

Introductory Statistics Using R

Contents

1. INTRODUCTION TO R.....	3
2. INTRODUCTION TO RSTUDIO	6
3. USING R AS A CALCULATOR.....	8
4. VECTORS	9
5. MATRICES	11
6. INPUT DATA INTO R.....	14
7. LISTING AND REMOVING OBJECTS FROM THE ENVIRONMENT.....	18
8. READING AND WRITING ASCII DATA	19
9. UNIVARIATE DESCRIPTIVE STATISTICS.....	20
10. BIVARIATE DESCRIPTIVE STATISTICS.....	22
11. PLOTS.....	26
12. SOME DISCRETE AND CONTINUOUS DISTRIBUTIONS	32
13. ASSESSING NORMALITY	37
14. LOOPS AND CONDITIONAL STATEMENTS	39
15. FUNCTIONS.....	41
16. MONTE CARLO SIMULATIONS.....	43
17. CONFIDENCE INTERVAL ESTIMATIONS.....	50
17.1 Confidence Intervals of a Population Mean	50
17.2 Confidence Intervals of a Population Proportion.....	52
17.3 Confidence Intervals of a Population Variance.....	53
17.4 Understanding Confidence Intervals	55
18. ONE-SAMPLE AND TWO-SAMPLE STATISTICAL INFERENCES.....	57
18.1 Population means.....	57
18.1.1 Z-TEST AND T-TEST OF A POPULATION MEAN.....	57
18.1.2 Z-TEST OF THE DIFFERENCE BETWEEN TWO POPULATION MEANS.....	58
18.1.3 T-TEST OF THE DIFFERENCE BETWEEN TWO INDEPENDENT POPULATION MEANS.....	58
18.1.4 PAIRED T-TEST	61
18.2 Population proportions.....	62
18.2.1 ONE-SAMPLE TEST	62
18.2.2 TWO-SAMPLE TEST.....	63

18.3 Population variances	66
18.3.1 ONE-SAMPLE TEST - CHI-SQUARE TEST OF A POPULATION VARIANCE	66
18.3.2 TWO-SAMPLE TEST – F-TEST OF THE DIFFERENCE BETWEEN TWO POPULATION VARIANCES	67
18.4 A Simulation Study of Probability of Type II error	68
19. CATEGORICAL DATA ANALYSIS	71
19.1 Goodness-of-fit Test	71
19.2 Analysis of Contingency Tables	72
20. ANALYSIS OF VARIANCE	74
20.1 One-Way ANOVA	74
20.2 Two-Way ANOVA for Randomized Block Design	83
20.3 Two-Way ANOVA for Factorial Design	90
21. LINEAR REGRESSION MODELS	98
21.1 Linear correlation	98
21.2 Understanding Simple Linear Regression Models	100
21.3 Analysis of Simple Linear Regression Models	102
21.4 Multiple Linear Regression Models	112
22. NONPARAMETRIC TESTS	124
22.1 Sign Test	124
22.2 Wilcoxon Signed-Rank Test	125
22.3 Wilcoxon Rank-Sum Test	126
22.4 Kruskal-Wallis Test	128
22.5 Friedman Rank-Sum Test	129
22.6 Spearman's Rank Correlation Test	131

1. INTRODUCTION TO R

R is a **free** software environment for statistical computing and graphics (<https://www.r-project.org>). It is a GNU project (an operating system that is free software) 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. The name S refers to statistics. A commercial version of S is marketed as S-PLUS by the company Insightful (<http://www.insightful.com>). The name R was chosen because R is the letter in the alphabet next to S, and because it was developed by two people whose first names started with the letter R (Robert Gentleman and Ross Ihaka, at the University of Auckland, New Zealand). The interfaces of R and S-PLUS are very different. R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS. For more information, see <https://www.r-project.org/about.html>.

The major web resource for R is <http://www.r-project.org/>. The latest version is R-3.5.1 (2018-7-2). The download sites are available with mirrors in many countries (<https://cran.r-project.org/mirrors.html>). For example, some sites in USA

https://cran.cnr.berkeley.edu/	University of California, Berkeley, CA
http://ftp.ussg.iu.edu/CRAN/	Indiana University
http://cran.case.edu/	Case Western Reserve University, Cleveland, OH
http://lib.stat.cmu.edu/R/CRAN/	Statlib, Carnegie Mellon University, Pittsburgh, PA
https://cran.fhcrc.org/	Fred Hutchinson Cancer Research Center, Seattle, WA

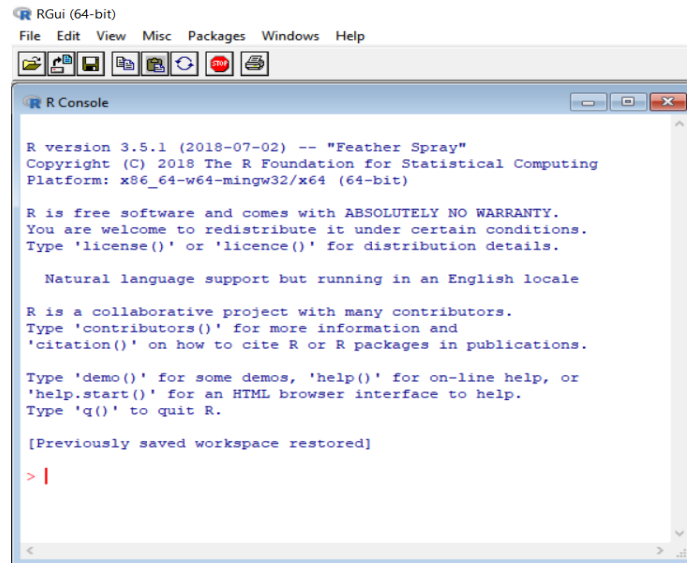
One of the great benefits of R is its community of users and the many user-contributed packages they're willing to share with the rest of us. Currently, the CRAN package repository features about 15,000 available packages (<https://cran.r-project.org/web/packages/>). The purpose of this introduction is to let you familiarize the software and be able to do statistical analysis using R. For more information, use the help facility in R and GOOGLE.

The following shows you how to install R and R packages.

Step 1: Download R by going to one of the mirrors such as <https://cloud.r-project.org/>. Choose the Linux, Mac or Windows version depending on your operating system. You then install R by double click the downloaded file. Accept all the defaults during installation.

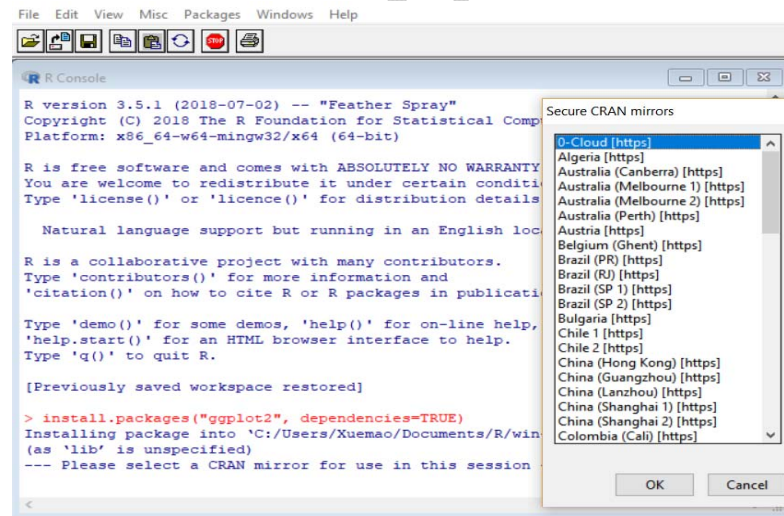
Step 2: Start R as administrator (Windows users need to right-click the R icon and click the corresponding item); the following is the R interface. You can enter commands one at a time at

the command prompt (`>`) or run a set of commands from a source file. Keep in mind that R is case-sensitive.

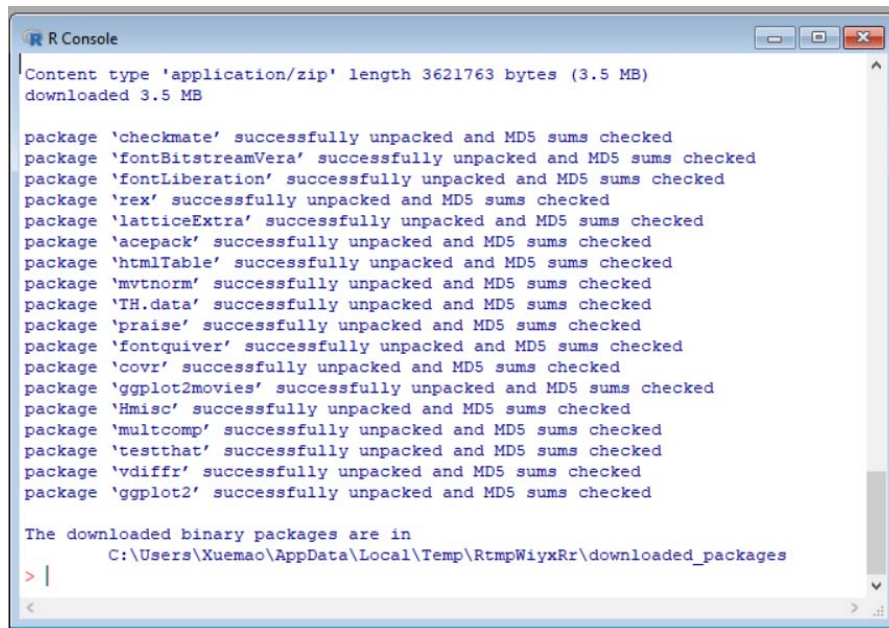


To install an R package, for example `ggplot2`, you can run the following command after the command prompt (`>`):

```
install.packages("ggplot2", dependencies=TRUE)
```



The option `dependencies=TRUE` lets you install uninstalled packages which these packages depend on/link to/import/suggest (and so on recursively). Then you choose any mirror and click “OK” to continue.



```

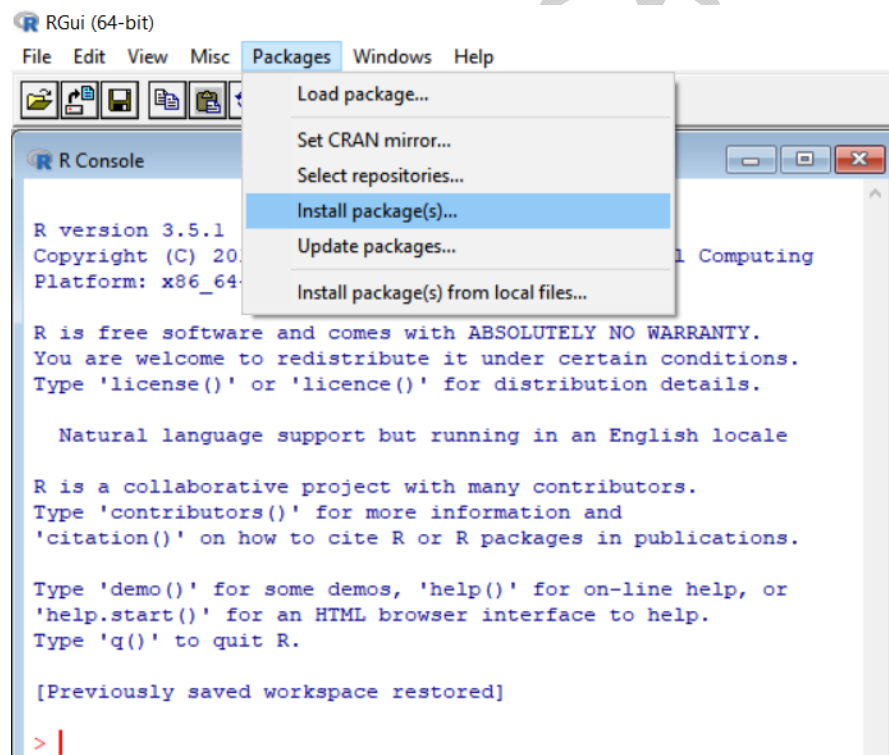
R Console
Content type 'application/zip' length 3621763 bytes (3.5 MB)
downloaded 3.5 MB

package 'checkmate' successfully unpacked and MD5 sums checked
package 'fontBitstreamVera' successfully unpacked and MD5 sums checked
package 'fontLiberation' successfully unpacked and MD5 sums checked
package 'rex' successfully unpacked and MD5 sums checked
package 'latticeExtra' successfully unpacked and MD5 sums checked
package 'acepack' successfully unpacked and MD5 sums checked
package 'htmlTable' successfully unpacked and MD5 sums checked
package 'mvtnorm' successfully unpacked and MD5 sums checked
package 'TH.data' successfully unpacked and MD5 sums checked
package 'praise' successfully unpacked and MD5 sums checked
package 'fontquiver' successfully unpacked and MD5 sums checked
package 'covr' successfully unpacked and MD5 sums checked
package 'ggplot2movies' successfully unpacked and MD5 sums checked
package 'Hmisc' successfully unpacked and MD5 sums checked
package 'multcomp' successfully unpacked and MD5 sums checked
package 'testthat' successfully unpacked and MD5 sums checked
package 'vdiff' successfully unpacked and MD5 sums checked
package 'ggplot2' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\Xuemao\AppData\Local\Temp\RtmpWiyxRr\downloaded_packages
> |

```

Another way to install a package is to use the R menu shown below.



```

RGui (64-bit)
File Edit View Misc Packages Windows Help

Load package...
Set CRAN mirror...
Select repositories...
Install package(s)...
Update packages...
Install package(s) from local files...

R Console
R version 3.5.1
Copyright (C) 2018
Platform: x86_64

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]
> |

```

For Mac users, if you do not have the latest software, you should first update your Mac software and secondly install XQuartz.app (XQuartz, <https://xquartz.macosforge.org/>) if you do not have X11.app because installation of lots of R packages requires XQuartz to be installed. If you update your macOS version, you should re-install R (and perhaps XQuartz). For more information, see <https://cran.r-project.org/doc/manuals/r-release/R-admin.html>.

To use the package `ggplot2`, type the following after the command prompt (`>`):

```
library(ggplot2)
```

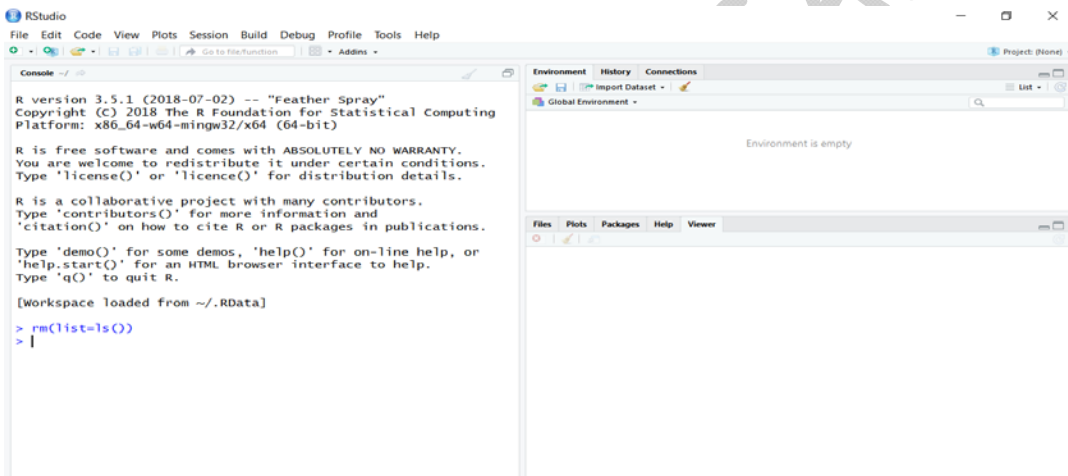
If you have installation problems, see the guide to installation and administration for R:

<https://cran.r-project.org/doc/manuals/r-release/R-admin.html>

2. INTRODUCTION TO RSTUDIO

RStudio (<https://www.rstudio.com>) is a popular **IDE** (Integrated Development Environment) for R programming. RStudio makes R easier to use. It is a powerful editor for R coding and debugging. It is also a powerful generator for HTML, PDF, dynamic documents and slide shows. Download and install it from <https://www.rstudio.com/products/rstudio/download/>. Again, choose the appropriate installer depending on your operating system.

RStudio includes a code editor, debugging & visualization tools. When you first start up RStudio, it will likely look something like the following figure.



The area on the left is an interactive console which is just the R console.

The two panes on the right become useful as you create and run your code.

- The “Environment” tab shows your R Environment – what objects are loaded into your session at the moment. If you’re new to programming, don’t worry; this will make more sense once you start coding.
- The “History” tab shows your command history. So if you typed `3+5` into the console, if you go to the History tab, you should see that in the history tab. You can select one or more lines in the history tab and then click the “To Console” button at the top of the pane to send the line(s) back to the console, or the “To Source” option to send them to the top-left script pane. You can search the history pane as well (you should see a search box at the upper right).

The lower right pane has several different, useful tabs.

- The “Files” tab, similar to Windows File Explorer or Mac Finder. Although not quite as robust as those, this area is convenient for quickly renaming or deleting files, opening files, or changing your working directory.
- The “Plots” tab is where you can view graphs and other data visualizations you create in R.
- The “Packages” tab shows what packages are 1) available for you to use and 2) actually loaded into your working session. Anything listed is on your system; anything with a check mark to the left of the name is currently loaded in memory.
- The “Help” tab is where you can view help files for functions and packages. You’ll likely be using that a lot, no matter how expert you become in R. If you click the home button on the help tab (it’s the house icon), you’ll see links to a lot of general R and RStudio information – some of it for beginners, some considerably more advanced. But you can also ask for more specific help in the R console for functions and packages, and search for help by keyword.
- The “Viewer” pane can be used to view local web content.

RStudio is what’s known as an IDE, or Integrated Development Environment. One common feature of IDEs is projects. In RStudio, opening a project automatically sets you up in the project’s working directory, making it easier to find files stored in that directory. You can create a new project by going to File > New Project. You can create a new project by going to File > New Project. You’ll be given three choices: New Directory (for a brand new project with nothing in it), Existing Project (to create a project from an existing directory that might already have files in it), and Version Control.

3. USING R AS A CALCULATOR

Start R by clicking on the R icon that is on your desktop or in “Programs”. Try the following and see what happens.

Remark. R uses a semi colon as a separator between commands.

```
>2+3 [enter]
```

```
[1] 5
```

Try out $3*2$, $3**2$, 3^2 , $\log(3)$, $\exp(3)$ as well.

```
> help(factorial)
```

(This asks for information about factorials.)

(You should get a screen of information.)

```
> ?factorial
```

(This is shorter way of asking for help.)

```
> example(factorial)
```

(This will give some examples)

```
>factorial(5)
```

(we know $5!=120$)

```
[1] 120
```

```
> pi
```

(pi is a built-in constant in R)

```
[1] 3.141593
```

```
> sin(pi)
```

```
[1] 1.224606e-16
```

(R has functions like sin, cos, etc. Here $\sin(\pi)$ is zero, but doesn't look like it. Too bad.)

```
> round(sin(pi),4)
```

(Round $\sin(\pi)$ to four decimal places. Now it is 0.)

```
[1] 0
```

Complex numbers are a bit strange. For example, one types “ $2+1i$ ” or “ $3-2i$ ” rather than the expected “ $2+i$ ” or “ $3-2*i$.”

```
>sqrt(2-3i)
```

```
[1] 1.674149-0.895977i
```

Write several expressions on one line using semicolons. It saves space.

```
>a=2; b=7; c=5;
```

(We have assigned values to a,b,c.)

```
>q( )
```

(Use this or simply close the window to quit.)

4. VECTORS

```
x <- c(2,2,3,4)
```

(This enters a list of numbers into a vector called x. The “c” means concatenate and it is now usually replaced by “=”.)

```
> x
```

(This asks for a list of the components of x.)

```
[1] 2 2 3 4
```

```
> x = c(2,2,3,4)
```

(Using “=” is an easier way to enter the vector x.)

```
> x*x
```

(Componentwise multiplication of entries in x.)

```
[1] 4 4 9 16
```

```
> ls()
```

(This asks for a list of all named items.)

```
[1] "x"
```

```
> rm(x)
```

(“rm” means “remove.” This deletes the vector x.)

```
> x
```

```
Error: object "x" not found
```

(x really is deleted.)

```
> rm(list=ls(all=TRUE));
```

(This deletes all variables in the current workspace.)

```
> x=c(2,2,3,4)
```

(Start over.)

```
> x[1]
```

(This asks for the first component of vector x.)

```
[1] 2
```

```
> x=2:7
```

(This is a way of entering consecutive integers in x.)

```
> x
```

```
[1] 2 3 4 5 6 7
```

```
> x=seq(from=1,to=100,by=2)
```

or x=seq(1,100,2)

(This enters numbers 1,3,5,...,99 in x. “seq” means “sequence.”)

R also has commands like mean(x), max(x), min(x), or rev(x) to reverse the order of the vector, or rep(x, 3) which repeats the elements of x three times and thus makes a longer vector.

```
> x=c(2,4,7,5,4)
```

```
> sort(x)
```

```
[1] 2 4 4 5 7
```

```
> sort(x, decreasing=TRUE)
```

```
[1] 7 5 4 4 2
```

```
> sort(x,d=T)
```

```
[1] 7 5 4 4 2
```

```
> y=rev(x)
```

(This reverses the elements of x.)

```
> y
```

```
[1] 4 5 7 4 2
```

Here are a couple of built in vectors in R.

```
> letters
```

```
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
```

```
[20] "t" "u" "v" "w" "x" "y" "z"
```

```
> LETTERS
```

```
[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
```

```
[20] "T" "U" "V" "W" "X" "Y" "Z"
```

```
> letters[1:10]
```

(This asks for the first 10 letters.)

```
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
```

5. MATRICES

This will help you to do linear algebra problems☺

```
> y=matrix(c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),4,4);
      #(This gives a 4 by 4 matrix with entries 1:16)
      #(y=matrix(1:16,4,4) is faster. )
> y
      [,1] [,2] [,3] [,4]
[1,]  1   5   9  13
[2,]  2   6  10  14
[3,]  3   7  11  15
[4,]  4   8  12  16
```

We see that R always **puts the entries in columns**. That is unfortunate, but get used to it. We could use “`y=matrix(x,4,4,byrow=TRUE)`” to put entries in rows. Suppose we really want the transpose of `y`. That’s easy.

```
> w=t(y); # “t” means transpose
> w
      [,1] [,2] [,3] [,4]
[1,]  1   2   3   4
[2,]  5   6   7   8
[3,]  9  10  11  12
[4,] 13  14  15  16
```

Now if we want to construct the matrix

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 1 & 0 & 6 \end{bmatrix}$$

in R, we can do:

```
> A = matrix(c(1,0,1,2,4,0,3,5,6), nrow=3, ncol=3);
> A
      [,1] [,2] [,3]
[1,]  1   2   3
[2,]  0   4   5
[3,]  1   0   6
```

```
> A[2, 2]                                     #This gives the (2,2) element
[1] 4
```

```
> A[, 3]                                       #This gives the third column
[1] 3 5 6
```

```
> A[1, ]                                       #This gives the 1st row
[1] 1 2 3
```

Now concatenate our matrix A with itself. Just for fun.

```
> matrix(c(A,A), 3, 6);
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    2    3    1    2    3
[2,]    0    4    5    0    4    5
[3,]    1    0    6    1    0    6
```

To solve for x in the linear system $Ax=b$ (where A is a matrix and b is a vector or a matrix) and x is a matrix or a vector), use

```
> solve(A,b);
```

If the vector b is not specified, then it is taken to be an identity matrix by default, so `solve(A)` gives the inverse of A .

```
> B = solve(A);      B;                                     #This gives the inverse of A
      [,1]      [,2]      [,3]
[1,] 1.0909091 -0.54545455 -0.09090909
[2,] 0.2272727 0.13636364 -0.22727273
[3,] -0.1818182 0.09090909 0.18181818
```

```
> C = A%*%B;      C;                                       # “%*%” is matrix multiplication
      [,1]      [,2]      [,3]
[1,] 1.000000e+00 2.775558e-17 5.551115e-17
[2,] 5.551115e-17 1.000000e+00 -5.551115e-17
[3,] -1.110223e-16 5.551115e-17 1.000000e+00
```

```
> round(C, 4);
```

```
  [,1] [,2] [,3]
```

```
[1,]  1  0  0
```

```
[2,]  0  1  0
```

```
[3,]  0  0  1
```

#You can see B is the inverse of A

```
>D = matrix(c(2,2,3,4), 4,1); D;
```

#make the vector a 4×1matrix

```
  [,1]
```

```
[1,]  2
```

```
[2,]  2
```

```
[3,]  3
```

```
[4,]  4
```

```
>D%*%t(D);
```

#matrix D multiplied by its transpose

```
  [,1] [,2] [,3] [,4]
```

```
[1,]  4  4  6  8
```

```
[2,]  4  4  6  8
```

```
[3,]  6  6  9 12
```

```
[4,]  8  8 12 16
```

```
>A*A;
```

```
  [,1] [,2] [,3]
```

```
[1,]  1  4  9
```

```
[2,]  0 16 25
```

```
[3,]  1  0 36
```

“*” is doing component multiplication

```
>diag(4);
```

#This gives a 4 dimension identity matrix(easy and quick)

```
  [,1] [,2] [,3] [,4]
```

```
[1,]  1  0  0  0
```

```
[2,]  0  1  0  0
```

```
[3,]  0  0  1  0
```

```
[4,]  0  0  0  1
```

Other matrix commands to try are `det(A)`, `eigen(A)`.

6. INPUT DATA INTO R

Unlike SAS, which has DATA and PROC steps, R has data structures (vectors, matrices, arrays, data frames) that you can operate on through functions that perform statistical analyses and create graphs. In this way, R is similar to PROC IML in SAS.

As seen in section 3, we can enter the univariate-data into R in the form of **vectors**. For example,

```
> x1 = c(1,2,5.3,6,-2,4);      # numeric vector
> x2 = c("one","two","three"); # character vector
> x3= c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE); #logical vector
```

Another example: `x=c(10,20,30,40,50);`

Or we can try

```
x=c(1:5); x=10*x;
```

For multivariate data, we can enter the data into R in the form of **matrices** as seen in section 4. For example, the following is for inputting 3 observations to three variables X_1 , X_2 , and X_3 .

```
> data = matrix(c(1,0,1,2,4,0,3,5,6), nrow=3, ncol=3);
```

We also can try

```
> x1 = c(1,0,1);
> x2 = c(2,4,0);
> x3 = c(3,5,6);
> data= cbind(x1,x2,x3);      # Create a matrix by binding
columns
```

We also can use

```
> r1=c(1,2,3);
> r2=c(0,4,5);
> r3=c(1,0,6);
> data= rbind(r1,r2,r3);      # Create a matrix by binding rows
```

We then label the names of the variables and the number of observations of the data.

```
> rownames(data) <- c('1', '2', '3');
> colnames(data) <- c('x1', 'x2', 'x3');
> data;
```

Another data type in R is **arrays**. Arrays are similar to matrices but can have more than two dimensions. See `help(array)` for details.

One limitation of matrices is that all columns and rows must be of the same type. Commonly in our dataset we may have variables of different types, e.g., some categorical variables (e.g. sex), some characters (e.g., name), and some numerical measurements (e.g., height and weight). Data-frames allow you to have variables of different type within one matrix-type array. The following example creates a **data frame**.

```
x=as.factor(c("low", "high", "medium", "low")); #encode a vector as a factor
y=c(1.1, 2.4, 3.1, 0.5);
myData<-data.frame(treatment=x, outcome=y); #combine and label the
variables
str(myData);          # look at the structure of myData
myData;
```

Indexing can be used in data frames in the same way as in matrices. Additionally, you can access individual columns using the **dollar sign** and the variable name. The following example illustrates this.

```
myData[1,];
myData[c(1,2),];
myData[,1];
myData$treatment;
myData$outcome;
```

We have seen that a data frame is created implicitly by the function `read.table`; it is also possible to create a data frame with the function `data.frame`. The vectors so included in the data frame must be of the same length, or if one of the them is shorter, it is “recycled” a whole number of times.

```
> x = 1:4; n = 10; M = c(10, 35); y = 2:4;
> data.frame(x, n);
  x  n
1 1 10
2 2 10
3 3 10
4 4 10

> data.frame(x, M);
```

```

x  M
1 1 10
2 2 35
3 3 10
4 4 35

```

```

> data.frame(x, y);
Error in data.frame(x, y) :
  arguments imply differing number of rows: 4, 3

```

If a data set is available in the format of text file, Excel file or other formats, we may prefer to import the data into R.

Suppose we have a **space delimited** Text file (grades.txt) which looks like

Name	Exam1	Exam2	Exam3	Letter
john	23	46	35	F
mary	42	31	36	F
sam	58	22	55	F
oksana	81	88	79	P
tom	11	19	18	F
peter	55	64	69	P
larisa	81	78	52	P

We need to enter R and give the command

```

grades =
read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/grades.txt",
header=TRUE); # note the / instead of \ on mswindows systems;
grades;

```

The “header=TRUE” part means that there is a header (Name Exam1 Exam2 Exam3) in the file. So the first line should be treated as a header.

Then grades will contain exactly the table of grades. R treats the contents of the file like an object so R is called an object oriented language.


```
> grades[,1]                                     #(This asks for the first column.)
[1] john   mary   sam    oksana tom    peter larisa
Levels: john larisa mary oksana peter sam tom
```

```
> grades[1,]                                     #(This asks for the first row.)
  Name Exam1 Exam2 Exam3 Letter
1 john    23    46    35      F
```

If the data set is a **comma delimited** Text file (grades.csv), we use

```
grades =
read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/grades.csv",
header=TRUE, sep=",");
grades;
```

Or we can use

```
grades =
read.csv("E:/ESUTeaching/2019Spring/Math402/RHandout/grades.csv",
header=TRUE, sep=",");
grades;
```

One of the best ways to read an Excel file is to export it to a space delimited or comma delimited file and import it using the methods above. Alternatively we can use the [xlsx](#) package to access Excel files. The first row should contain variable/column names.

For Stata and Systat data, use the [foreign](#) package. For SPSS and SAS, it is recommended to use the [Hmisc](#) package for ease and functionality. See the Quick-R section on [packages](#), for information on obtaining and installing these packages.

7. LISTING AND REMOVING OBJECTS FROM THE ENVIRONMENT

We can list the objects available in the working environment using the functions `ls()` or `objects()`. The function `rm()` can be used to remove an object (or a list of objects) from the environment. We can quit R by closing the console or by typing `quit()`. Use `quit(save='yes')`, `quit(save='no')`, to quit saving or without saving the environment, respectively. The following example illustrate these functions.

```
> x=seq(from=1,to=100,by=2);
> y=2;

> ls();                                #list the objects in the environment
[1] "x" "y"

> rm(list=ls());                       # cleans the environment
> ls()
character(0)

> # Now let's create another object
> x2<-1:10;
> ls();
[1] "x2"

> quit(save='yes');

# Now open R and type ls()
```

8. READING AND WRITING ASCII DATA

The functions `read.table()` and `write.table()` can be used to read and write data in table-format. In the following example we first load a data frame (`oats`) available in the MASS package (<https://cran.r-project.org/web/packages/MASS/index.html>) and then write it to the hard drive as an ASCII file and read the data again back into the R session.

```
> rm(list=ls());
> library(MASS);      #loads the MASS package
> data(oats);
> str(oats);          # shows the structure of an object
'data.frame':   72 obs. of  4 variables:
 $ B: Factor w/ 6 levels "I","II","III",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ V: Factor w/ 3 levels "Golden.rain",...: 3 3 3 3 1 1 1 1 2 2 ...
 $ N: Factor w/ 4 levels "0.0cwt","0.2cwt",...: 1 2 3 4 1 2 3 4 1 2 ...
 $ Y: int   111 130 157 174 117 114 161 141 105 140 ...

> fix(oats);          # displays data (can also modify/edit/create variables)
> # writing data to hard drive
>
write.table(x=oats,file='E:/ESUTeaching/2019Spring/Math402/RHandout/oats.txt', sep=' '); # space-delimited
>
write.table(x=oats,file='E:/ESUTeaching/2019Spring/Math402/RHandout/oats.csv', sep=','); # comma-delimited

> # reading data
>
myData1=read.table('E:/ESUTeaching/2019Spring/Math402/RHandout/oats.txt', sep=' ',header=TRUE);

>
myData2=read.table('E:/ESUTeaching/2019Spring/Math402/RHandout/oats.csv', sep=',',header=TRUE);

> #returns the first parts of the data
> head(myData1);
> head(myData2);
```

9. UNIVARIATE DESCRIPTIVE STATISTICS

The summary function will produce some descriptive statistics. The output produced by summary() depends on the nature of the object. For discrete variables, the table function is useful (\$ operator references a certain variable in a data frame).

For the grades data in the last section,

```
> grades =
read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/grades.txt",
header=TRUE);

> table(grades$Letter);           # look at counts for P and F

F P
4 3

> summary(grades)
  Name      Exam1      Exam2      Exam3      Letter
john   :1  Min.   :11.00  Min.   :19.00  Min.   :18.00  F:4
larisa:1 1st Qu.:32.50  1st Qu.:26.50  1st Qu.:35.50  P:3
mary   :1  Median :55.00  Median :46.00  Median :52.00
oksana:1  Mean    :50.14  Mean    :49.71  Mean    :49.14
peter  :1 3rd Qu.:69.50  3rd Qu.:71.00  3rd Qu.:62.00
sam    :1  Max.    :81.00  Max.    :88.00  Max.    :79.00
tom    :1
```

For **numerical** variables, we will use the mean, standard deviation (sd), variance (var), quantile, and histogram functions. For example, consider the variable Exam1 in the data grades.

```
> x=grades$Exam1;
> x;
[1] 23 42 58 81 11 55 81

> mean(x);           # This finds the mean of variable x.
[1] 50.14286

> sd(x);             #This gives sample standard deviation.
[1] 26.8479

> var(x);            #This gives sample variance.
[1] 720.8095
```

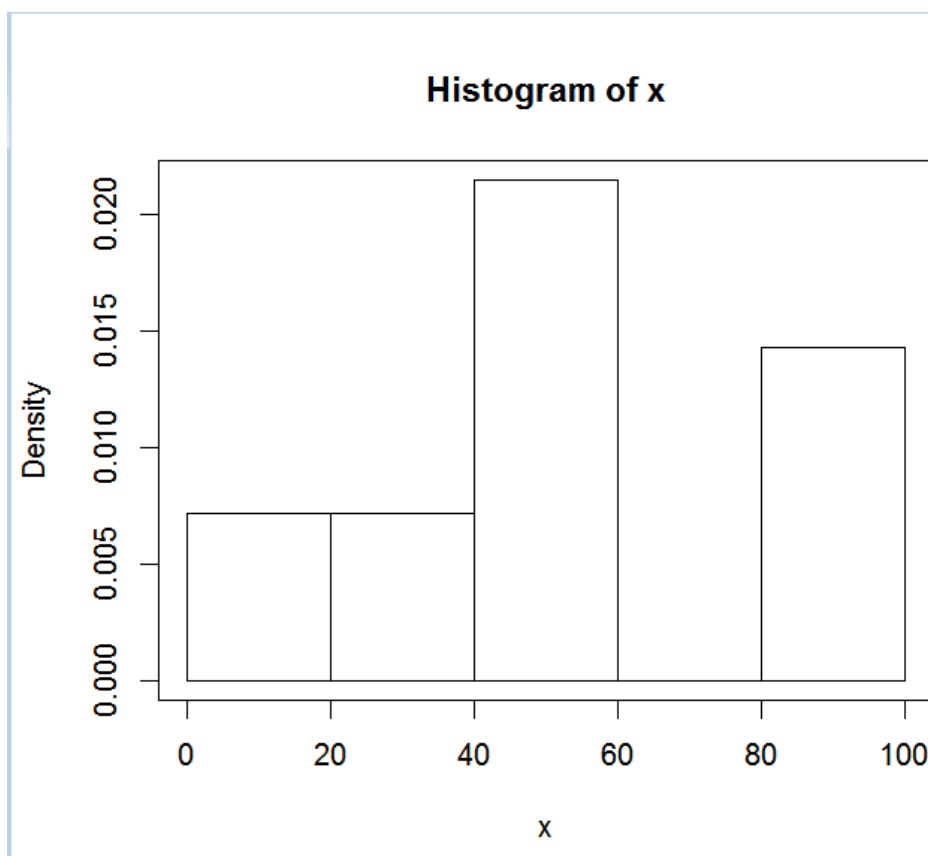
```

> quantile(x);           #This gives quantiles of the variable
  0%   25%   50%   75%  100%
11.0  32.5  55.0  69.5  81.0

> summary(x);           #This gives max, min, quartiles, median, mean.
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 11.00  32.50   55.00   50.14  69.50   81.00

> hist(x, plot=TRUE, freq=FALSE, right=FALSE); box();

```



```

>n=length(x);
>sum(x)/n;           #compare this with mean(x);
[1] 50.14286

>xbar = mean(x);
>varx=sum( (x- xbar)^2 )/( n-1);   #compare this with var(x);
>varx
[1] 720.8095

>sqrt(varx);        #compare this with sd(x);
[1] 26.8479

```

10. BIVARIATE DESCRIPTIVE STATISTICS

In the previous example we described features of the distribution of a random variable. Now we turn into description of the bi-variate distribution of two random variables. First we look at an example of two discrete random variables from the oats dataset in the MASS package

```
> rm(list=ls());
> library(MASS);      #loads the MASS package
> data(oats);

> # Contingency tables for two discrete random variables
> table(oats$V,oats$B); # table for bivariate discrete stats
```

	I	II	III	IV	V	VI
Golden.rain	4	4	4	4	4	4
Marvellous	4	4	4	4	4	4
Victory	4	4	4	4	4	4

```
> xtabs(formula=~oats$V + oats$B) # Create a contingency table
```

	oats\$B					
oats\$V	I	II	III	IV	V	VI
Golden.rain	4	4	4	4	4	4
Marvellous	4	4	4	4	4	4
Victory	4	4	4	4	4	4

```
> highYield = oats$Y>median(oats$Y); # binary high yield
```

```
> highYield
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE FALSE FALSE FALSE FALSE FALSE
[18] TRUE TRUE TRUE FALSE TRUE TRUE TRUE FALSE FALSE TRUE
FALSE FALSE FALSE FALSE FALSE FALSE TRUE
[35] TRUE TRUE FALSE FALSE FALSE TRUE FALSE TRUE TRUE TRUE
FALSE FALSE TRUE TRUE FALSE FALSE FALSE
[52] TRUE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE
FALSE TRUE TRUE FALSE FALSE FALSE TRUE
[69] FALSE FALSE TRUE TRUE
```

```
> table(oats$V,highYield); # counts of high-yielding plots by variety
```

	highYield	
	FALSE	TRUE
Golden.rain	12	12
Marvellous	9	15
Victory	15	9

```
> xtabs(formula=~oats$V+highYield);
```

	highYield	
oats\$V	FALSE	TRUE
Golden.rain	12	12
Marvellous	9	15
Victory	15	9

Now we describe the association between two continuous random variables (Gas=gas consumption, and Temp=Temperature, of the whiteside dataset, also available with the MASS package) using the covariance, correlation and plot functions.

```
> # Two continuous random variables #####
```

```
> library(MASS);
```

```
> data(whiteside);
```

```
> str(whiteside);
```

```
'data.frame': 56 obs. of 3 variables:
```

```
$ Insul: Factor w/ 2 levels "Before","After": 1 1 1 1 1 1 1 1 1 1 1 ...
```

```
$ Temp : num -0.8 -0.7 0.4 2.5 2.9 3.2 3.6 3.9 4.2 4.3 ...
```

```
$ Gas : num 7.2 6.9 6.4 6 5.8 5.8 5.6 4.7 5.8 5.2 ...
```

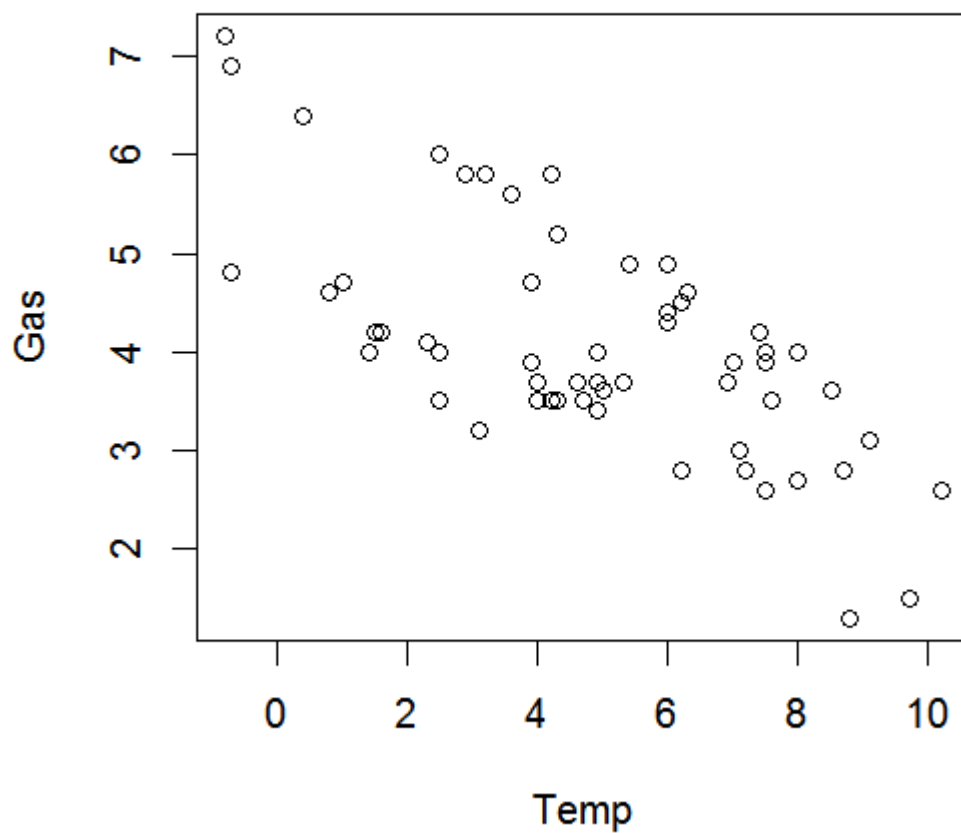
```
> head(whiteside);
```

	Insul	Temp	Gas
1	Before	-0.8	7.2
2	Before	-0.7	6.9
3	Before	0.4	6.4
4	Before	2.5	6.0
5	Before	2.9	5.8
6	Before	3.2	5.8

```
> # variance-covariance matrix
> var(whiteside[,2:3]);
      Temp      Gas
Temp  7.560091 -2.194000
Gas   -2.194000  1.363896

> # correlation matrix
> cor(whiteside[,2:3]);
      Temp      Gas
Temp  1.0000000 -0.6832545
Gas   -0.6832545  1.0000000

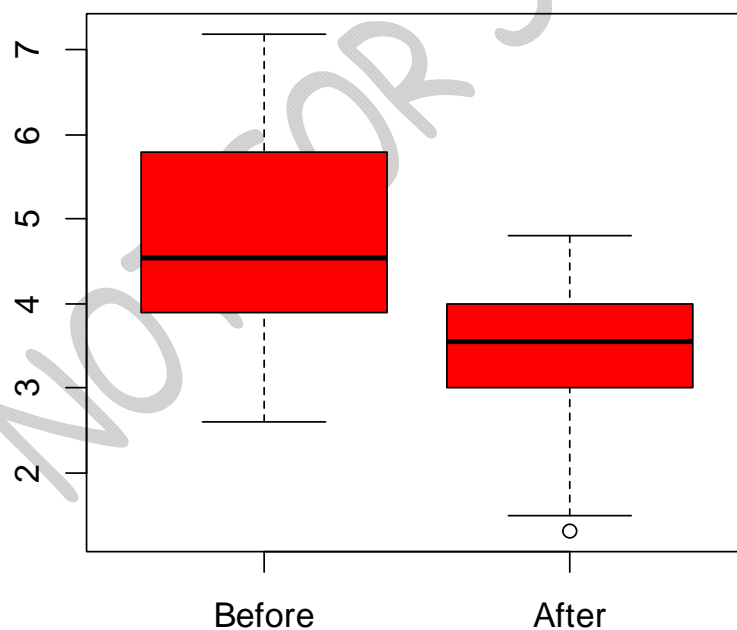
> # scatter plot for visualization
> plot(Gas~Temp,data=whiteside);
```



Now, let's look at an example of one continuous (Gas) and one discrete random variable (Ins=Insulation) also from the whiteside dataset. In this case we describe the joint distribution by first calculating the **conditional mean** gas consumption given insulation, and using a **box-plot**, which provides quantiles of a continuous random variable by level of the discrete random variable.

```
> # One continuous versus one discrete random variable
> # Conditional mean
> tapply(FUN=mean, X=whiteside$Gas, INDEX=whiteside$Insul);
  Before      After 
4.750000  3.483333

> # boxplot, which displays several quantiles
> boxplot(Gas~Insul,col='red',data=whiteside);
```



11. PLOTS

R is widely known for its ability to produce high quality, customized graphics. We have seen histograms, box plots and scatter plots in the last two sections. In what follows, several R plots typically used in preliminary data analysis are illustrated by examples. The detailed grammar of each function is copied from R help.

(1) Stem-and-Leaf Plots

`stem` produces a stem-and-leaf plot of the values in `x`. The parameter `scale` can be used to expand the scale of the plot. A value of `scale=2` will cause the plot to be roughly twice as long as the default.

Usage

```
stem(x, scale = 1, width = 80, atom = 1e-08)
```

Arguments

`x` a numeric vector.
`scale` This controls the plot length.
`width` The desired width of plot.
`atom` a tolerance.

Examples

```
>x = c(90, 70, 70, 70, 75, 70, 65, 68, 60,74, 70, 95, 75, 70, 68,
65, 40, 65);
>stem(x);
```

The decimal point is 1 digit(s) to the right of the |

```
4 | 0
6 | 055588000000455
8 | 05
```

```
> stem(x, scale=1);
```

The decimal point is 1 digit(s) to the right of the |

```
4 | 0
5 |
6 | 055588
7 | 000000455
8 |
9 | 05
```

(2) Histogram Plots

`hist` produces a (relative) frequencies of the values in `x`.

Usage

```
hist(x, breaks = "Sturges",
     freq = NULL, probability = !freq,
     include.lowest = TRUE, right = TRUE,
     density = NULL, angle = 45, col = NULL, border = NULL,
     main = paste("Histogram of", xname),
     xlim = range(breaks), ylim = NULL,
     xlab = xname, ylab,
     axes = TRUE, plot = TRUE, labels = FALSE,
     nclass = NULL, ...)
```

Arguments

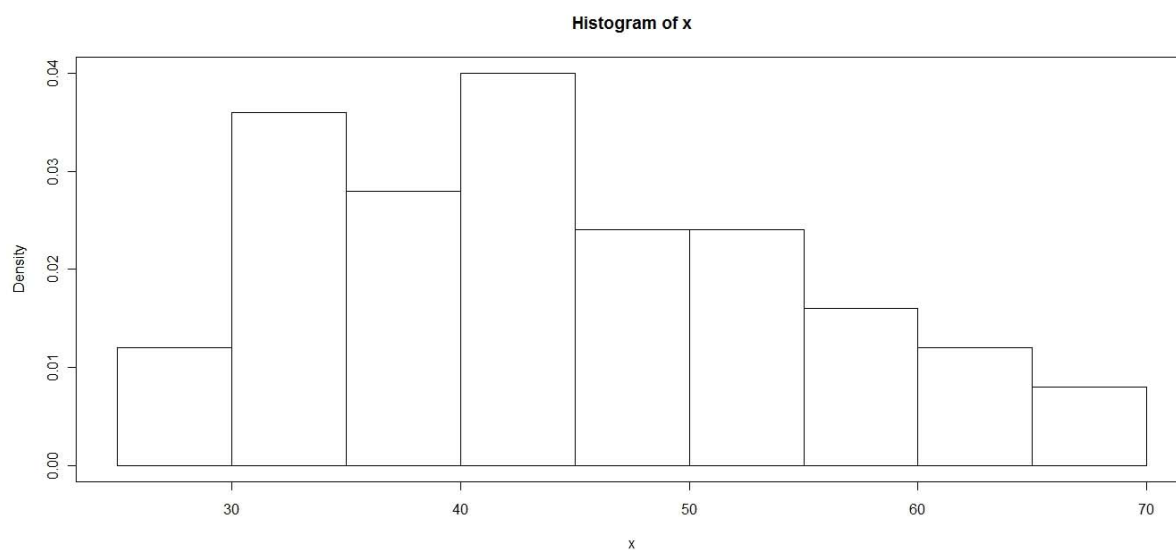
`freq` logical; if `TRUE`, the histogram graphic is a representation of frequencies, the counts component of the result; if `FALSE`, probability densities, component `density`, are plotted (so that the histogram has a total area of one). Defaults to `TRUE` *if and only if* `breaks` are equidistant (and `probability` is not specified).

`plot` logical. If `TRUE` (default), a histogram is plotted, otherwise a list of breaks and counts is returned. In the latter case, a warning is used if (typically graphical) arguments are specified that only apply to the `plot = TRUE` case.

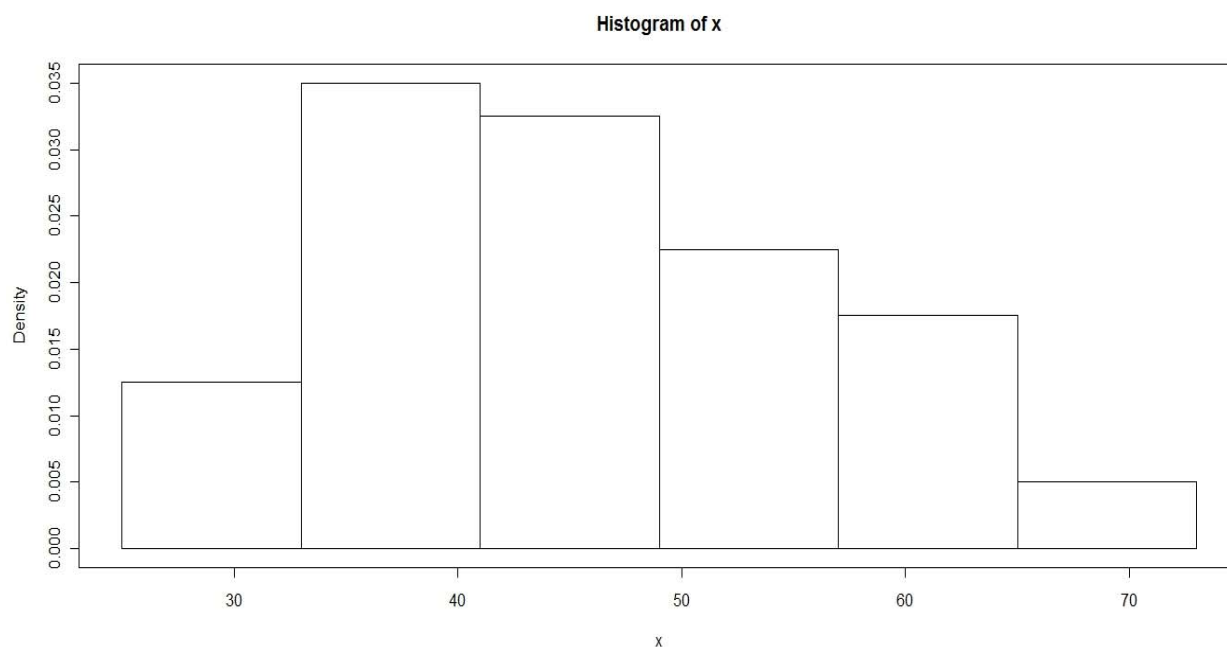
Example

```
>x=c(34, 48, 70, 63, 52, 52, 35, 50, 37, 43, 53, 43, 52, 44, 42,
31, 36, 48, 43, 26, 58, 62, 49, 34, 48, 53, 39, 45, 34, 59, 34, 66,
40, 59, 36, 41, 35, 36, 62, 34, 38, 28, 43, 50, 30, 43, 32, 44, 58,
53);

> hist(x, plot=TRUE, freq=FALSE, right=FALSE);
box();
```



```
> hist(x, plot=TRUE, freq=FALSE, breaks=c(25, 33, 41, 49,
57, 65, 73), right=FALSE); box();
```



(3) Dot Plots

dotchart produces a dot plot of the values in x.

Usage

```
dotchart(x, labels = NULL, groups = NULL, gdata = NULL,
  cex = par("cex"), pch = 21, gpch = 21, bg = par("bg"),
  color = par("fg"), gcolor = par("fg"), lcolor = "gray",
  xlim = range(x[is.finite(x)]),
```

```
main = NULL, xlab = NULL, ylab = NULL, ...)
```

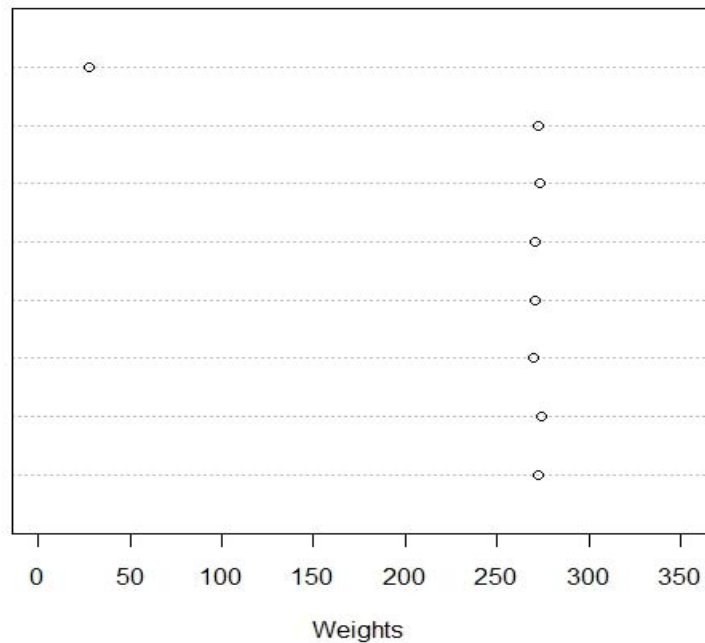
Arguments

`xlim` horizontal range for the plot

`xlab` Label of the x-axis

Example

```
>x = c(272, 274, 270, 271, 271, 273, 272, 27.2);
>dotchart(x,xlab="Weights",xlim=c(0,350));
```



Remark: for pie chart and bar chart, please see `pie()` and `barplot()`.

(4) Lines/Points Plots**Usage**

```
plot(x, y, ...)
```

Arguments

`x` the coordinates of points in the plot. Alternatively, a single plotting structure, function or *any R object with a plot method* can be provided.

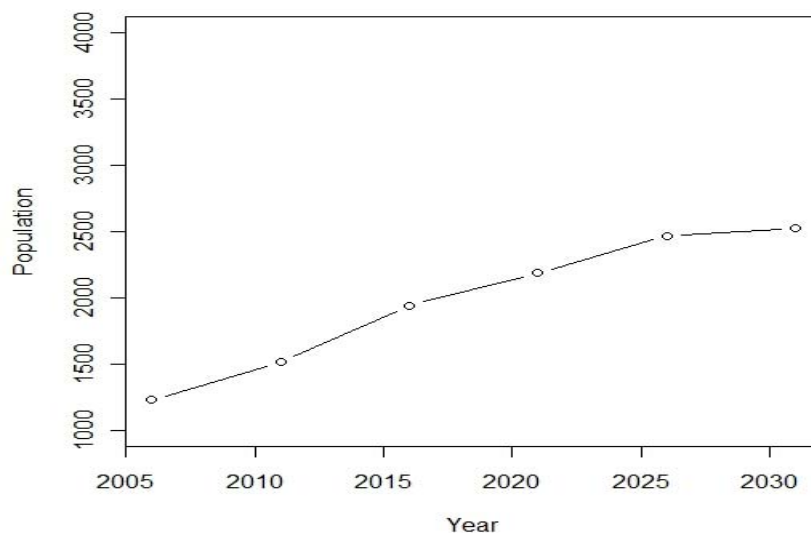
`y` the y coordinates of points in the plot, *optional* if `x` is an appropriate structure.

`...` Arguments to be passed to methods, such as graphical parameters (see [par](#)).

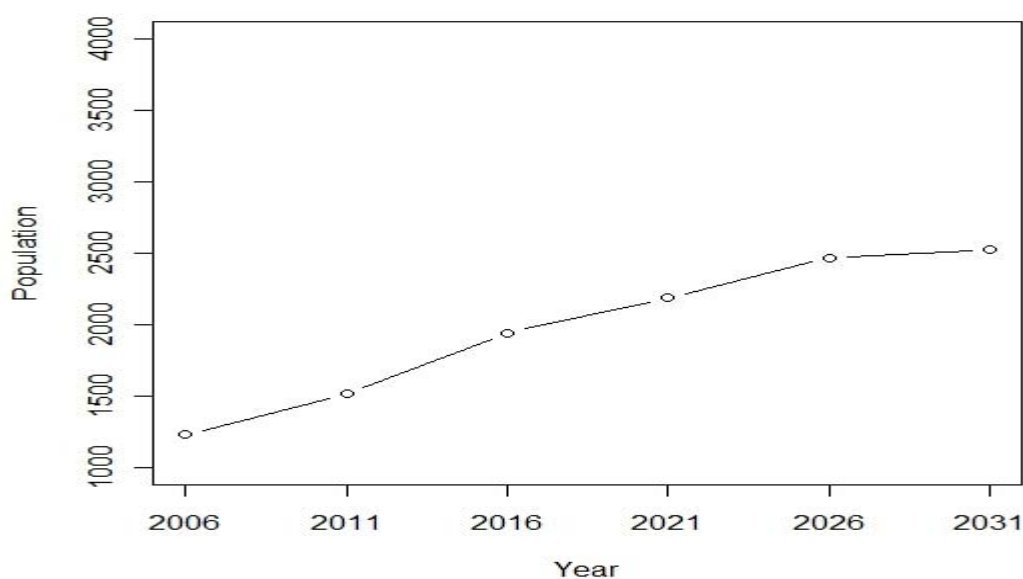
Examples

```
>y = c(1227.3, 1513.1, 1942.1, 2184.7, 2466.6, 2527.6);
>x = c(2006, 2011, 2016, 2021, 2026, 2031);
```

```
> plot(x,y, type="b", xlab="Year", ylab="Population", xlim=c(2006,
2031), ylim=c(1000, 4000));
```

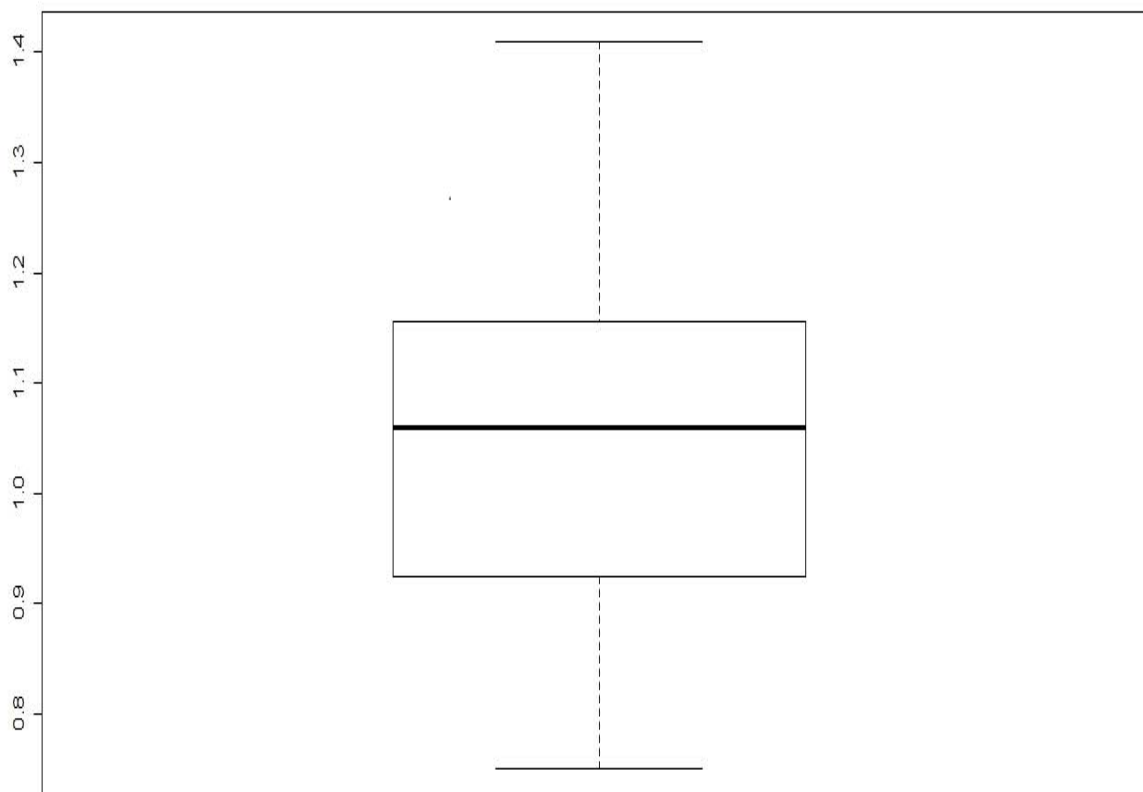


```
> plot(x,y, type="b", xlab="Year", ylab="Population", xlim=c(2006,
2031), ylim=c(1000, 4000), axes=FALSE);
> axis(1, at= seq(from = 2006, to = 2031, by=5),
labels=c("2006","2011","2016","2021","2026","2031"));
> axis(2, at= seq(from = 1000, to = 4000, by=500), labels=c("1000",
"1500", "2000", "2500", "3000", "3500" , "4000"));
> box();
```



(5) Box Plots (five-number summary)**Usage**`boxplot(x, ...)`**Example**

```
> x=c(0.75, 0.93, 1.08, 1.18, 0.83, 0.96, 1.08, 1.18, 0.87, 0.96,  
1.12, 1.24, 0.89, 0.97, 1.12, 1.28, 0.89, 0.98, 1.14, 1.38, 0.89,  
0.99, 1.14, 1.41, 0.92, 1.06, 1.17) ;  
> summary(x) ; # (This gives  
max, min, quartiles, median, mean.)  
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.     
 0.750  0.925   1.060   1.052   1.155   1.410    
>boxplot(x);
```



12. SOME DISCRETE AND CONTINUOUS DISTRIBUTIONS

In this section, we list three discrete distributions: Binomial, Poisson and Hyper-geometric distributions and four typical continuous distributions: Normal, T, Chi-square and F distributions. For each distribution, there are four available functions:

(1) d-: probability mass function of a discrete random variable or probability density function of a continuous random variable.

(2) p-: cumulative distribution function of a random variable (denoted by X) which is the probability of that X is less than or equal to a specified value x_0 , denoted by $F(x_0) = P(X \leq x_0)$. It is often called “Lower tail” probability. Thus the “Upper tail” probability is $1 - F(x_0) = P(X > x_0)$.

(3) q-: quantile function of a random variable. A **quantile** is the inverse of $F(x_0)$. That is, if $P(X \leq x_0) = p$ is known, then x_0 is the required quantile corresponding to the cumulative (Lower tail) probability p .

(4) r-: random generation function of a random variable. It is used to generate random numbers (a random sample) from a population with the specified distribution.

Remark. A **critical value** is a value of a random variable X separating unlikely values from those that are likely to occur. The notation X_α denotes that the area to its right is α . That is, the “Upper tail” probability is α : $P(X \geq X_\alpha) = \alpha$.

Examples. Let $Y \sim \text{Normal}(\mu=0.8, \sigma=1.15)$.

(1) $P(0.3 \leq Y < 1.5) = ?$

```
> pnorm(1.5, mean=0.8, sd=1.15) - pnorm(0.3, mean=0.8, sd=1.15);
[1] 0.3967768
```

(2) $Y_{0.017} = ?$ That is, $P(Y > y_0) = 0.017$, then $y_0 = ?$

```
> qnorm(p=1-0.017, mean = 0.8, sd = 1.15);
[1] 3.238082
```

(3) Generate a random sample of size 10 from this population.

```
> rnorm(n=10, mean = 0.8, sd = 1.15);
[1] -0.10162306  0.21868090  1.04254639 -0.83113610  2.04652208
[6] -1.05015109  0.52182539  0.06790268  0.38681307 -0.48160470
```


Please use R help for more examples!**Binomial Distribution****Description**

Density, distribution function, quantile function and random generation for the binomial distribution with parameters `size` and `prob`.

Usage

```
dbinom(x, size, prob, log = FALSE)
pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
rbinom(n, size, prob)
```

Arguments

`x`, `q` vector of quantiles.
`p` vector of probabilities.
`n` number of observations. If `length(n) > 1`, the length is taken to be the number required.
`size` number of trials (zero or more).
`prob` probability of success on each trial.
`log`, `log.p` logical; if TRUE, probabilities `p` are given as $\log(p)$.
`lower.tail` logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

Poisson Distribution**Description**

Density, distribution function, quantile function and random generation for the Poisson distribution with parameter `lambda`.

Usage

```
dpois(x, lambda, log = FALSE)
ppois(q, lambda, lower.tail = TRUE, log.p = FALSE)
qpois(p, lambda, lower.tail = TRUE, log.p = FALSE)
rpois(n, lambda)
```

Arguments

`x` vector of (non-negative integer) quantiles.
`q` vector of quantiles.
`p` vector of probabilities.
`n` number of random values to return.
`lambda` vector of (non-negative) means.
`log`, `log.p` logical; if TRUE, probabilities `p` are given as $\log(p)$.
`lower.tail` logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

Hypergeometric Distribution**Description**

Density, distribution function, quantile function and random generation for the hypergeometric distribution.

Usage

```
dhyper(x, m, n, k, log = FALSE)
phyper(q, m, n, k, lower.tail = TRUE, log.p = FALSE)
qhyper(p, m, n, k, lower.tail = TRUE, log.p = FALSE)
rhyper(nn, m, n, k)
```

Arguments

`x`, `q` vector of quantiles representing the number of white balls drawn without replacement from an urn which contains both black and white balls.

`m` the number of white balls in the urn.

`n` the number of black balls in the urn.

`k` the number of balls drawn from the urn.

`p` probability, it must be between 0 and 1.

`nn` number of observations. If `length(nn) > 1`, the length is taken to be the number required.

`log`, `log.p` logical; if TRUE, probabilities `p` are given as $\log(p)$.

`lower.tail` logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

Normal distribution**Description**

Density, distribution function, quantile function and random generation for the normal distribution with mean equal to `mean` and standard deviation equal to `sd`.

Usage

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

Arguments

`x`, `q` vector of quantiles.

`p` vector of probabilities.

`n` number of observations. If `length(n) > 1`, the length is taken to be the number required.

`mean` vector of means.

`sd` vector of standard deviations.

`log`, `log.p` logical; if TRUE, probabilities `p` are given as $\log(p)$.

`lower.tail` logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.

t – distribution

Description

Density, distribution function, quantile function and random generation for the t distribution with `df` degrees of freedom (and optional non-centrality parameter `ncp`).

Usage

```
dt(x, df, ncp, log = FALSE)
pt(q, df, ncp, lower.tail = TRUE, log.p = FALSE)
qt(p, df, ncp, lower.tail = TRUE, log.p = FALSE)
rt(n, df, ncp)
```

Arguments

`x, q` vector of quantiles.
`p` vector of probabilities.
`n` number of observations. If `length(n) > 1`, the length is taken to be the number required.
`df` degrees of freedom (> 0 , maybe non-integer). `df = Inf` is allowed.
`ncp` non-centrality parameter *delta*; currently except for `rt()`, only for `abs(ncp) <= 37.62`. If omitted, use the central t distribution.
`log, log.p` logical; if TRUE, probabilities `p` are given as `log(p)`.
`lower.tail` logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

Chi-Square distribution

Description

Density, distribution function, quantile function and random generation for the chi-squared (χ^2) distribution with `df` degrees of freedom and optional non-centrality parameter `ncp`.

Usage

```
dchisq(x, df, ncp=0, log = FALSE)
pchisq(q, df, ncp=0, lower.tail = TRUE, log.p = FALSE)
qchisq(p, df, ncp=0, lower.tail = TRUE, log.p = FALSE)
rchisq(n, df, ncp=0)
```

Arguments

`x, q` vector of quantiles.
`p` vector of probabilities.
`n` number of observations. If `length(n) > 1`, the length is taken to be the number required.
`df` degrees of freedom (non-negative, but can be non-integer).
`ncp` non-centrality parameter (non-negative).

`log`, `log.p` logical; if TRUE, probabilities p are given as $\log(p)$.

`lower.tail` logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

F- distribution

Description

Density, distribution function, quantile function and random generation for the F distribution with `df1` and `df2` degrees of freedom (and optional non-centrality parameter `ncp`).

Usage

```
df(x, df1, df2, ncp, log = FALSE)
pf(q, df1, df2, ncp, lower.tail = TRUE, log.p = FALSE)
qf(p, df1, df2, ncp, lower.tail = TRUE, log.p = FALSE)
rf(n, df1, df2, ncp)
```

Arguments

`x`, `q` vector of quantiles.

`p` vector of probabilities.

`n` number of observations. If `length(n) > 1`, the length is taken to be the number required.

`df1`, `df2` degrees of freedom. Inf is allowed.

`ncp` non-centrality parameter. If omitted the central F is assumed.

`log`, `log.p` logical; if TRUE, probabilities p are given as $\log(p)$.

`lower.tail` logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

13. ASSESSING NORMALITY

Histogram can be used to assess normality visually. If the histogram does not depart dramatically from a bell shape and there is at most one outlier, we can use R to generate a normal quantile plot. The sample is from a normal population distribution if the pattern of the points is reasonably close to a straight line and the points do not show some systematic pattern that is not a straight-line pattern.

1. QQ Plot:

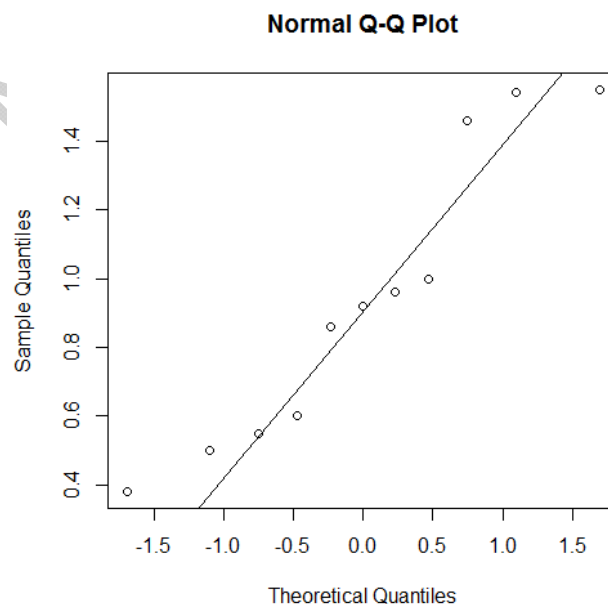
`qqnorm` is a generic function the default method of which produces a normal QQ plot of the values in the data. `qqline` adds a line to a “theoretical”, by default normal, quantile-quantile plot which passes through the probs quantiles, by default the first and third quartiles.

Example. Consider the measured radiation emissions (in W/kg) corresponding to a sample of cell phones. See the data in the table.

0.38	0.55	1.54	1.55	0.50	0.60	0.92	0.96	1.00	0.86	1.46
------	------	------	------	------	------	------	------	------	------	------

```
> x=c(0.38, 0.55, 1.54, 1.55, 0.50, 0.60, 0.92, 0.96, 1.00,
0.86, 1.46);
> qqnorm(x);
> qqline(x);
```

The QQ plot is shown in the figure below.



2. Statistical tests

Sometimes, the conclusions from the histogram and the QQ plot might differ. Then we can conduct formal statistical tests to check normality of the data.

We are testing

H_0 : data are from a normal population versus H_a : data are not from a normal population.

One test method we can use is Shapiro-Wilk test:

```
> shapiro.test(x)
```

```
Shapiro-Wilk normality test
```

```
data: x
```

```
W = 0.90192, p-value = 0.1951
```

In this example, we fail to reject the null hypothesis due to the large p-value.

If we use the nortest package (<https://cran.r-project.org/web/packages/nortest/index.html>), other normality test methods can be used. For example, Anderson-Darling normality test, Cramér-von Mises test for normality, Pearson chi-square test for normality, and Shapiro-Francia test for normality.

14. LOOPS AND CONDITIONAL STATEMENTS

Loops are used to repeat tasks. For example, the `for` loop in R can be used to run a task over a pre-defined index set. Here we have one simple example.

```
for(i in 1:10)
{
  print(i);
}
```

Look at the output:

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

Besides the `for` loop, the other two loops available in R are `while` loop and `repeat` loop where conditional statements are generally involved.

Conditional statement can be used to execute an operation if some variable is equal to `TRUE`. Let's look at an example.

```
> (3>2)
[1] TRUE
> (3<2)
[1] FALSE

> condition1<-c(TRUE,FALSE,TRUE,FALSE);
> condition2<-c(FALSE,TRUE,FALSE,TRUE);
> # AND
> condition1&condition2
[1] FALSE FALSE FALSE FALSE
> # OR
> condition1|condition2
[1] TRUE TRUE TRUE TRUE
```

```
> # IF
  for(i in 1:4)
  {
    if(i<2)
    {
      print(i);
    }
    else
    {
      print(-i);
    }
  }

> # Another example of if ... else
x = 5;
if (x < 0)
{
  print("Negative number");
} else if (x > 0)
{
  print("Positive number");
} else
  print("Zero");

> # ifelse(condition, action if true, action if false)
> ifelse(condition1, 'a', 'b');
[1] "a" "b" "a" "b"
```


15. FUNCTIONS

Most objects in R are functions. A function takes some arguments as input, perform some internal computations and, usually, returns an object. For instance, the function `mean()` takes as argument a numeric vector and returns an integer. You can easily create your own R-functions. This allows you to automate blocks of code that can be later on used as a black-box. The following examples illustrate how to create a very simple function.

EXAMPLE 1. (sum of the first n integers) We begin with the sum a equal to zero and then keep adding i , for i from one to n . Then we print out the final sum a .

```
summ<-function(n)
{
  a=0;
  for(i in 1:n)
  {a=a+i;}
  return(a);
}
```

```
> summ(10)
[1] 55
```

EXAMPLE 2. (power of a vector or matrix)

```
getPower<-function(x ,power)
{
  out<-x^power;
  return(out);
}
```

```
> getPower(x=3,power=2);
[1] 9
> getPower(x=c(1,2,3),power=0);
[1] 1 1 1
```

EXAMPLE 3. (find the 1st and 3rd quartiles of a data set):

There is no universal agreement on the method of calculating quartiles of a sample of size n , and different computer programs often yield different results. In the following, we define a function `MyQuartiles` to calculate Q_1 (with position $0.25 \times (n+1)$ and value obtained by interpolation) and Q_3 (with position $0.75 \times (n+1)$ and value obtained by interpolation).

```
MyQuartiles<-function(x)
{
  x = sort(x);
  n = length(x);
  q1position = 0.25*(n+1);
  q3position = 0.75*(n+1);

  n1 = floor(q1position);          #n1 = trunc(q1position);
  n3 = floor(q3position);          #n3 = trunc(q3position);

  q1 = x[n1] +(x[n1+1]-x[n1])*(q1position-n1);
  q3 = x[n3] +(x[n3+1]-x[n3])*(q3position-n3);
  quartiles = c(q1, q3);

  return(quartiles);
}

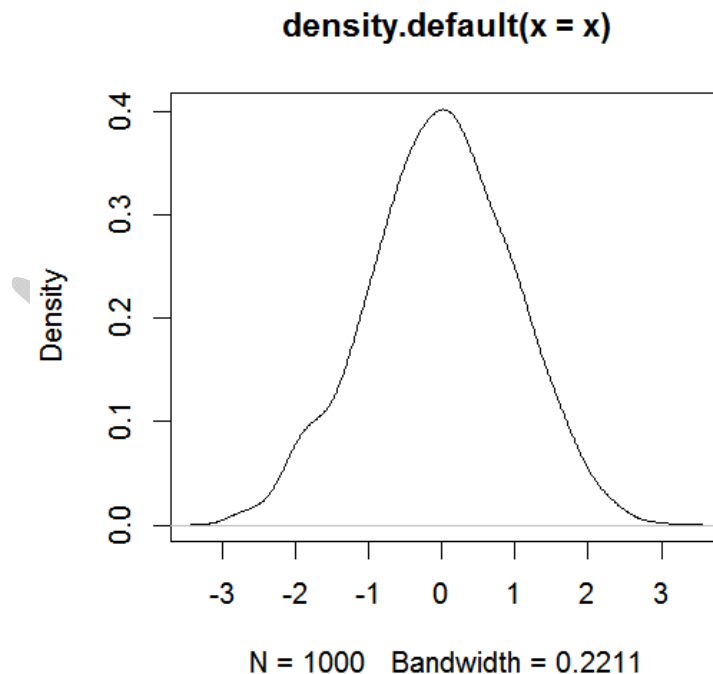
> x=rnorm(n=35, mean = 0, sd = 1);
> MyQuartiles(x)
[1] -0.8205387  0.5680713
```

16. MONTE CARLO SIMULATIONS

Monte Carlo (MC) simulations are commonly used in statistics to estimate features of distributions that may not have closed form. For instance, it can be used to estimate the power of a test statistics whose distribution over repeated sampling is unknown. Some simulation examples related to estimations and hypothesis testing will be illustrated later.

The base package of R offers functions that can be used to obtain “random” draws from several distributions. For each distribution there are usually four functions: one, whose name usually starts with **r** for “random”, which can be used to obtain random draws, one, whose name usually starts with **d** for “density”, that evaluates the density function for a give value of the random variable, one, whose name usually starts with **p** for “probability”, that will give the cumulative distribution function (CDF) at a given quantile and one, whose name usually starts with **q** for “quantile” that gives the quantile corresponding to a value of the CDF. Below we have an example of selecting a random sample of size 1000 from a population which is standard normal.

```
> # Random draws  
> x<-rnorm(n=1000,sd=1,mean=0);  
> plot(density(x));
```



There are many other functions that can be used to draw numbers from other distribution with continuous (e.g., `rgamma`, `rexp`, `runif`) or discrete support (e.g., `rbinom`, `rpoiss`).

The function `sample()` can be used to draw numbers from a bag of labels with or without replacement. For example,

```
> # Random draws
> sample(x=c("a","b","c","d"),size=3,replace=TRUE);
[1] "a" "a" "c"

> sample(x=c("a","b","c","d"),size=3,replace=FALSE);
[1] "c" "d" "a"
```

We use computers to mimic random processes. Although the numbers generated by functions such as `sample()` or `rnorm()` look like random they are indeed deterministic. You can see this by controlling the seed of random number generator. The seed is an integer that controls the sequence of number of generated by the random generator. The following example illustrates this.

```
> set.seed(1295490)
> runif(3)
[1] 0.8562950 0.2772573 0.4433361

> set.seed(1295490)
> runif(3)
[1] 0.8562950 0.2772573 0.4433361

> set.seed(12954)
> runif(3)
[1] 0.9507285 0.8522783 0.2263756
```

Monte Carlo Estimates. Here we have an example of a MC estimate of the mean, standard deviation and 95% quantile of a normal density using numbers randomly generated from a standard normal density (with mean zero and variance equal to one).

```
> n=1000;
> z<-rnorm(n,sd=1,mean=0);

> # estimating the mean
> mean(z);
[1] 0.017429
```

```
> # estimating the sd (should be close to 1)
> sd(z)
[1] 0.9909837

> # estimating the probability of  $z < 1.96$ , should be close to 0.975
> mean(z < 1.96);
[1] 0.974
```

In practice, we do not use MC methods to estimate the mean, standard deviation and quantiles of the standard normal density. These methods are used mostly when the distribution of the random variable is unknown. To illustrate, suppose X is a random variable which is the product of Z_1 and Z_2 , where Z_1 is a standard normal random variable and Z_2 is a random variable having an exponential density with rate parameter equal to 1. The density function of X does not have a closed form. However, we can estimate features of the distribution of X using MC methods. An example is provided below.

```
> m<-100000;
> z1<-rnorm(n=m,sd=1,mean=0);
> z2<-rexp(n=m,rate=1);
> x<-z1*z2;

> # estimating the mean
> mean(x)
[1] -0.005213986

> # estimating the variance
> var(x)
[1] 2.002479

> # estimating the probability of  $z > 1$ 
> mean(z > 1)
[1] 0.168
```

Here we have a more elaborated example.

EXAMPLE 3. (Simulation of sampling distribution: Rolling a balanced die 5 times)

Consider repeating this process: **Roll a balanced die 5 times.** Find the mean, variance, and the probability of *odd* numbers of the results.

Let X be the random variable of the results of rolling the die. Before studying the distribution of X , we conduct a MC simulation of repeating the process (of rolling the die 5 times) 10,000 times.

```
iter = 10000;

n=5;          #sample size
means=numeric();
variances=numeric();
proportions=numeric();

# Define a function which counts the number of odd numbers in a vector
num.odd <- function(x)
{
  count = 0;
  d=length(x);
  for (i in 1:d)
  {
    if(x[i] %% 2 != 0) count=count+1; # %% is modulo division
  }

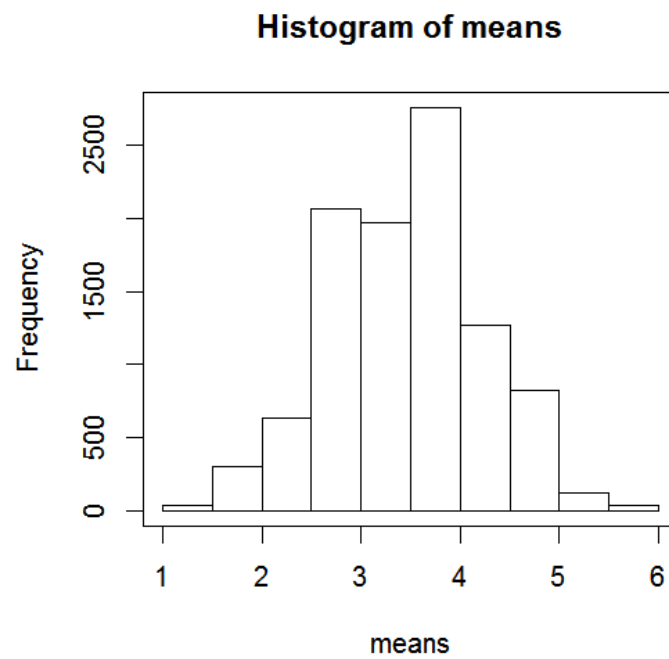
  return(count);
}

for (i in 1:iter)
{
  x=sample(1:6, n, replace=T);
  xbar=mean(x);
  means[i]=xbar;

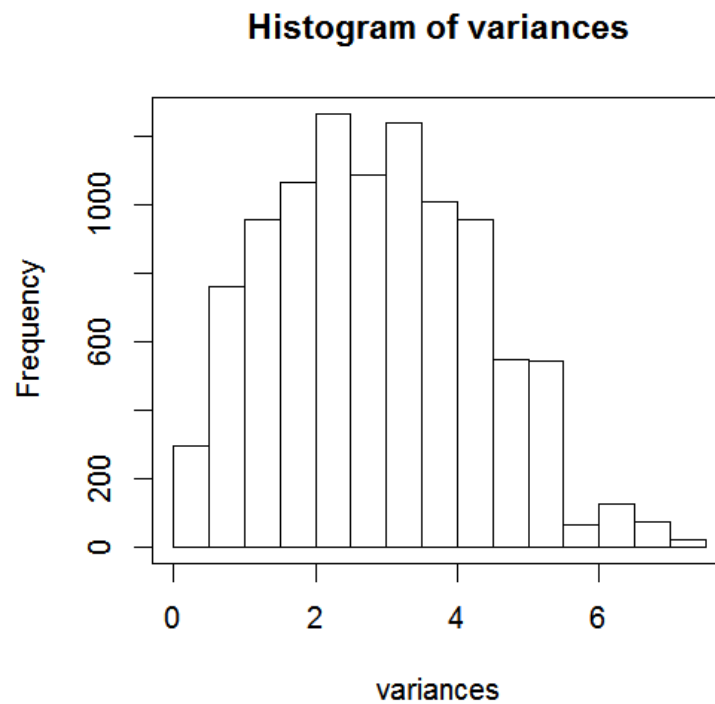
  varx=var(x);
  variances[i]=varx;

  proportions[i]=num.odd(x)/n;
}

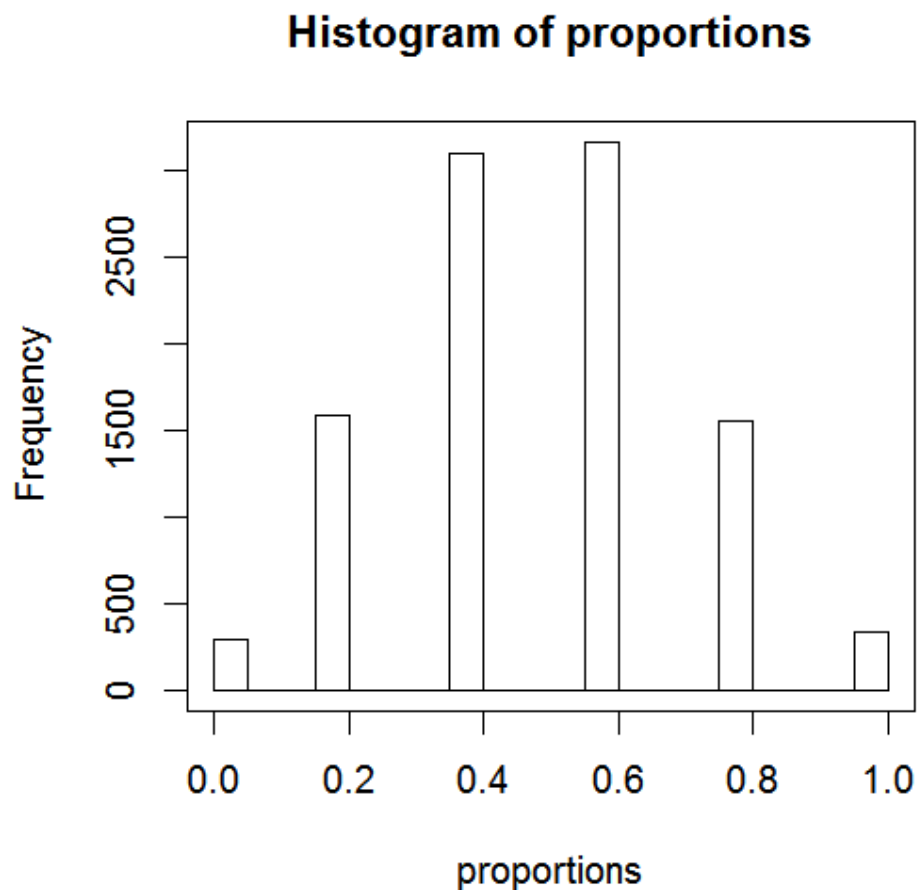
> mean(means);
[1] 3.50346
> hist(means); box();
```



```
> mean(variances);  
[1] 2.91225  
> hist(variances); box();
```



```
> mean(proportions);
[1] 0.50142
> hist(proportions); box();
```



We then compare the simulation results with the theoretical results by using the probability distribution of X . Since the die is balanced, we have

$$P(x) = P(X = x) = \frac{1}{6}, \quad x = 1, 2, 3, 4, 5, 6.$$

Thus,

$$\mu = \sum xP(x) = 21/6 = 3.5.$$

$$\sigma^2 = \sum x^2P(x) - \mu^2 \approx 15.167 - 3.5^2 = 2.917.$$

The proportion of odd numbers is $\frac{1}{6} + \frac{1}{6} + \frac{1}{6} = 0.5$

The population mean is 3.5; the mean of the 10,000 trials was 3.50346. If continued **indefinitely**, the sample mean will be 3.5. Also, notice the distribution is “approximately normal.”

The population variance is about 2.917; the mean of the 10,000 trials was 2.91225. If continued **indefinitely**, the sample variance will be 2.917. Also, notice the distribution is “skewed to the right.”

The population proportion of odd numbers is 0.50; the proportion for the 10,000 trials was 0.50142 . If continued **indefinitely**, the mean of sample proportions will be exactly 0.50. Also, notice the distribution is “approximately normal.”

NOT FOR SALE

17. CONFIDENCE INTERVAL ESTIMATIONS

In this chapter, we consider one-sample inference problems only using R.

17.1 Confidence Intervals of a Population Mean

If the population is either normal or the sample size is large ($n > 30$) and the population standard deviation σ is known, a $100(1-\alpha)$ % confidence interval of the population mean μ is of the form

$$\left(\bar{x} - Z_{\alpha/2} \frac{\sigma}{\sqrt{n}}, \quad \bar{x} + Z_{\alpha/2} \frac{\sigma}{\sqrt{n}} \right).$$

We need the sample mean and the standard normal critical value.

In reality, it is **very rare** to estimate a population mean or test a claim about an unknown population mean while the population standard deviation is known. Thus we construct a confidence interval of a population mean using t -distribution. We require that the population is either normal or the sample size is large ($n > 30$), and σ is unknown.

A $100(1-\alpha)$ % confidence interval of the population mean μ is of the form

$$\left(\bar{x} - t_{\alpha/2} \frac{s}{\sqrt{n}}, \quad \bar{x} + t_{\alpha/2} \frac{s}{\sqrt{n}} \right),$$

where the t -distribution has $df = n - 1$. Again the t -critical value needs to be found. An easier way can let you obtain the confidence interval immediately without having to find critical values.

T Test:

`t.test()` performs one and two sample t -tests on vectors of data..

Example

Consider the measured radiation emissions (in W/kg) corresponding to a sample of cell phones (see the data set **Emissions.xls**).

0.38	0.55	1.54	1.55	0.50	0.60	0.92	0.96	1.00	0.86	1.46
------	------	------	------	------	------	------	------	------	------	------

First, histogram and QQ plot shows the assumption of normality of the population from which the sample is selected is reasonable (DIY).

Next we construct a 96% confidence interval of the population mean using the following code.

```
> x=c(0.38, 0.55, 1.54, 1.55, 0.50, 0.60, 0.92, 0.96, 1.00,  
0.86, 1.46);  
> t.test(x, conf.level=0.96);
```

The following is the output.

One Sample t-test

```
data:  x  
t = 7.3583, df = 10, p-value = 2.43e-05  
alternative hypothesis: true mean is not equal to 0  
96 percent confidence interval:  
 0.6373712 1.2389924  
sample estimates:  
mean of x  
0.9381818
```

The required 96% confidence interval is given by (0.6374, 1.2390).

17.2 Confidence Intervals of a Population Proportion

We use one example to illustrate how to construct a confidence interval of a population proportion using R.

Example. Find the sample proportion of M&Ms that are green. Use the result to test the claim of Mars, Inc., that 16% of its plain M&M candies are green (**M&M_2.xls**).

The original data set looks like the following graph.

A	B	C	D	E	F	G	A
Red	Orange	Yellow	Brown	Blue	Green		Colors
0.751	0.735	0.883	0.696	0.881	0.925		Green
0.841	0.895	0.769	0.876	0.863	0.914		Green
0.856	0.865	0.859	0.855	0.775	0.881		Green
0.799	0.864	0.784	0.806	0.854	0.865		Green
0.966	0.852	0.824	0.840	0.810	0.865		Green
0.859	0.866	0.858	0.868	0.858	1.015		Green
0.857	0.859	0.848	0.859	0.818	0.876		Green
0.942	0.838	0.851	0.982	0.868	0.809		Green
0.873	0.863			0.803	0.865		Green
0.809	0.888			0.932	0.848		Green
0.890	0.925			0.842	0.940		Green
0.878	0.793			0.832	0.833		Green
0.905	0.977			0.807	0.845		Green
	0.850			0.841	0.852		Green
	0.830			0.932	0.778		Green
	0.856			0.833	0.814		Green
	0.842			0.881	0.791		Green
	0.778			0.818	0.810		Green
	0.786			0.864	0.881		others
	0.853			0.825			others
	0.864			0.855			others
	0.873			0.942			others
	0.880			0.825			others
	0.882			0.869		=COUNT(A2:F28)	others
	0.931			0.912			others
				0.887			others
				0.886			others

We are interested in the proportion of green candies and edit the data using one variable, colors, only as shown in the figure above.

Q. Find a 95% confidence interval of the population proportion of green M&Ms.

An easy way to find the confidence interval is count the sample size (100) and the number of green candies (19) using COUNT() function in Excel and write one line of R code.

```
> prop.test(19, 100, conf.level = 0.95, correct = FALSE)
```

The following is the output.

```

1-sample proportions test without continuity correction

data: 19 out of 100, null probability 0.5
X-squared = 38.44, df = 1, p-value = 5.646e-10
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.1251475 0.2777885
sample estimates:
      p
0.19

```

The required 95% confidence interval of the population proportion is given by (0.125, 0.278).

17.3 Confidence Intervals of a Population Variance

If the data are from a normal population, the $100(1-\alpha)\%$ confidence interval of the population variance is given by

$$\left(\frac{(n-1)S^2}{\chi^2_{\alpha/2}}, \frac{(n-1)S^2}{\chi^2_{1-\alpha/2}} \right).$$

This can be obtained manually with the help of R without using an R package. Or we can write our own R function as the following.

```

var.interval = function(data, conf.level)
{
  df = length(data) - 1;
  chilower = qchisq((1 - conf.level)/2, df);
  chiupper = qchisq((1 - conf.level)/2, df, lower.tail = FALSE);
  ssq = var(data);
  interval = c(df * ssq/chiupper, df * ssq/chilower);
  return(interval);
}

```

Example. Listed below are the heights (inches) for a simple random sample of ten supermodels. Use a $\alpha = 0.01$ significance level to test the claim that supermodels have heights with a standard deviation that is less than 2.6 inches.

70	71	69.25	68.5	69	70	71	70	70	69.5
----	----	-------	------	----	----	----	----	----	------

Construct a 98% confidence interval of σ^2

Solution.

With $df=10-1=9$, the two critical values of the Chi-square distributions are

$\chi^2_{0.01} = 21.66599$ and $\chi^2_{0.99} = 2.087901$, where these two values are two Chi-square quantiles. For example $\chi^2_{0.01}$ is obtained by the qchisq function

```
> qchisq(p=0.01, df=9, lower.tail = FALSE)
[1] 21.66599
```

Therefore, a level $100(1-2\alpha)\%=98\%$ confidence interval of σ^2 is given by

$$\left(\frac{(n-1)s^2}{\chi^2_{0.01}}, \frac{(n-1)s^2}{\chi^2_{0.99}} \right) = \left(\frac{9 \times 0.6395833}{21.66599}, \frac{9 \times 0.6395833}{2.087901} \right) = (0.266, 2.757),$$

where the sample variance can be obtained using the var function in R.

And thus a 98% confidence interval of σ is

$$\left(\sqrt{\frac{9 \times 0.6395833}{21.66599}}, \sqrt{\frac{9 \times 0.6395833}{2.087901}} \right) = (0.515, 1.660).$$

If we use our own function defined above, we simple use the next two lines to get a 98% confidence interval of σ^2

```
> x=c(70, 71, 69.25, 68.5, 69, 70, 71, 70, 70, 69.5);
> var.interval(data=x, conf.level=0.98);
[1] 0.2656813 2.7569558
```

17.4 Understanding Confidence Intervals

The formulas of the confidence intervals we derived in class are random. However, they are not random any more when we construct a confidence interval from an observed sample since the statistic(s) in the confidence interval now is(are) replaced by its(their) realized/observed value calculated from the specific sample. Therefore, we should avoid using probability language in interpreting a confidence interval. We use a simulation study in this section to better understand the concept of confidence intervals. Let us consider the two-sided symmetric confidence interval of a population mean.

Let X_1, \dots, X_n be a random sample from a population with σ unknown. If the population is normal or the sample size is large ($n \geq 30$). Then a $100(1-\alpha)\%$ symmetric confidence interval of μ is

$$\left(\bar{X} - t_{\alpha/2} \frac{S}{\sqrt{n}}, \quad \bar{X} + t_{\alpha/2} \frac{S}{\sqrt{n}} \right),$$

where S is the sample standard deviation and the t-distribution has $df=n-1$. The correct interpretation of the confidence interval is:

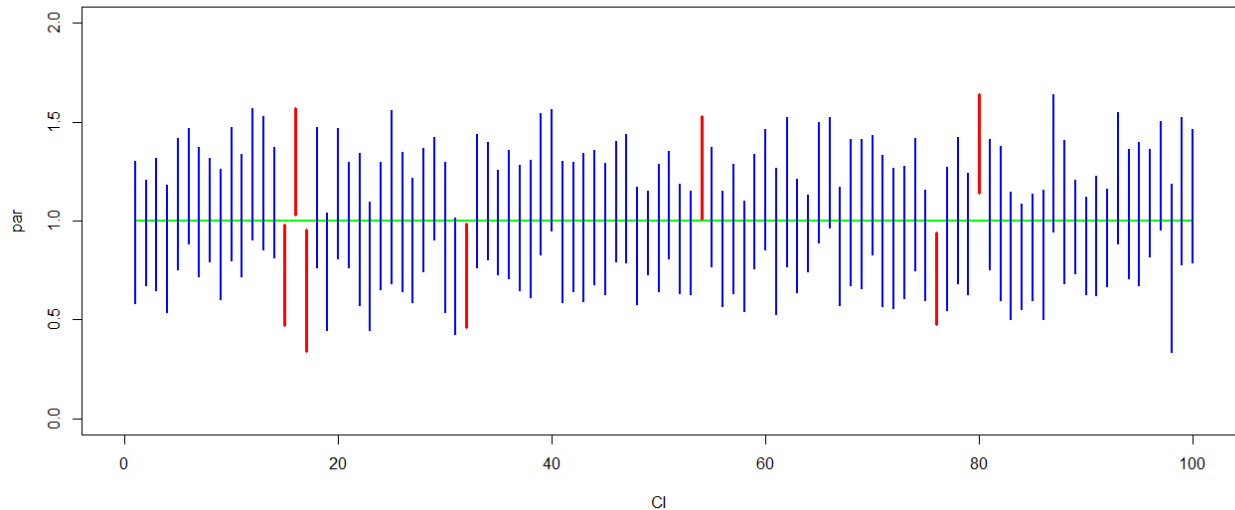
“If we repeat the sampling and compute the $100(1-\alpha)\%$ confidence interval of μ from each sample following the formula, then in the long run, $100(1-\alpha)\%$ such confidence intervals will actually contain the true mean μ .”

Simulation Scheme

- Specify the confidence level $100(1-\alpha)\% = 95\%$;
- Specify a population: Uniform[0,2] which is not normal. Then, $\mu=1$;
- Choose sample size $n=15$;
- Generate 100,000 samples of size n , and construct a 95% confidence interval of $\mu=1$ based on each sample;
- Empirical coverage = # of intervals containing μ divided by 100,000;
- Repeat the above using $n=50$.

When $n=15$, I got the empirical coverage = 94.846%; when $n=50$, I got the empirical coverage = 94.989%. When the sample size is larger, the coverage of the confidence intervals is closer to the confidence level 95%. This is because $\frac{\bar{X}-\mu}{S/\sqrt{n}}$ approximately has a t-distribution only if the sample size n is large, $n \geq 30$.

The following graph shows the simulation results using 100 samples. Among the 100 confidence intervals, only 93 of them contain the true population mean $\mu=1$.



The following is the **R code**.

```
n = 15;    mu=1;
iter=100;      #number of samples
count=0;      #number of intervals containing mu
L=numeric(); U=numeric(); CI=1:iter; par=rep(mu,iter);
plot(CI,par,type="l",xlim=c(0,iter),ylim=c(0,2),lwd=2,col="green");
for (i in 1:iter)
{
  #Sys.sleep(0.2);      #suspend 0.2 second
  x=runif(n, min = 0, max = 2);      #Generation of samples
  interval = t.test(x, conf.level=0.95)$conf.int;      #C.I.
  L[i]= interval[1];    U[i]=interval[2];
  if (L[i]<=mu && U[i]>=mu) count=count+1;
  time=c(i,i); bds=c(L[i], U[i]);
  if (L[i]>mu | U[i]<mu)
  {lines(time,bds,type="l", col="red", lwd=3);
  } else {
  lines(time,bds,type="l", col="blue", lwd=2);
  }
}
count; count/iter;
```


18. ONE-SAMPLE AND TWO-SAMPLE STATISTICAL INFERENCES

In this chapter, we consider one-sample and two-sample inference problems using R. Confidence interval estimation and hypothesis testing are two statistical inference methods, and confidence interval estimation can be used for hypothesis test as well. However, the confidence interval estimation for one-sample data is discussed in last Chapter, thus it will not be repeated in this chapter.

18.1 Population means

18.1.1 Z-TEST AND T-TEST OF A POPULATION MEAN

When the population standard deviation σ is known (the population is either normal or the sample size $n > 30$), z-test is used.

Example. Consider the preceding example with data set **Emissions.xls** and assume that the population standard deviation is known, $\sigma = 0.4$. Calculate the p-value of the z-test of $H_0: \mu = \mu_0 = 1.0$ versus $H_a: \mu < 1.0$.

We first calculate the test statistic

$$\frac{\bar{x} - \mu_0}{\sigma / \sqrt{n}}$$

which is -0.5125694 and then calculate the p-value of the test by finding standard normal tail probability using the R function `pnorm`.

To do a t-test, we use the `t.test()` function in R. It will be a two-sided test by default. So we need to change this depending on our problem.

```
> x=c(0.38, 0.55, 1.54, 1.55, 0.50, 0.60, 0.92, 0.96, 1.00,
0.86, 1.46)
> t.test(x, alternative="less", mu=1.0)
```

The following is the output.

One Sample t-test

```
data: x
t = -0.48485, df = 10, p-value = 0.3191
alternative hypothesis: true mean is less than 1
95 percent confidence interval:
```

```

      -Inf 1.169269
sample estimates:
mean of x
0.9381818

```

The p-value of the t-test is given by 0.3191 and thus we fail to reject H_0 .

18.1.2 Z-TEST OF THE DIFFERENCE BETWEEN TWO POPULATION MEANS

When the two population standard deviations are known (the populations are either normal or the sample sizes >30), z-test can be used. To test

$H_0 : \mu_1 - \mu_2 = D_0$ versus $H_A : \text{one or two sided}$ at level α ,
again, we first calculate the test statistic manually

$$z_0 = \frac{(\bar{x}_1 - \bar{x}_2) - D_0}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

and then calculate the p-value of the test by finding standard normal tail probability using the R function `pnorm`.

18.1.3 T-TEST OF THE DIFFERENCE BETWEEN TWO INDEPENDENT POPULATION MEANS

One method to do t-test (**independent two samples with equal population variances**)

$H_0 : \mu_1 - \mu_2 = D_0$ versus $H_A : \text{one or two sided}$ at level α ,
is to first construct the test statistic manually

$$t = \frac{(\bar{x}_1 - \bar{x}_2) - D_0}{\sqrt{\frac{s_p^2}{n_1} + \frac{s_p^2}{n_2}}}, \text{ where}$$

$$s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{(n_1 - 1) + (n_2 - 1)}, \text{ } s_1^2 \text{ and } s_2^2 \text{ are sample variances for sample 1 and 2, respectively.}$$

and then calculate the p-value of the test by finding tail probability of the t-distribution with $df = n_1 + n_2 - 2$ using the `pt` function in R. If the population variances are not equal, the test statistic should be

$$t = \frac{(\bar{x}_1 - \bar{x}_2) - D_0}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}.$$

This can be done using R as well.

Example. Consider the data set **Longevity.xls** describe below.

Longevity Listed below are the numbers of years that popes and British monarchs (since 1690) lived after their election or coronation (based on data from *Computer-Interactive Data Analysis*, by Lunn and McNeil, John Wiley & Sons). Treat the values as simple random samples from a larger population. Use a 0.01 significance level to test the claim that the mean longevity for popes is less than the mean for British monarchs after coronation.

Popes: 2 9 21 3 6 10 18 11 6 25 23 6
 2 15 32 25 11 8 17 19 5 15 0 26

Kings and Queens: 17 6 13 12 13 33 59 10 7 63 9 25 36 15

Denote the mean longevity for popes and the mean for British monarchs by μ_1 and μ_2 respectively. We are testing

$$H_0 : \mu_1 - \mu_2 = 0 \text{ versus } H_A : \mu_1 < \mu_2 \text{ at level } \alpha = 0.01.$$

Let's skip the requirements check (DIY).

Solution.

```
> x1=c(2,9,21,3,6,10,18,11,6,25,23,6,2,15,32,25,11,8,17,19,5,
      15,0,26);
> x2=c(17,6,13,12,13,33,59,10,7,63,9,25,36,15);
```

Step 1. Since the t-test has two versions with assumption of equality of population variances and inequality of population variances, we need to perform F-test of **equality of variances** and check *p*-value (*p*-value = 0.00225) to determine if equal variances assumption is acceptable. In general, we conduct a two-sided F test using the `var.test` function in R.

```
> var.test(x1, x2, alternative = "two.sided");
```

And the following is the output.

F test to compare two variances

data: x1 and x2

F = 0.232, num df = 23, denom df = 13, p-value = 0.00225

alternative hypothesis: true ratio of variances is not equal to 1

95 percent confidence interval:

```

0.0798588 0.5871520
sample estimates:
ratio of variances
0.2320022

```

In general, if p -value is greater than 0.05, the equal variances assumption would be acceptable (at 5% level of significance). Now equal variances assumption is rejected since the p -value is 0.00225. Thus we reject the null hypothesis of equality of variances.

Step 2. We perform two independent samples t-test using the function `t.test` function in R without assuming equal variances.

```
> t.test(x1,x2, alternative='less', var.equal=FALSE);
```

The following is the output. We fail to reject the null hypothesis since the p -value is larger than the significance level 0.01.

```

Welch Two Sample t-test

data:  x1 and x2
t = -1.8101, df = 16.585, p-value = 0.04422
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
 -Inf -0.3603019
sample estimates:
mean of x mean of y
13.12500 22.71429

```

18.1.4 PAIRED T-TEST

Again, paired t-test can be done manually by first calculating the test statistics and then finding the p-value of the test by the function `pt`. `T.test` in R can do this type of test in a much easier way.

Example – Matched pairs of heights of U.S. presidents and heights of their main opponents (**Height.xls**). We want to test $H_0: \mu_1 = \mu_2$ versus the two sided alternative $H_A: \mu_1 \neq \mu_2$.

Dependent Samples: Matched pairs of heights of U.S. presidents and heights of their main opponents.

Height (cm) of President	189	173	183	180	179
Height (cm) of Main Opponent	170	185	175	180	178

Again, let's skip the normality requirement check (DIY).

Sol.

```
> x1=c(189, 173, 183, 180, 179);
> x2=c(170, 185, 175, 180, 178);
> t.test(x1,x2, alternative='two.sided', paired=TRUE);
```

The following is the output.

```
Paired t-test

data:  x1 and x2
t = 0.6283, df = 4, p-value = 0.5639
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -10.9408  17.3408
sample estimates:
mean of the differences
      3.2
```

18.2 Population proportions

18.2.1 ONE-SAMPLE TEST

We use one example to illustrate how to test one population proportions using R.

Example. Find the sample proportion of M&Ms that are green. Use the result to test the claim of Mars, Inc., that 16% of its plain M&M candies are green (**M&M_2.xls**).

We need to perform the two-sided test $H_0: p = 0.16$ versus $H_a: p \neq 0.16$. The original data set looks like the following graph.

A	B	C	D	E	F	G		A
Red	Orange	Yellow	Brown	Blue	Green			Colors
0.751	0.735	0.883	0.696	0.881	0.925		1	Green
0.841	0.895	0.769	0.876	0.863	0.914		2	Green
0.856	0.865	0.859	0.855	0.775	0.881		3	Green
0.799	0.864	0.784	0.806	0.854	0.865		4	Green
0.966	0.852	0.824	0.840	0.810	0.865		5	Green
0.859	0.866	0.858	0.868	0.858	1.015		6	Green
0.857	0.859	0.848	0.859	0.818	0.876		7	Green
0.942	0.838	0.851	0.982	0.868	0.809		8	Green
0.873	0.863			0.803	0.865		9	Green
0.809	0.888			0.932	0.848		10	Green
0.890	0.925			0.842	0.940		11	Green
0.878	0.793			0.832	0.833		12	Green
0.905	0.977			0.807	0.845		13	Green
	0.850			0.841	0.852		14	Green
	0.830			0.932	0.778		15	Green
	0.856			0.833	0.814		16	Green
	0.842			0.881	0.791		17	Green
	0.778			0.818	0.810		18	Green
	0.786			0.864	0.881		19	Green
	0.853			0.825			20	others
	0.864			0.855			21	others
	0.873			0.942			22	others
	0.880			0.825			23	others
	0.882			0.869		=COUNT(A2:F28)	24	others
	0.931			0.912			25	others
				0.887			26	others
				0.886			27	others
							28	others
							29	others

We are interested in the proportion of green candies and edit the data using one variable, colors, only as shown in the figure above. We simply use Excel to count the sample size n and x , the number of successes (greens). Here, $x=19$ and $n=100$.

```
> prop.test(19, 100, p=0.16, alternative = "two.sided", correct = FALSE);
```

The following is the output.

```
1-sample proportions test without continuity correction

data: 19 out of 100, null probability 0.16
X-squared = 0.66964, df = 1, p-value = 0.4132
alternative hypothesis: true p is not equal to 0.16
95 percent confidence interval:
 0.1251475 0.2777885
sample estimates:
      p 
0.19
```

You can get more information about the function `prop.test()` in R using R help.

18.2.2 TWO-SAMPLE TEST

We use one example to illustrate how to test one population proportions using R.

Example. Independent random samples of $n_1=80$ and $n_2=78$ observations were randomly selected from binary populations 1 and 2, respectively. Sample 1 had 34 successes, and sample 2 had 41 successes. Test if there is a significant difference between the two proportions.

We need to perform the two-sided test $H_0: p_1=p_2$ versus $H_a: p_1 \neq p_2$.

Again, the easy way to do this is to use function `prop.test()` in R. The R code is `prop.test(x=c(34,41), n=c(80,78), alternative="two.sided", correct = FALSE)`

The following will be the output.

```
2-sample test for equality of proportions without continuity
correction

data: c(34, 41) out of c(80, 78)
X-squared = 1.6042, df = 1, p-value = 0.2053
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.25560697 0.05432492
sample estimates:
 prop 1    prop 2 
0.425000 0.525641
```

Now suppose the original data set has two groups. Sample 1 has response S (success) and F(failure); sample 2 has response S (success) and F(failure). See the data set **TwoProportions.xls** or **TwoProportions.csv**.

Then we can use R to construct a 2×2 table.

	Number of Successes	Number of Failures
Sample 1	x_1	$n_1 - x_1$
Sample 2	x_2	$n_2 - x_2$

In this example, we have

	Number of Failures	Number of Successes
Sample 1	46	34
Sample 2	37	41

```
>myData=read.table('E:/ESUTeaching/2019Spring/Math402/RHandout/Two
Proportions.csv', sep=',', header=TRUE);
> DataTab=xtabs(formula=~myData$Response + myData$Samples);
> DataTab;
              myData$Samples
myData$Response  1    2
               F 46 37
               S 34 41
> prop.test(DataTab, alternative='two.sided', correct=FALSE);
```

The following is the output.

2-sample test for equality of proportions without continuity correction

```
data: DataTab
X-squared = 1.6042, df = 1, p-value = 0.2053
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.05444815  0.25621521
sample estimates:
   prop 1    prop 2 
0.5542169 0.4533333
```


Note that here the success is defined as “F” and the failure is defined as “S”. Or we can conduct the analysis by first manually constructing the 2×2 table.

```
> A = matrix(c(34,46,41,37), byrow=TRUE, ncol=2);  
> A.table <- as.table(A);  
> A.table  
      A  B  
A 34 46  
B 41 37  
> prop.test(A.table, alternative='two.sided', correct=FALSE);
```

The following is the output.

```
2-sample test for equality of proportions without continuity  
correction  
  
data:  A.table  
X-squared = 1.6042, df = 1, p-value = 0.2053  
alternative hypothesis: two.sided  
95 percent confidence interval:  
 -0.25560697  0.05432492  
sample estimates:  
   prop 1   prop 2  
0.425000 0.525641
```

18.3 Population variances

18.3.1 ONE-SAMPLE TEST - CHI-SQUARE TEST OF A POPULATION VARIANCE

Chi-square test of one population variance is not designed in a R function. To test $H_0: \sigma^2 = \sigma_0^2$ versus $H_a: \text{one or two sided at level } \alpha$, we need to calculate the test static manually

$$x_0^2 = \frac{(n-1)s^2}{\sigma_0^2}$$

and obtain the p -value of the test using the R function `pchisq`. Similarly, a $100(1-\alpha)\%$ confidence interval of a population variance can be manually constructed using this R function `pchisq`.

Example. Listed below are the heights (inches) for a simple random sample of ten supermodels. Use a $\alpha = 0.01$ significance level to test the claim that supermodels have heights with a standard deviation that is less than 2.6 inches (see the data set **ModelHeight.xls**).

70	71	69.25	68.5	69	70	71	70	70	69.5
----	----	-------	------	----	----	----	----	----	------

We are testing $H_0: \sigma = 2.6$ versus $H_a: \sigma < 2.6$.

Solution. First, the sample variance is calculated using R as

$$s^2 \approx 0.6395833$$

And thus the test statistic is calculated as

$$\frac{(n-1)s^2}{\sigma^2} = \frac{(10-1) \times 0.6395833}{2.6^2} \approx 0.8515.$$

With $df=10-1=9$, the p -value of the one-sided test is obtained by

```
pchisq(c(0.8515), df=9, lower.tail=TRUE)
```

We use “Lower tail” probability since the alternative hypothesis is left-tailed. The resulting p -value of the test is 0.0002897 as shown in the following output.

```
> pchisq(c(0.8515), df=9, lower.tail=TRUE)
[1] 0.0002897207
```

The second method is to construct a two-tailed level $100(1-2\alpha)\% = 98\%$ confidence interval of σ^2 . With $df=10-1=9$, the two critical values of the Chi-square distributions are obtained using the `qchisq` function

$$\chi^2_{0.01} = 21.66599 \text{ and } \chi^2_{0.99} = 2.087901,$$

where these two values are two Chi-square quantiles. For example $\chi^2_{0.01}$ is obtained by

```
> qchisq(p=0.01, df=10-1, lower.tail=FALSE)
[1] 21.66599
```

Therefore, a level $100(1-2\alpha)\%=98\%$ confidence interval of σ^2 is given by

$$\left(\frac{(n-1)s^2}{\chi^2_{0.01}}, \frac{(n-1)s^2}{\chi^2_{0.99}} \right) = \left(\frac{9 \times 0.6395833}{21.66599}, \frac{9 \times 0.6395833}{2.087901} \right).$$

And thus a 98% confidence interval of σ is

$$\left(\sqrt{\frac{9 \times 0.6395833}{21.66599}}, \sqrt{\frac{9 \times 0.6395833}{2.087901}} \right) = (0.515, 1.660).$$

H_0 is rejected since the confidence interval does not contain the $\sigma=2.6$ value under H_0 .

Or we can define our own R function to obtain a two-sided confidence interval and hypothesis test of a population variance. This is left as a homework.

18.3.2 TWO-SAMPLE TEST – F-TEST OF THE DIFFERENCE BETWEEN TWO POPULATION VARIANCES

Example. Use the weights of red M&Ms and orange M&Ms. Test the claim that the two samples are from populations with the same amount of variation (**M&M_3.xls**).

We are testing $H_0: \sigma_1 = \sigma_2$ versus $H_a: \sigma_1 \neq \sigma_2$ which is a two-sided test. And we conduct the two-sided F test using the `var.test` function in R.

```
> x1 = c(0.751,0.841,0.856,0.799,0.966,0.859,0.857,0.942,
        0.873,0.809, 0.890,0.878, 0.905);
>
x2=c(0.735,0.895,0.865,0.864,0.852,0.866,0.859,0.838,0.863,0.888,0
.925,0.793,0.977,0.850,0.830,0.856,0.842,0.778,0.786,0.853,0.864,0
.873,0.880,0.882,0.931);
> var.test(x1, x2, alternative = "two.sided");
```

And the following is the output.

```
      F test to compare two variances

data:  x1 and x2
F = 1.3213, num df = 12, denom df = 24, p-value = 0.5399
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.5199758 3.9887301
sample estimates:
ratio of variances
 1.321335
```

In general, if p -value is greater than 0.05, the equal variances assumption would be acceptable (at 5% level of significance). Now equal variances assumption is NOT rejected since the p -value is 0.05399. Thus we fail to reject the null hypothesis of equality of variances.

18.4 A Simulation Study of Probability of Type II error

In this section, we conduct a simulation study to investigate the relationship between probability of Type I error and probability of Type II error of a test.

p -value:

The p -value of a hypothesis test of H_0 versus an alternative H_A measures the strength of the evidence from a sample against H_0 . The smaller the p -value is, the more convincing the evidence is against the null hypothesis. That is, H_0 should be rejected only if the p -value of the test is small.

Question. Use R simulations to understand the relationship between probability of type I error and probability of type II error in a hypothesis test.

- (1) Generate a random sample of size 15 from a Normal population with mean $\mu_1=1.5$ and variance 2.
- (2) Test $H_0: \mu = \mu_0 = 1.0$ versus $H_A: \mu > 1.0$ at the given significance level (probability of type I error) $\alpha=0.05$ using “t.test()” in R. Assume that the population variance is unknown.
- (3) Repeat (1) and (2) 5000 times using the “for” loop in R and count the number of times that the p -value of the test is greater than $\alpha=0.05$ using “if” in R. (Hint: get the p -value of the t test using “t.test()\$p.value”).

Answer the following questions.

- (a) Calculate the empirical $\beta = \text{count}/5000$ which is the simulated probability of type II error. Compare this empirical β with the average value of the **true β** 's (you need to derive the formula) with each obtained from a single sample when $\alpha=0.05$ for the above test.
- (b) Repeat the simulation using sample size 60. Compare the empirical β with the result obtained in (a) and summarize your findings about how sample sizes affect the power of a test.

Solution.

We first derive the formula of $\beta = P(\text{Type II error})$.

Now, we are testing

$$H_0 : \mu = 1 \text{ against } H_A : \mu > 1$$

at significance level $\alpha = 0.05$

The test statistic is given by

$$t_0 = \frac{\bar{x} - 1}{s/\sqrt{n}}.$$

Notice that under $H_A: \mu = 1.5$. Therefore, for any sample size n ,

$$\begin{aligned} \beta &= P(\text{Type II error}) = P(\text{we fail to reject } H_0 \mid H_A) \\ &= P\left(\frac{\bar{x} - 1}{s/\sqrt{n}} < t_{0.05} \mid \mu = 1.5\right) \\ &= P\left(\frac{\bar{x} - 1.5 + 0.5}{s/\sqrt{n}} < t_{0.05} \mid \mu = 1.5\right) \\ &= P\left(\frac{\bar{x} - 1.5}{s/\sqrt{n}} + \frac{0.5}{s/\sqrt{n}} < t_{0.05} \mid \mu = 1.5\right), \end{aligned}$$

where

$$\frac{\bar{x} - 1.5}{s/\sqrt{n}}$$

has a t distribution with $df = n-1$ under $H_A: \mu = 1.5$.

Therefore,

$$\beta = P\left(t < t_{0.05} - \frac{0.5}{s/\sqrt{n}}\right)$$

which depends on the sample size n and sample standard deviation s .

The following page is an example R code for the problem. You can use the simulation results to answer the questions.

```
n = 50;
alpha = 0.05;      Counter = 0;
iter = 5000;      #Looping 5000 times
beta_true = numeric();
df = n-1;

for (i in 1:iter)
{
  x = rnorm(n, mean=1.5, sd=sqrt(2)); #Generation of random samples
  s = sd(x);                        #m = mean(x);
  se = s/sqrt(n);                    #standard error

  t.05=qt(p=0.05, df=n-1, lower.tail = FALSE);
  beta_true[i] = pt(t.05- (0.5 / se), df = n-1, lower.tail = TRUE);
  #Finding true beta

  test = t.test(x, alternative = "greater", mu=1.0); #Conducting t-test

  #Raising counter if we fail to reject H0: p-value of test statistic is greater than significance level
  if(test$p.value > alpha) {Counter = Counter + 1;}
}

writeLines("P-Value counter: ")
Counter;

#Calculating empirical beta
beta = Counter / 5000;

writeLines("Empirical beta: ")
beta;

writeLines("True beta: ")
mean(beta_true); #average values of the true betas
```

19. CATEGORICAL DATA ANALYSIS

In this chapter, we consider categorical data analysis by examples using R.

19.1 Goodness-of-fit Test

Example. A random sample of 100 weights of Californians is obtained, and the last digits of those weights are summarized in the following table (**LastDigits.xls**). Test the claim that the sample is from a population of weights in which the last digits do not occur with the **same frequency**. The raw data are of the form

Last Digit	Frequency
0	46
1	1
2	2
3	3
4	3
5	30
6	4
7	0
8	8
9	3

We are testing

H_0 : all the probabilities are the same $p_0=p_1=\dots=p_9=1/10$ versus

H_a : at least one of the probabilities p_0, p_1, \dots, p_9 is different from others.

The rejection region for the Chi-square test is right-tailed. This test of equal probabilities can be easily done by writing two lines of code.

```
> x=c(46,1,2,3,3,30,4,0,8,3);
> chisq.test(x, p=rep(1/10,10));
```

The second line can also be written as

```
> chisq.test(x, p=c(1/10,1/10,1/10,1/10,1/10,1/10,1/10,1/10,1/10, 1/10));
```

The following is the output.

```
Chi-squared test for given probabilities
data:  x
X-squared = 212.8, df = 9, p-value < 2.2e-16
```

The small p -value warrants the rejection of H_0 .

19.2 Analysis of Contingency Tables

Similar as the Goodness-of-fit test, we can test dependence or homogeneity of two classifications by Chi-square test.

Example. Does it appear that the choice of treatment affects the success of the treatment for the foot procedures? Use a 0.05 level of significance to test the claim that success is independent of treatment group.

	Success	Failure
Surgery	54	12
Weight-Bearing Cast	41	51
Non-Weight-Bearing Cast for 6 Weeks	70	3
Non-Weight-Bearing Cast Less Than 6 Weeks	17	5

We first enter the data summary manually and then conduct the Chi-square test.

```
> DataTable0 = as.table(cbind(c(54,41,70,17),c(12,51,3,5)));
> chisq.test(DataTable0, correct=FALSE);
```

The following is the output.

Pearson's Chi-squared test

```
data:  DataTable0
X-squared = 58.393, df = 3, p-value = 1.296e-12
```

We can make the table nicer by adding row variable name, column variable names, and the corresponding categories.

```
> DataTable = as.table(cbind(c(54,41,70,17),c(12,51,3,5)));
> dimnames(DataTable)=list(treatment=c("1","2","3","4"),result=c("S",
", "F")); # rownames(DataTable) = c('1', '2', '3', '4');
colnames(DataTable) = c('S', 'F');
> DataTable; # Counts
      result
treatment S  F
      1 54 12
      2 41 51
      3 70  3
      4 17  5
> Test = chisq.test(DataTable, correct=FALSE);
```



```
> Test
```

The following is the output same as above.

```
Pearson's Chi-squared test

data:  DataTable
X-squared = 58.3933, df = 3, p-value = 1.296e-12
```

Furthermore, we can get the expected count in each cell.

```
> Test$expected # Expected Counts
      1      2
1 47.47826 18.521739
2 66.18182 25.818182
3 52.51383 20.486166
4 15.82609  6.173913

> remove(Test);
```

We can conduct the Fisher's exact test as well.

```
> fisher.test(DataTable)

Fisher's Exact Test for Count Data

data:  DataTable
p-value = 1.817e-13
alternative hypothesis: two.sided

> remove(DataTable);
```

The p -value of the Pearson's Chi-squared test is 1.296×10^{-12} which is very small and thus the null hypothesis of independence of the two variables should be rejected. Therefore, it appears that the choice of treatment affects the success of the treatment for the foot procedures.

If the raw data set is available, the data set should be imported. And then we can use the `table()` function or `xtabs()` function to obtain the contingency table.

20. ANALYSIS OF VARIANCE

In this chapter, we consider analysis of variance by examples using R.

20.1 One-Way ANOVA

Example. Is the attention span of children affected by whether or not they had a good breakfast? Twelve children were randomly divided into three groups and assigned to a different meal plan. The response was attention span in minutes during the morning reading time (**breakfast.csv**).

No Breakfast	Light Breakfast	Full breakfast
8	14	10
7	16	12
9	12	16
13	17	15

One-way analysis is an extension of comparing two independent population means.

(1) Analysis of Variance and Multiple Comparisons

Step 1. The data must be arranged in a single column and with Group variable takes on 1, 2, 3 or 4 placed in the column next to it denote which sample the value is from (See figure below).

response	treatment
8	1
7	1
9	1
13	1
14	2
16	2
12	2
17	2
10	3
12	3
16	3
15	3

Then we import the data of this format into R. Another way is to manage the summary of the data using R.

```
> Breakfast
=read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/breakfast.
csv", header=TRUE, sep=",");
> Breakfast
```

```

      trt1 trt2 trt3
1       8   14   10
2       7   16   12
3       9   12   16
4      13   17   15

> resp = c(t(as.matrix(Breakfast))));      # response data
> resp
[1]  8 14 10  7 16 12  9 12 16 13 17 15

> f = c("Trt1", "Trt2", "Trt3")      # treatment levels
> k = 3; # number of treatment levels
> n = 4; # observations per treatment
> Trt = gl(k, 1, n*k, factor(f)); # matching treatments; treatments are factors
> Trt
[1] Trt1 Trt2 Trt3 Trt1 Trt2 Trt3 Trt1 Trt2 Trt3 Trt1 Trt2 Trt3
Levels: Trt1 Trt2 Trt3

> Breakfast=as.data.frame(cbind(resp, Trt));
> Breakfast$Trt =factor(Breakfast$Trt); #make "Trt" a factor
> Breakfast
      resp Trt
1       8   1
2      14   2
3      10   3
4       7   1
5      16   2
6      12   3
7       9   1
8      12   2
9      16   3
10     13   1
11     17   2
12     15   3

```

Remark. If the data are raw data like above with the variable “Trt” (treatment) taking numerical values (see **breakfast2.csv**). Again, we need to make the variable “Trt” a factor using factor() function.

```

> Breakfast2
=read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/breakfast2
.csv", header=TRUE, sep=",");
> Breakfast=as.data.frame(cbind(resp=Breakfast2$resp, Trt=
Breakfast2$Trt));
> Breakfast$Trt =factor(Breakfast$Trt); #make "Trt" a factor

```

Step 2. Then we conduct One-way ANOVA for the “formal” data which should be the raw data. Before doing this, we can use descriptive statistics and a box plot to compare the three treatment means.

```
> tapply(Breakfast$resp, Breakfast$Trt, summary);
```

\$Trt1

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
7.00	7.75	8.50	9.25	10.00	13.00

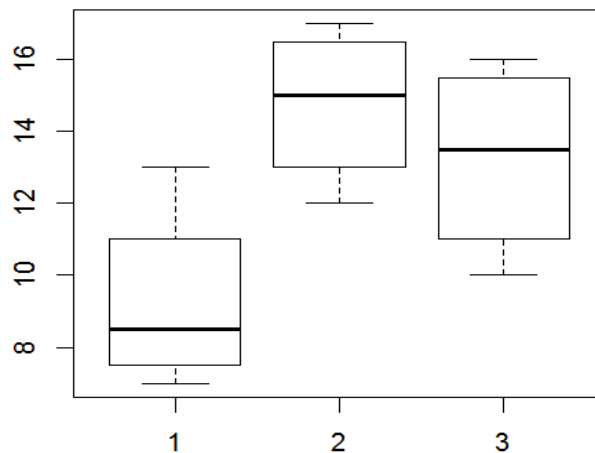
\$Trt2

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
12.00	13.50	15.00	14.75	16.25	17.00

\$Trt3

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
10.00	11.50	13.50	13.25	15.25	16.00

```
> boxplot(resp ~ Trt, data= Breakfast);
```



Several functions are available in R for One-way ANOVA.

(1) **aov()** function. `aov()` function is designed for balanced designs

```
> Modell1 = aov(resp ~ Trt, data= Breakfast); # Fit the one-way ANOVA
model;
```

```
> anova(Modell1); #Obtain the ANOVA table
Analysis of Variance Table
```

```
Response: resp
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Trt	2	64.667	32.333	4.9957	0.03472 *
Residuals	9	58.250	6.472		

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> summary(Modell1);
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Trt	2	64.67	32.33	4.996	0.0347 *
Residuals	9	58.25	6.47		

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the above output, the p-value of the test of equality of means is 0.0347 and thus the null hypothesis of equality of means is rejected. Therefore, we need multiple comparisons. For example, we conduct Tukey multiple comparisons of means.

```
> TukeyHSD(Modell1, conf.level = 0.95); # Tukey multiple comparisons of means
Tukey multiple comparisons of means
95% family-wise confidence level
```

```
Fit: aov(formula = resp ~ Trt, data = Breakfast)
```

```
$Trt
```

	diff	lwr	upr	p adj
Trt2-Trt1	5.5	0.4774069	10.522593	0.0330568
Trt3-Trt1	4.0	-1.0225931	9.022593	0.1202290
Trt3-Trt2	-1.5	-6.5225931	3.522593	0.6926057

It can be seen from the multiple comparisons by Tukey's method that treatment 1 and 2 are significantly different.

To conduct multiple comparisons by **Bonferroni's method**, we can use the `pairwise.t.test()` function..

```
> pairwise.t.test(x=Breakfast$resp, g=Breakfast$Trt, p.adj = "bonf");
```

The following is the Output.

```
Pairwise comparisons using t tests with pooled SD

data: Breakfast$resp and Breakfast$Trt

    1      2
2 0.041 -
3 0.160 1.000

P value adjustment method: bonferroni
```

It can be seen from the multiple comparisons by **Bonferroni's method** that treatment 1 and 2 are significantly different since the p-value of the test is 0.041. The following is the results by LSD method.

```
> pairwise.t.test(x=Breakfast$resp, g=Breakfast$Trt, p.adj =
"none");
```

```
Pairwise comparisons using t tests with pooled SD

data: Breakfast$resp and Breakfast$Trt

    1      2
2 0.014 -
3 0.053 0.426

P value adjustment method: none
```

For more information about multiple comparisons of means using R, please refer to the multcomp (<https://cran.r-project.org/web/packages/multcomp/index.html>) package.

(2) **oneway.test()** function.

```
> oneway.test(resp ~ Trt, data= Breakfast, var.equal = TRUE);
```

One-way analysis of means

data: resp and Trt

F = 4.9957, num df = 2, denom df = 9, p-value = 0.03472

(3) **lm()** function.

```
> Model2 = lm(resp ~ Trt, data= Breakfast);
```

```
> summary(Model2);
```

Call:

```
lm(formula = resp ~ Trt, data = Breakfast)
```

Residuals:

Min	1Q	Median	3Q	Max
-3.250	-1.500	-0.500	1.875	3.750

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	9.250	1.272	7.272	4.7e-05 ***
Trt2	5.500	1.799	3.057	0.0136 *
Trt3	4.000	1.799	2.224	0.0533 .

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.544 on 9 degrees of freedom

Multiple R-squared: 0.5261, Adjusted R-squared: 0.4208

F-statistic: 4.996 on 2 and 9 DF, p-value: 0.03472

```
> anova(Model2);
```

Analysis of Variance Table

Response: resp

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Trt	2	64.667	32.333	4.9957	0.03472 *
Residuals	9	58.250	6.472		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

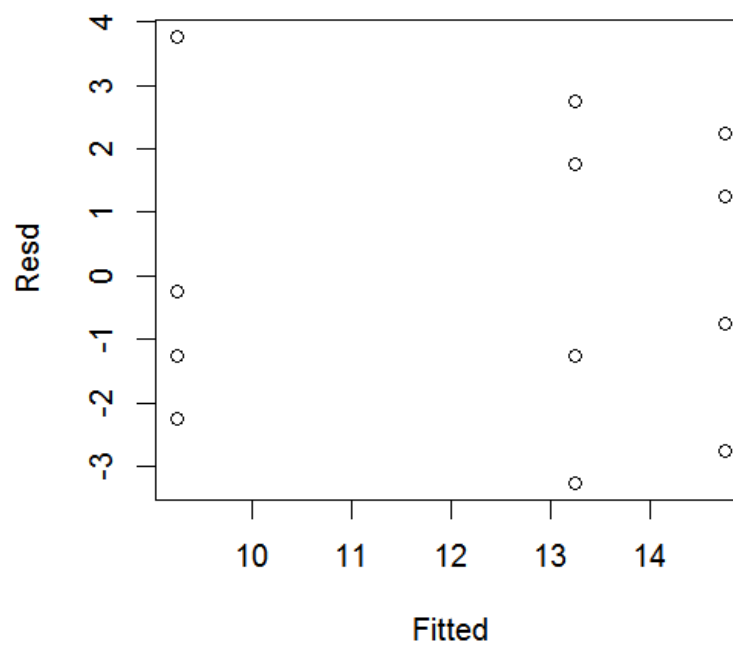
Model adequacy check. We need to check the fitted values and residuals to see if the three model assumptions are satisfied.

To get the fitted values, we use `Model1$fitted` or `Model2$fitted` in R.

```
> Fitted= Model1$fitted;
> Resd= Model1$residuals; #Resd=Breakfast$resp-Fitted;
> Breakfast=as.data.frame(cbind(Breakfast, Fitted, Resd));
#update the data by adding fitted values and residulas
> Breakfast$Trt =factor(Breakfast$Trt); #make "Trt" a factor
> str(Breakfast);
'data.frame':  12 obs. of  4 variables:
 $ resp  : num  8 14 10 7 16 12 9 12 16 13 ...
 $ Trt   : Factor w/ 3 levels "1","2","3": 1 2 3 1 2 3 1 2 3 1 ...
 $ Fitted: num  9.25 14.75 13.25 9.25 14.75 ...
 $ Resd  : num  -1.25 -0.75 -3.25 -2.25 1.25 ...
Fitted
```

(a) Independence and constant variance assumptions. The scatter plot of fitted values versus residuals is obtained using the updated data set.

```
> plot(Resd~ Fitted, data=Breakfast);
```



From the scatter plot, it can be seen that the residuals are randomly distributed and thus no pattern can be found. Therefore, the independence assumption seems reasonable. However, we can see that the spread of the residuals has a decreasing trend. But the departure from constant variance is not serious. We need statistical test, such as Levene test, to further check the constant variance assumption. That is, we are testing

$$H_0: \sigma_1^2 = \sigma_2^2 = \sigma_3^2 \text{ versus } H_A: \text{Not all variances are equal}$$

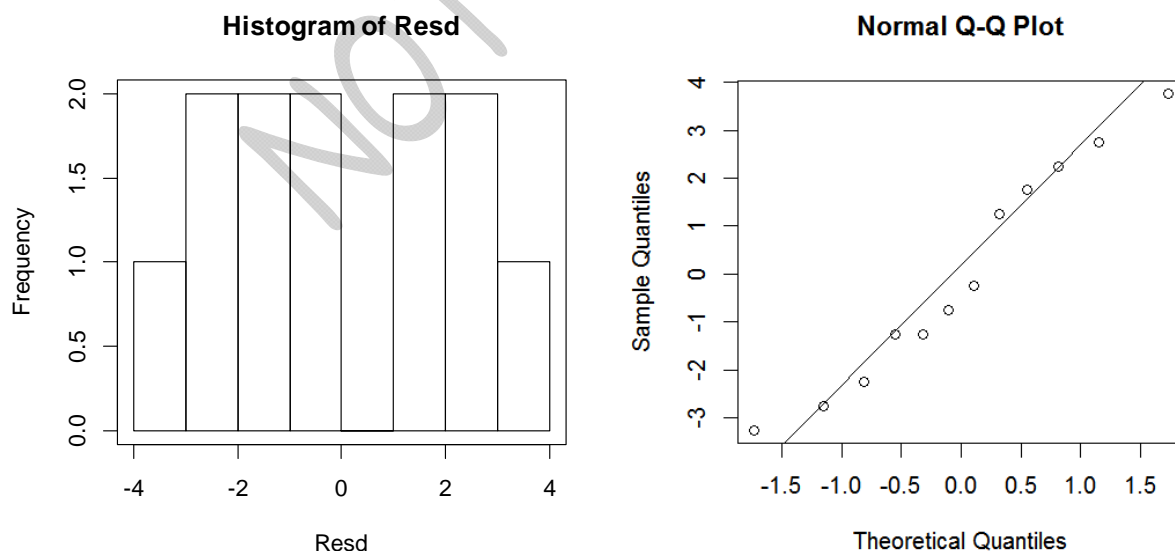
To conduct Levene Test of equality of variances of more than two groups, we use the `leveneTest()` function in the `car` package. Make sure you have the package installed.

```
> library(car);
> leveneTest(Resd ~ Trt, data=Breakfast);
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  2  0.1846 0.8345
      9
```

We fail to reject the null hypothesis because the p-value of the test is large.

(b) Normality. Normality assumption can be checked using histogram and QQ plot of the residuals.

```
> hist(Resd); box();
> qqnorm(Resd); qqline(Resd);
```



From the histogram of the residuals, it can be seen that the shape is not very bell-shaped. But the QQ plot seems to be supporting Normality.

We then conduct Shapiro-Wilk test for normality (Recall: a small p-value of the test results in rejection of Normality).

```
> shapiro.test(Resd);
```

```
Shapiro-Wilk normality test
```

```
data: Resd
```

```
W = 0.95013, p-value = 0.639
```

NOT FOR SALE

20.2 Two-Way ANOVA for Randomized Block Design

Example.

We want to investigate the effect of 3 methods of soil preparation on the growth of seedlings. Each method is applied to seedlings growing at each of 4 locations and the average first year growth is recorded. Is the average growth different for the 3 soil preps?

	Location			
Soil Prep	1	2	3	4
A	11	13	16	10
B	15	17	20	12
C	10	15	13	10

The summary data set is of the form

	1	2	3	4
A	11	13	16	10
B	15	17	20	12
C	10	15	13	10

So we change it to the format of raw data (**Soil.csv**)

growth	soil	location
11	A	1
15	B	1
10	C	1
13	A	2
17	B	2
15	C	2
16	A	3
20	B	3
13	C	3
10	A	4
12	B	4
10	C	4

Remark. In this section, we use the `lm()` function to fit the general linear model for the block design. `aov()` function will produce the same results.

Analysis of Variance and Multiple Comparisons.

The locations are labelled as numbers and they should be converted into factor variables.

```
> seedlings
=read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/Soil.csv",
header=TRUE, sep=",");
> seedlings$location =factor(seedlings$location); #make "location" a
factor

> LinearModell <- lm(growth ~ location +soil, data=seedlings);
> summary(LinearModell);
```

Call:

```
lm(formula = growth ~ location + soil, data = seedlings)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.8333	-0.6250	0.1667	0.7083	1.5000

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	11.0000	0.9718	11.319	2.85e-05	***
location2	3.0000	1.1222	2.673	0.03686	*
location3	4.3333	1.1222	3.862	0.00835	**
location4	-1.3333	1.1222	-1.188	0.27966	
soilB	3.5000	0.9718	3.601	0.01135	*
soilC	-0.5000	0.9718	-0.514	0.62530	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.374 on 6 degrees of freedom

Multiple R-squared: 0.8979, Adjusted R-squared: 0.8128

F-statistic: 10.55 on 5 and 6 DF, p-value: 0.006199

```
> anova(LinearModel1);
Analysis of Variance Table

Response: growth
          Df Sum Sq Mean Sq F value    Pr(>F)
location   3 61.667  20.5556   10.882 0.007693 **
soil        2 38.000  19.0000   10.059 0.012124 *
Residuals   6 11.333   1.8889
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It can be seen from the ANOVA table that blocking is necessary and the null hypothesis of equality of three means of soil preps is rejected. Furthermore, we need to conduct multiple comparisons to see which mean(s) are different from others.

(i) Bonferroni's method.

```
> pairwise.t.test(seedlings$growth, seedlings$soil, p.adj = "bonf");

Pairwise comparisons using t tests with pooled SD

data:  seedlings$growth and seedlings$soil

   A      B
B 0.35 -
C 1.00 0.23

P value adjustment method: bonferroni
```

(ii) LSD method

```
> pairwise.t.test(seedlings$growth, seedlings$soil, p.adj = "none");

Pairwise comparisons using t tests with pooled SD

data:  seedlings$growth and seedlings$soil

   A      B
B 0.116 -
C 0.809 0.078

P value adjustment method: none
```

(iii) Tukey's method.

```
> LinearModel2=aov(growth ~ location + soil, data = seedlings);
> TukeyHSD(LinearModel2, "soil");
```

The following is the output.

```
Tukey multiple comparisons of means
 95% family-wise confidence level
```

```
Fit: aov(formula = growth ~ location + soil, data = seedlings)
```

```
$soil
      diff      lwr      upr      p adj
B-A   3.5   0.5181732  6.481827 0.0263670
C-A  -0.5  -3.4818268  2.481827 0.8672718
C-B  -4.0  -6.9818268 -1.018173 0.0147233
```

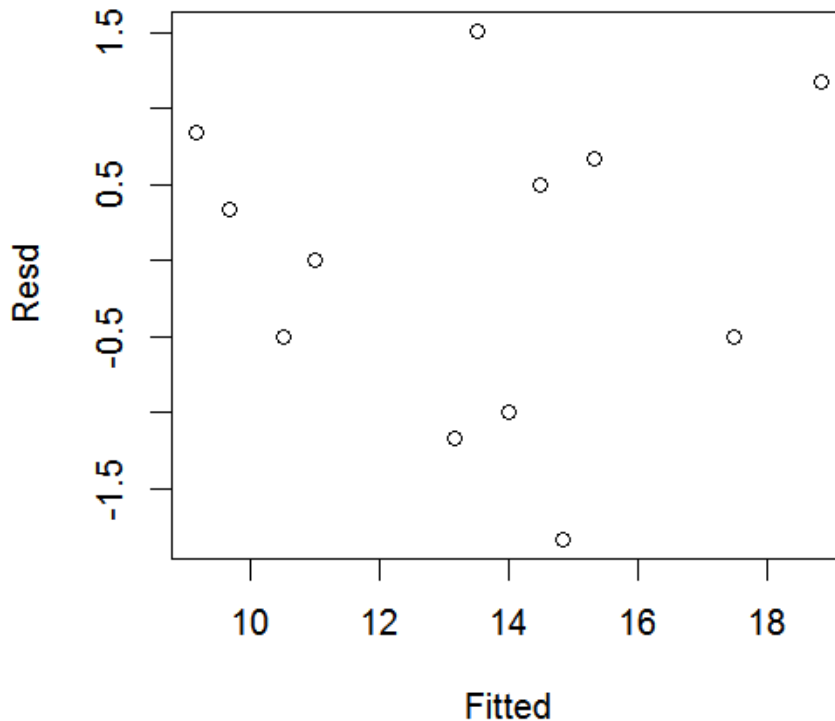
Model adequacy check.

We need to check the fitted values and residuals to see if the three model assumptions are satisfied. These are to be done in a similar way as those in one-way ANOVA.

```
> Fitted=LinearModel1$fitted;
> Resd=LinearModel1$residuals; #Resd= seedlings$growth-Fitted;
> seedlings =as.data.frame(cbind(seedlings, Fitted, Resd)); #update
the data by adding fitted values and residuals
> seedlings$location =factor(seedlings$location);
> str(seedlings);
'data.frame':  12 obs. of  5 variables:
 $ growth  : int  11 15 10 13 17 15 16 20 13 10 ...
 $ soil    : Factor w/ 3 levels "A","B","C": 1 2 3 1 2 3 1 2 3 1 ...
 $ location: Factor w/ 4 levels "1","2","3","4": 1 1 1 2 2 2 3 3 3 4 ...
 $ Fitted  : num  11 14.5 10.5 14 17.5 ...
 $ Resd    : num  5.33e-15 5.00e-01 -5.00e-01 -1.00 -5.00e-01 ...
```

- (a) Independence and constant variance assumptions. The scatter plot of fitted values versus residuals is obtained using the updated data set.

```
> plot(Resd~ Fitted, data= seedlings);
```



From the scatter plot, it can be seen that the residuals are randomly distributed and thus no pattern can be found. Therefore, the independence assumption seems reasonable. Moreover, we can see that the spread of the residuals does not have an increasing or decreasing trend. The constant variance assumption seems valid.

Now we use Levene test to further check the constant variance assumptions. That is, we are testing

$$H_0: \sigma_A^2 = \sigma_B^2 = \sigma_C^2 \text{ versus } H_A: \text{Not all variances are equal}$$

and

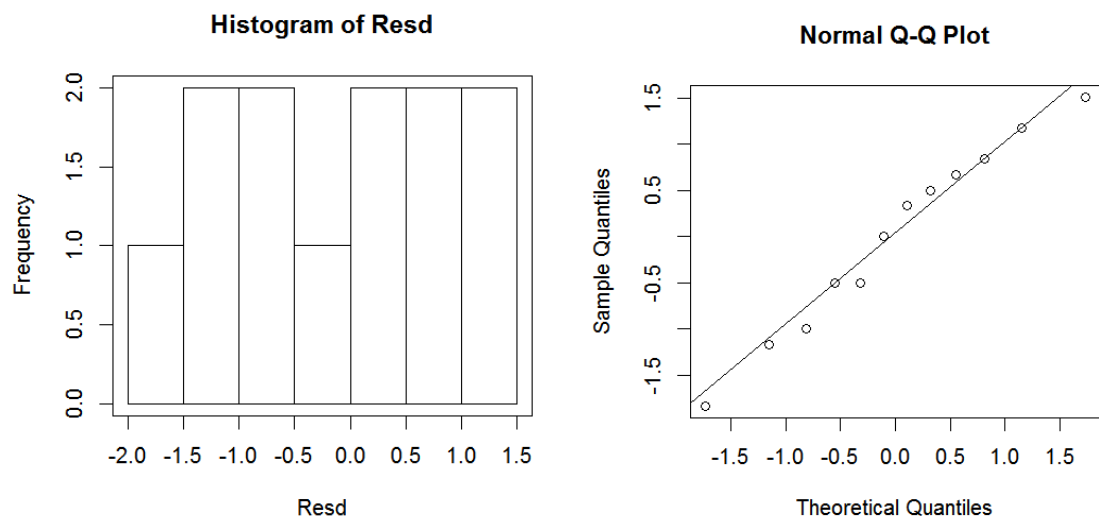
$$H_0: \sigma_1^2 = \sigma_2^2 = \sigma_3^2 = \sigma_4^2 \text{ versus } H_A: \text{Not all variances are equal}$$

```
> library(car);
> leveneTest(Resd ~ soil, data=seedlings);
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  2  1.7143  0.234
      9
> leveneTest(Resd ~ location, data=seedlings);
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  3  0.2778  0.84
      8
```

We fail to reject both null hypotheses because the p-values of both tests are large.

(b) Normality. Normality assumption can be checked using histogram and QQ plot of the residuals.

```
> hist(Resd); box();
> qqnorm(Resd); qqline(Resd);
```



From the histogram of the residuals, it can be seen that the shape is not very bell-shaped. But the QQ plot seems to be supporting Normality.

We then conduct Shapiro-Wilk test for normality (Recall: a small p-value of the test results in rejection of Normality).

```
> shapiro.test(Resd);
```

```
Shapiro-Wilk normality test
```

```
data: Resd
```

```
W = 0.97178, p-value = 0.9285
```

From the Shapiro-Wilk normality test, we fail to reject Normality as well.

NOT FOR SALE

20.3 Two-Way ANOVA for Factorial Design

Example.

Given the performance IQ scores in the table below, use two-way ANOVA to test for an interaction effect, an effect from the row factor of gender, and an effect from the column factor of blood lead level. The data in the table are categorized with two factors: (1) Sex: Male or Female and (2) Blood Lead Level: Low, Medium, or High. And the response variable is IQ score.

The following is the summarized two way table.

Measures of Performance IQ			
	Blood Lead Level		
	Low	Medium	High
Male	85	78	93
	90	107	97
	107	90	79
	85	83	97
	100	101	111
Female	64	97	100
	111	80	71
	76	108	99
	136	110	85
	99	97	93

We should enter the raw data in R in the following format (see **IQPerformance.csv**)

IQ	LeadLevel	Sex
85	Low	M
90	Low	M
107	Low	M
85	Low	M
100	Low	M
78	Medium	M
107	Medium	M
90	Medium	M
83	Medium	M
101	Medium	M
93	High	M

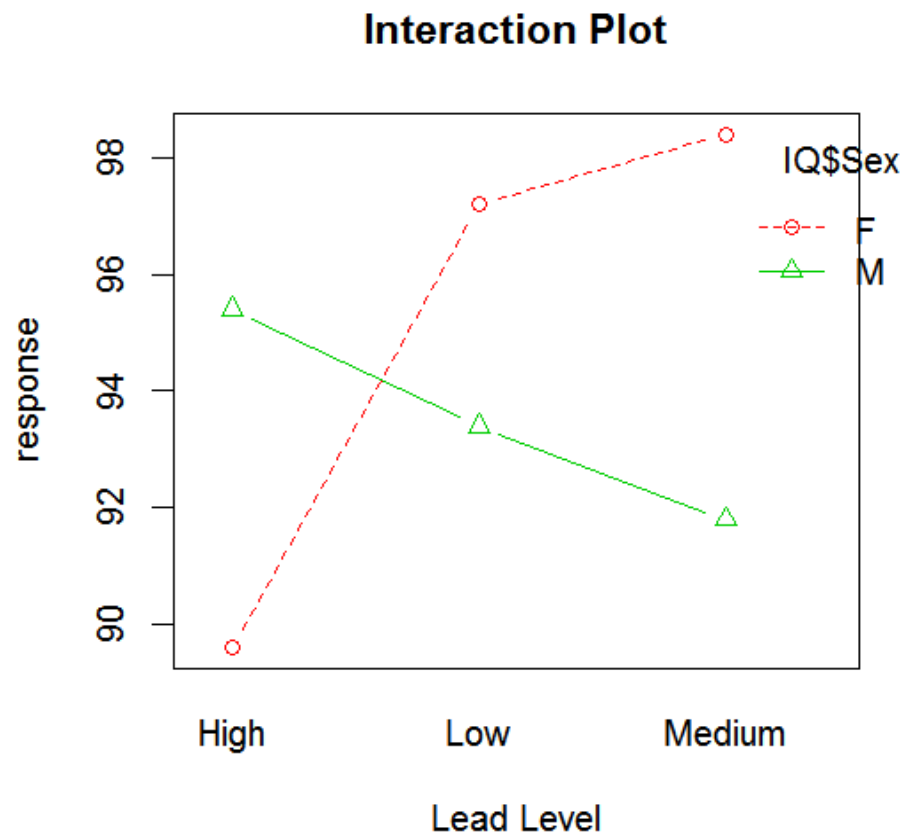
97	High	M
79	High	M
97	High	M
111	High	M
64	Low	F
111	Low	F
76	Low	F
136	Low	F
99	Low	F
97	Medium	F
80	Medium	F
108	Medium	F
110	Medium	F
97	Medium	F
100	High	F
71	High	F
99	High	F
85	High	F
93	High	F

Remark. Again, we use the `lm()` function to fit the general linear model for the factorial design in this section.

```
> IQ
=read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/IQPerformance.csv", header=TRUE, sep=",");
> str(IQ);
'data.frame': 30 obs. of 3 variables:
 $ Score: int 85 90 107 85 100 78 107 90 83 101 ...
 $ Lead : Factor w/ 3 levels "High","Low","Medium": 2 2 2 2 2 3 3 3 3 3 ...
 $ Sex : Factor w/ 2 levels "F","M": 2 2 2 2 2 2 2 2 2 2 ...
```

Before conducting ANOVA to check the interaction effect, we first explore the IQ data in the table by calculating the mean for each cell and constructing an interaction graph.

```
> interaction.plot(x.factor=IQ$Lead, trace.factor=IQ$Sex,
response=IQ$Score, fun=mean, type="b", legend=T, xlab="Lead Level",
ylab="response", main="Interaction Plot", pch=c(1,2), col =
c(2,3) );
```



From the interaction plot, it appears there is an interaction effect. Females with high lead exposure appear to have lower IQ scores, while males with high lead exposure appear to have high IQ scores. To get more reliable result by conducting a hypothesis test, we need analysis of variance.

Analysis of Variance and Multiple Comparisons

```
> fit1 = lm(Score ~ Lead*Sex, data=IQ);
  # fit1 = lm(Score ~ Lead +Sex +Lead*Sex, data=IQ);
> summary(fit1);
```

Call:

```
lm(formula = Score ~ Lead * Sex, data = IQ)
```

Residuals:

Min	1Q	Median	3Q	Max
-33.20	-8.40	0.10	9.55	38.80

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	89.600	7.004	12.793	3.29e-12 ***
LeadLow	7.600	9.905	0.767	0.450
LeadMedium	8.800	9.905	0.888	0.383
SexM	5.800	9.905	0.586	0.564
LeadLow:SexM	-9.600	14.008	-0.685	0.500
LeadMedium:SexM	-12.400	14.008	-0.885	0.385

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.66 on 24 degrees of freedom

Multiple R-squared: 0.04508, Adjusted R-squared: -0.1539

F-statistic: 0.2266 on 5 and 24 DF, p-value: 0.9473

The following is the results of the ANOVA.

```
> anova(fit1);
```

Analysis of Variance Table

Response: Score

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Lead	2	48.8	24.400	0.0995	0.9057
Sex	1	17.6	17.633	0.0719	0.7909
Lead:Sex	2	211.5	105.733	0.4311	0.6547
Residuals	24	5886.4	245.267		

From the ANOVA table above, we see that the p -value of the test of

H_0 : there is no interaction between the Sex and Lead factor
against

H_A : there is an interaction between the Sex and Lead factor

is 0.6547 and thus there is no sufficient evidence to show interaction between the two factors. Therefore, we can continue our analysis by checking if the population means for different lead levels are equal and if the population means for male and female are equal.

From the above ANOVA table, the two null hypotheses of equalities cannot be rejected. And thus we **do not** need to conduct multiple comparisons.

For illustration, let's consider multiple comparisons of population means for different lead levels by writing R programming code.

(i) Bonferroni's method.

```
> pairwise.t.test(IQ$Score, IQ$Lead, p.adj = "bonf");
```

The following is the output.

```
Pairwise comparisons using t tests with pooled SD

data:  IQ$Score and IQ$Lead

      High Low
Low    1    -
Medium 1    1

P value adjustment method: bonferroni
```

(ii) LSD method.

```
> pairwise.t.test(IQ$Score, IQ$Lead, p.adj = "none");
```

```
Pairwise comparisons using t tests with pooled SD

data:  IQ$Score and IQ$Lead

      High Low
Low    0.68 -
Medium 0.70 0.98

P value adjustment method: none
```

(iii) Tukey's method.

```
> fit2=aov(Score ~ Lead+Sex+ Lead*Sex,, data = IQ);
> TukeyHSD(fit2, "Lead");
  Tukey multiple comparisons of means
    95% family-wise confidence level
```

```
Fit: aov(formula = Score ~ Lead + Sex + Lead * Sex, data = IQ)
```

```
$Lead
```

	diff	lwr	upr	p adj
Low-High	2.8	-14.69052	20.29052	0.9159779
Medium-High	2.6	-14.89052	20.09052	0.9270803
Medium-Low	-0.2	-17.69052	17.29052	0.9995505

Model adequacy check.

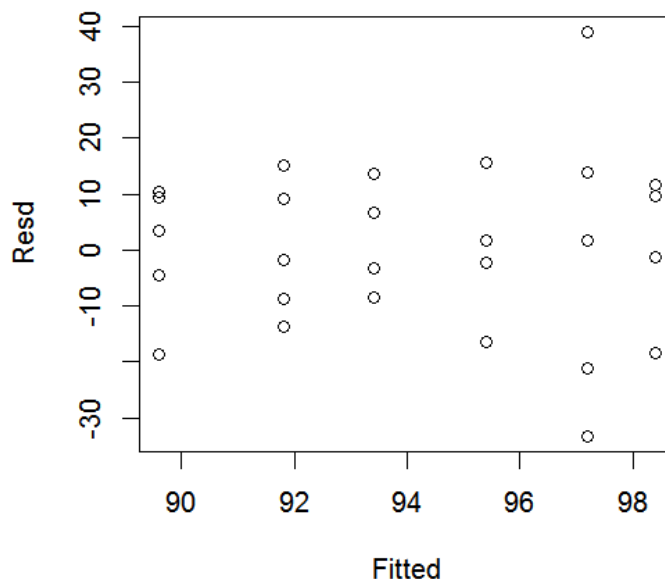
Again, after analysis of variance, we need to check the fitted values and residuals to see if the three model assumptions are satisfied. These are to be done in a similar way as those in one-way ANOVA and two-way ANOVA for block designs.

```
> Fitted=fit1$fitted;
> Resd=fit1$residuals;
> IQ =as.data.frame(cbind(IQ, Fitted, Resd)); #update the data by adding
fitted values and residuals
> str(IQ);
'data.frame': 30 obs. of 5 variables:
 $ Score : int 85 90 107 85 100 78 107 90 83 101 ...
 $ Lead : Factor w/ 3 levels "High","Low","Medium": 2 2 2 2 2 3 3 3 3
3 ...
 $ Sex : Factor w/ 2 levels "F","M": 2 2 2 2 2 2 2 2 2 2 ...
 $ Fitted: num 93.4 93.4 93.4 93.4 93.4 91.8 91.8 91.8 91.8 91.8 ...
 $ Resd : num -8.4 -3.4 13.6 -8.4 6.6 ...
```

- (a)** Independence and constant variance assumptions. The scatter plot of fitted values versus residuals is obtained using the updated data set.

From the scatter plot, it can be seen that the residuals are randomly distributed and thus no pattern can be found. Therefore, the independence assumption seems reasonable. Moreover, we can see that the spread of the residuals does not have an increasing or decreasing pattern. The constant variance assumption seems valid.

```
> plot(Resd~ Fitted, data= IQ);
```



Now we use Levene test to further check the constant variance assumptions. That is, we are testing

$$H_0: \sigma_L^2 = \sigma_M^2 = \sigma_H^2 \text{ versus } H_A: \text{Not all variances are equal}$$

and

$$H_0: \sigma_M^2 = \sigma_F^2 \text{ versus } H_A: \text{Not all variances are equal}$$

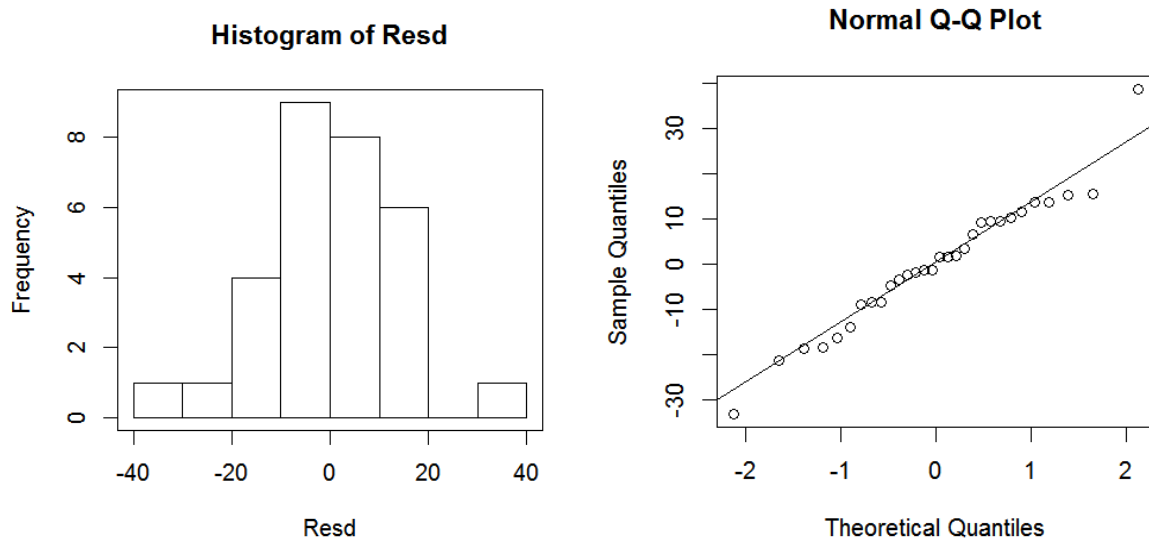
```
> library(car);
> leveneTest(Resd ~ Lead, data=IQ);
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  2  1.6661 0.2078
27
```

```
> leveneTest(Resd ~ Sex, data=IQ);
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  1  1.9607 0.1724
28
```

We fail to reject both null hypotheses because the p-values of both tests are large.

(b) Normality. Normality assumption can be checked using histogram and QQ plot of the residuals.

```
> hist(Resd); box();
> qqnorm(Resd); qqline(Resd);
```



From the histogram of the residuals, it can be seen that the shape is about bell-shaped. Furthermore, the QQ plot seems to be supporting Normality. Therefore, we conclude the normality assumption is reasonable.

Again, the statistical test of normality, Shapiro-Wilk normality test (a small p -value of the test results in rejection of Normality), can be very helpful in determining normality.

```
> shapiro.test(Resd);
```

Shapiro-Wilk normality test

data: Resd

W = 0.97069, p-value = **0.5581**

From the Shapiro-Wilk normality test, we fail to reject Normality as well because the p -value of the test is large.

21. LINEAR REGRESSION MODELS

In this chapter, we consider data analysis using simple linear regression and multiple linear regression models by examples using R.

21.1 Linear correlation

Example. Refer to Data Set **IQBRAIN.csv** and use paired data consisting brain volume (VOL) and IQ score (IQ). Construct a scatter plot and find the value of the linear correlation coefficient and determine if there is sufficient evidence to support the claim of a linear correlation between the two variables.

```
> IQBRAIN
=read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/IQBRAIN.csv",
header=TRUE, sep=",");
> str(IQBRAIN);
'data.frame': 20 obs. of 9 variables:
 $ PAIR : int 1 1 2 2 3 3 4 4 5 5 ...
 $ SEX  : int 2 2 2 2 2 2 2 2 2 2 ...
 $ ORDER: int 1 2 1 2 1 2 1 2 1 2 ...
 $ IQ   : int 96 89 87 87 101 103 103 96 127 126 ...
 $ VOL  : int 1005 963 1035 1027 1281 1272 1051 1079 1034 1070 ...
 $ AREA : num 1914 1685 1902 1860 2264 ...
 $ CCSA : num 6.08 5.73 6.22 5.8 7.99 8.42 7.44 6.84 6.48 6.43 ...
 $ CIRC : num 54.7 54.2 53 52.9 57.8 56.9 56.6 55.3 53.1 54.8 ...
 $ WT   : num 57.6 59 64.2 58.5 64 ...
```

We first construct a scatter plot.

```
> plot(IQ~VOL, data=IQBRAIN);
```

The following scatter-plot will be shown in the R graph window. It can be seen that the **linear** pattern between the two variables is very **weak**.

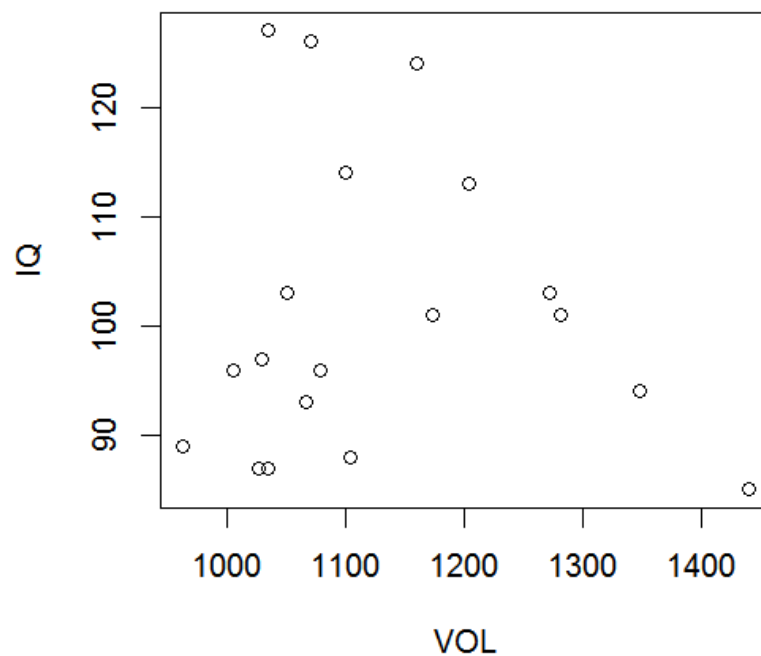
Let ρ be the population correlation coefficient between the two variables. We consider the two-sided test

$$H_0: \rho = 0 \text{ versus } H_a: \rho \neq 0.$$

```
> cor(IQBRAIN$IQ, IQBRAIN$VOL);
[1] -0.06339202
```

The Pearson correlation coefficient is -0.06339202 by the output.

```
> cor.test(IQBRAIN$IQ, IQBRAIN$VOL, alternative="two.sided",
method="pearson");
```



In the following output, the Pearson correlation coefficient is calculated and the p -value of the test is given.

Pearson's product-moment correlation

```
data: IQBRAIN$IQ and IQBRAIN$VOL
t = -0.26949, df = 18, p-value = 0.7906
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.4921080  0.3900712
sample estimates:
      cor
-0.06339202
```

21.2 Understanding Simple Linear Regression Models

Assume that a visual examination of the scatter plot confirms that the points $(x_1, y_1), \dots, (x_n, y_n)$ approximate a straight-line pattern

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

This model is called a **deterministic** mathematical model because it does not allow for any error in predicting y as a function of x .

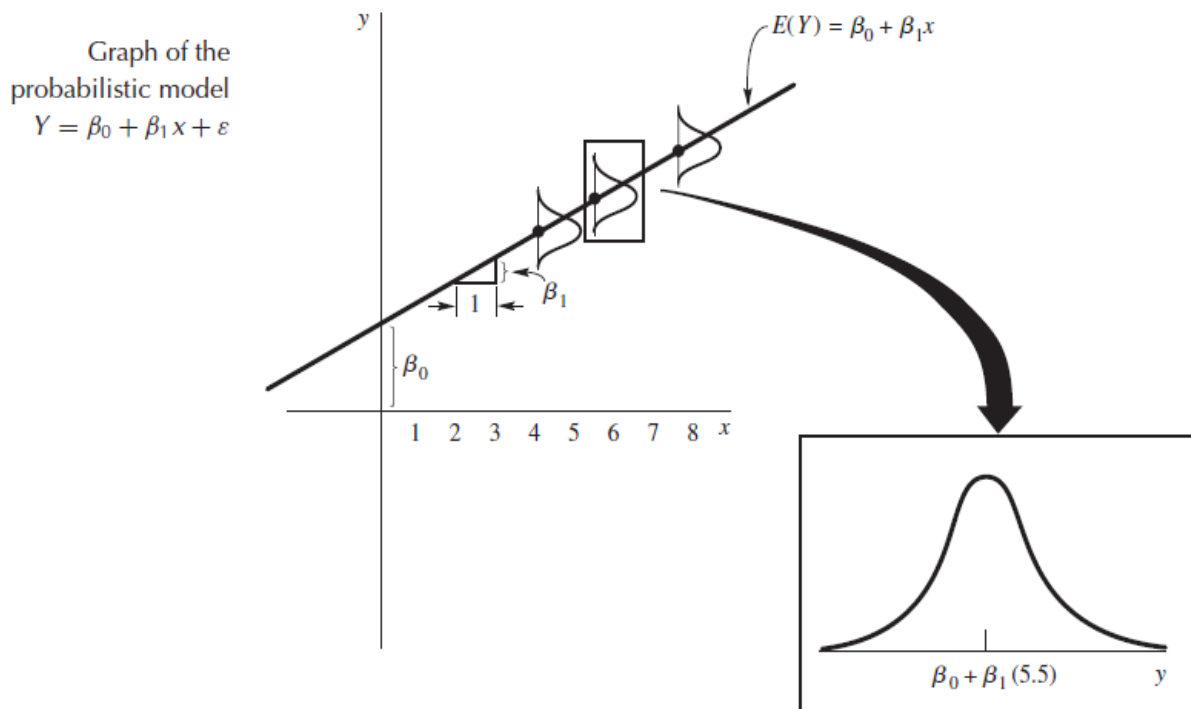
However, the bivariate measurements that we observe do not generally fall exactly on a straight line, we choose to use a probabilistic model: for any fixed value of $x = x_i$,

$$E(Y_i) = \beta_0 + \beta_1 x_i, i=1, \dots, n.$$

Furthermore, the distribution of each Y_i is specified as

$$E(Y_i | X=x_i) = \beta_0 + \beta_1 x_i + \varepsilon_i, i=1, \dots, n,$$

where each ε_i is assumed to have a probability distribution with mean 0. In general, we assume that ε_i 's are independent normal random variables with common variance σ^2 .



We use a simulation study in this section to understand simple linear regression models. In the scatterplot of the simulated data, the line is the true line of means, $y = \beta_0 + \beta_1 x$. We estimate the population parameters β_0 and β_1 using sample information.

Simulation Scheme

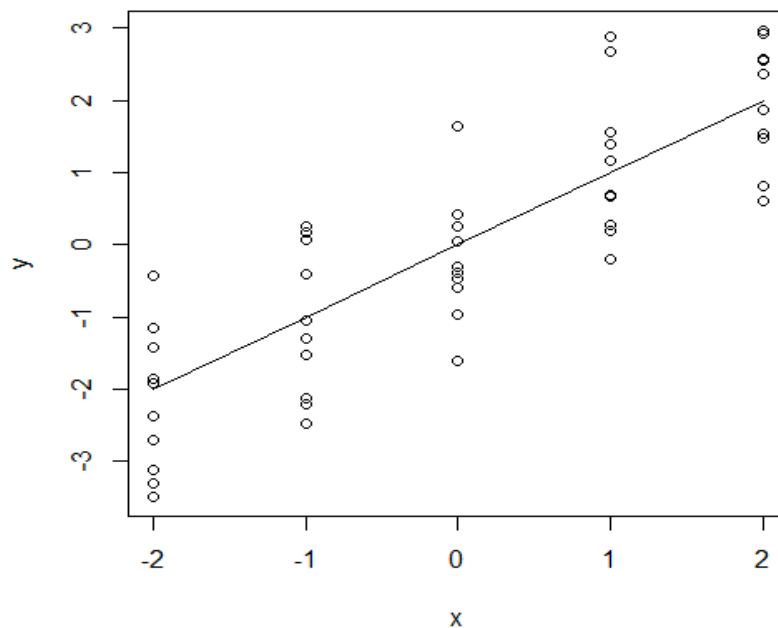
- Specify the parameter values: $\beta_0=0$, $\beta_1=1$, and $\sigma=1$.
- Specify 5 values of the independent variable x : -2, -1, 0, 1, 2.
- Choose a sample size $n=5 \times 10=50$;
- Generate a sample of size $k=50/5=10$, from each of the 5 populations
- Scatterplot of (x, y)
- Plot the line of means: $y = \beta_0 + \beta_1 x$

R code:

```

beta0=0;  beta1=1;  xi=c(-2, -1, 0, 1, 2);
x=numeric(); y=numeric();
k=10;
for (i in 1:5)
{
  x = rbind(x, matrix(rep(xi[i],k), k,1));
  yi = rnorm(k, mean=beta0+beta1*xi[i], sd=1);
  y = rbind(y, matrix(yi,k,1));      # Data generation
}
plot(x, y);                          #scatterplot
lines(x, beta0+beta1*x);             #plot the line of means

```



21.3 Analysis of Simple Linear Regression Models

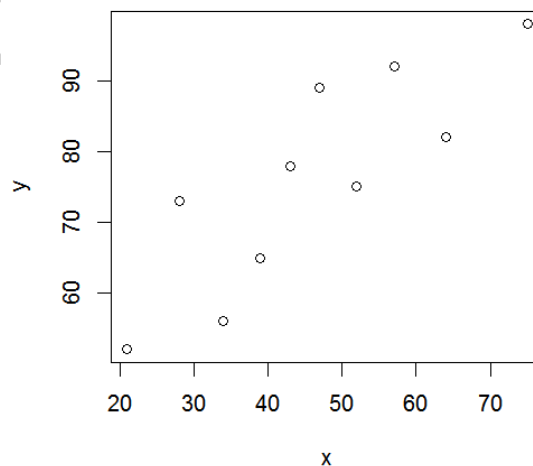
Example. Fit a simple linear regression model using the data, where y is the response variable and x is the explanatory variable (**slrex.txt**).

x	y
39	65
43	78
21	52
64	82
57	92
47	89
28	73
75	98
34	56
52	75

Before a simple linear regression model is fitted to the data, a scatter plot should be constructed to see if there is a linear relationship between the two variables. If there is no linear relationship, a simple linear regression model should not be developed.

```
> slrex
=read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/slrex.txt",
header=TRUE, sep=" ");
> str(slrex);
'data.frame':  10 obs. of  2 variables:
 $ x: int  39 43 21 64 57 47 28 75 34 52
 $ y: int  65 78 52 82 92 89 73 98 56 75

> plot(y~x,data= slrex);
```



From the scatter plot, it can be seen that there is a negative linear relationship between x and y . Before fitting a linear regression model using the least squares method, we obtain the least absolute error estimates of β_0 and β_1 , respectively. The following is the R code.

Step 1. We define a function which returns the sum of absolute errors. The function contains the two unknown parameters b_0 and b_1 .

```
SAE <- function(theta)
{
  b0 = theta[1];
  b1 = theta[2];
  x = slrex$x;
  y = slrex$y;
  sae = sum(abs(y-b0-b1*x));
  return(sae);
}
```

Step 2. We obtain the minimizer for the SAE function using the `optim` function in R.

```
> start =c(1,1);    #initial values
> optim(par=start, fn=SAE);
$par
[1] 34.1111097  0.8518518

$value
[1] 65.92593

$counts
function gradient
      243      NA

$convergence
[1] 0

$message
NULL)
```

Therefore, the least absolute error estimates of β_0 and β_1 are 34.111 and 0.852, respectively

Now we fit the linear regression model using the `lm()` function.

Model Fit.

```
> RegModel1 <- lm(y~x, data=slrex);
> names(RegModel1); #these are attributes available in the lm object
[1] "coefficients" "residuals"      "effects"      "rank"
[5] "fitted.values" "assign"          "qr"           "df.residual"
[9] "xlevels"      "call"           "terms"        "model"
> RegModel1Summary=summary(RegModel1);
> RegModel1Summary;
```

```
Call:
lm(formula = y ~ x, data = slrex)
```

Residuals:

Min	1Q	Median	3Q	Max
-10.813	-5.629	-2.531	6.758	12.234

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	40.7842	8.5069	4.794	0.00137 **
x	0.7656	0.1750	4.375	0.00236 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.704 on 8 degrees of freedom

Multiple R-squared: 0.7052, Adjusted R-squared: 0.6684

F-statistic: 19.14 on 1 and 8 DF, p-value: 0.002365

```
> coef(RegModel1);
(Intercept)          x
 40.7841552    0.7655618
```

The F test statistic is stored in the `fstatistic` component of the summary object.

```
> RegModel1Summary$fstatistic;
      value      numdf      dendf
19.14076    1.00000    8.00000
```


Now, the estimates of the intercept β_0 and slope β_1 are given. Furthermore, the test statistics and the p-values of the hypothesis tests

$H_0: \beta_0=0$ vs $H_a: \beta_0 \neq 0$ and $H_0: \beta_1=0$ vs $H_a: \beta_1 \neq 0$ are given, respectively.

To get confidence intervals for the parameters we use `confint()` function. For example, to get the 99% confidence intervals of the parameters,

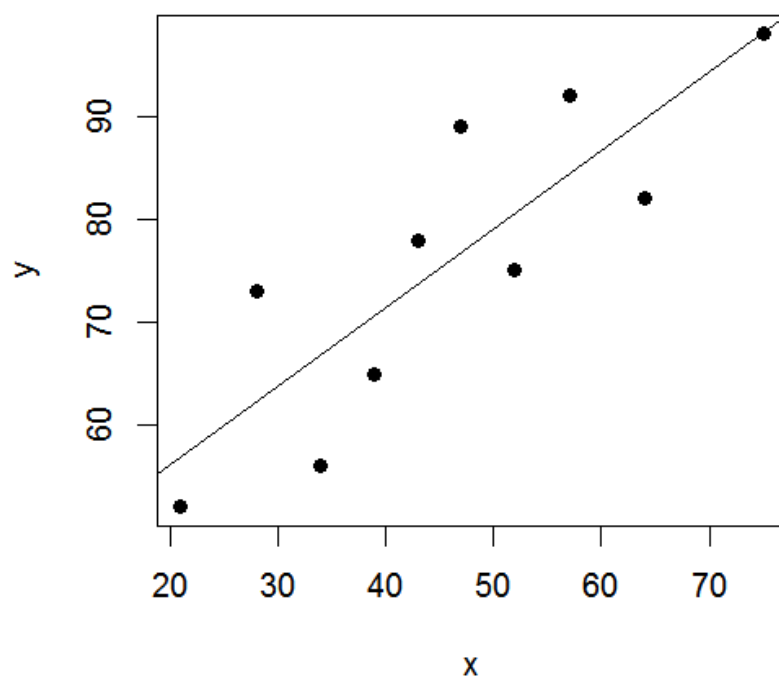
```
> confint(RegModel1, level = 0.99);
```

	0.5 %	99.5 %
(Intercept)	12.2403403	69.327970
x	0.1784195	1.352704

The coefficient of determination, R^2 , is equal to 0.7052.

The F-test statistic for the test $H_0: \beta_1=0$ vs $H_a: \beta_1 \neq 0$ is 19.14 and the p-value of the test is 0.002365.

```
> plot(y ~ x, data = slrex, pch = 16);
> abline(coef(RegModel1)); #scatter plot and regression line plot
```



ANOVA Table. The following is the ANOVA table of the test.

```
> anova(RegModel1);
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value    Pr(>F)
x           1 1449.97  1449.97   19.141 0.002365 **
Residuals   8   606.03    75.75
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
However, it is noticed that the totals are not given. We should manually construct the complete ANOVA table as
```

Source of variation	Sum Sq	Df	Mean Sq	F value	Pr(>F)
x	1449.97	1	1449.97	19.141	0.002365
Residuals	606.03	8	75.75		
Total	2056	9			

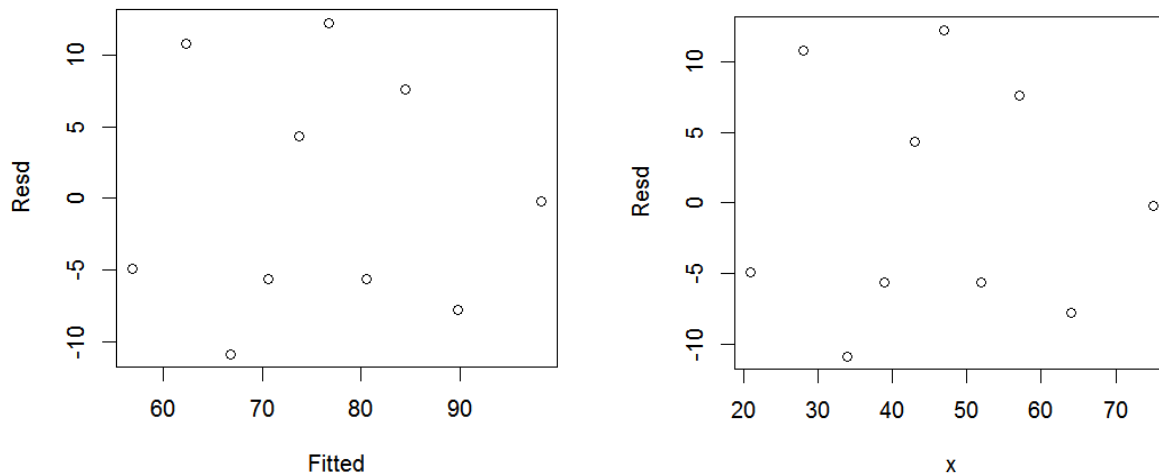
Model adequacy check. We need to check the fitted values and residuals to see if the three model assumptions are satisfied. As what we have done in ANOVA for experimental designs, we update the raw data set by adding the fitted values and residuals.

```
> Fitted= RegModel1$fitted;
> Resd= RegModel1$residuals;
> slrex =as.data.frame(cbind(slrex, Fitted, Resd));
> str(slrex)
'data.frame':  10 obs. of  4 variables:
 $ x      : int  39 43 21 64 57 47 28 75 34 52
 $ y      : int  65 78 52 82 92 89 73 98 56 75
 $ Fitted: num  70.6 73.7 56.9 89.8 84.4 ...
 $ Resd  : num  -5.64 4.3 -4.86 -7.78 7.58 ...
```

(a) Independence and constant variance assumptions.

The scatter plot of residuals versus fitted values is obtained using the updated data set. We can draw the scatter plot of residuals versus the independent variable as well. The two scatter plots are very similar.

```
> plot(Resd~ Fitted, data= slrex);
> plot(Resd~ x, data= slrex);
```



From these two scatter plots, it can be seen that the residuals are randomly distributed and thus no pattern can be found. Therefore, the independence assumption seems reasonable. Furthermore, we cannot see the spread of the residuals has decreasing or increasing trend. Constant variance assumptions is valid as well. Statistically, we are testing

$$H_0: \sigma_1^2 = \sigma_2^2 = \dots = \sigma_n^2 \text{ versus } H_A: \text{Not all variances are equal}$$

To check the constant variance assumption using a statistical test, the `ncvTest()` function in the `car` package (<https://cran.r-project.org/web/packages/car/index.html>) can be used.

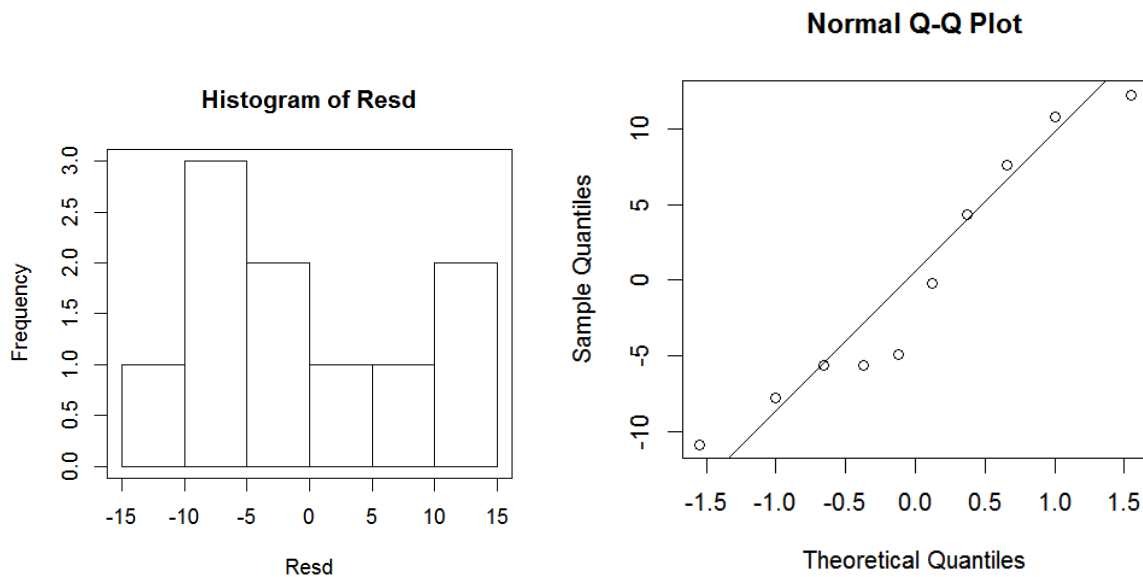
```
> library(car);
> ncvTest(RegModel1);
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 0.2920722    Df = 1    p = 0.5888957
```

We fail to reject the null hypothesis because the p-value of the test is large. The conclusion is consistent with that from the residual plot.

(b) Normality.

Normality assumption can be checked using histogram and QQ plot of the residuals.

```
> hist(Resd); box();
> qqnorm(Resd); qqline(Resd);
```



From the histogram of the residuals, it can be seen that the shape is approximately bell-shaped with some positive skewness and most points are close to the straight line in the QQ plot. Therefore, we conclude the normality assumption is reasonable.

We then conduct Shapiro-Wilk test for normality (Recall: a small p-value of the test results in rejection of Normality).

```
> shapiro.test(Resd);

      Shapiro-Wilk normality test

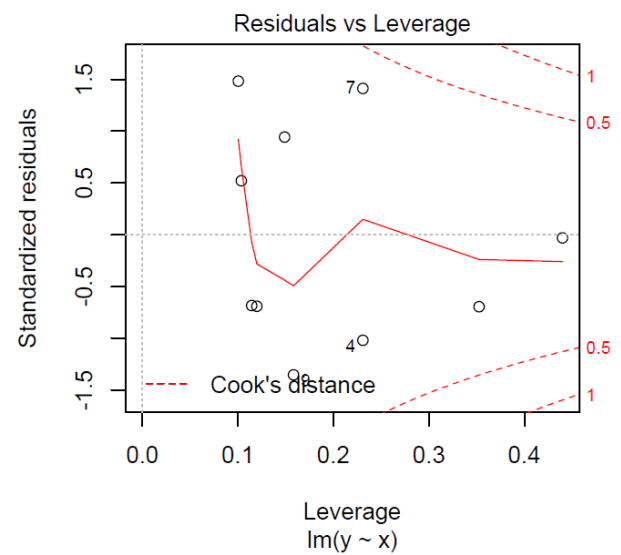
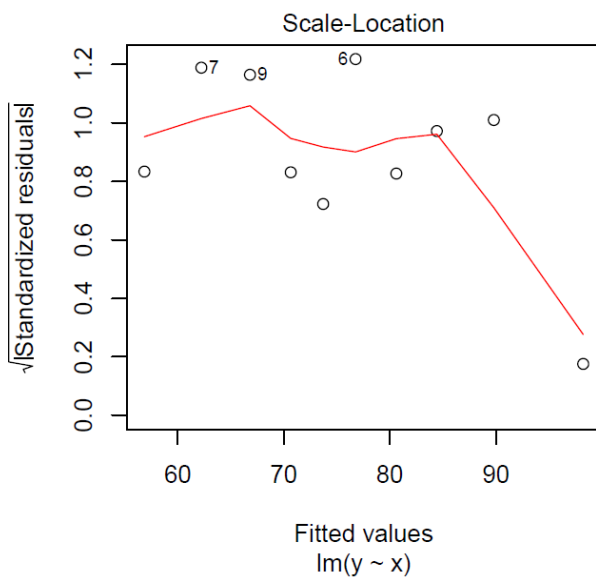
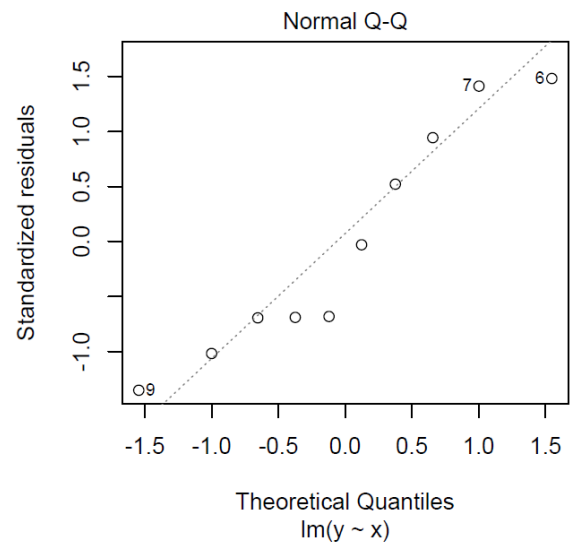
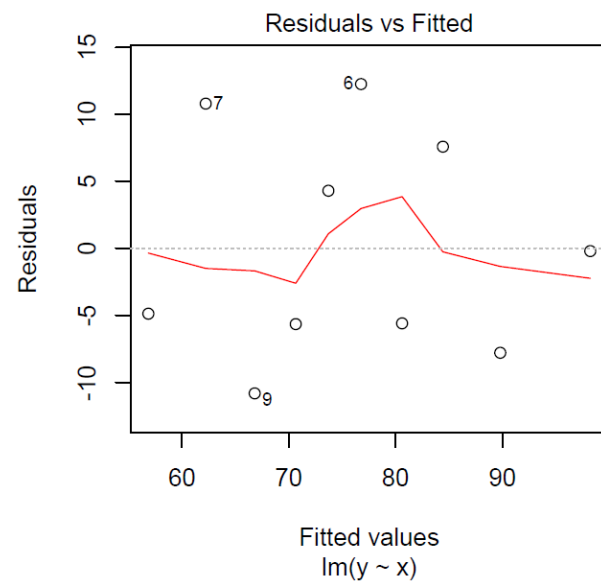
data:  Resd
W = 0.91619, p-value = 0.3263
```

From this test, we fail to reject Normality as well.

Remark. To get all diagnostic plots, we just need the code “plot(the lm object)”.

```
> plot(RegModel1);
```

After the above code is run, we will get the following four graphs.



Confidence intervals and prediction confidence intervals.

Suppose there are two future observation $x_1=50$ and $x_2=60$.

- (a) Find a 99% **confidence interval** of the average value of the y variable for the given x value $x_1=50$ and $x_2=60$, respectively.

We only need two lines of code to solve the problem.

```
new =data.frame( x=c(50, 60) ); #x is the symbol for the independent variable.
predict(RegModel1, newdata=new, interval="confidence", level=0.99);
```

Here, 0.99 is the level of confidence. We have the out put

	fit	lwr	upr
1	79.06225	69.53316	88.59133
2	86.71787	74.35437	99.08136

That is, the required 99% confidence intervals are (69.53316, 88.59133) and (74.35437, 99.08136), respectively.

- (b) Find a 99% **prediction confidence interval** of the true value of the y variable for the given x value $x_1=50$ and $x_2=60$, respectively.

We just need one more line of code to solve the problem:

```
predict(RegModel1, newdata=new, interval="prediction", level=0.99);
```

And the output is given by

	fit	lwr	upr
1	79.06225	48.34286	109.7816
2	86.71787	55.00457	118.4312

That is, the required 99% prediction confidence intervals are (48.34286, 109.7816) and (55.00457, 118.4312), respectively. It can be seen that prediction confidence intervals are much wider.

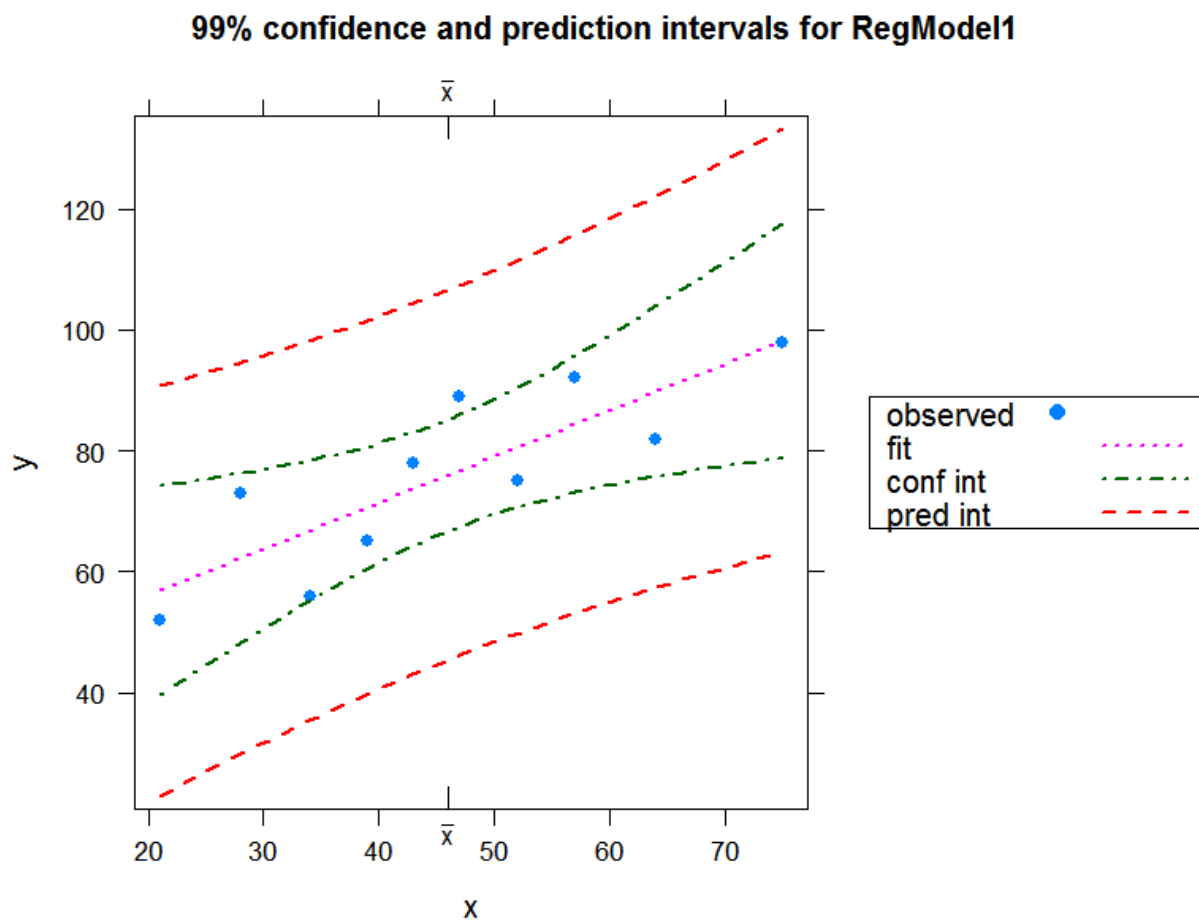
Note. We are entering a future observation of the independent variable. The notation x is the name of the independent variable.

Graphing the Confidence and Prediction Bands.

We expect the confidence intervals for the mean to be narrower than the prediction confidence intervals for a new observation. A close look at the standard deviations in the formulas of the two confidence intervals confirms this. We might want to plot a graph to see the difference of the two types of confidence intervals. We may plot the confidence and prediction intervals using the `ci.plot()` function from the HH package (<https://cran.r-project.org/web/packages/HH/index.html>).

```
> library(HH);
> ci.plot(RegModel1, conf.level=0.99);
```

Notice that the bands curve outward away from the regression line as the x values move away from the center. This is expected once we notice the $(x^* - \bar{x})^2$ in the standard deviation term in the formulas of the two confidence intervals.



21.4 Multiple Linear Regression Models

The multiple linear regression model is a generalization of the simple linear regression model by allowing for more than one independent variable. Data analysis using multiple linear regression models can be done similarly as in section 21.2.

Example. How do real estate agents decide on the asking price for a newly listed condominium? A computer database in a small community contains the listed selling price y (in thousands of dollars), the amount of living area x_1 (in thousands of square feet), and the number of floors x_2 , bedrooms x_3 , and bathrooms x_4 , for $n=15$ randomly selected condos currently on the market. The data are shown in the following table (**condo.txt**).

An investigator considered the multiple linear regression model

$$Y|x_1, x_2, x_3, x_4 = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \varepsilon,$$

where ε are iid $N(0, \sigma^2)$ distributed.

Observations	List Price (y)	Living Area (x_1)	# of Floors (x_2)	# of Bedrooms (x_3)	# of Baths (x_4)
1	69.0	0.56	1	2	1
2	118.5	0.93	1	2	2
3	116.5	0.93	1	3	2
4	125.0	1.02	1	3	2
5	129.9	1.21	1	3	1.7
6	135.0	1.21	2	3	2.5
7	139.9	1.21	1	3	2
8	147.9	1.58	2	3	2.5
9	160.0	1.77	2	3	2
10	169.9	1.67	1	3	2
11	134.9	1.21	1	4	2
12	155.0	1.67	1	4	2
13	169.9	1.58	2	4	3
14	194.5	1.86	2	4	3
15	209.9	0.95	2	4	3

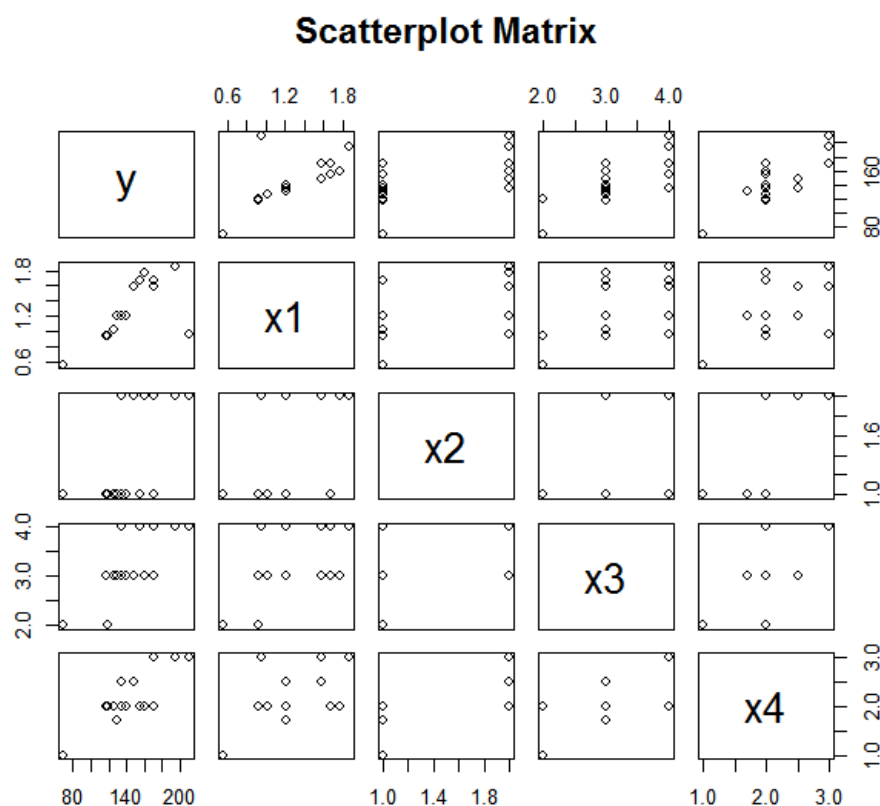

```

< condo
=read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/condo.txt",
header=TRUE, sep="");
> str(condo)
'data.frame': 15 obs. of 5 variables:
 $ y : num 69 118 116 125 130 ...
 $ x1: num 0.56 0.93 0.93 1.02 1.21 1.21 1.21 1.58 1.77 1.67 ...
 $ x2: int 1 1 1 1 1 2 1 2 2 1 ...
 $ x3: int 2 2 3 3 3 3 3 3 3 3 ...
 $ x4: num 1 2 2 2 1.7 2.5 2 2.5 2 2 ...

```

To see the relationship between any two variables, it is useful to have a look at the Scatter plot matrix.

```
> pairs(~y+x1+x2+x3+x4,data=condo, main="Scatterplot Matrix")
```



We fit the regression model proposed by the investigator and conduct relevant data analysis using R.

Model Fit.

```
> RegModel2 = lm(y~x1+x2+x3+x4, data=condo);
> RegModel2Summary=summary(RegModel2);
> RegModel2Summary;
```

Call:

```
lm(formula = y ~ x1 + x2 + x3 + x4, data = condo)
```

Residuals:

Min	1Q	Median	3Q	Max
-19.836	-11.832	-2.171	5.054	35.765

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.7991	23.9827	0.033	0.9741
x1	24.7647	15.6418	1.583	0.1444
x2	-3.3967	15.6390	-0.217	0.8324
x3	11.0755	10.7981	1.026	0.3292
x4	37.4335	17.7941	2.104	0.0617

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.83 on 10 degrees of freedom

Multiple R-squared: 0.806, Adjusted R-squared: 0.7284

F-statistic: 10.38 on 4 and 10 DF, p-value: 0.001383

summary(RegModel2) gives us the fitted model. That is, the estimates of the intercept β_0 and slopes $\beta_1, \beta_2, \beta_3$ and β_4 are given. Furthermore, the test statistics and the p-values of the hypothesis tests

$H_0: \beta_0=0$ vs $H_a: \beta_0 \neq 0$ and $H_0: \beta_i=0$ vs $H_a: \beta_i \neq 0$ ($i=1, 2, 3, 4$) are given.

The multiple coefficient of determination, R^2 , is equal to 0.806; and the adjusted R^2 is given by 0.7284.

The F-test statistic for the test of usefulness of the multiple linear regression model $H_0: \beta_1 = \beta_2 = \beta_3 = \beta_4 = 0$ vs H_a : At least one $\beta_i \neq 0$ ($i=1, 2, 3, 4$) is 10.38 and the p-value of the test is 0.001383.

To get some parts of the output, we can do the following.

```
> residuals(RegModel2)
```

1	2	3	4	5	6
-1.855134	1.048387	-12.027063	-5.755888	5.668876	-15.781290
7	8	9	10	11	12
4.438816	-12.044236	14.067234	23.047046	-11.636634	-2.928405
13	14	15			
-19.836453	-2.170574	35.765319			

```
> RegModel2Summary$coeff
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.7991185	23.98274	0.03332057	0.97407465
x1	24.7647179	15.64183	1.58323672	0.14444981
x2	-3.3966606	15.63901	-0.21719159	0.83242627
x3	11.0754502	10.79815	1.02568079	0.32921305
x4	37.4335336	17.79407	2.10370794	0.06169151

```
> RegModel2Summary$r.squared;
[1] 0.8059679
```

```
> RegModel2Summary$adj.r.squared;
[1] 0.7283551
```

The F test statistic is stored in the `fstatistic` component of the summary object as well.

```
> RegModel2Summary$fstatistic;
      value      numdf      dendf
10.38447    4.00000  10.00000
```

To get confidence intervals for the parameters we use `confint()` function. For example, to get the 99% confidence intervals of the parameters,

```
> confint(RegModel2, level = 0.99);
```

	0.5 %	99.5 %
(Intercept)	-75.20872	76.80696
x1	-24.80850	74.33794
x2	-52.96094	46.16762
x3	-23.14682	45.29772
x4	-18.96074	93.82780

ANOVA Table.

Next, we get the ANOVA table.

```
> anova(RegModel2);
```

Analysis of Variance Table

Response: y

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
x1	1	6965.8	6965.8	21.9141	0.0008659 ***
x2	1	2007.7	2007.7	6.3162	0.0307433 *
x3	1	2823.3	2823.3	8.8819	0.0138016 *
x4	1	1406.7	1406.7	4.4256	0.0616915 .
Residuals	10	3178.7	317.9		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

However, it is noticed that the totals are not given. We should manually construct the complete ANOVA table as

Source of variation	SS	df	MS	F	p-value
Regression	13203.5	4	3300.875	10.384	?
Residual	3178.7	10	317.87		
Total	16382.2	14			

Here, $13203.5 = 6965.8 + 2007.7 + 2823.3 + 1406.7$ and the p-value of the F-test is given by `summary(RegModel2)` which is 0.001383.

Model Adequacy Check.

We need to check the fitted values and residuals to see if the three model assumptions are satisfied. As what we have done for simple linear regression models, we update the raw data set by adding the fitted values and residuals.

```
> Fitted= RegModel2$fitted;
> Resd= RegModel2$residuals;
> condo =as.data.frame(cbind(condo, Fitted, Resd));
> str(condo);
```

'data.frame': 15 obs. of 7 variables:

\$ y : num 69 118 116 125 130 ...

```

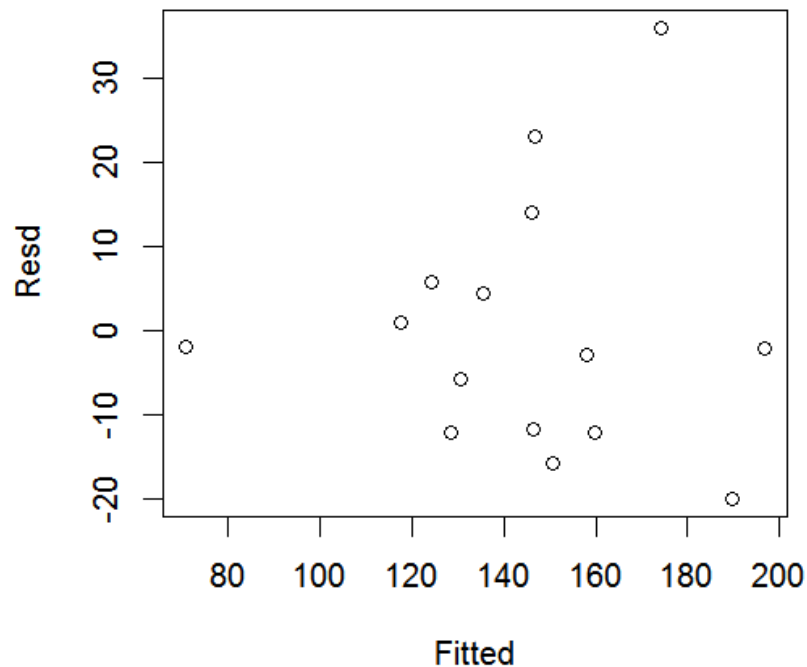
$ x1      : num  0.56 0.93 0.93 1.02 1.21 1.21 1.21 1.58 1.77
1.67 ...
$ x2      : int   1 1 1 1 1 2 1 2 2 1 ...
$ x3      : int   2 2 3 3 3 3 3 3 3 3 ...
$ x4      : num   1 2 2 2 1.7 2.5 2 2.5 2 2 ...
$ Fitted: num  70.9 117.5 128.5 130.8 124.2 ...
$ Resd    : num  -1.86 1.05 -12.03 -5.76 5.67 ...

```

(a) **Independence and constant variance assumptions.**

We now draw the scatter plot of residuals versus the fitted values.

```
> plot(Resd~ Fitted, data=condo);
```



From the scatter plot, it can be seen that the residuals are randomly distributed and thus no pattern can be found. Therefore, the independence assumption seems reasonable. However, we can see that the spread of the residuals has an increasing trend. The constant variance assumption seems invalid. Therefore, **weighted least squares** method should be used to estimate the parameters in the model. Further discussions will not be given in this manual.

Statistically, we are testing the constant variance

$$H_0: \sigma_1^2 = \sigma_2^2 = \dots = \sigma_n^2 \text{ versus } H_A: \text{Not all variances are equal}$$

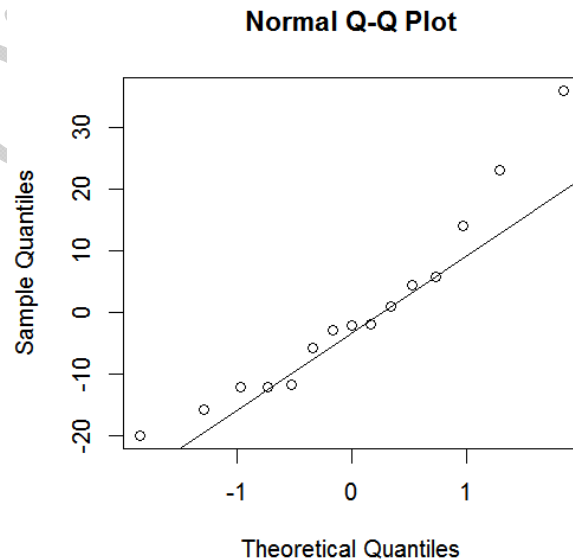
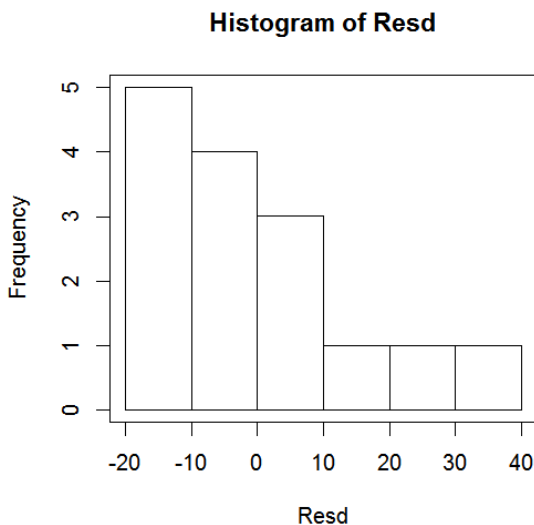
Again, we use the `ncvTest()` function in the `car` package to check the constant variance assumption.

```
> library(car);
> ncvTest(RegModel2);
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 2.648764    Df = 1    p = 0.1036304
```

(b) Normality.

Normality assumption can be checked using histogram and QQ plot of the residuals.

```
> hist(Resd); box();
> qqnorm(Resd); qqline(Resd);
```



From the histogram of the residuals, it can be seen that the shape is not very bell-shaped. But the QQ plot seems to be supporting Normality. Therefore, we conclude the normality assumption is reasonable.

We then conduct Shapiro-Wilk test for normality (Recall: a small p-value of the test results in rejection of Normality).

```
> shapiro.test(Resd);
```

Shapiro-Wilk normality test

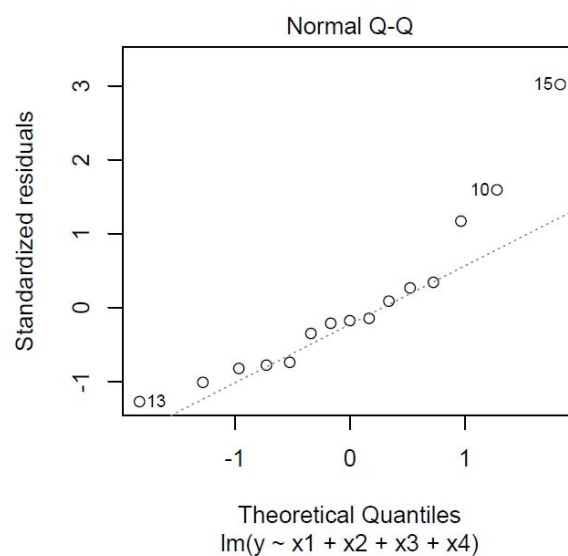
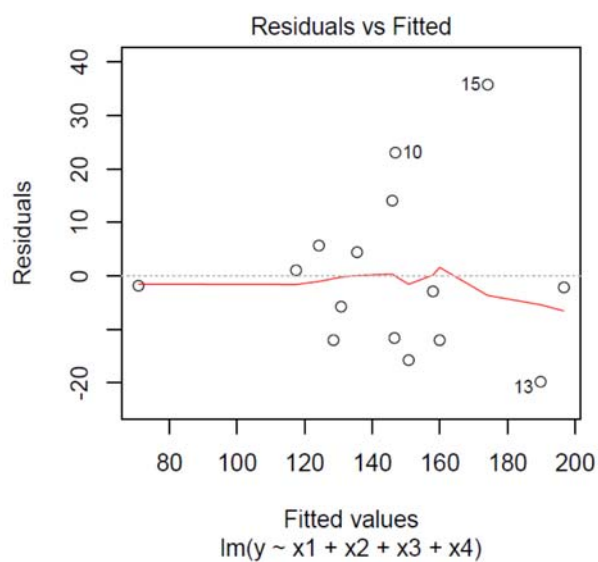
```
data: Resd
W = 0.92187, p-value = 0.2057
```

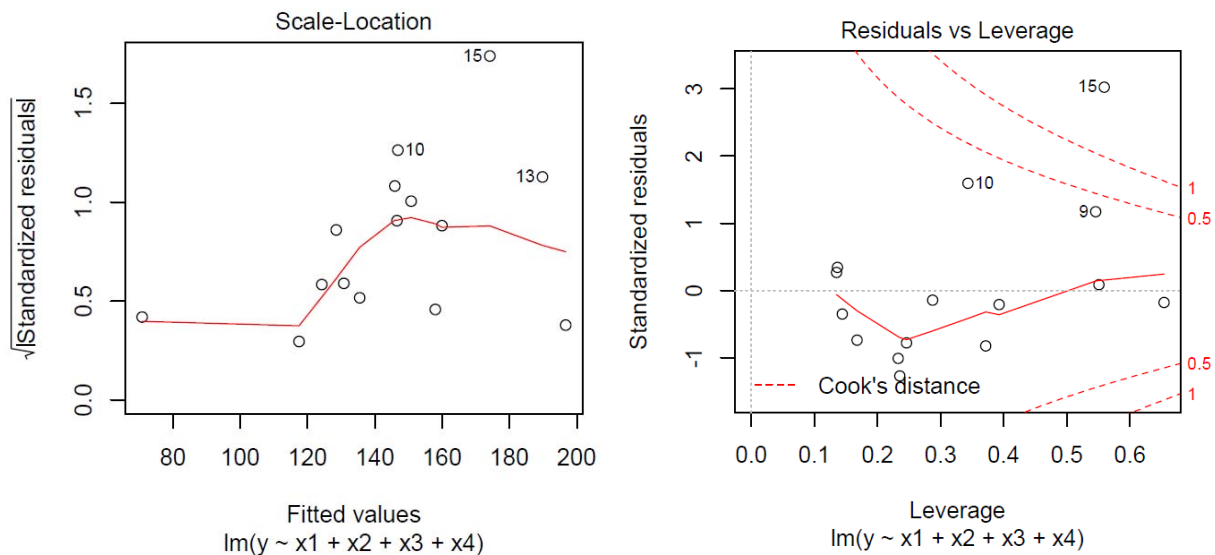
From this test, we fail to reject Normality as well.

Remark. To get all diagnostic plots, we just need the code “plot(the lm object)”.

```
> plot(RegModel2);
```

After the above code is run, we will get the following four graphs.





Confidence intervals and prediction confidence intervals.

Suppose there is a future observation of living area $x_1=1.75$, number of floors $x_2=2$, number of bedrooms $x_3=4$ and number of bathrooms $x_4=2.5$.

- (a) Find a 95% confidence interval of the average price of these condos.

We need two lines of code to solve the problem:

```
new =data.frame(x1=1.75, x2=2, x3=4, x4=2.5);
predict(RegModel2, newdata=new, interval="confidence", level=0.95);
```

Here, 0.95 is the level of confidence. The output is given by

	fit	lwr	upr
1	175.2297	151.9813	198.4781

That is, the required 95% confidence interval is (151.9813, 198.4781).

- (b) Find a 95% prediction confidence interval of the true price of a condo with given living area $x_1=1.75$, number of floors $x_2=2$, number of bedrooms $x_3=4$ and number of bathrooms $x_4=2.5$.

We need one more line of code to solve the problem:

```
predict(RegModel2, newdata=new, interval="prediction", level=0.95);
```

And the output is given by

	fit	lwr	upr
1	175.2297	129.2018	221.2576

That is, the required 95% prediction confidence interval is (129.2018, 221.2576). It can be seen that the prediction confidence interval is much wider.

The best multiple regression model.

Consider the **general form** of a multiple linear regression (MLR) model given by

$$Y_i | x_1=x_{1i}, \dots, x_p=x_{pi} = \beta_0 + \beta_1 Z_1(x_{1i}, \dots, x_{pi}) + \dots + \beta_k Z_k(x_{1i}, \dots, x_{pi}) + \varepsilon_i, i = 1, \dots, n,$$

where Y = response/dependent variable, X_1, \dots, X_p are p independent variables and the **regressors** $Z_1(X_1, \dots, X_p), \dots, Z_k(X_1, \dots, X_p)$ are k functions/transformations of X_1, \dots, X_p , $\beta_0, \beta_1, \dots, \beta_k$ are the regression model parameters, and ε_i 's are iid random error terms.

Once the k regressors are determined, the most important question then is the selection of the best regression model among the 2^k possible models. We consider stepwise methods to select a best model. In general, the following 4 stepwise selection methods will be considered:

- Backward elimination;
- Forward selection;
- Backward stepwise (Backward/Forward);
- Forward stepwise (Forward/Backward).

We consider the `stepAIC()` function in the MASS package (<https://cran.r-project.org/web/packages/MASS/index.html>) in which the first three methods are available..

For example, we use the forward selection method.

```
> library(MASS);
> stepAIC(RegModel2, direction="forward");
Start:  AIC=90.34
y ~ x1 + x2 + x3 + x4
```

Call:

```
lm(formula = y ~ x1 + x2 + x3 + x4, data = condo)
```

Coefficients:

(Intercept)	x1	x2	x3	x4
0.7991	24.7647	-3.3967	11.0755	37.4335

We use the backward selection method:

```
> stepAIC(RegModel2, direction="backward");
Start:  AIC=90.34
y ~ x1 + x2 + x3 + x4
```

	Df	Sum of Sq	RSS	AIC
- x2	1	14.99	3193.7	88.413
- x3	1	334.40	3513.1	89.843
<none>			3178.7	90.343
- x1	1	796.78	3975.4	91.698
- x4	1	1406.75	4585.4	93.839

Step: AIC=88.41

y ~ x1 + x3 + x4

	Df	Sum of Sq	RSS	AIC
- x3	1	443.42	3637.1	88.363
<none>			3193.7	88.413
- x1	1	795.46	3989.1	89.749
- x4	1	2576.78	5770.4	95.287

Step: AIC=88.36

y ~ x1 + x4

	Df	Sum of Sq	RSS	AIC
<none>			3637.1	88.363
- x1	1	1362.4	4999.4	91.135
- x4	1	5779.3	9416.4	100.632

Call:

lm(formula = y ~ x1 + x4, data = condo)

Coefficients:

(Intercept)	x1	x4
13.70	29.59	42.74

We use the Stepwise Backward selection method:

```
> stepAIC(RegModel2, direction="both");
```

Start: AIC=90.34

y ~ x1 + x2 + x3 + x4

	Df	Sum of Sq	RSS	AIC
- x2	1	14.99	3193.7	88.413
- x3	1	334.40	3513.1	89.843
<none>			3178.7	90.343
- x1	1	796.78	3975.4	91.698
- x4	1	1406.75	4585.4	93.839

Step: AIC=88.41

y ~ x1 + x3 + x4

	Df	Sum of Sq	RSS	AIC
- x3	1	443.42	3637.1	88.363
<none>			3193.7	88.413
- x1	1	795.46	3989.1	89.749
+ x2	1	14.99	3178.7	90.343
- x4	1	2576.78	5770.4	95.287

Step: AIC=88.36

y ~ x1 + x4

	Df	Sum of Sq	RSS	AIC
<none>			3637.1	88.363
+ x3	1	443.4	3193.7	88.413
+ x2	1	124.0	3513.1	89.843
- x1	1	1362.4	4999.4	91.135
- x4	1	5779.3	9416.4	100.632

Call:

lm(formula = y ~ x1 + x4, data = condo)

Coefficients:

(Intercept)	x1	x4
13.70	29.59	42.74

It can be seen that the forward selection method leads to the full model. For both Backward Elimination method and Backward Stepwise method, we start with the full model

$$E(Y | x_1, x_2, x_3, x_4) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4,$$

and end with a final model

$$E(Y | x_1, x_4) = \beta_0 + \beta_1 x_1 + \beta_4 x_4.$$

22. NONPARAMETRIC TESTS

In this chapter, we consider nonparametric tests by examples using R.

22.1 Sign Test

Sign test is essentially an exact binomial test. It is used to decide whether a binomial distribution has the equal chance of success and failure.

Example. The following table includes taxi-out times and taxi-in times for a sample of American Airlines Flight 21. Use the sign test to test the claim that there is no difference between taxi-out and taxi-in times.

Table 13-3 Taxi-Out Times and Taxi-In Times for American Airlines Flight 21

Taxi-out time (min)	13	20	12	17	35	19	22	43	49	45	13	23
Taxi-in time (min)	13	4	6	21	29	5	27	9	12	7	36	12
Sign of difference	0	+	+	-	+	+	-	+	+	+	-	+

We have 8 positive signs, 3 negative signs, and 1 difference of 0. The test of equality of medians is equivalent to testing whether or not the numbers of positive and negative signs are approximately equal.

The hypotheses are:

H_0 : The median of the differences is equal to 0

H_a : The median of the differences is not equal to 0

Here, we have $n=11$ and $x=3$. In R command prompt we write

```
binom.test(x=3, n=11, p = 0.5, alternative = c("two.sided"));
```

The following is the output.

```
Exact binomial test
```

```
data: 3 and 11
```

```
number of successes = 3, number of trials = 11, p-value = 0.2266
```

```
alternative hypothesis: true probability of success is not equal  
to 0.5
```

```
95 percent confidence interval:
```

```
0.06021773 0.60974256
```

```
sample estimates:
```

```
probability of success
```

```
0.2727273
```

22.2 Wilcoxon Signed-Rank Test

The Wilcoxon Signed-Rank Test, compared to Sign test, is a more powerful nonparametric procedure that can be used to test one population median or compare two population medians when the samples consist of paired observations

Example. Again, consider the example of taxi-out times and taxi-in times for a sample of American Airlines Flight 21. Use the Wilcoxon Signed-Rank test to test the claim that there is no difference between taxi-out and taxi-in times.

After importing the data set into R in the format of

Taxiout	Taxiin
13	13
20	4
12	6
17	21
35	29
19	5
22	27
43	9
49	12
45	7
13	36
23	12,

See taxi.csv.

```
> taxi
=read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/taxi.csv",
header=TRUE, sep=",");
```

If we choose “Default” type of test in the Options tab, we have the following output.

```
> Taxiout = taxi$Taxiout;
> Taxiin = taxi$Taxiin;
> median(Taxiout - Taxiin, na.rm=TRUE); # median difference
[1] 8.5
```

```
> wilcox.test(Taxiout, Taxiin, alternative='two.sided', paired=TRUE);
```

Wilcoxon signed rank test with continuity correction

data: Taxiout and Taxiin

V = 55, p-value = 0.05581

alternative hypothesis: true location shift is not equal to 0

Warning messages:

1: In wilcox.test.default(Taxiout, Taxiin, alternative = "two.sided", :
cannot compute exact p-value with ties

2: In wilcox.test.default(Taxiout, Taxiin, alternative = "two.sided", :
cannot compute exact p-value with zeroes

If we choose “Normal approximation” type of test, then we have

```
> wilcox.test(Taxiout, Taxiin, alternative='two.sided', correct=FALSE,  
exact=FALSE, paired=TRUE);
```

Wilcoxon signed rank test

data: Taxiout and Taxiin

V = 55, p-value = 0.05035

alternative hypothesis: true location shift is not equal to 0

22.3 Wilcoxon Rank-Sum Test

The Wilcoxon rank-sum test is a nonparametric test that uses ranks of sample data from two independent populations. It is used to test the null hypothesis H_0 that the two independent samples come from populations with equal medians.

Example. The following table lists pulse rates of samples of males and females (**pulserates.csv**). Test the claim that males and females have the same median pulse rate.

rates	gender
60	M
74	M
86	M
54	M
90	M
80	M
66	M
68	M
68	M
56	M
80	M
62	M
78	F
80	F
68	F
56	F
76	F
78	F
78	F
90	F
96	F
60	F
98	F

```
> pulse
=read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/pulserates
.csv", header=TRUE, sep=",");
> rates= pulse$rates; gender= pulse$gender;
> tapply(rates, gender, median, na.rm=TRUE);
  F  M
78 68

> wilcox.test(rates ~ gender, alternative="two.sided", data=pulse)
```

Wilcoxon rank sum test with continuity correction

```
data: rates by gender
W = 86.5, p-value = 0.2166
alternative hypothesis: true location shift is not equal to 0
```

Warning message:

```
In wilcox.test.default(x = c(78L, 80L, 68L, 56L, 76L, 78L, 78L,
cannot compute exact p-value with ties
```

22.4 Kruskal-Wallis Test

The Kruskal-Wallis Test is a nonparametric procedure that can be used to compare more than two population medians in a completely randomized design.

Example. The following table lists IQ scores from a sample of subjects with low, medium, and high lead exposure (**IQScores.csv**). Test the claim that the three sample medians come from populations with medians that are all equal.

Scores	LeadLevel
85	L
90	L
107	L
85	L
100	L
97	L
101	L
64	L
78	M
97	M
107	M
80	M
90	M
83	M
93	H
100	H
97	H
79	H
97	H

```
> IQ
=read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/IQScores.csv", header=TRUE, sep=",");
> Scores = IQ$Scores; LeadLevel = IQ$LeadLevel;

> tapply(Scores, LeadLevel, median, na.rm=TRUE)
      H      L      M
97.0  93.5  86.5
```



```
> kruskal.test(Scores ~ LeadLevel, data=IQ);
```

```
Kruskal-Wallis rank sum test
```

```
data: Scores by LeadLevel
```

```
Kruskal-Wallis chi-squared = 0.70311, df = 2, p-value = 0.7036
```

22.5 Friedman Rank-Sum Test

The Friedman rank-sum test is a nonparametric procedure that can be used to compare more than two population medians in a randomized complete block design.

Consider one example analyzed using the parametric method.

Example.

We want to investigate the effect of 3 methods of soil preparation on the growth of seedlings. Each method is applied to seedlings growing at each of 4 locations and the average first year growth is recorded. Is the average growth different for the 3 soil preps?

	Location			
Soil Prep	1	2	3	4
A	11	13	16	10
B	15	17	20	12
C	10	15	13	10

Since we are comparing three methods of soil preparation, we need the data in this format.

A	B	C
11	15	10
13	17	15
16	20	13
10	12	10

```

> Soil
=read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/Soil.csv",
header=TRUE, sep=",");
> x1=subset(Soil, soil=="A");
> x2=subset(Soil, soil=="B");
> x3=subset(Soil, soil=="C");
> x1
  growth soil location
1      11    A        1
4      13    A        2
7      16    A        3
10     10    A        4
> x2
  growth soil location
2      15    B        1
5      17    B        2
8      20    B        3
11     12    B        4
> x3
  growth soil location
3      10    C        1
6      15    C        2
9      13    C        3
12     10    C        4
> MyData = as.table(cbind(x1$growth, x2$growth, x3$growth));
> dimnames(MyData)=list(location=c("1","2","3","4" ),
method=c("A","B","C"));
> MyData;
      method
location A  B  C
1  11 15 10
2  13 17 15
3  16 20 13
4  10 12 10

> friedman.test(MyData);
      Friedman rank sum test

data:  MyData
Friedman chi-squared = 6.5333, df = 2, p-value = 0.03813

```

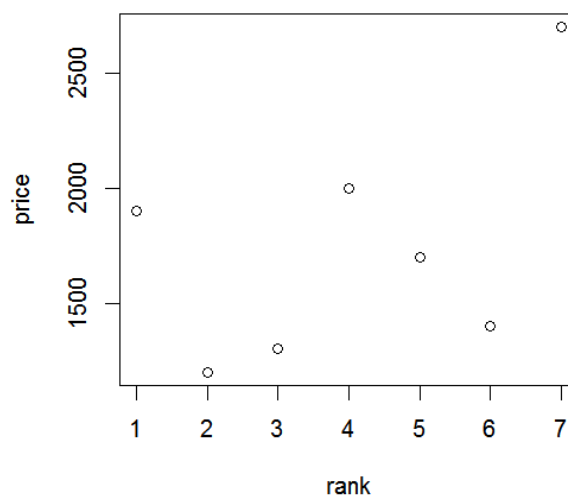
22.6 Spearman's Rank Correlation Test

The rank correlation test (or Spearman's rank correlation test) is a non-parametric test that uses ranks of sample data consisting of matched pairs. It is used to test for the strength of a linear or nonlinear association between two variables.

Example. The following table lists quality rankings and prices of 37-inch LCD televisions (**televisions.csv**). Find the value of the rank correlation coefficient and use it to determine whether or not there is a correlation between quality and price. Based on the result, does it appear that you can get better quality by spending more?

rank	price
1	1900
2	1200
3	1300
4	2000
5	1700
6	1400
7	2700

```
> TV
=read.table("E:/ESUTeaching/2019Spring/Math402/RHandout/televisions.csv", header=TRUE, sep=",");
> str(TV);
'data.frame':  7 obs. of  2 variables:
 $ rank : int  1 2 3 4 5 6 7
 $ price: int  1900 1200 1300 2000 1700 1400 2700
> plot(price~rank,data=TV);
```



It can be seen from the scatter-plot that the pattern of the association between the two variables is not linear.

Therefore, we test

$$H_0: \rho_s = 0 \text{ against } H_A: \rho_s \neq 0,$$

where ρ_s is the rank correlation coefficient between quality and price for all the population data.

```
> price=TV$price; rank=TV$rank;
> cor.test(price, rank, alternative="two.sided",
method="spearman");
```

Spearman's rank correlation rho

```
data: price and rank
S = 32, p-value = 0.3536
alternative hypothesis: true rho is not equal to 0
sample estimates:
rho
0.4285714
```

The estimate of ρ_s is 0.4285714. We fail to reject H_0 due to the large p-value of the test, 0.3536. Based on the test result, it does not appear that we can get better quality by spending more.