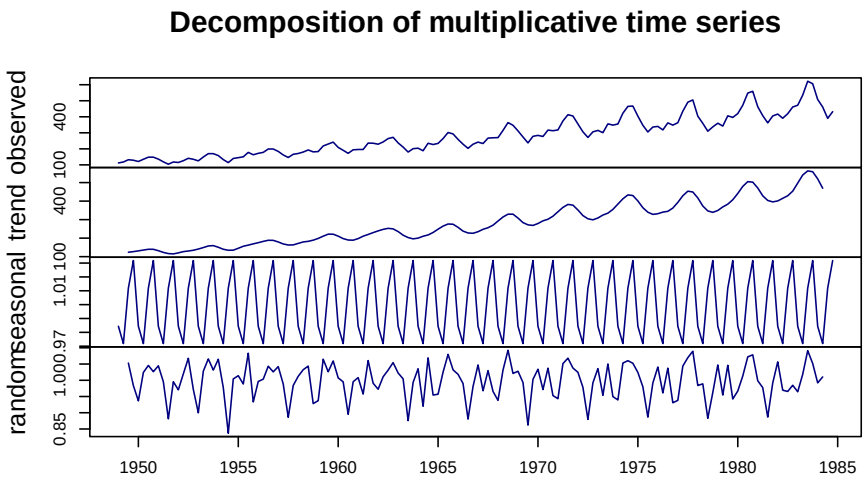
Example 6: Plot the components of the Australian beer production data.

```
decomp.beer = decompose(ausbeer.ts, "additive")
## plot the decomposed components
par(mar=c(2,2,2,2), oma=c(0,2,2,0))
plot(decomp.beer, col="darkred", xlab="")
```



Example 7: Plot the components of the Airline Passenger Data.

```
decomp.air = decompose(AirPassengers.ts, "multiplicative")
## plot the decomposed components
par(mar=c(2,2,2,2), oma=c(0,2,2,0))
plot(decomp.air, col= "navy", xlab="")
```



Th R function **decompose()** can be used to extract the individual components from a given additive and multiplicative model using the following code.

Example 8: Decomposing Australian beer production data using **decompose()**.

```
decomp.beer = decompose(ausbeer.ts, "additive")
# the four components can be extracted by
seasonal.beer = decomp.beer$seasonal
trend.beer = decomp.beer$trend
error.beer = decomp.beer$random
```

Example 9: Decomposing airline passengers data using **decompose()**.

```
decomp.air = decompose(AirPassengers.ts, "multiplicative")
# the four components can be extracted by
seasonal.air = decomp.air$seasonal
trend.air = decomp.air$trend
error.air = decomp.air$random
```

Concluding Remark: There are other decomposition methods. among them, **X11** is commonly used in econometrics. The recently developed method **STL()** that used LOESS algorithm to estimate the trend can also be used to extract components from more general time series. We will outline this decomposition method to forecast future values.

15.3 Forecasting with Decomposing

Several benchmark forecasting methods were introduced in the previous module. Next, we use these benchmark methods to forecast the deseasonalized series. Since the seasonality of a time series is a **scalar**, we can forecast the deseasonalized series through decomposition and then adjust the forecasted values.

15.3.1 Forecasting Additive Models with Decomposing

Since the multiplicative models can be converted to an additive model by

$$\log(y_t) = \log(S_t) + \log(T_t) + \log(E_t).$$

So we only restrict our discussion in this module to additive models. Assuming an additive decomposition, the decomposed time series can be written as

$$y_t = \hat{S}_t + \hat{A}_t$$

where $\hat{A}_t = \hat{T} + \hat{E}_t$ is the seasonally adjusted component. We can forecast the future values based on $\hat{A}_t$.

Example 10: Forecasting based on the seasonally adjusted series with Australian beer production data. We introduced four benchmark forecasting methods in the previous module. For a time series with a trend, naive, seasonal naive, and drift method is more accurate than the moving average. The issue is that none of the benchmark methods forecast the trend. As an illustrative example, we use the naive method to forecast the deseasonalized series and then add the seasonal adjustment to the forecast values.

```
decomp.beer = decompose(ausbeer.ts, "additive")
# the four components can be extracted by
seasonal.beer = decomp.beer$seasonal
trend.beer = decomp.beer$trend
error.beer = decomp.beer$random
seasonal.adj = trend.beer + error.beer
deseasonalized.pred = as.data.frame(rwf(na.omit(seasonal.adj), h = 6))
seasonality =  matrix(rep(seasonal.beer[3:8],5), nco=5, byrow=F)
seasonal.adj.pred <- deseasonalized.pred  + seasonality
kable(seasonal.adj.pred, caption = "Forecasting with decomposing - drift method")
```

Since the deseasonalized series has two missing values in the beginning and two in the end. we remove the missing values before using the drift methods to forecast the next 6 periods (qtr3, 1973 - qtr 4, 1874). The forecast values are given in the above table.

Table 15.1: Forecasting with decomposing - drift method

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
1973 Q3	404.7167	386.3725	423.0608	376.6617	432.7717
1973 Q4	486.6167	460.6741	512.5593	446.9409	526.2924
1974 Q1	437.1500	405.3769	468.9231	388.5573	485.7427
1974 Q2	387.0000	350.3116	423.6884	330.8900	443.1100
1974 Q3	404.7167	363.6978	445.7355	341.9838	467.4496
1974 Q4	486.6167	441.6828	531.5505	417.8962	555.3371

```
par(mar=c(2,2,2,2))
plot(1:62, as.vector(ausbeer.ts)[-c(63,64)], type="l", xlim=c(1,70), ylim=c(200, 570),
     xlab="", ylab="Beer Production", main="Forecast with classical decomposing")
lines(62:64,  as.vector(ausbeer.ts)[c(62,63,64)], col="red")
points(62:64,  as.vector(ausbeer.ts)[c(62,63,64)], col="red", pch=21)
lines(63:68,seasonal.adj.pred[,1], col="blue")
points(63:68,seasonal.adj.pred[,1], col="blue", pch = 16)
```

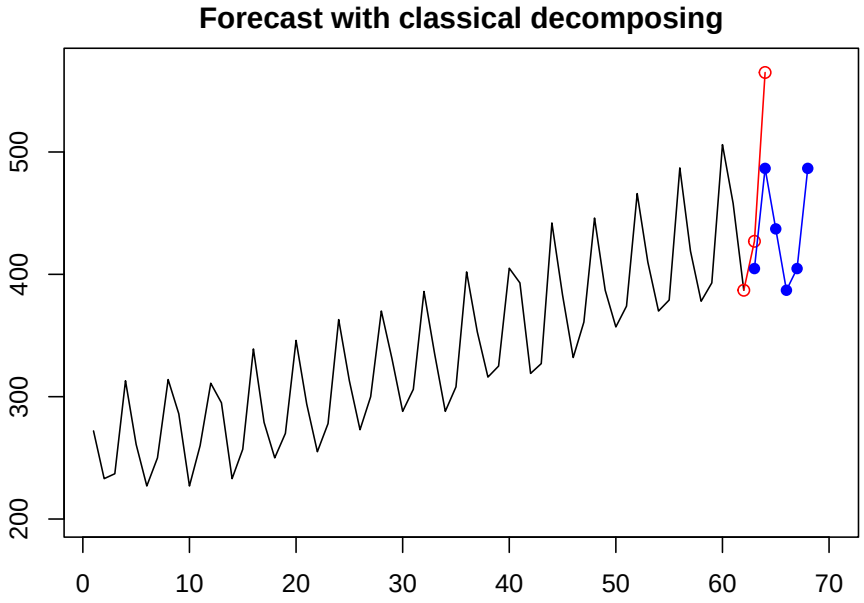


Figure 15.8: forecasting with classical decomposing

The plot of the forecast values and the original values. The red plot represented quarters 3-4 of 1973 and forecast values in quarters 3 - quarters 4, 1974, are plotted in **blue**. We can see that

15.3.2 Concepts of Seasonal and Trend Decomposition Using Loess (STL)

Seasonal and Trend decomposition using LOESS (STL) combines the classical time series decomposition and the **modern** locally estimated scatterplot smoothing (LOESS). The LOESS has developed about 40 years ago and is a modern computational algorithm. The seasonal trend in a time series is a **fixed** pattern. The real benefit of STL is to use the LOESS to estimate the nonlinear trend more accurately. We will not discuss the technical development of the STL. Instead, we will use its R implementation to decompose time series and use it to forecast future values.

```
stl.beer = stl(ausbeer.ts, "periodic")
seasonal.stl.beer   <- stl.beer$time.series[,1]
trend.stl.beer     <- stl.beer$time.series[,2]
random.stl.beer  <- stl.beer$time.series[,3]
###
par(mfrow=c(4,1), mar=c(2,2,2,2))
plot(ausbeer.ts)
plot(as.ts(seasonal.stl.beer))
plot(trend.stl.beer)
plot(random.stl.beer)
```

We can also plot the above-decomposed components in a single step as follows with the STL model.

```
plot(stl.beer)
```

15.3.3 Forecast with STL decomposing

For the additive model, we can use the R function **stl()** to decompose the series into three components. It uses a more robust non-parametric smoothing method (LOESS) to estimate the nonlinear trend.

```
fit <- stl(ausbeer.ts,s.window="periodic")
par(mar=c(2,2,2,2))
plot(forecast(fit,h=6, method="rwdrift"))
```

15.3.4 Length of Time Series

The length of the time series impacts the performance of the forecasting. In general, a very long time series (for example, more than 200 observations) usually does not work well for most of the existing models partly because the existing models were not built for **very long** series. Intuitively, future values are dependent on recent historical values. If including too old observations that have no predictive power in the model will bring bias and noise to the underlying model and, hence, negative impacts on the performance of the model.

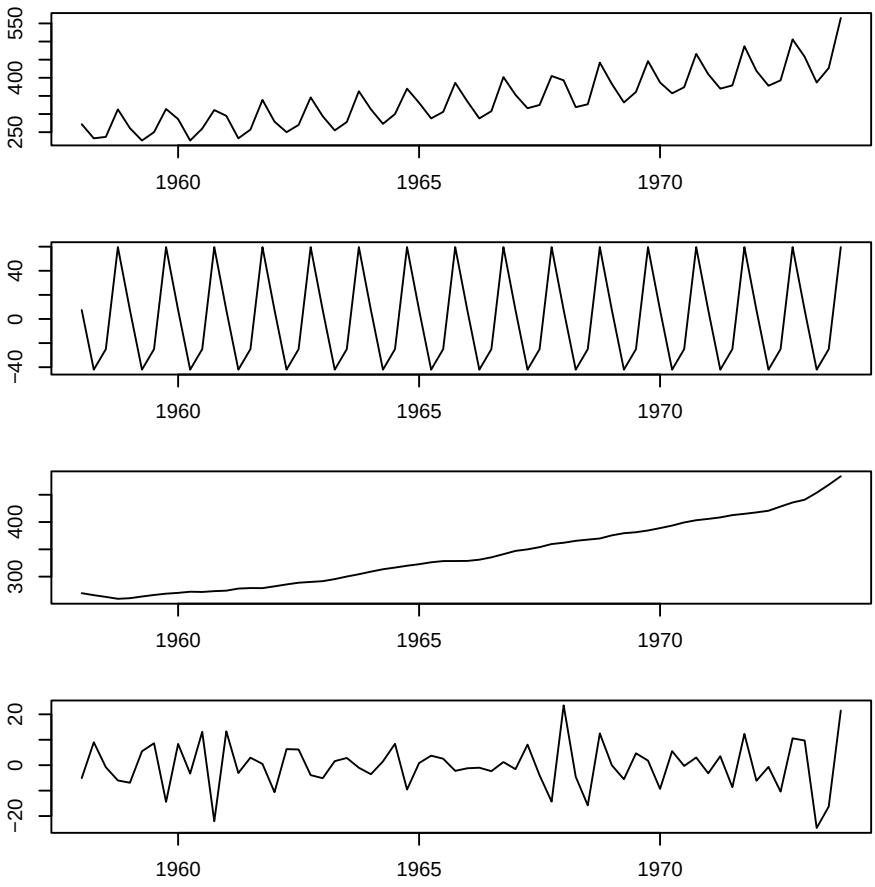


Figure 15.9: Decomposing with STL approach

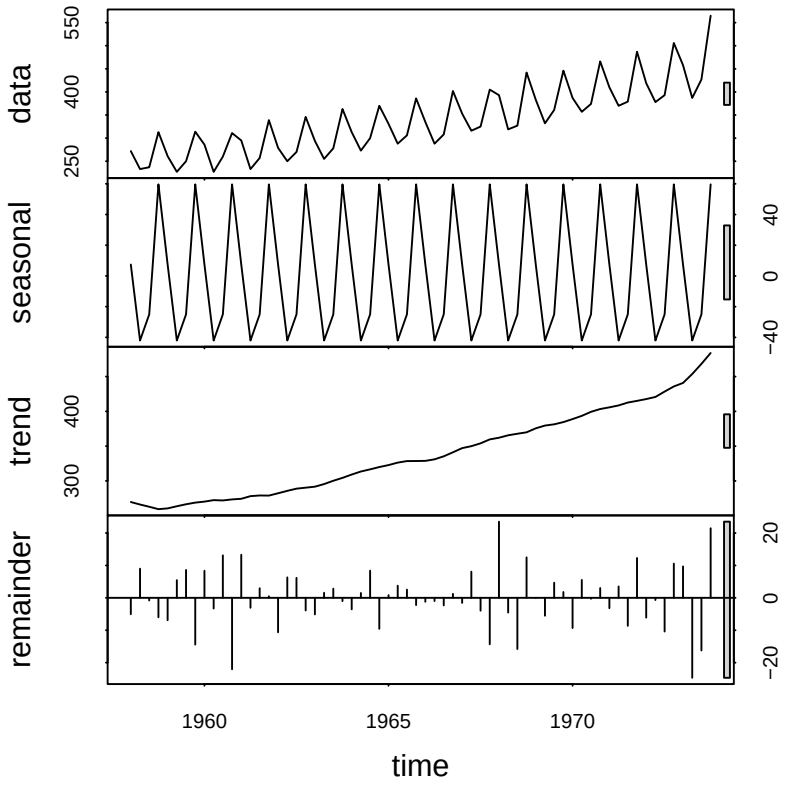


Figure 15.10: plot component panel with STL object

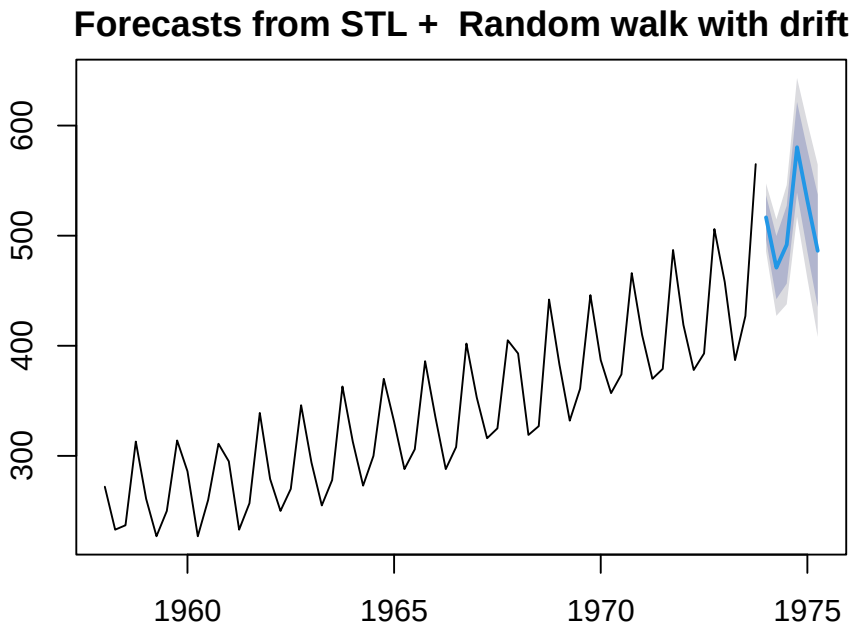


Figure 15.11: Forecas with STL decomposing

There are a lot of discussions in literature and practice about the minimum size required for building a good time series model. It seems that 60 is the suggested minimum size. The actual minimum size depends on the situation and the level of accuracy.

In this class, we recommend the sample size be between 60 and 200. Therefore, in the assignment, if the original time series data has more than 200 observations, we can use only 150-200 most recent data values for analysis.

15.4 Case Study

15.4.1 Data description and

The time series used in this case study is chosen from https://datahub.io/search: 10-year nominal yields on US government bonds from the Federal Reserve. The 10-year government bond yield is considered a standard indicator of long-term interest rates. The data contains monthly rates. There are 808 months of data between April 1943 and July 2020. We only use look at monthly data between January 2008 and July 2020.

```
us.bond=read.csv("https://datahub.io/core/bond-yields-us-10y/r/monthly.csv")
n.row = dim(us.bond)[1]
```

```
data.us.bond = us.bond[(n.row-150):n.row, ]
```

15.4.2 Define time series object

Since this is monthly data, frequency =12 will be used the define the time series object.

```
usbond.ts = ts(data.us.bond[,2], frequency = 12, start = c(2008, 1))
par(mar=c(2,2,2,2))
plot(usbond.ts, main="US Bond Rates Between Jan, 2008 and July, 2020", ylab="Monthly Ra
```

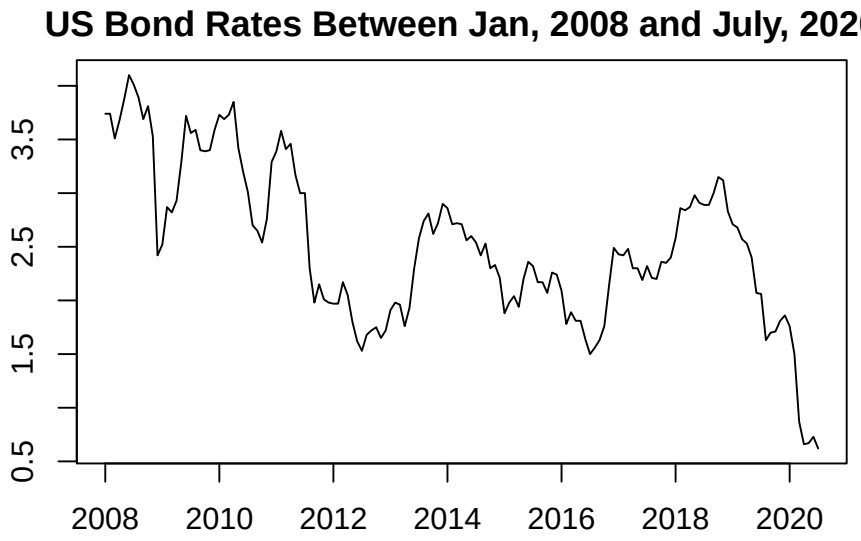


Figure 15.12: US bond monthly rates

15.4.3 Forecasting with Decomposing

Notice that the classical decomposition method does not work as well as the STL method due to the robustness of the LOESS component. The following visual representations show the different behaviors of the two methods of decomposition.

```
cls.decomp = decompose(usbond.ts)
par(mar=c(2,2,2,2))
plot(cls.decomp, xlab="")
```

```
stl.decomp=stl(usbond.ts, s.window = 12)
par(mar=c(2,2,2,2))
plot(stl.decomp)
```

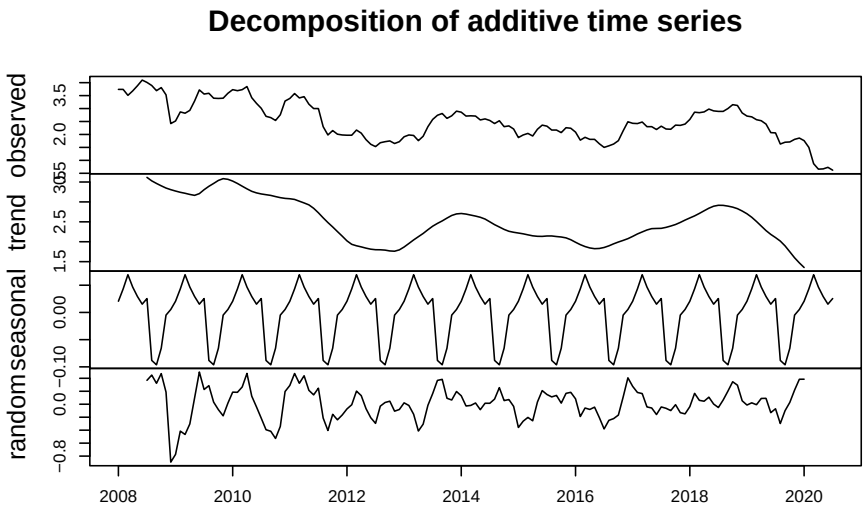


Figure 15.13: Classical decomposition of additive time series

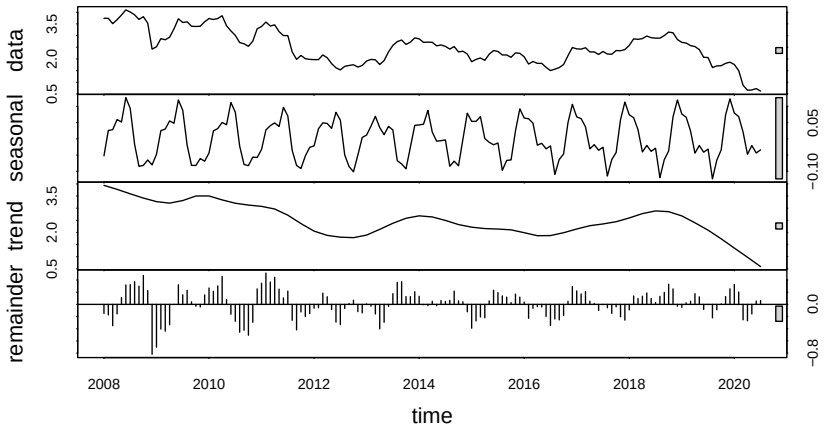


Figure 15.14: STL decomposition of additive time series

Training and Testing Data

We hold up the **last 7 periods** of data for testing. The rest of the historical data will be used to train the forecast model.

To evaluate the effect of different sizes in training the time series, We define different training data sets with different sizes. Three training set sizes used in this example are 144, 109, 73, and 48. The same test set with size 7 will be used to calculate the prediction error.

```
ini.data = data.us.bond[,2]
n0 = length(ini.data)
##
train.data01 = data.us.bond[1:(n0-7), 2]
train.data02 = data.us.bond[37:(n0-7), 2]
train.data03 = data.us.bond[73:(n0-7), 2]
train.data04 = data.us.bond[97:(n0-7), 2]
## last 7 observations
test.data = data.us.bond[(n0-6):n0,2]
##
train01.ts = ts(train.data01, frequency = 12, start = c(2008, 1))
train02.ts = ts(train.data02, frequency = 12, start = c(2011, 1))
train03.ts = ts(train.data03, frequency = 12, start = c(2014, 1))
train04.ts = ts(train.data04, frequency = 12, start = c(2016, 1))
##
stl01 = stl(train01.ts, s.window = 12)
stl02 = stl(train02.ts, s.window = 12)
stl03 = stl(train03.ts, s.window = 12)
stl04 = stl(train04.ts, s.window = 12)
## Forecast with decomposing
fcst01 = forecast(stl01,h=7, method="naive")
fcst02 = forecast(stl02,h=7, method="naive")
fcst03 = forecast(stl03,h=7, method="naive")
fcst04 = forecast(stl04,h=7, method="naive")
```

We next perform error analysis.

```
## To compare different errors, we will not use the percentage for MAPE
PE01=(test.data-fcst01$mean)/fcst01$mean
PE02=(test.data-fcst02$mean)/fcst02$mean
PE03=(test.data-fcst03$mean)/fcst03$mean
PE04=(test.data-fcst04$mean)/fcst04$mean
###
MAPE1 = mean(abs(PE01))
MAPE2 = mean(abs(PE02))
MAPE3 = mean(abs(PE03))
MAPE4 = mean(abs(PE04))
###
```

Table 15.2: Error comparison between forecast results with different sample sizes

	MSE	MAPE
n.144	0.7967685	0.4518108
n.109	0.7718715	0.4463052
n. 73	0.7665760	0.4449924
n. 48	0.8055530	0.4649921

```
E1=test.data-fcst01$mean
E2=test.data-fcst02$mean
E3=test.data-fcst03$mean
E4=test.data-fcst04$mean
##
MSE1=mean(E1^2)
MSE2=mean(E2^2)
MSE3=mean(E3^2)
MSE4=mean(E4^2)
###
MSE=c(MSE1, MSE2, MSE3, MSE4)
MAPE=c(MAPE1, MAPE2, MAPE3, MAPE4)
accuracy=cbind(MSE=MSE, MAPE=MAPE)
row.names(accuracy)=c("n.144", "n.109", "n. 73", "n. 48")
kable(accuracy, caption="Error comparison between forecast results with different sample sizes")
```

```
plot(1:4, MSE, type="b", col="darkred", ylab="Error", xlab="",
     ylim=c(0.4,.85),xlim = c(0.5,4.5), main="Error Curves", axes=FALSE)
labs=c("n=144", "n=109", "n=73", "n=48")
axis(1, at=1:4, label=labs, pos=0.4)
axis(2)
lines(1:4, MAPE, type="b", col="blue")
text(1:4, MAPE+0.03, as.character(round(MAPE,4)), col="blue", cex=0.7)
text(1:4, MSE-0.03, as.character(round(MSE,4)), col="darkred", cex=0.7)
legend(1.5, 0.63, c("MSE", "MAPE"), col=c("darkred","blue"), lty=1, bty="n", cex=0.7)
```

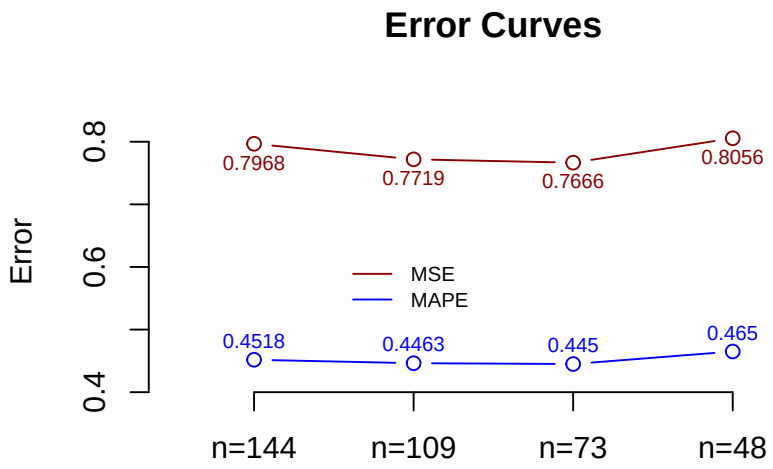


Figure 15.15: Comparing forecast errors

We trained the same algorithm with different sample sizes and compared the resulting accuracy measures. Among four training sizes 144, 109, 73, and 48. training data size 73 yields the best performance.

As anticipated, forecasting with STL smoothing does not yield decent results. However, our case study still accomplishes the main learning goals. To be more specific, we have learned how to

- decompose a time series
- distinguish the graphical patterns of additive and multiplicative time series models
- use the non-parametric smoothing LOESS method in time series forecast-

ing;

- use the technique of machine learning to tune the training size to identify the optimal training size to achieve the best accuracy. In this case, the training size is considered a tuning parameter (hyper-parameter).

We will start building actual and practical forecasting models in the next module - exponential smoothing models.

15.5 Analysis Assignment

This assignment focuses on the conceptual understanding of decomposing time series and forecasting with decomposing. As mentioned in the closing paragraph of the case study in the class note, the goals of this assignment are (1) to enhance your conceptual understanding of methods of decomposition and forecasting; (2) to find the appropriate training size to produce the best performance.

The following websites contain some time series data. You can find one time series that has at least 150 observations. If your data set has more than 150, you choose 150-200 most recent observations to complete the assignment. The analysis and components in the analysis should be similar to that in the case study in the class note.

1. https://www.census.gov/econ/currentdata/datasets/index
2. https://datahub.io/search
3. https://lionbridge.ai/datasets/17-best-finance-economic-datasets-for-machine-learning/

You can also find your data set from other sites for this assignment.

Chapter 16

Exponential Smoothing Methods

Exponential smoothing methods are a family of algorithms that forecast future values by using exponentially decreasing weights of historical observations to forecast new values. Therefore as observations get older, the importance of these values diminishes exponentially. The more recent the observation the higher the associated weight. The exponentially decreasing weights are controlled by several smoothing coefficients based on the patterns of the underlying time series.

There are some obvious **advantages** of exponential smoothing methods:

- Exponential smoothing is very simple in concept and structure. It is also very easy to understand.
- Exponential smoothing is very powerful because of its exponentially-decayed weighting process.
- Exponential smoothing methods including Holt-Winters methods are appropriate for non-stationary data. In fact, they are only really appropriate if the data are non-stationary. Using an exponential smoothing method on stationary data is not wrong but is sub-optimal.
- Because exponential smoothing relies on only two pieces of data: (1). the last period's actual value; (2). the forecast value for the same period. This minimizes the use of random access memory (RAM).
- Exponential smoothing requires minimum intervention in terms of model maintenance, it is can be adapted to make large-scale forecasting.

There are also **limitations** of exponential smoothing methods:

- The method is useful for short-term forecasting only. It assumes that future patterns and trends will not change significantly from the current patterns and trends. This kind of assumption may sound reasonable in the short term. However, it creates problems for the long-term forecast.
- Exponential smoothing will lag. In other words, the forecast will be behind, as the trend increases or decreases over time.
- Exponential smoothing will fail to account for the dynamic changes at work in the real world, and the forecast will constantly require updating to respond to new information.

To avoid getting bogged down in too much technical detail for various smoothing methods, we only outline the basic components in exponential smoothing models built on several reliable smoothing algorithms under the ETS framework in the following sections.

16.1 ETS Framework

The general exponential smoothing methods combine **error (E)**, **trend (T)**, and **seasonal(S)** components in such a way that the resulting functional forms and relevant smoothing coefficients best fit the historical data. **Each term** can be combined in one of three different ways: additive (**A**), multiplicative (**M**), and None (**N**). These three terms (**E**rror, **T**rend, and **S**eason) together with the ways of combinations are referred to as the **ETS** framework. **ETS** is also called the abbreviation of **E**xponen**T**ial **S**moothing.

The following table summarizes the possible ways to construct smoothing algorithms and forecasting models within the ETS framework.

Error (E)	Trend (T)	Seasonality (S)
Additive: **A**	Additive: **A**	Additive: **A**
Multiplicative: **M**	Multiplicative: **M**	Multiplicative: **M**
	None: **N**	None: **N**
	Additive Damped **Ad**	
	Multiplicative damped: **Md**	

16.1.1 Exponential Smoothing Methods

The idea of exponential smoothing is to smooth out the random fluctuations to see a clearer signal (trend and cycle, and seasonality). Depending on the pattern of the underlying time series, there are 15 different smoothing methods.

The patterns in different smoothing methods (except for the damped additive trend) are sketched in the following figure.

ETS Smoothing Methods

Trend Component		Seasonal Component N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	A,A	A,M
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

Figure 16.1: ETS smoothing methods

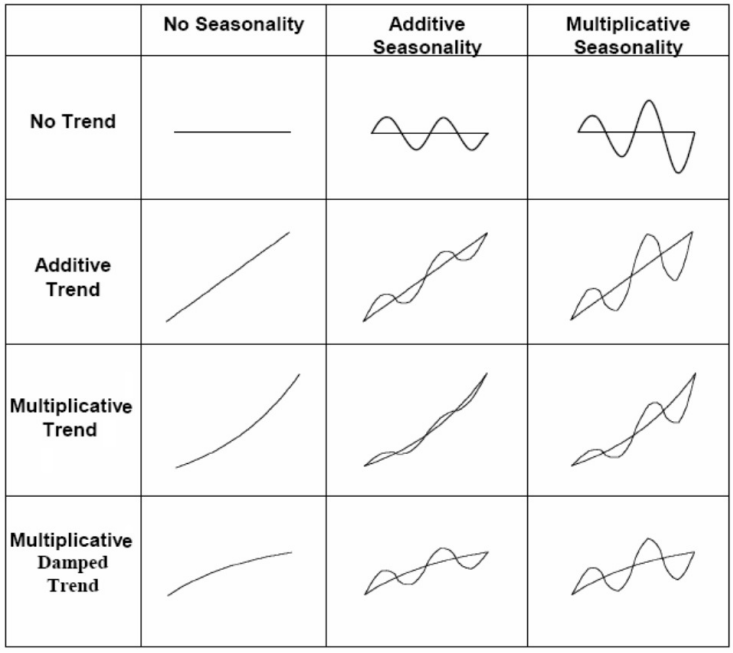


Figure 16.2: ETS smoothing charts

These patterns can be used in selecting specific smoothing models from the complete list of all possible models outlined in the following section.

16.1.2 Exponential Smoothing Models

The smoothing methods can only produce a point forecast. We can attach random error to the smoothing methods to build statistical forecast models. We can attach the error term to the combined trend and seasonality in the form of addition or multiplication. There are 30 exponential smoothing models.

ETS Smoothing Models

Error Component	Trend Component		Seasonal Component N (None)	A (Additive)	M (Multiplicative)
A (Additive)	N	(None)	A, N,N	A, N,A	A, N,M
	A	(Additive)	A, A,N	A, A,A	A, A,M
	A_d	(Additive damped)	A, A_d,N	A, A_d,A	A, A_d,M
	M	(Multiplicative)	A, M,N	A, M,A	A, M,M
	M_d	(Multiplicative damped)	A, M_d,N	A, M_d,A	A, M_d,M
M (Multiplicative)	N	(None)	M, N,N	M, N,A	M, N,M
	A	(Additive)	M, A,N	M, A,A	M, A,M
	A_d	(Additive damped)	M, A_d,N	M, A_d,A	M, A_d,M
	M	(Multiplicative)	M, M,N	M, M,A	M, M,M
	M_d	(Multiplicative damped)	M, M_d,N	M, M_d,A	M, M_d,M

Figure 16.3: ETS smoothing models

We can use the notation **ETS(error, trend, seasonality)** to represent different smoothing models. For example, (1) **ETS(A,N,N)** - simple exponential smoothing model with additive errors, (2) **ETS(A,A,N)** - additive trend with additive errors (so Holt's linear method with additive errors).

Since some of the ETS smoothing models are unstable, in this note, we only focus on a few commonly used smoothing models:

- **ETS(A,N,N)**: Simple exponential smoothing with additive errors.
- **ETS(A,A,N)**: Holt's linear method with additive errors.
- **ETS(A,A,A)**: Additive Holt-Winters' method with additive errors.
- **ETS(M,A,M)**: Multiplicative Holt-Winters' method with multiplicative errors.
- **ETS(A,A_d,N)**: Damped trend method with additive errors.

16.1.3 Estimation Smoothing Parameters in Smoothing Models

There are different ways to estimate the coefficient of the smoothing parameter. One way is to find the smoothing parameters by minimizing the means square

error (MSE). This is similar to the least square method. This is a distribution-free method and can be used to automate the exponential smoothing models.

The other method is the **likelihood** approach. After we include a random error with specific parametric distribution, we can estimate the coefficient parameters by using the likelihood method. We will not discuss estimation methods in detail.

Most of the smoothing models are implemented in the R library **forecast**. We will use this library to perform data analysis in this note.

16.2 Simple Exponential Smoothing

Exponential smoothing is a very popular scheme to produce a smoothed time series and assigns exponentially decreasing weights as the observation gets older. That is, more recent data points affect the forecast trend more heavily than older data points. This unequal weighting is accomplished by using one or more smoothing constants.

Assume we have historical data $\{Y_1, Y_2, \cdots, Y_T\}$, the forecast value of the next period is

$$Y_{(T+1)|T} = \alpha Y_T + \alpha(1-\alpha)Y_{T-1} + \alpha(1-\alpha)^2 Y_{T-2} + \alpha(1-\alpha)^3 Y_{T-3} + \cdots$$

$0 < \alpha < 1$ is called **smoothing coefficients**. The forms of the coefficients in the above expression

It provides a forecasting method that is most effective when the components like trend and seasonal factors of the time series may change over time. Several equivalent formulations of simple exponential smoothing:

- $Y_{(t+1)|t} = \alpha Y_t + (1-\alpha)Y_{t|(t-1)}$.
- Forecasting form: $\hat{Y}_{t+1|t} = \ell_t$
- Smoothing equation: $\ell_t = \alpha Y_t + (1-\alpha)\ell_{t-1}$
- Error correction form: $\ell_t = \ell_{t-1} + \alpha(y_t - \ell_{t-1}) = \ell_{t-1} + \alpha e_t$, where $e_t = y_t - \ell_{t-1} = y_t - \hat{y}_{t|t-1}$.

Example We use stock price data to build three models with different smoothing coefficients and then use the accuracy measures to choose the optimal smoothing model. The optimal smoothing coefficient will be identified by the built-in algorithm and will be reported in the output (if the α is not specified in the model formula).

```
stock=read.table("https://raw.githubusercontent.com/pengdsci/sta321/main/ww13/w13-stockprice.txt'
price=stock$V1[1:35]
```

```
fit1 = ses(price, alpha=0.2, initial="optimal", h=3)
fit2 = ses(price, alpha=0.6, initial="simple", h=3)
fit3 = ses(price, h=3)  ## alpha is unspecified, it will be estimated
plot(fit1,  ylab="Stock Price",
  xlab="Time", main="", fcol="white", type="o", lwd=2, cex=0.5)
title("Comparing SES Models: Different Smoothinh Coefficients")
lines(fitted(fit1), col="blue", type="o", cex=0.5)
lines(fitted(fit2), col="red", type="o", cex=0.5)
lines(fitted(fit3), col="darkgreen", type="o", cex=0.5)
points(fit1$mean, col="blue", pch=16) ## plot forecast values
points(fit2$mean, col="red", pch=18)
points(fit3$mean, col="darkgreen", pch=21)
legend("bottomleft",lty=1, col=c(1,"blue","red","darkgreen"),
  c("data", expression(alpha == 0.2), expression(alpha == 0.6),
  expression(alpha ==  0.9332)),pch=1)
```

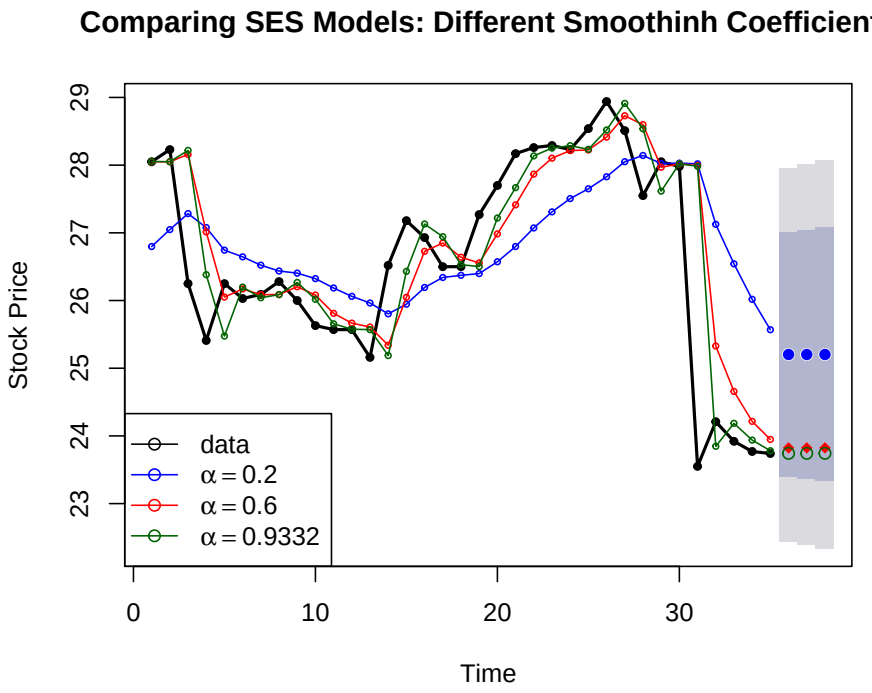


Figure 16.4: Comparing simple exponential models with various smoothing coefficients.

The following table summarizes the accuracy measures based on various smooth-

Table 16.2: The accuracy measures of simple exponential smoothing models with different smoothing coefficients.

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
alpha=0.2	-0.2278	1.3663	1.0369	-1.1287	4.0286	1.9361	0.6838
alpha=0.6	-0.2013	0.9974	0.5895	-0.8835	2.2956	1.1006	0.3024
optimal alpha = 0.09332	-0.1320	0.9421	0.5158	-0.5833	1.9991	0.9630	-0.0126

ing models with different smoothing coefficients.

```
accuracy.table = round(rbind(accuracy(fit1), accuracy(fit2), accuracy(fit3)),4)
row.names(accuracy.table)=c("alpha=0.2", "alpha=0.6", "optimal alpha = 0.09332")
kable(accuracy.table, caption = "The accuracy measures of simple exponential
      smoothing models with different smoothing coefficients.")
```

Remark: The above measures are based on the training data (i.e., based on the observed values and the fitted values). We can also hold up test data to calculate the actual accuracy measures.

16.3 Holt's Smoothing Model: Linear (additive) Trend

Holt generalized simple exponential smoothing by adding a trend parameter to allow the forecasting of data with a linear trend. The model components are given in the following:

- Forecast function: $\hat{y}_{t+h|t} = l_t + hb_t$.
- Level: $l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1})$.
- Trend: $b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1}$.
- Error: $e_t = y_t - l_{t-1} + b_t = y_t - \hat{y}_{t|t-1}$.

where α is the smoothing coefficient for level and β^* the smoothing coefficient of trend. The smoothing coefficients are estimated by minimizing the sum of the squared error (SSE) of the model.

The R function **holt()** computes the smoothing coefficients and forecasts future values for a given h period of future values.

Example 2: We apply Holt's method to annual passenger numbers for Australian airlines from 1990 to 2016.

```
ausair=read.table("https://raw.githubusercontent.com/pengdsci/sta321/main/ww13/w13-ausair.txt")[,
air = ts(ausair,start=1990,end=2016)
fit0 = holt(air, initial="simple", exponential=TRUE,h=5)  ### optimal alpha and beta
fit1 = holt(air, alpha=0.8, beta=0.2, exponential=TRUE, initial="simple", h=5)
```

```
###### Plot the original data
plot(fit0, lwd=2,type="o", ylab="Air Passengers", xlab="Time",
     fcol="white", ylim=c(10, 110))
lines(fitted(fit0), col="red")
lines(fitted(fit1), col="blue")
#points(fit0, col="black", pch=1)
points(fitted(fit0), col="red", pch=16, cex=0.6)
points(fitted(fit1), col="blue", pch=22, cex=0.6)
########################
lines(fit0$mean, col="red", type="o")
lines(fit1$mean, col="blue", type="o")
legend("topleft", lty=1, col=c("red","black","blue"),pch=c(16,1,22),
   c("Holt's Exp Trend(optimal)","Data","Holt's Exp trend"), bty="n")
```

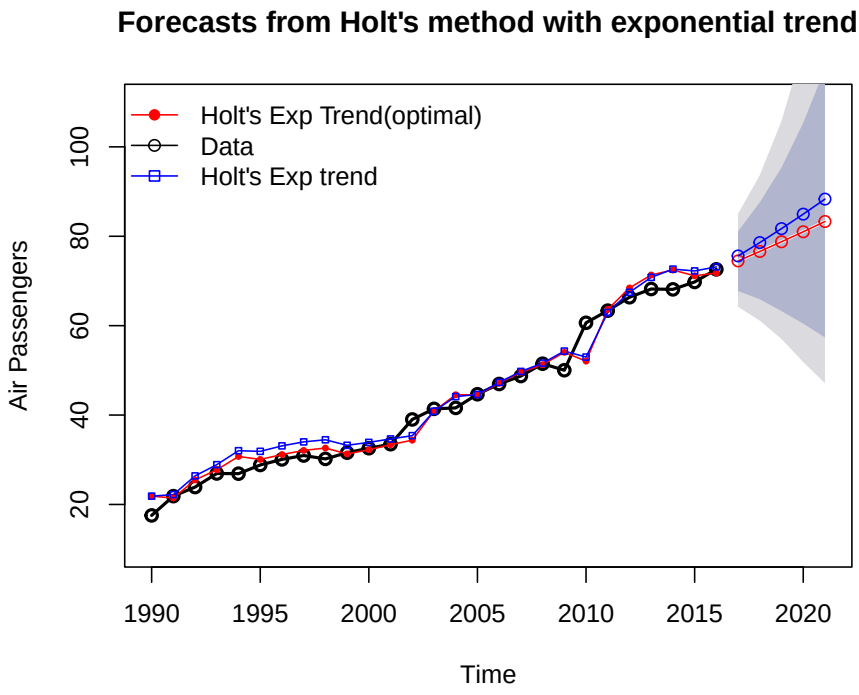


Figure 16.5: Comparing Holt's exponential trend models with various smoothing coefficients.

As expected, we can see from the above chart that

- the smoothing model with exponential trend works equally well as the linear trend model for air passenger data;

- the smoothing model with exponential trend works better than the linear trend model for the beverage data;

16.4 Damped Trend Methods

Empirical evidence indicates that these methods tend to over-forecast, especially for longer forecast horizons. Gardner's damped trend models are shown to be effective in improving accuracy for prediction.

To capture the damped trend (the trend component curve flattens over time instead of being linear), in addition to the two smoothing parameters α and β^* in linear and exponential trend models, Gardener added a third parameter ϕ $(0 < \phi < 1)$ that damps the trend as h gets bigger.

For an additive (linear) trend model, the additional parameter is added in the model component in the following form.

- Forecast Model: $\widehat{y}_{t+h|t} = \mathfrak{l}_t + (1 + \phi + \phi^2|\cdots + \phi^h)b_t$
- Level: $\mathfrak{l}_t = \alpha y_t + (1-\alpha)(\mathfrak{l}_{t-1} + \phi b_{t-1})$
- Trend: $b_t = \beta^*(\mathfrak{l}_t - \mathfrak{l}_{t-1}) + (1-\beta^*)\phi b_{t-1}$. where ϕ is called the **damping parameter**.

Let error $e_t = y_t - (\mathfrak{l}_{t-1} + b_{t-1}) = y_t - \widehat{y}_{t|t-1}$, then the level and trend can be re-expressed as

- Level: $\mathfrak{l}_t = \mathfrak{l}_{t-1} + \phi b_{t-1} + \alpha e_t$.
- Additive Trend: $b_t = \phi b_{t-1} + \alpha\beta^* e_t$.

For a multiplicative (exponential) trend mode, the additional parameter is added in the model component in the following form.

- Forecast Model: $\widehat{y}_{t+h|t} = \mathfrak{l}_t b_t^{(1+\phi+\phi^2|\cdots+\phi^h)}$.
- Level: $\mathfrak{l}_t = \alpha y_t + (1-\alpha)\mathfrak{l}_{t-1} b_{t-1}^{\phi}$
- Multiplicative Trend: $b_t = \beta^*\mathfrak{l}_t/\mathfrak{l}_{t-1} + (1-\beta^*)b_{t-1}^{\phi}$.

Since this is a modification of the smoothing model with the exponential trend, b_t and h are interpreted in the same way as in the exponential trend smoothing model and ϕ is between 0 and 1, exclusively.

Let error $e_t = y_t - \widehat{y}_{t-1|t} = y_t - \mathfrak{l}_{t-1}b_{t-1}$. The corresponding error correction form is given by

- Level: $\mathfrak{l}_t = \alpha e_t + \mathfrak{l}_{t-1} b_{t-1}^{\phi}$.
- Multiplicative Trend: $b_t = b_{t-1}^{\phi} + \alpha\beta^* e_t/\mathfrak{l}_{t-1}$

Table 16.3: The accuracy measures of various exponential smoothing models

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
SES	22.8967	38.5522	32.1076	0.4518	0.6392	0.9738	0.5930
Holt's	-1.1647	31.0537	26.0678	-0.0248	0.5227	0.7906	0.6132
Exponential	-0.6055	31.2007	26.2099	-0.0140	0.5247	0.7949	0.6100
Additive Damped	-0.8253	31.4965	25.5226	-0.0232	0.5113	0.7740	0.6218
Multiplicative Damped	0.6962	31.3254	25.7591	0.0084	0.5160	0.7812	0.6189

Example 3: We use Australian GDP data as an example to illustrate the application of the R function holt() for the damped trend models (both additive and multiplicative models).

```
ausgdp = read.table("https://raw.githubusercontent.com/pengdsci/sta321/main/ww13/w13-au
ausgdp0=ts(ausgdp$V1,frequency=4,start=1971+2/4)
ausgdp1=ausgdp$V1
fit1 = ses(ausgdp1)
fit2 = holt(ausgdp1)
fit3 = holt(ausgdp1,exponential=TRUE)
fit4 = holt(ausgdp1,damped=TRUE)      ## additive damping
fit5 = holt(ausgdp1,exponential=TRUE,damped=TRUE)  ## multiplicative dampling
###
plot(fit3, type="o", ylab="Australia GDP",flwd=1,
main="Comparison of various smoothing models")
lines(fitted(fit1),col=2)
lines(fitted(fit2),col=3)
lines(fitted(fit4),col=5)
lines(fitted(fit5),col=6)
######
lines(fit1$mean,col=2)
lines(fit2$mean,col=3)
lines(fit4$mean,col=5)
lines(fit5$mean,col=6)
legend("topleft", lty=1, pch=1, col=1:6,
    c("Data","SES","Holt's","Exponential",
      "Additive Damped","Multiplicative Damped"), cex=0.8, bty="n")
```

```
accuracy.table = round(rbind(accuracy(fit1), accuracy(fit2), accuracy(fit3),
                             accuracy(fit4), accuracy(fit5)),4)
row.names(accuracy.table)=c("SES","Holt's","Exponential",
                    "Additive Damped","Multiplicative Damped")
kable(accuracy.table, caption = "The accuracy measures of various exponential
      smoothing models")
```

From the above accuracy table, we can see the additive and multiplicative

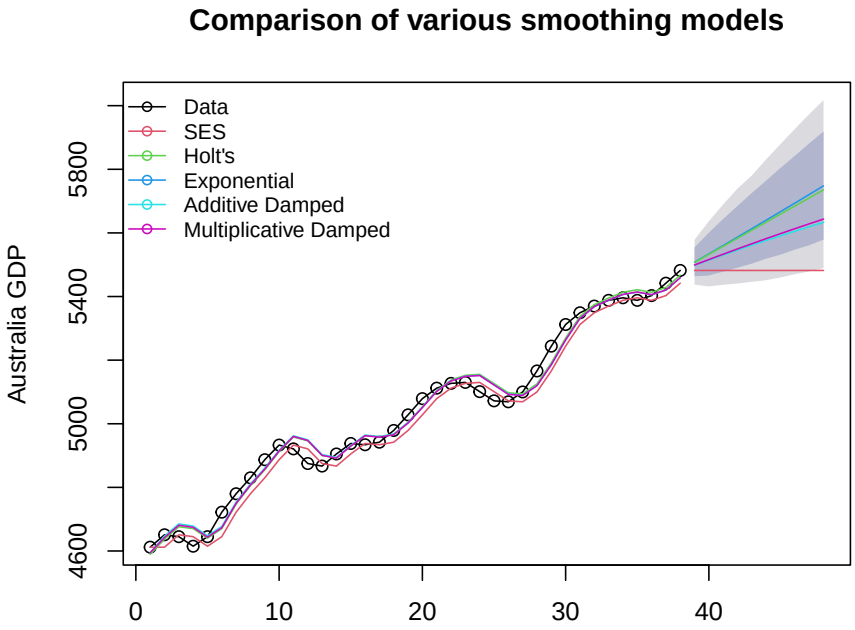


Figure 16.6: Comparing Holt's exponential damped trend models.

damped trend models are better than other models. The additive damped model is marginally better than the multiplicative damped model.

16.5 Holt-Winters Model for Trend and Seasonal Data

Holt and Winter's exponential smoothing method is used to deal with time series containing both trend and seasonal variation. There are two Holt-Winter (HW) models: Additive and Multiplicative models.

The additive method is preferred when the seasonal variations are roughly constant through the series, while **the multiplicative method** is preferred when the seasonal variations are changing proportionally to the level of the series.

The Holt-Winters seasonal method comprises the forecast equation and three smoothing equations — one for the level l_t, one for trend b_t, and one for the seasonal component denoted by s_t, with smoothing parameters α, β^* and γ.

Additive Holt-Winters Model Components of additive HW model:

- Level: $l_t = \alpha(y_t - s_{t-s}) + (1-\alpha)(l_{t-1} + b_{t-1})$.
- Trend: $b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1}$.
- Seasonality: $s_t = \gamma(y_t - l_{t-1}) + (1-\gamma)s_{t-s}$.
- Forecast Model: $\widehat{y}_{t+h|t} = l_t + hb_t + s_{t-s+h}$.

where s is the length of the seasonal cycle, for $0 \leq \alpha \leq 1$, $0 \leq \beta^* \leq 1$, and $0 \leq \gamma \leq 1$. h is the number of periods to be predicted.

Let $e_t = \widehat{y}_t - \widehat{y}_{t|t-1} = y_t - (l_{t-1} + b_{t-1} + s_{t-1})$ be the one-step training error, we can re-express level, trend and seasonality equation in the following form:

- Level: $l_t = l_{t-1} + b_{t-1} + \alpha e_t$.
- Trend: $b_t = b_{t-1} + \alpha\beta^* e_t$.
- Seasonality: $s_t = s_{t-2} + \gamma e_t$.

Interpretations of individual components:

Level: the current level is the weighted average of the difference between the current observation and previous seasonality and the sum of the previous level and trend.

Trend: the current trend is the weighted average of the previous trend and the difference between the current level and the previous level.

Seasonality: the current seasonality is the weighted average of the previous seasonality and the difference between the current observation and the current

level.

Multiplicative Holt-Winters Model: An alternative Holt-Winter's model multiplies the forecast by a seasonal factor. Its equations are:

- Level: $l_t = \alpha y_t / s_{t-s} + (1-\alpha)(l_{t-1} + b_{t01})$.
- Trend: $b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1}$.
- Seasonality: $s_t = \gamma y_t/(l_{t-1} + b_{t-1}) + (1-\gamma)s_{t-s}$.
- Forecast Model: $\widehat{y}_{t+h|t} = (l_t + hb_t)s_{t-s-h}$.

where s is the length of the seasonal cycle, for $0 \leq \alpha \leq 1$, $0 \leq \beta^* \leq 1$, and $0 \leq \gamma \leq 1$. h is the number of periods to be predicted.

Similarly, we denote the one-step error for the multiplicative model to be $e_t = \widehat{y}_t - \widehat{y}_{t|t-1} = y_t - (l_{t-1} + b_{t-1} + s_{s-t})$, the error correction representation of the model can be written as

- Level: $l_t = l_{t-1} + b_{t-1} + \alpha e_t / s_{t-s}$.
- Trend: $b_t = b_{t-1} + \alpha\beta^* e_t / s_{t-s}$.
- Seasonality: $s_t = s_{t-s} + \gamma e_t/(l_{t-1} + b_{t-1})$.

Holt-Winters Model with a Damped Trend and Multiplicative Seasonality. This is a simple modification of the HW multiplicative model with a damped trend. The model formulation is given below

- Level: $l_t = \alpha y_t / s_{t-s} + (1-\alpha)(l_{t-1} + \phi b_{t-1})$.
- Trend: $b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)\phi b_{t-1}$.
- Seasonality: $s_t = \gamma y_t/(l_{t-1} + \phi b_{t-1}) + (1-\gamma)s_{t-s}$.
- Forecast Model: $\widehat{y}_{t+h|t} = \left[l_t + (1 + \phi + \phi^2 + \cdots + \phi^h)b_t\right] s_{t-s+h}$.

where $0 < \phi < 1$.

Three R functions can be used to fit Holt-Winters model: **hw()**,**ets()**, and **HoltWinters()** in package **{forecast}**.

- The estimation of parameters used in function **HoltWinters()** uses **optim()** which requires the initial values for the parameters α, β^* and γ. The default initial values of these parameters are 0.3, 0.1, and 0.1. You can provide your own more accurate values. **HoltWinters()** is using heuristic values for the initial states and then estimating the smoothing parameters by optimizing the MSE.
- **ets()** also use **optim()** to estimate the parameters and the initial states by optimizing the likelihood function (which is only equivalent to optimizing the MSE for the linear additive models).
- **hw()** has the option to fit a HW model with a damped trend. **HoltWinters()** does not have the option.

Table 16.4: The accuracy measures of various Holt-Winter's exponential smoothing models

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Holt Winters' Additive	0.2052	12.5660	9.9395	-4.4789	13.3916	0.4483	0.3564
Holt Winters' Multiplicative	0.1987	14.6071	11.8053	-4.5487	16.6051	0.5325	0.4842
HW Additive Damped Trend	1.2919	12.6380	10.0841	-3.1055	13.2677	0.4549	0.3544

Example 4: We use a simulated time series to illustrate how to R functions to fit various HW models.

HW models: additive, multiplicative and damped HW models with R function **hw()**.

```
dat01=read.table("https://raw.githubusercontent.com/pengdsci/sta321/main/ww13/w13-ts3.t
datset=ts(dat01$V1+55, start=2000, frequency=12)
## Model building
fit1 = hw(datset,h=12, seasonal="additive")   # default h = 10
fit2 = hw(datset,h=12, seasonal="multiplicative")
fit3 = hw(datset,h=12, seasonal="additive",damped=TRUE)
### plots
plot(fit2,ylab="Artificial Series",main="Various Holt-Winters Models",
     type="o", fcol="white", xlab="Year", ylim=c(0,300), cex = 0.6)
lines(fitted(fit1), col="red", lty=2, cex = 0.6)
lines(fitted(fit2), col="green", lty=2, cex = 0.6)
lines(fitted(fit3), col="blue", lty=2, cex = 0.6)
##
lines(fit1$mean, type="o", col="red", cex = 0.6)
lines(fit2$mean, type="o", col="green", cex = 0.6)
lines(fit3$mean, type="o", col="blue", cex = 0.6)
###
legend("topleft",lty=1, pch=1, col=c("black", "red", "green", "blue"),
  c("data","Holt Winters' Additive","Holt Winters' Multiplicative",
    "HW Additive Damped Trend"), cex=0.7, bty = "n")
```

The accuracy measures of the three models are summarized in the following table.

```
accuracy.table = round(rbind(accuracy(fit1), accuracy(fit2), accuracy(fit3)),4)
row.names(accuracy.table)=c("Holt Winters' Additive","Holt Winters' Multiplicative",
    "HW Additive Damped Trend")
kable(accuracy.table, caption = "The accuracy measures of various Holt-Winter's
        exponential smoothing models")
```

We can see from the above table that Holt-Winter's additive model outperformed the other two models.

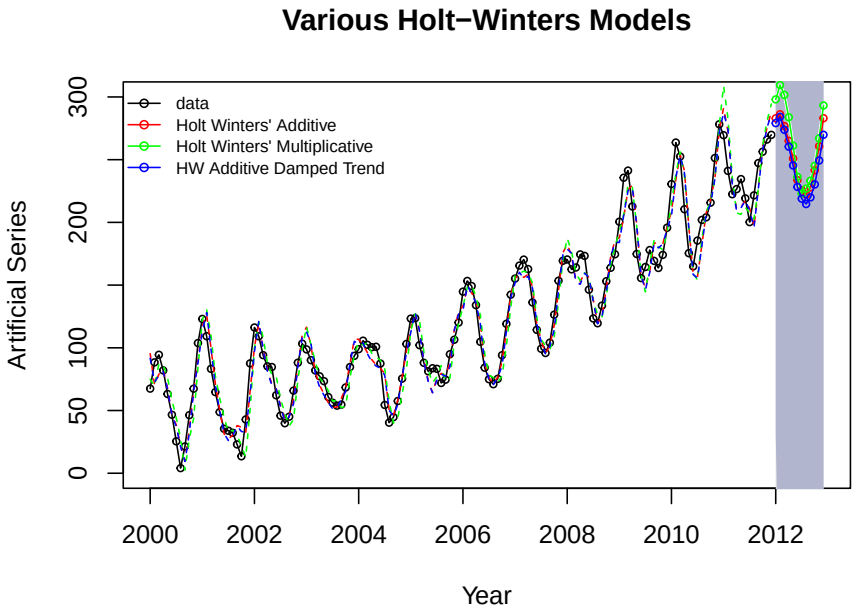


Figure 16.7: Comparing Holt-Winter's (damped) trend and seasonal models.

HW models: additive, multiplicative, and damped HW models with R function HoltWinters(). Since **HoltWinters()** does not have the option to make predict values directly. We need to use the R function **predict.HoltWinters()** to make the prediction.

```
dat01=read.table("https://raw.githubusercontent.com/pengdsci/sta321/main/ww13/w13-ts3.t
datset=ts(dat01$V1+55, start=2000, frequency=12)
## Model building
fit1 = HoltWinters(datset, seasonal="additive")   # default h = 10
fit2 = HoltWinters(datset, seasonal="multiplicative")
### plots
plot(fit1,ylab="Artificial Series",main="Various Holt-Winters Models",
     type="o", xlab="Year", cex = 0.7)
#lines(fitted(fit1)[,1], col="red", lty=2)
lines(fitted(fit2)[,1], col="blue", lty=2, cex = 0.7)
###
legend("topleft",lty=1, pch=1, col=c("black", "red", "blue"),
  c("data","Holt Winters' Additive","Holt Winters' Multiplicative"),
  cex = 0.7, bty = "n")
```

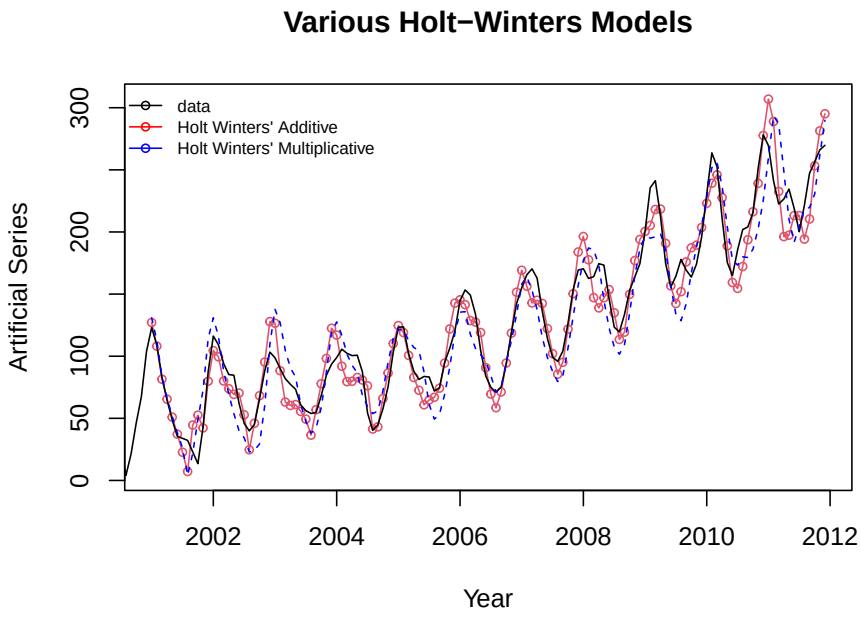


Figure 16.8: Comparing Holt's exponential trend and seasonal models.

The accuracy measures of the three models are summarized in the following

table.

```
pred = predict(fit1, 50, prediction.interval = TRUE)
pred2 = predict(fit2, 50, prediction.interval = TRUE)
plot(fit1, pred, lwd=2, main="HW Models with R Function HoltWinters()")
lines(pred[,2], col="red", lwd=1, lty=3)
lines(pred[,3], col="red", lwd=1, lty=3)
lines(fitted(fit2)[,1], col="blue", lty=1, lwd=2)
lines(pred2[,2], col="blue", lty=2)
lines(pred2[,1], col="blue", lty=1, lwd=2)
lines(pred2[,3], col="blue", lty=2)
legend("topleft",lty=1, col=c("black", "red", "blue"),
  c("data","Holt Winters' Additive","Holt Winters' Multiplicative"),
  cex = 0.7, bty = "n")
```

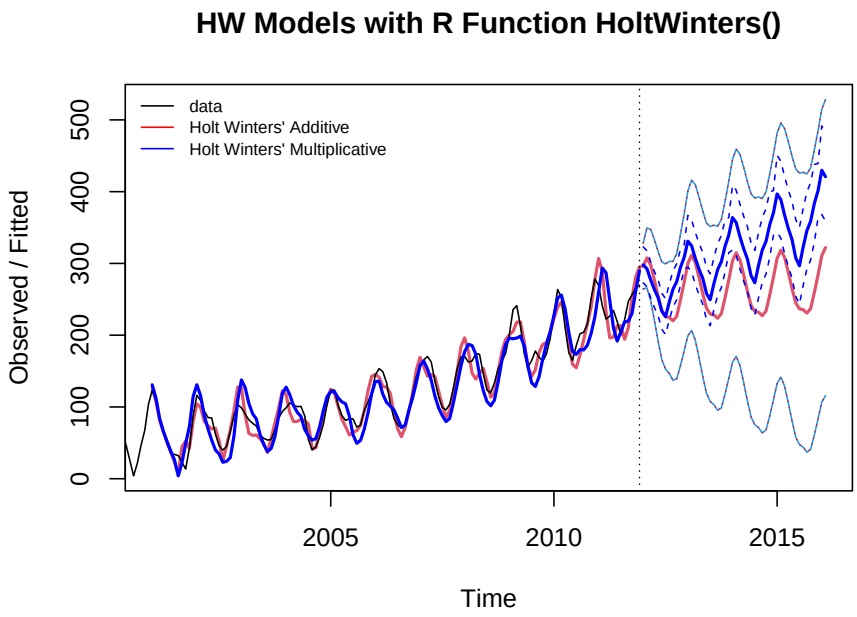


Figure 16.9: Exponential damped trend models with Holt-Winters Filtering.

The above figure shows that the original series is a typical exponential trend with a seasonal pattern. As expected, the Holt-Winters additive model performed poorly (a very wide prediction band).

Table 16.5: The accuracy measures of various exponential smoothing models based on the training data

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
SES	16.5017	309.4093	257.6429	0.1583	5.0742	1.3090	0.0043
Holt Linear	-2.5111	312.4926	262.2489	-0.2495	5.2023	1.3324	-0.0007
Holt Add. Damped	10.8546	311.8373	260.8383	0.0082	5.1661	1.3252	-0.0016
Holt Exp. Damped	11.6379	312.2482	261.0031	0.0305	5.1699	1.3261	-0.0029
HW Add.	-8.3236	117.5757	91.9541	-0.2135	1.7920	0.4672	-0.0215
HW Exp.	2.3822	121.7157	95.9449	-0.0051	1.8727	0.4875	0.1052
HW Add. Damp	11.7728	118.5088	93.8735	0.1786	1.8225	0.4769	-0.0376
HW Exp. Damp	11.9044	118.6752	92.4620	0.1635	1.7994	0.4698	-0.0900

16.6 Case study

In this case study, we use the beverage consumption data to compare various models introduced in this note.

```
beverage = read.table("https://raw.githubusercontent.com/pengdsci/sta321/main/ww13/w13
test.bev = beverage$V1[169:180]
train.bev = beverage$V1[1:168]
bev=ts(beverage$V1[1:168], start=2000, frequency = 12)
fit1 = ses(bev, h=12)
fit2 = holt(bev, initial="optimal", h=12)              ## optimal alpha and beta
fit3 = holt(bev,damped=TRUE, h=12 )                    ## additive damping
fit4 = holt(bev,exponential=TRUE, damped=TRUE, h =12) ## multiplicative damp
fit5 = hw(bev,h=12, seasonal="additive")               ## default h = 10
fit6 = hw(bev,h=12, seasonal="multiplicative")
fit7 = hw(bev,h=12, seasonal="additive",damped=TRUE)
fit8 = hw(bev,h=12, seasonal="multiplicative",damped=TRUE)
```

```
accuracy.table = round(rbind(accuracy(fit1), accuracy(fit2), accuracy(fit3), accuracy(
                             accuracy(fit5), accuracy(fit6), accuracy(fit7), accuracy(
row.names(accuracy.table)=c("SES","Holt Linear","Holt Add. Damped", "Holt Exp. Damped"
                            "HW Add.","HW Exp.","HW Add. Damp", "HW Exp. Damp")
kable(accuracy.table, caption = "The accuracy measures of various exponential smoothing
      based on the training data")
```

The above table shows that the HW additive seems to be the most appropriate.

```
par(mfrow=c(2,1), mar=c(3,4,3,1))
###### plot the original data
pred.id = 169:180
plot(1:168, train.bev, lwd=2,type="o", ylab="Beverage", xlab="",
     xlim=c(1,180), ylim=c(2500, 7500), cex=0.3,
     main="Non-seasonal Smoothing Models")
```

```
lines(pred.id, fit1$mean, col="red")
lines(pred.id, fit2$mean, col="blue")
lines(pred.id, fit3$mean, col="purple")
lines(pred.id, fit4$mean, col="navy")
##
points(pred.id, fit1$mean, pch=16, col="red", cex = 0.5)
points(pred.id, fit2$mean, pch=17, col="blue", cex = 0.5)
points(pred.id, fit3$mean, pch=19, col="purple", cex = 0.5)
points(pred.id, fit4$mean, pch=21, col="navy", cex = 0.5)
#points(fit0, col="black", pch=1)
legend("bottomright", lty=1, col=c("red","blue","purple", "navy"),pch=c(16,17,19,21),
   c("SES","Holt Linear","Holt Linear Damped", "Holt Multiplicative Damped"),
   cex = 0.7, bty="n")
############
plot(1:168, train.bev, lwd=2,type="o", ylab="Beverage", xlab="",
     xlim=c(1,180), ylim=c(2500, 7500), cex=0.3,
     main="Holt-Winterd Teend and Seasonal Smoothing Models")
lines(pred.id, fit5$mean, col="red")
lines(pred.id, fit6$mean, col="blue")
lines(pred.id, fit7$mean, col="purple")
lines(pred.id, fit8$mean, col="navy")
##
points(pred.id, fit5$mean, pch=16, col="red", cex = 0.5)
points(pred.id, fit6$mean, pch=17, col="blue", cex = 0.5)
points(pred.id, fit7$mean, pch=19, col="purple", cex = 0.5)
points(pred.id, fit8$mean, pch=21, col="navy", cex = 0.5)
###
legend("bottomright", lty=1, col=c("red","blue","purple", "navy"),pch=c(16,17,19,21),
   c("HW Additive","HW Multiplicative","HW Additive Damped", "HW Multiplicative Damped"),
   cex = 0.7, bty="n")
```

We can see from the above accuracy table that HW's linear trend with an additive seasonal model is the best of the eight smoothing models. This is consistent with the patterns in the original serial plot.

Since we train the model with the training data and identify the best model using both training and testing data. Both methods yield the same results. To use the model for real-forecast, we need to refit the model using the entire data to update the smoothing parameters in the final working model.

```
acc.fun = function(test.data, mod.obj){
  PE=100*(test.data-mod.obj$mean)/mod.obj$mean
  MAPE = mean(abs(PE))
  ###
  E=test.data-mod.obj$mean
  MSE=mean(E^2)
```

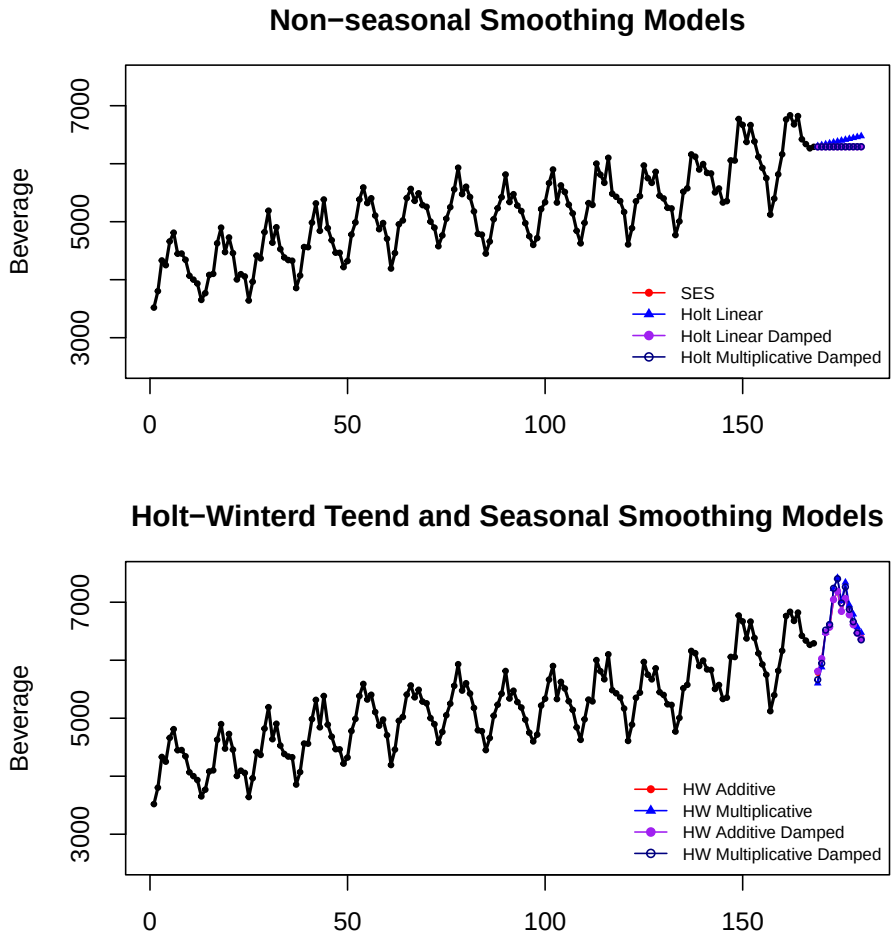


Figure 16.10: Case study: Comparing various exponential smoothing models.

Table 16.6: The accuracy measures of various exponential smoothing models based on the testing data

	MSE	MAPE
SES	268795.08	6.325197
Holt.Add	221347.08	5.411444
Holt.Add.Damp	268829.31	6.326006
Holt.Exp	267988.88	6.306752
HW.Add	49429.46	3.055574
HW.Exp	87301.79	3.607590
HW.Add.Damp	51295.11	3.091598
HW.Exp.Damp	72440.01	3.422931

```
  ###
  accuracy.metric=c(MSE=MSE, MAPE=MAPE)
  accuracy.metric
}
```

```
pred.accuracy = rbind(SES =acc.fun(test.data=test.bev, mod.obj=fit1),
                      Holt.Add =acc.fun(test.data=test.bev, mod.obj=fit2),
                      Holt.Add.Damp =acc.fun(test.data=test.bev, mod.obj=fit3),
                      Holt.Exp =acc.fun(test.data=test.bev, mod.obj=fit4),
                      HW.Add =acc.fun(test.data=test.bev, mod.obj=fit5),
                      HW.Exp =acc.fun(test.data=test.bev, mod.obj=fit6),
                      HW.Add.Damp =acc.fun(test.data=test.bev, mod.obj=fit7),
                      HW.Exp.Damp =acc.fun(test.data=test.bev, mod.obj=fit8))
kable(pred.accuracy, caption="The accuracy measures of various exponential smoothing models
      based on the testing data")
```

We can see from the above accuracy table that HW's linear trend with an additive seasonal model is the best of the eight smoothing models. This is consistent with the patterns in the original serial plot.

In the previous analysis, we train the model with the training data and identify the best model using both training and testing data. **In real-forecast, we need to refit the model at the very end using the entire data to update the smoothing parameters in the final working model**.

```
beverage = read.table("https://raw.githubusercontent.com/pengdsci/sta321/main/ww13/w13-beverage01
bev=ts(beverage$V1[1:180], start=2000, frequency = 12)
final.model = hw(bev,h=12, seasonal="additive")
smoothing.parameter = final.model$model$par[1:3]
kable(smoothing.parameter, caption="Estimated values of the smoothing parameters in
      Holt-Winters linear trend with additive seasonality")
```

Table 16.7: Estimated values of the smoothing parameters in Holt-Winters linear trend with additive seasonality

	x
alpha	0.3443447
beta	0.0001004
gamma	0.0003443

In summary, the updated values of the three smoothing parameters in the Holt-Winters linear trend and with additive seasonality using the entire data are given in the above table.