Introduction to R Programming
Lecture 1

Minghao Wu
minghaowu_2015@163.com

April 25, 2015

1 Some Reference Material

R Cookbook: http://www.cookbook-r.com/
R in Action: http://www.amazon.com/R-Action-Robert-Kabacoff/dp/1935182390
ggplot2: Elegant Graphics for Data Analysis (Use R!): http://www.amazon.com/ggplot2-Elegant-Graphics-Data-Analysis/dp/0387981403
Advanced R: http://adv-r.had.co.nz/ **&** http://www.amazon.com/Advanced-Chapman-Hall-CRC-Series/dp/1466586966

2 Installing and Loading Packages

Installing: install.packages('ggplot2')
Loading: library(ggplot2)
Updating: update.packages()

3 R language basics

Create a vector: v = c(1,4,4,3,2,2,3) or w = c("apple","banana","orange")
Return certain elements: v[c(2,3,4)] or v[2:4] or v[c(2,4,3)]
Delete certain element: v = v[-2] or v = v[-2:-4]
Extract elements: v[v<3]
Find elements: which(v==3) **Note**: the returns are the indices of elements

4 Numbers

Random Number: a = runif(3, min=0, max=100)
Rounding of Numbers: floor(a) or ceiling(a) or round(a,4)
Random Numbers from Other Distributions: rnorm(), rexp(), rbinom(), rgeom(), rnbinom() and so on.
Repeatable Random Numbers: set.seed()

5 Data Input

Loading Local Data: ?read.csv(); read.csv(file=" /documents/rugby.txt") or
read.table(file=" /documents/rugby.txt")
Loading Online Data: read.csv("http://www.macalester.edu/ kaplan/ISM/datasets/swim100m.csv")
Attach: attach()

6 Graphs

Plot:plot()
Histograms: hist()
Density Plot:plot(density())
Scatter Plot: plot()
Box Plot: boxplot(time sex)
Q-Q Plot: qqnorm(), qqline() and qqplot()

Introduction to R Programming
Lecture 2

Minghao Wu
minghaowu_2015@163.com

April 12, 2015

1 Understanding the Dataset

1.1 Vector

Vectors are **one-dimensional** arrays that can hold numeric data, character data, or logical data. The combine function **c()** is used to form the vector.

```
> a = c(1, 2, 5, 3, 6, -2, 4)
> b = c("one", "two", "three")
> c = c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)
```

1.2 Matrix

Matrix is a **two-dimensional** array where each element has the same mode (numeric, character, or logical). Matrices are created with the **matrix()** function.

```
> x = matrix(1:20, nrow=5, ncol=4, byrow=TRUE)
> x

     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13   14   15   16
[5,]   17   18   19   20

> y = matrix(1:20, nrow=5, ncol=4, byrow=FALSE)
> x

     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
```

```
[3,]    9   10   11   12
[4,]   13   14   15   16
[5,]   17   18   19   20
```

```
> x[2,]
```

```
[1] 5 6 7 8
```

```
> x[,2]
```

```
[1]  2  6 10 14 18
```

```
> x[1,4]
```

```
[1] 4
```

```
> x[2,c(2,4)]
```

```
[1] 6 8
```

```
> x[3:5, 2]
```

```
[1] 10 14 18
```

```
> rnames=c("apple","banana","orange","melon","corn")
> cnames=c("cat","dog","bird","pig")
> x = matrix(1:20, nrow=5, ncol=4, byrow=TRUE)
> rownames(x)=rnames
> colnames(x)=cnames
> x
```

```
       cat dog bird pig
apple    1   2    3   4
banana   5   6    7   8
orange   9  10   11  12
melon   13  14   15  16
corn    17  18   19  20
```

1.3 Array

Arrays are similar to matrices but can have **more than two dimensions**. They are created with an **array()** function.

```
> dim1 = c("A1", "A2")
> dim2 = c("B1", "B2", "B3")
> dim3 = c("C1", "C2", "C3", "C4")
> dim4 = c("D1", "D2", "D3")
> z = array(1:72, c(2, 3, 4, 3), dimnames=list(dim1, dim2, dim3, dim4))
> z
```

```
, , C1, D1

   B1 B2 B3
A1  1  3  5
A2  2  4  6

, , C2, D1

   B1 B2 B3
A1  7  9 11
A2  8 10 12

, , C3, D1

   B1 B2 B3
A1 13 15 17
A2 14 16 18

, , C4, D1

   B1 B2 B3
A1 19 21 23
A2 20 22 24

, , C1, D2

   B1 B2 B3
A1 25 27 29
A2 26 28 30

, , C2, D2

   B1 B2 B3
A1 31 33 35
A2 32 34 36

, , C3, D2

   B1 B2 B3
A1 37 39 41
A2 38 40 42

, , C4, D2

   B1 B2 B3
A1 43 45 47
```

```
A2 44 46 48

, , C1, D3

   B1 B2 B3
A1 49 51 53
A2 50 52 54

, , C2, D3

   B1 B2 B3
A1 55 57 59
A2 56 58 60

, , C3, D3

   B1 B2 B3
A1 61 63 65
A2 62 64 66

, , C4, D3

   B1 B2 B3
A1 67 69 71
A2 68 70 72
```

```
> z[1,2,3,]
```

```
D1 D2 D3
15 39 63
```

1.4 Data Frame

A data frame is more general than a matrix in that different columns can contain different modes of data (numeric, character, etc.). It is similar to the datasets you would typically see in SAS, SPSS, and Stata. Data frames are the most common data structure you will deal with in R.

```
> patientID = c(1, 2, 3, 4)
> age = c(25, 34, 28, 52)
> diabetes = c("Type1", "Type2", "Type1", "Type1")
> status = c("Poor", "Improved", "Excellent", "Poor")
> patientdata = data.frame(patientID, age, diabetes, status)
> patientdata
```

```
  patientID age diabetes    status
1         1  25    Type1      Poor
```

```
2          2  34    Type2  Improved
3          3  28    Type1 Excellent
4          4  52    Type1      Poor
```

```
> swim = read.csv("http://www.macalester.edu/~kaplan/ISM/datasets/swim100m.csv")
> patientdata[1:2]
```

```
  patientID age
1         1  25
2         2  34
3         3  28
4         4  52
```

```
> patientdata[1:3]
```

```
  patientID age diabetes
1         1  25    Type1
2         2  34    Type2
3         3  28    Type1
4         4  52    Type1
```

```
> patientdata[1,1:3]
```

```
  patientID age diabetes
1         1  25    Type1
```

```
> patientdata[c(1,3),1:3]
```

```
  patientID age diabetes
1         1  25    Type1
3         3  28    Type1
```

1.5 Attach and Detach

The attach() function adds the data frame to the R search path.
The detach() function removes the data frame from the search path.

1.6 List

Lists are the most complex of the R data types. Basically, a list is an ordered collection of objects (components). A list allows you to gather a variety of (possibly unrelated) objects under one name.

```
> mylist = list(patientdata, swim, x)
> mylist
```

```
[[1]]
  patientID age diabetes    status
1         1  25    Type1      Poor
2         2  34    Type2  Improved
3         3  28    Type1 Excellent
4         4  52    Type1      Poor

[[2]]
   year  time sex
1  1905 65.80   M
2  1908 65.60   M
3  1910 62.80   M
4  1912 61.60   M
5  1918 61.40   M
6  1920 60.40   M
7  1922 58.60   M
8  1924 57.40   M
9  1934 56.80   M
10 1935 56.60   M
11 1936 56.40   M
12 1944 55.90   M
13 1947 55.80   M
14 1948 55.40   M
15 1955 54.80   M
16 1957 54.60   M
17 1961 53.60   M
18 1964 52.90   M
19 1967 52.60   M
20 1968 52.20   M
21 1970 51.90   M
22 1972 51.22   M
23 1975 50.59   M
24 1976 49.44   M
25 1981 49.36   M
26 1985 49.24   M
27 1986 48.74   M
28 1988 48.42   M
29 1994 48.21   M
30 2000 48.18   M
31 2000 47.84   M
32 1908 95.00   F
33 1910 86.60   F
34 1911 84.60   F
35 1912 78.80   F
36 1915 76.20   F
37 1920 73.60   F
```

```
38 1923 72.80   F
39 1924 72.20   F
40 1926 70.00   F
41 1929 69.40   F
42 1930 68.00   F
43 1931 66.60   F
44 1933 66.00   F
45 1934 65.40   F
46 1936 64.60   F
47 1956 62.00   F
48 1958 61.20   F
49 1960 60.20   F
50 1962 59.50   F
51 1964 58.90   F
52 1972 58.50   F
53 1973 57.54   F
54 1974 56.96   F
55 1976 55.65   F
56 1978 55.41   F
57 1980 54.79   F
58 1986 54.73   F
59 1992 54.48   F
60 1994 54.01   F
61 2000 53.77   F
62 2004 53.52   F

[[3]]
       cat dog bird pig
apple    1   2    3   4
banana   5   6    7   8
orange   9  10   11  12
melon   13  14   15  16
corn    17  18   19  20
```

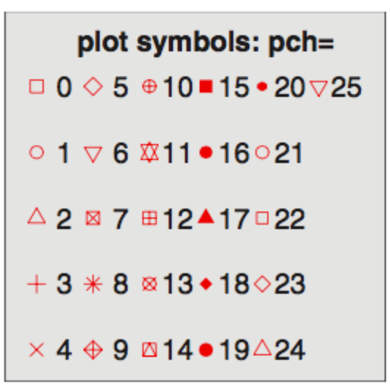
2 Graphs

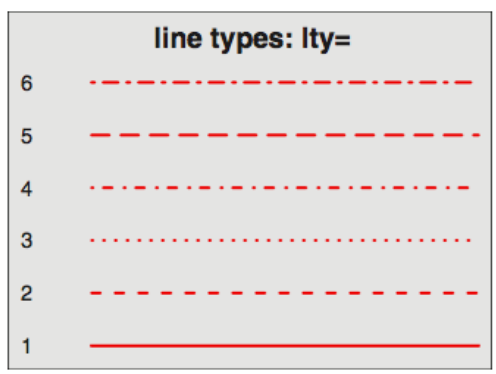
2.1 Graphical parameters

You can customize many features of a graph (fonts, colors, axes, titles) through options called graphical parameters. They are specified with an **par()** function.

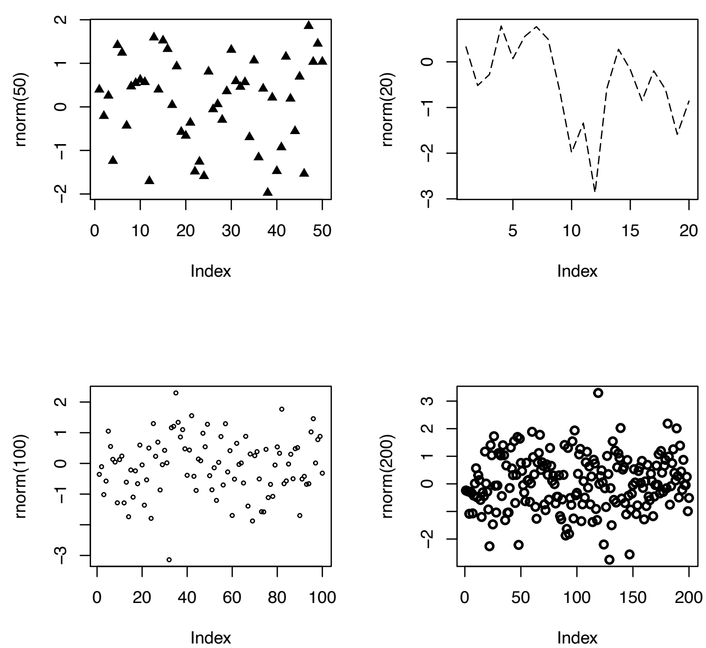
```
> par(mfrow=c(2,2))
> plot(rnorm(50),pch=17)
> plot(rnorm(20),type="l",lty=5)
> plot(rnorm(100),cex=0.5)
```

Parameter	Description
`pch`	Specifies the symbol to use when plotting points (see figure 3.4).
`cex`	Specifies the symbol size. `cex` is a number indicating the amount by which plotting symbols should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, and so forth.
`lty`	Specifies the line type (see figure 3.5).
`lwd`	Specifies the line width. `lwd` is expressed relative to the default (default=1). For example, `lwd=2` generates a line twice as wide as the default.





```
> plot(rnorm(200),lwd=2)
>
```



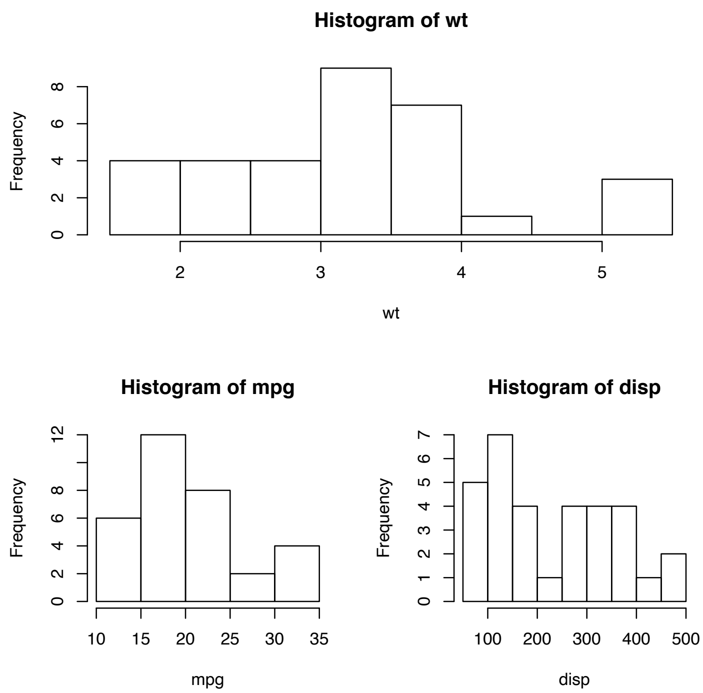
2.2 Text, Axes, and Legends

title()
axis()
legend()

2.3 Layout

The layout() function has the form layout(mat) where mat is a matrix object speci- fying the location of the multiple plots to combine.

```
> attach(mtcars)
> layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
> hist(wt)
> hist(mpg)
> hist(disp)
> detach(mtcars)
```



3 Next Topic

Operators, Control Flow & User-defined Function

Introduction to R Programming
Lecture 3

Minghao Wu
minghaowu_2015@163.com

April 15, 2015

1 Last Lecture - List

```
> x = matrix(1:20, nrow=5, ncol=4, byrow=TRUE)
> swim = read.csv("http://www.macalester.edu/~kaplan/ISM/datasets/swim100m.csv")
> mylist=list(swim,x)
> mylist[[2]][2]

[1] 5

> mylist[[2]][1:2]

[1] 1 5

> mylist[[2]][1:2,3]

[1] 3 7

> mylist[[2]][1:2,]

     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
```

2 Operators

>, >=, <, <=, ==, !=, x | y, x & y

3 Control Flow

3.1 Repetition and looping

Looping constructs repetitively execute a statement or series of statements until a condition is not true. These include the **for** and **while** structures.

```
> #For-Loop
> for(i in 1:10){
+   print(i)
+ }

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

> #While-Loop
> i = 1
> while(i <= 10){
+   print(i)
+   i=i+1
+ }

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

3.2 Conditional Execution

In conditional execution, a statement or statements are only executed if a specified condition is met. These constructs include **if**, **if-else**, and **switch**.

```
> #If statement
> i = 1
> if(i == 1){
+   print("Hello World")
+ }

[1] "Hello World"
```

```
> #If-else statement
> i = 2
> if(i == 1){
+   print("Hello World!")
+ }else{
+   print("Goodbye World!")
+ }
```

```
[1] "Goodbye World!"
```

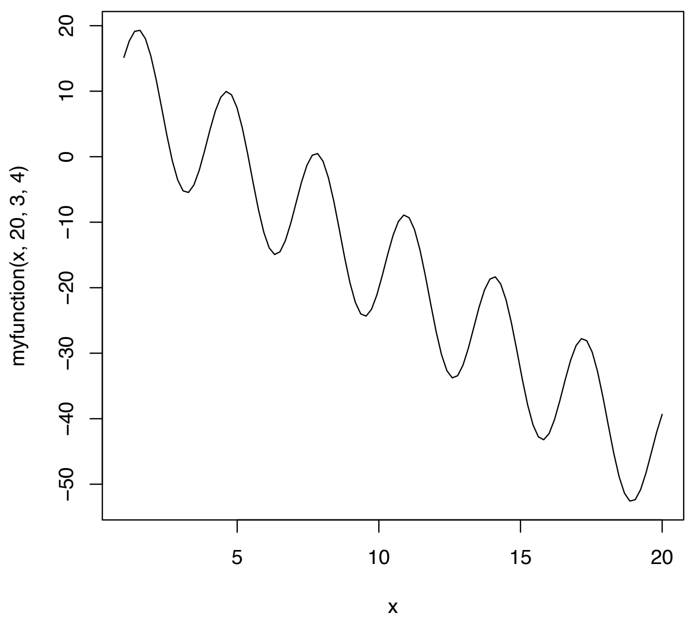
```
> #switch
> #switch(expression, cnoditions)
> feelings = c("sad", "afraid")
> for (i in feelings){
+   print(
+     switch(i,
+            happy  = "I am glad you are happy",
+            afraid = "There is nothing to fear",
+            sad    = "Cheer up",
+            angry  = "Calm down now"
+            )
+     )
+ }
```

```
[1] "Cheer up"
[1] "There is nothing to fear"
```

4 User-defined Function

One of greatest strengths of R is the ability to add functions. In fact, many of the functions in R are functions of existing functions.

```
> myfunction = function(x,a,b,c){
+   return(a*sin(x)^2 - b*x + c)
+ }
> curve(myfunction(x,20,3,4),xlim=c(1,20))
```




```
> myfeeling = function(x){
+     for (i in x){
+       print(
+         switch(i,
+                happy  = "I am glad you are happy",
+                afraid = "There is nothing to fear",
+                sad    = "Cheer up",
+                angry  = "Calm down now"
+                )
+         )
+     }
+ }
> feelings = c("sad", "afraid")
> myfeeling(feelings)

[1] "Cheer up"
[1] "There is nothing to fear"
```

Introduction to R Programming
Lecture 4

Minghao Wu
minghaowu_2015@163.com

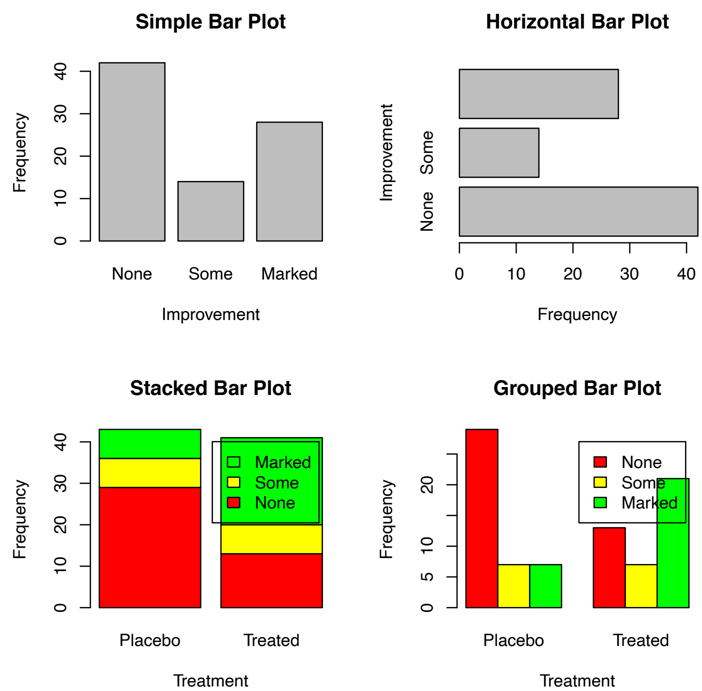
April 16, 2015

1 Baisc Graph

1.1 Bar Plot

```
> library(vcd)
> counts <- table(Arthritis$Improved)
> counts

  None   Some Marked
    42     14     28

> par(mfrow=c(2,2))
> barplot(counts,
+         main="Simple Bar Plot",
+         xlab="Improvement", ylab="Frequency")
> barplot(counts,
+         main="Horizontal Bar Plot",
+         xlab="Frequency", ylab="Improvement",
+         horiz=TRUE)
> counts <- table(Arthritis$Improved, Arthritis$Treatment)
> counts

         Placebo Treated
  None        29      13
  Some         7       7
  Marked       7      21

> barplot(counts,
+         main="Stacked Bar Plot",
+         xlab="Treatment", ylab="Frequency",
+         col=c("red", "yellow","green"),
+         legend=rownames(counts))
> barplot(counts,
```

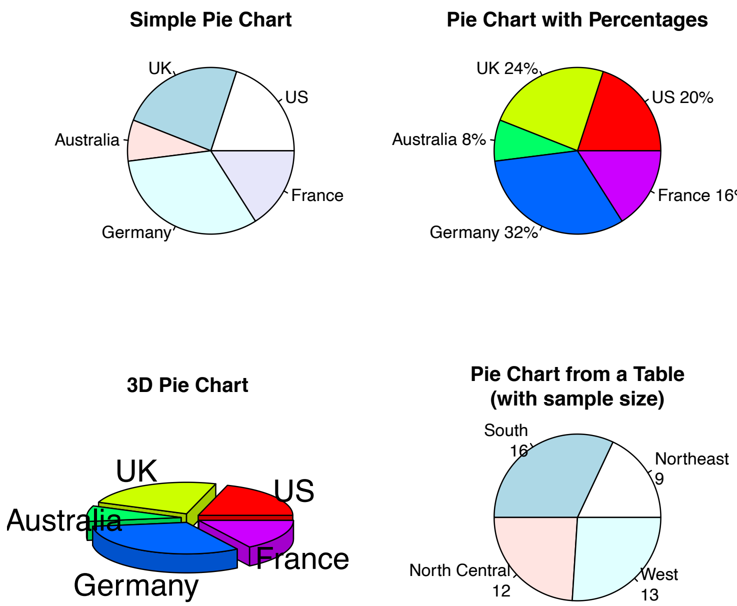
```
+          main="Grouped Bar Plot",
+          xlab="Treatment", ylab="Frequency",
+          col=c("red", "yellow", "green"),
+          legend=rownames(counts), beside=TRUE)
>
```



1.2 Pie Chart

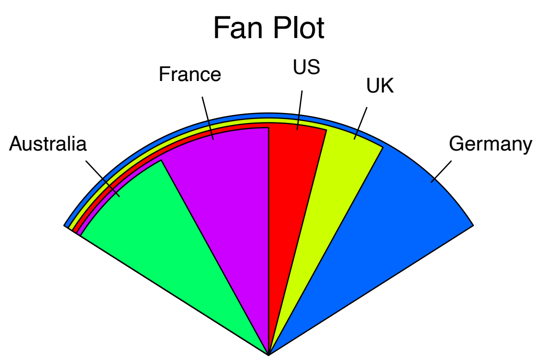
```
> library(plotrix)
> par(mfrow=c(2,2))
> slices <- c(10, 12,4, 16, 8)
> lbls <- c("US", "UK", "Australia", "Germany", "France")
> pie(slices, labels = lbls,main="Simple Pie Chart",edges=300,radius=1)
> pct <- round(slices/sum(slices)*100)
> lbls2 <- paste(lbls, " ", pct, "%", sep="")
> pie(slices, labels=lbls2, col=rainbow(length(lbls2)),
+      main="Pie Chart with Percentages",edges=300,radius=1)
> pie3D(slices, labels=lbls,explode=0.1,
+        main="3D Pie Chart ",edges=300,radius=1)
> mytable <- table(state.region)
> lbls3 <- paste(names(mytable), "\n", mytable, sep="")
> pie(mytable,labels=lbls3,
```

```
+     main="Pie Chart from a Table\n(with sample size)",
+     edges=300,radius=1)
> slices <- c(10, 12,4, 16, 8)
> lbls <- c("US", "UK", "Australia", "Germany", "France")
>
```



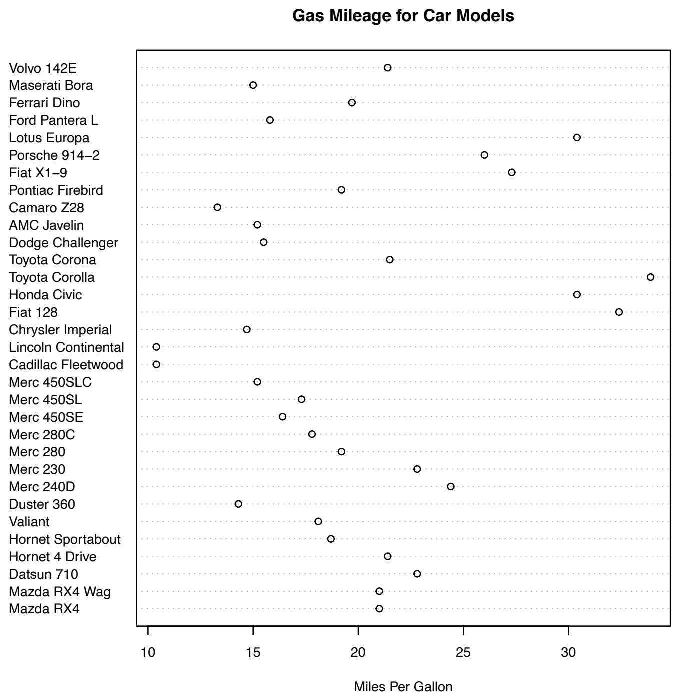
1.3 Fan Plot

```
> fan.plot(slices, labels = lbls, main="Fan Plot")
```



1.4 Dot Plot

```
> dotchart(mtcars$mpg,
+          labels=row.names(mtcars),cex=0.7,
+          main="Gas Mileage for Car Models",
+          xlab="Miles Per Gallon")
```



2 Basic Statistics

2.1 Descriptive statistics

```
> head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
> summary(mtcars)
```

```
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
```

```
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am              gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :0.0000   Median :4.000   Median :2.000  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :1.0000   Max.   :5.000   Max.   :8.000  
```

2.2 Frequency and contingency tables

```
> attach(mtcars)
> table(cyl)

cyl
 4  6  8 
11  7 14 

> summary(mpg)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  10.40   15.42   19.20   20.09   22.80   33.90 

> table(cut(mpg,seq(10,34,by=2)))

(10,12] (12,14] (14,16] (16,18] (18,20] (20,22] (22,24] (24,26] (26,28] (28,30] 
      2       1       7       3       5       5       2       2       1       0 
(30,32] (32,34] 
      2       2 
```

2.3 Correlations

```
> states = state.x77[,1:6]
> cov(states)

              Population      Income   Illiteracy     Life Exp      Murder
Population 19931683.7588 571229.7796  292.8679592 -407.8424612 5663.523714
Income       571229.7796 377573.3061 -163.7020408  280.6631837 -521.894286
Illiteracy      292.8680   -163.7020    0.3715306   -0.4815122    1.581776
Life Exp       -407.8425    280.6632   -0.4815122    1.8020204   -3.869480
```

```
Murder         5663.5237   -521.8943    1.5817755   -3.8694804   13.627465
HS Grad       -3551.5096   3076.7690   -3.2354694    6.3126849  -14.549616
                  HS Grad
Population -3551.509551
Income      3076.768980
Illiteracy    -3.235469
Life Exp       6.312685
Murder       -14.549616
HS Grad       65.237894
```

```
> var(states)
```

```
             Population      Income   Illiteracy     Life Exp      Murder
Population 19931683.7588 571229.7796  292.8679592 -407.8424612 5663.523714
Income       571229.7796 377573.3061 -163.7020408  280.6631837 -521.894286
Illiteracy      292.8680   -163.7020    0.3715306   -0.4815122    1.581776
Life Exp       -407.8425    280.6632   -0.4815122    1.8020204   -3.869480
Murder         5663.5237   -521.8943    1.5817755   -3.8694804   13.627465
HS Grad       -3551.5096   3076.7690   -3.2354694    6.3126849  -14.549616
                  HS Grad
Population -3551.509551
Income      3076.768980
Illiteracy    -3.235469
Life Exp       6.312685
Murder       -14.549616
HS Grad       65.237894
```

```
> cor(states)
```

```
            Population     Income Illiteracy    Life Exp     Murder     HS Grad
Population  1.00000000  0.2082276  0.1076224 -0.06805195  0.3436428 -0.09848975
Income      0.20822756  1.0000000 -0.4370752  0.34025534 -0.2300776  0.61993232
Illiteracy  0.10762237 -0.4370752  1.0000000 -0.58847793  0.7029752 -0.65718861
Life Exp   -0.06805195  0.3402553 -0.5884779  1.00000000 -0.7808458  0.58221620
Murder      0.34364275 -0.2300776  0.7029752 -0.78084575  1.0000000 -0.48797102
HS Grad    -0.09848975  0.6199323 -0.6571886  0.58221620 -0.4879710  1.00000000
```

2.4 T-test

```
> x = rnorm(100, mean = 10, sd = 1)
> y = rnorm(100, mean = 30, sd = 10)
> t.test(x, y, alt = "two.sided",paired=TRUE)
```

```
        Paired t-test

data:  x and y
t = -20.8901, df = 99, p-value < 2.2e-16
```

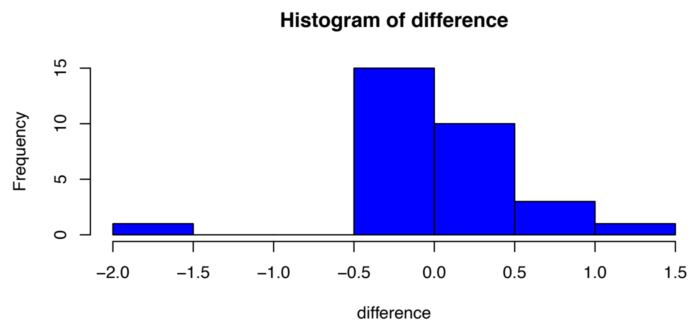
```
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -21.35301 -17.64851
sample estimates:
mean of the differences
              -19.50076
```

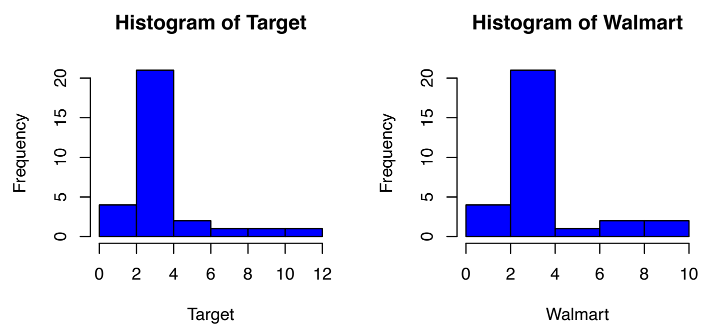
2.5 Nonparametric tests of group differences

```
> wilcox.test(x,y,alt="less")

        Wilcoxon rank sum test with continuity correction

data:  x and y
W = 61, p-value < 2.2e-16
alternative hypothesis: true location shift is less than 0
```

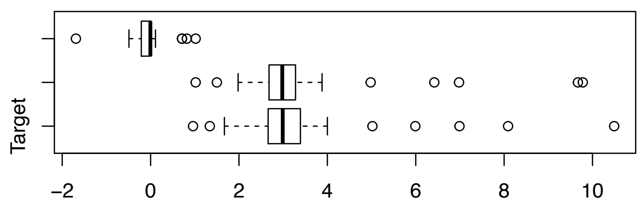
3 Practical Example

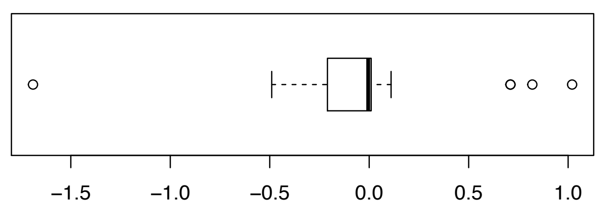
```
> data=read.csv("~/documents/R Programming/project/STAT.csv")
> attach(data)
> layout(matrix(c(1,1,2,3),2,2,byrow=TRUE))
> hist(difference,col="blue")
> hist(Target,col="blue")
> hist(Walmart,col="blue")
```





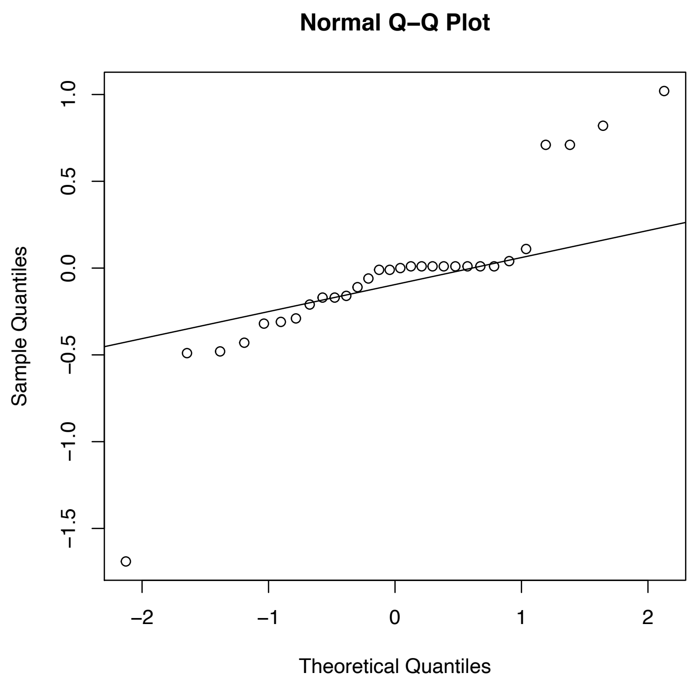

```
> par(mfrow=c(2,1))
> boxplot(data[2:4],horizontal=TRUE)
> boxplot(difference,horizontal=TRUE)
```






```
> par(mfrow=c(1,1))
> qqnorm(difference)
> qqline(difference)
> binom.test(length(difference[difference>=0]),
+            length(difference),
+            alter="two.sided")

        Exact binomial test

data:  length(difference[difference >= 0]) and length(difference)
number of successes = 15, number of trials = 30, p-value = 1
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.3129703 0.6870297
sample estimates:
probability of success
                   0.5

> wilcox.test(difference,alter="two.sided")

        Wilcoxon signed rank test with continuity correction

data:  difference
```

```
V = 174.5, p-value = 0.3557
alternative hypothesis: true location is not equal to 0

> t.test(difference,alter="two.sided")

        One Sample t-test

data:  difference
t = -0.5432, df = 29, p-value = 0.5911
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.2255485  0.1308819
sample estimates:
  mean of x
-0.04733333
```




```
> library(nortest)
> ad.test(difference)

        Anderson-Darling normality test

data:  difference
A = 2.1234, p-value = 1.552e-05
```

Introduction to R Programming
Lecture 5

Minghao Wu
minghaowu_2015@163.com

April 25, 2015

1 Matrix Algebra

R for MATLAB users: http://mathesaurus.sourceforge.net/octave-r.html

```
> set.seed(123)
> A = matrix(sample(100,15), nrow=5, ncol=3)
> set.seed(234)
> B = matrix(sample(100,15), nrow=5, ncol=3)
> set.seed(321)
> X = matrix(sample(100,25), nrow=5, ncol=5)
> set.seed(213)
> b = matrix(sample(100,5),nrow=5, ncol=1)
> A

     [,1] [,2] [,3]
[1,]   29    5   87
[2,]   79   50   98
[3,]   41   83   60
[4,]   86   51   94
[5,]   91   42    9

> B

     [,1] [,2] [,3]
[1,]   75   62   51
[2,]   78   88   49
[3,]    2   67   52
[4,]   76   86   90
[5,]    7   26    1

> X
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]   96   33   55   18   86
[2,]   93   43   95   54   79
[3,]   24   27   68   34   73
[4,]   25   42    4   98  100
[5,]   38   74   51   52   44
```

```
> b
```

```
     [,1]
[1,]    3
[2,]   35
[3,]   62
[4,]   57
[5,]   41
```

```
> #  + - * / ^
> #Element-wise addition, subtraction, multiplication, division
> #, and exponentiation, respectively.
> A + 2
```

```
     [,1] [,2] [,3]
[1,]   31    7   89
[2,]   81   52  100
[3,]   43   85   62
[4,]   88   53   96
[5,]   93   44   11
```

```
> A * 2
```

```
     [,1] [,2] [,3]
[1,]   58   10  174
[2,]  158  100  196
[3,]   82  166  120
[4,]  172  102  188
[5,]  182   84   18
```

```
> A ^ 2
```

```
     [,1] [,2] [,3]
[1,]  841   25 7569
[2,] 6241 2500 9604
[3,] 1681 6889 3600
[4,] 7396 2601 8836
[5,] 8281 1764   81
```

```
> #Matrix multiplication
> t(A) %*% B
```

```
      [,1]  [,2]  [,3]
[1,] 15592 21259 15313
[2,]  8611 15749 11653
[3,] 21496 26356 20828
```

```
> #Returns a vector containing the column means of A.
> colMeans(A)
```

```
[1] 65.2 46.2 69.6
```

```
> #Returns a vector containing the column sums of A.
> colSums(A)
```

```
[1] 326 231 348
```

```
> #Returns a vector containing the row means of A.
> rowMeans(A)
```

```
[1] 40.33333 75.66667 61.33333 77.00000 47.33333
```

```
> #Returns a vector containing the row sums of A.
> rowSums(A)
```

```
[1] 121 227 184 231 142
```

```
> #Matrix Crossproduct
> # A'A
> crossprod(A)
```

```
      [,1]  [,2]  [,3]
[1,] 24440 15706 21628
[2,] 15706 13779 15487
[3,] 21628 15487 29690
```

```
> # A'B
> crossprod(A,B)
```

```
      [,1]  [,2]  [,3]
[1,] 15592 21259 15313
[2,]  8611 15749 11653
[3,] 21496 26356 20828
```

```
> #Inverse of A where A is a square matrix.
> solve(X)
```

```
             [,1]         [,2]          [,3]          [,4]         [,5]
[1,]  0.005613920  0.010231429 -0.0169120729  0.0005830023 -0.002609067
[2,]  0.008482937 -0.016566915  0.0004314519 -0.0039590679  0.021446920
[3,] -0.009636337  0.009519483  0.0107155725 -0.0063872060 -0.001518871
[4,] -0.016672892  0.019685497 -0.0106940350  0.0088482442 -0.005123760
[5,]  0.011758574 -0.015272319  0.0140983398  0.0031012668 -0.003273400
```

```
> #Solves for vector x in the equation b = Ax.
> # b = Xv
> v = solve(X, b)
> v

           [,1]
[1,] -0.7473474
[2,]  0.1260136
[3,]  0.5422939
[4,]  0.2702193
[5,]  0.4174044

> #Returns a vector containing the elements of the principal diagonal
> diag(X)

[1] 96 43 68 98 44

> #Creates a diagonal matrix with the elements of x in the principal diagonal.
> diag(c(1,2,3,4))

     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    2    0    0
[3,]    0    0    3    0
[4,]    0    0    0    4

> #If k is a scalar, this creates a k x k identity matrix.
> diag(5)

     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    1    0    0    0
[3,]    0    0    1    0    0
[4,]    0    0    0    1    0
[5,]    0    0    0    0    1

> #Eigenvalues and eigenvectors of A.
> eigen(X)

$values
[1] 277.41449+ 0.00000i  58.39588+ 6.55948i  58.39588- 6.55948i
[4] -22.60313+29.96419i -22.60313-29.96419i

$vectors
             [,1]                       [,2]                       [,3]
[1,] 0.4488450+0i  0.74412580+0.00000000i  0.74412580+0.00000000i
[2,] 0.5594059+0i  0.19946808+0.03821859i  0.19946808-0.03821859i
[3,] 0.3416482+0i -0.38355207+0.09959650i -0.38355207-0.09959650i
```

```
[4,] 0.4364984+0i -0.46905017-0.15636245i -0.46905017+0.15636245i
[5,] 0.4223136+0i -0.05844621+0.01112305i -0.05844621-0.01112305i
                        [,4]                  [,5]
[1,] -0.2318343-0.1214549i -0.2318343+0.1214549i
[2,]  0.0164659+0.5611941i  0.0164659-0.5611941i
[3,] -0.2372177-0.1165173i -0.2372177+0.1165173i
[4,] -0.3706598-0.2584861i -0.3706598+0.2584861i
[5,]  0.5850122+0.0000000i  0.5850122+0.0000000i
```

2 Afterword

Google & English

The R Project (http://www.r-project.org/): The official R website and your first stop for all things R. The site includes extensive documentation, including An Introduction to R, The R Language Definition, Writing R Extensions, R Data Import/Export, R Installation and Administration, and The R FAQ.

The R Journal (http://journal.r-project.org/): A freely accessible refereed journal containing articles on the R project and contributed packages.

R Bloggers (http://www.r-bloggers.com/): A central hub (blog aggregator) collecting content from bloggers writing about R. Contains new articles daily. I am addicted to it.

Planet R (http://planetr.stderr.org): Another good site-aggregator, including information from a wide range of sources. Updated daily.

R Graph Gallery (http://addictedtor.free.fr/graphiques/): A collection of innovative graphs, along with their source code.

R Graphics Manual (http://bm2.genes.nig.ac.jp/): A collection of R graphics from all R packages, arranged by topic, package, and function. At last count, there were 35,000+ images!

Journal of Statistical Software (http://www.jstatsoft.org/): A freely accessible refereed journal containing articles, book reviews, and code snippets on statistical computing. Contains frequent articles about R.

Quick-R (http://www.statmethods.net): The website of R in Action author.