R for Beginners

Shakhawat Hossain

Department of Mathematics and Statistics
University of Winnipeg

June 19, 2013

Outline

- Preliminaries
- Packages and libraries
- Some basic functions and assignment statements
- Matrices and data frames
- Graphics

Preliminaries

—Why is **R**?

1. It is open-source and it is free!
2. It is cross-platform (Windows, Mac, Linux).
3. It is very popular among scientists
4. It has a large active, helpful, and friendly user base. It is updated regularly.
5. It has excellent graphical capabilities.

1. Navigate to the CRAN website at http://cran.r-project.org/.
2. Click on the appropriate link for your operating system
   - Download R for Linux
   - Download R for (Mac) OS X
   - Download R for Windows
3. Download and install it. Then you are done.

Note: R is updated regularly (roughly 3 times a year). These updates contain improvements, bugfixes, and new features. Newly developed or updated packages are not backward compatible. So keep your R version up-to-date.

Preliminaries

—R can be used as a calculator

```
> 5+8 # for addition
[1] 13
> 7/(9+2.8) # for division
[1] 0.5932203
> 3*7  # for multiplication
[1] 21
> (0.7842)^(2/5) # for exponent
[1] 0.9073414
> 9.098-2.456 # for substraction
[1] 6.642
> log(293847) # for logarithm
[1] 12.59081
```

Preliminaries—some basic functions

R knows the following mathematical functions:

- print()—prints objects
- log()—computes logarithms
- exp()—computes the exponential function
- sqrt()—takes the square root
- abs()—returns the absolute value
- sin()—returns the sine
- cos()—returns the cosine
- tan()—returns the tangent
- asin()—returns the arc-sine
- factorial()—returns the factorial
- sign()—returns the sign (negative or positive)
- round()—rounds the input to the desired digit

Preliminaries

—logical operators

- < less than
- <= less than or equal to
- > greater than
- >= greater than or equal to
- == equal
- ! = not equal
- & means and
- | means or

Preliminaries

—logical operators

```
> 5==5
[1] TRUE
> 6==6
[1] TRUE
> 6 !=5
[1] TRUE
> 7 <= 9 & 5 <= 8
[1] TRUE
> 5 == 5|1 == 2
[1] TRUE
> 1 > 1|1 > 2 & 3 == 3
[1] FALSE
> 4 > 4 & 5 > 7 & 2 > 1
[1] FALSE
```

Packages and library

R has become popular due to the vast array of packages. These packages are available in the comprehensive R archive network (CRAN). Suppose you want to install the 'MASS' package. Write down the following command on the **R console** and press ENTER

- install.packages ("MASS")

Write down the following command on the **R editor** when you want to use any function from MASS package.

- library(MASS)

To get help, use the following command on the **R-console** and press ENTER:

- library(help ="MASS")

Assignment statements and real vectors

```
> XX=2+3 # assignment statement
> XX
[1] 5
> XX*5
[1] 25
# remove all objects from the active memory
rm(list = ls ())
# assignment statement and vectors
> VE=c(3,2,6,11,9,4,7,10,5,7)
> VE*2;
 [1]  6  4 12 22 18  8 14 20 10 14
> KK=VE^2
> KK
 [1]   9   4  36 121  81  16  49 100  25  49
> KK[4:5]
[1] 121  81
```

Assignment statements and real vectors

```
> kk=seq(0,4, length=6) #6 numbers with equal spacing
> kk
[1] 0.0 0.8 1.6 2.4 3.2 4.0
> x1=c('x','y','z','t'); x1 # character vector
[1] "x" "y" "z" "t"
> x=seq(0,20,length=9)
> x[1:5] # select first 5 elements
[1]  0.0  2.5  5.0  7.5 10.0
> x[-(1:5)] # exclude first 5 elements
[1] 12.5 15.0 17.5 20.0
> fruit = c(5, 10, 1, 20)
> names(fruit)=c('orange','banana','apple','peach')
> lunch = fruit[c('apple','orange')]
> lunch
 apple orange
     1      5
```

Assignment statements and real vectors

```
> x=c(2,4,6,4,5,8)
> y=c(3,4,6,8,1,2)
> x+y
[1]  5  8 12 12  6 10
> log(x+y)
[1] 1.60943 2.07944 2.48490 2.48490 1.79175 2.30258
> 3*(x+y)
[1] 15 24 36 36 18 30
> round(log(x+y), digits=2)
[1] 1.61 2.08 2.48 2.48 1.79 2.30
> d=round(log(x+y), digits=2)
> floor(d)
[1] 1 2 2 2 1 2
> ceiling(d)
[1] 2 3 3 3 2 3
> round(d,digits=0)
[1] 2 2 2 2 2 2
```

Useful functions

Function	Description
sum()	sums of the elements of a vector
prod()	product of the elements of a vector
min()	minimum of the elements of a vector
max()	maximum of the elements of a vector
mean()	mean of the elements
median()	median of the elements
range()	range of the vector
sd()	standard deviation
var()	variance
cov()	covariance (takes two inputs cov(x,y))
cor()	correlation coefficient (takes two imputs cor(x,y))
sort()	sorts the vector (argument: decreasing = FALSE)
length()	returns the length of the vector
summary()	returns summary statistics
unique()	returns a vector of all the unique elements of the input

Useful functions

```
> kk=c(12, 8, 15, 25, 17, 13, 9, 12, 18)
> sum(kk)
[1] 129
> prod(kk)
[1] 15466464000
> min(kk)
[1] 8
> max(kk)
[1] 25
> mean(kk)
[1] 14.33333
> median(kk)
[1] 13
> range(kk)
[1]  8 25
```

Useful functions

```
> kk=c(12, 8, 15, 25, 17, 13, 9, 12, 18)
> sd(kk)
[1] 5.196152
> var(kk)
[1] 27
> sort(kk)
[1]  8  9 12 12 13 15 17 18 25
> length(kk)
[1] 9
> summary(kk)
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
8.00   12.00   13.00   14.33   17.00   25.00
> x=c(2,3,4,2,3,4)
> unique(x)
[1] 2 3 4
```

Covariance and correlation

The following data concerns the blood haemoglobin (Hb) levels and packed cell volumes (PCV) of 14 female blood bank donors. It is of interest to know if there is a relationship between the two variables Hb and PCV when considered in the female population.

```
# Read data from a local drive
data= read.table("D:/Lecture on R/data1.txt", header=T)
> Hb=data[,1];Hb
 [1] 15.5 13.6 13.5 13.0 13.3 12.4 11.1 13.1 16.1 16.4
 13.4 13.2 14.3 16.1
> PCV=data[,2]; PCV
 [1] 0.450 0.420 0.440 0.395 0.395 0.370 0.390 0.400
 0.445 0.470 0.390 0.400
 [13] 0.420 0.450
> cor(Hb, PCV)
 [1] 0.8770133
> cov(Hb, PCV)
 [1] 0.04067582
```

Delete NA, subset, and sample

```
>  kk= c(2, 3, 4, 3, NA , NA , 6, 6, 10, 11, 2, NA , 4, 3)
> summary(kk)
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
2.000   3.000   4.000   4.909   6.000  11.000       3
> kk1=na.omit(kk) # This function returns
the vector suppressing   the NAs
> summary(kk1)
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
2.000   3.000   4.000   4.909   6.000  11.000
> X=seq(1:200) # creating a vector from 1 to 200
>
> XX= subset(X, X%%6==0) # recall the modulo operator
> XX
 [1]6  12  18  24  30  36  42  48  54  60  66  72  78  84
[15]90  96 102 108 114 120 126 132 138 144 150 156 162 168
[29] 174 180 186 192 198
> sample(XX, size=10)
 [1]  24 150 108  84  54  36  60 138  48 174
```

Matrices

```
> rm( list=ls()) # delete all objects from active memory
> # create three vectors
> x=c(3,5,8,5,9,6)
> y=c(13, 9, 11, 8, 7, 5)
> z=c(8,11, 10, 13, 9, 5)
> data=cbind(x,y,z) # Make column bind
> M1= matrix(data, nrow=6, ncol=3)  # write matrix
> dim(M1) #  Dimension of matrix
[1] 6 3
> apply(data, 1, mean) # calculate row mean
[1] 8.0 8.33 9.67 8.67 8.33 5.33
> apply(data, 2, mean) # calculate column mean
x    y    z
6.0 8.83 9.33
> apply(data, 1, sd) # calculate row standard deviation
[1] 5.0 3.05 1.53 4.04 1.15 0.58
> apply(data, 2, sd) # calculate column standard deviation
x    y    z
2.19 2.86 2.73
```

Continued from last slide.....

```
> # extracts corresponding element in the first row and 3rd col
> data[1,3]
8
> # extracts corresponding element in the 2nd row and 3rd col
> data[2,3]
11
> data[,1] # extracts the first column
[1] 3 5 8 5 9 6
> data[2,]  # extracts the second row
 x  y  z
 5  9 11
> # extracts the first and second row
> data[1:2,]
     x  y  z
[1,] 3 13  8
[2,] 5  9 11
> # delete third element of vector x
> data[,1][-3]
[1] 3 5 5 9 6
> x # call  vector x
[1] 3 5 8 5 9 6
> x[x<6]
[1] 3 5 5
```

R can do matrix arithmetic. Below is a list of some basic operations:

$+, -, *$	standard scalar or by element operations
$\%*\%$	matrix multiplication
diag()	diagonal elements of a matrix
$t()$	transpose of a matrix
solve()	inverse of a matrix
det()	determinant of a matrix
chol()	cholesky decomposition
eigen()	eigenvalues and eigenvectors
crossprod()	cross product
$\%\times\%$	kronecker product

Matrices

—Mathematical Operations (see results of the following R codes in the next 3 slides)

```
# Define two vectors
M4=c(3, 7, 8, 5, 8, 9, 4, 7,6)
M5=c(6,3,7,5,9,10,9,4,7)
MAT1=matrix(M4,3,3); MAT1 # write 3 by 3 matrix
MAT2=matrix(M5,3,3);MAT2 # write 3 by 3 matrix
MAT1+MAT2  # sum of two matrices
MAT1-MAT2  # difference of two matrices
MAT1%*%MAT2 # Multiplication of two matrices
diag(MAT1) # diagonal elements of MAT1
sum(diag(MAT1))  #sum of diagonal elements of MAT1
t(MAT1) # transpose of MAT1
solve(MAT1) # inverse of MAT1
det(MAT1) # determinant of MAT1
LL=eigen(MAT1) # eigen values and eigen vectors
LL$values # eigenvalues
LL$vectors # eigenvectors
crossprod(MAT1, MAT2)# cross product of two matrics
M4%x%M5 # kronecker product product of two vectors
```

Matrices

—Mathematical Operations

```
> M4=c(3, 7, 8, 5, 8, 9, 4, 7,6)
> M5=c(6,3,7,5,9,10,9,4,7)
> MAT1=matrix(M4,3,3); MAT1 # write 3 by 3 matrix
     [,1] [,2] [,3]
[1,]    3    5    4
[2,]    7    8    7
[3,]    8    9    6
> MAT2=matrix(M5,3,3);MAT2 # write 3 by 3 matrix
     [,1] [,2] [,3]
[1,]    6    5    9
[2,]    3    9    4
[3,]    7   10    7
> MAT1+MAT2  # sum of two matrices
     [,1] [,2] [,3]
[1,]    9   10   13
[2,]   10   17   11
[3,]   15   19   13
> MAT1-MAT2  # difference of two matrices
     [,1] [,2] [,3]
[1,]   -3    0   -5
[2,]    4   -1    3
[3,]    1   -1   -1
```

Matrices

—Mathematical Operations

```
> MAT1%*%MAT2 # Multiplication of two matrices
     [,1] [,2] [,3]
[1,]   61  100   75
[2,]  115  177  144
[3,]  117  181  150
> diag(MAT1) # diagonal elements of MAT1
[1] 3 8 6
> sum(diag(MAT1))  #sum of diagonal elements of MAT1
[1] 17
> t(MAT1) # tanspose of MAT1
     [,1] [,2] [,3]
[1,]    3    7    8
[2,]    5    8    9
[3,]    4    7    6

> solve(MAT1) # inverse of MAT1
            [,1]       [,2]       [,3]
[1,] -0.71428571  0.2857143  0.1428571
[2,]  0.66666667 -0.6666667  0.3333333
[3,] -0.04761905  0.6190476 -0.5238095
> det(MAT1) # inverse of MAT1
[1] 21
```

Matrices

—Mathematical Operations

```
> LL=eigen(MAT1) # eigenvalues and eigenvectors
> LL$values # eigenvalues
[1] 19.1464459 -1.3077398 -0.8387061
> LL$vectors # Normalized eigenvectors
           [,1]       [,2]       [,3]
[1,] -0.3661686 -0.3740149  0.2223176
[2,] -0.6488467 -0.3611428 -0.7074185
[3,] -0.6670221  0.8542182  0.6709202

> crossprod(MAT1, MAT2)# cross product of two matrics
     [,1] [,2] [,3]
[1,]   95  158  111
[2,]  117  187  140
[3,]   87  143  106

> M4%x%M5 # kronecker product product of two vectors
 [1] 18  9 21 15 27 30 27 12 21 42 21 49 35 63 70 63 28 49 48
[20] 24 56 40 72 80 72 32 56 30 15 35 25 45 50 45 20 35 48 24
[39] 56 40 72 80 72 32 56 54 27 63 45 81 90 81 36 63 24 12 28
[58] 20 36 40 36 16 28 42 21 49 35 63 70 63 28 49 36 18 42 30
[77] 54 60 54 24 42
```

Matrices

—Application of matrices to regression

```
> # simple linear regression example
> Constant = rep (1, times = 10)
> Variable = c(1,2,3,1,2,3,5,6,7,8)
> X = cbind ( Constant , Variable )
> Y = seq (1, 10)
> beta.hat = solve(t(X)%*%X)%*%t(X)%*%Y
> colnames(beta.hat)="Estimate"
> beta.hat
         Estimate
Constant  1.34375
Variable  1.09375
########################################

> # replication function
> rep(c(2,3,4), 4)
 [1] 2 3 4 2 3 4 2 3 4 2 3 4
> rep("love", 6)
[1] "love" "love" "love" "love" "love" "love"
> rep(love, 6)
Error: object 'love' not found
> rep(0, 6)
[1] 0 0 0 0 0 0
```

Read data from a local drive (comma separated data)

Following are the comma separated data. To import this data set from the local drive, use the following R-code:

```
var1,var2,var3,var4
12,23,A,John
14,24,B,Dev
14,16,C,Terry
18,11,D,Bob
> rm(list=ls()) # clear your memory
> mydata="D:\\LectureNoteOnR\\program\\data2.txt"
> newdata=read.csv(mydata, sep=",", header=TRUE)
> newdata
  var1 var2 var3  var4
1   12   23    A  John
2   14   24    B   Dev
3   14   16    C Terry
4   18   11    D   Bob
```

Read data from a local drive (semi-colon separated data)

Following are the semi colon separated data. To import this data set from the local drive, use the following R-code:

```
var1;var2;var3;var4
12;23;A;John
14;24;B;Dev
14;16;C;Terry
18;11;D;Bob
> rm(list=ls()) # clear your memory
> mydata="D:\\LectureNoteOnR\\program\\data2.txt"
> newdata=read.table(mydata, sep=";", header=TRUE)
> newdata
  var1 var2 var3  var4
1   12   23    A  John
2   14   24    B   Dev
3   14   16    C Terry
4   18   11    D   Bob
```

Read data from a local drive (Excel data)

To import excel data set from the local drive to R, first save this data as a 'csv' format, then use the following R-code

```
> rm(list=ls())
> mydata="D:\\LectureNoteOnR\\program\\data2.csv"
> newdata=read.csv(mydata, header=TRUE)
> newdata
  var1 var2 var3  var4
1   12   23    A  John
2   14   24    B   Dev
3   14   16    C Terry
4   18   11    D   Bob
```

Extract variables from data set

```
> library(MASS) # Call library MASS
> data(Cars93)  # This data is in this library
> names(Cars93) # see the variable list
 [1] "Manufacturer"       "Model"              "Type"
 [4] "Min.Price"          "Price"              "Max.Price"
 [7] "MPG.city"           "MPG.highway"        "AirBags"
[10] "DriveTrain"         "Cylinders"          "EngineSize"
[13] "Horsepower"         "RPM"                "Rev.per.mile"
[16] "Man.trans.avail"    "Fuel.tank.capacity" "Passengers"
[19] "Length"             "Wheelbase"          "Width"
[22] "Turn.circle"        "Rear.seat.room"     "Luggage.room"
[25] "Weight"             "Origin"             "Make"
>
> # Extract variables
> price=Cars93$Price
> weight=Cars93$Weight
> Hmile=Cars93$MPG.highway
> Cmile= Cars93$MPG.city
> RPM= Cars93$RPM
```

Dataframe and subset of data set

```
> data1=data.frame(price,weight,Hmile,Cmile,RPM)
> data1[1:3,] # extract first 3 rows
  price weight Hmile Cmile  RPM
1  15.9   2705    31    25 6300
2  33.9   3560    25    18 5500
3  29.1   3375    26    20 5500
> price1=price>40
> data1[price1,] # subset of data when price>40
   price weight Hmile Cmile  RPM
11  40.1   3935    25    16 6000
48  47.9   4000    22    17 6000
59  61.9   3525    25    19 5500
> subset(data1, weight > 3000 & price < 16)# subset data
   price weight Hmile Cmile  RPM
14  15.1   3240    28    19 4600
15  15.9   3195    29    21 5200
21  15.8   3085    28    23 5000
27  15.6   3080    27    21 4800
61  14.9   3610    26    19 3800
65  15.7   3050    30    24 5600
```

Merge two data sets (see output in the next three slides)

```
# read data from local drive
mydata1="D:\\LectureNoteOnR\\program\\merge1.txt"
merge1=read.table(mydata1, header=TRUE)
merge1

mydata2="D:\\LectureNoteOnR\\program\\merge2.txt"
merge2=read.table(mydata2, header=TRUE)
merge2

# remove duplicate
merge1=subset(merge1, !duplicated(merge1[,1]))
merge1
merge2=subset(merge2, !duplicated(merge2[,1]))
merge2

newdata1=merge1[order(merge1[,1]),] # sort data by ID
newdata1
newdata2=merge2[order(merge2[,1]),] # sort data by ID
newdata2
#merge two data sets
merge(newdata1,newdata2)
```

Merge two data sets (see output in the next two slides)

```
> mydata1="D:\\LectureNoteOnR\\program\\merge1.txt"
> merge1=read.table(mydata1, header=TRUE) # read data from local drive
> merge1
   ID Sex age
1 234   F  24
2 345   M  56
3 123   F  22
4 634   M  34
5 224   M  36
6 834   M  45
7 345   M  56
> mydata2="D:\\LectureNoteOnR\\program\\merge2.txt"
> merge2=read.table(mydata2, header=TRUE) # read data from local drive
> merge2
   ID income SBP
1 234  40000  90
2 345  64000 100
3 123  25000 110
4 634  45000 115
5 224  70000 125
6 334  80000 110
7 234  23000 100
```

Merge two data sets

```
> # remove duplicate records
> merge1=subset(merge1, !duplicated(merge1[,1]))
> merge1
   ID Sex age
1 234   F  24
2 345   M  56
3 123   F  22
4 634   M  34
5 224   M  36
6 834   M  45
> merge2=subset(merge2, !duplicated(merge2[,1]))
> merge2
   ID income SBP
1 234  40000  90
2 345  64000 100
3 123  25000 110
4 634  45000 115
5 224  70000 125
6 334  80000 110
```

Merge two data sets

```
> newdata1=merge1[order(merge1[,1]),] # sort data by ID
> newdata1
   ID Sex age
3 123   F  22
5 224   M  36
1 234   F  24
2 345   M  56
4 634   M  34
6 834   M  45
> newdata2=merge2[order(merge2[,1]),] # sort data by ID
> newdata2
   ID income SBP
3 123  25000 110
5 224  70000 125
1 234  40000  90
6 334  80000 110
2 345  64000 100
4 634  45000 115
> merge(newdata1,newdata2) #merge two data sets
   ID Sex age income SBP
1 123   F  22  25000 110
2 224   M  36  70000 125
3 234   F  24  40000  90
4 345   M  56  64000 100
5 634   M  34  45000 115
```

```
> X=rnorm(20, 10,5)
> Test=t.test(X, alternative = "two.sided",
+ mu = 12, paired = FALSE, conf.level = 0.95)
>
> Test # Test output

        One Sample t-test
data:  X
t = -2.8189, df = 19, p-value = 0.01096
alternative hypothesis: true mean is not equal to 12
95 percent confidence interval:
  6.93148 11.25098
sample estimates:
mean of x
  9.09123
```

```
> Estimate=Test$estimate # estimated mean
> Estimate
mean of x
  9.09123
> statistic= Test$statistic # statistic value
> statistic
        t
-2.818902
> pvalue=Test$p.value # p-value
> pvalue
[1] 0.01096245
> CI=Test$conf.int # 95 percent confidence interval
> CI
[1]  6.93148 11.25098
attr(,"conf.level")
[1] 0.95
>2*pt(-2.818902, df=19) #calculate pvalue using pt functio
[1] 0.01096245
```

Two sample t-test

Example on Cold Medicine

Example: In an attempt to determine if two brands of cold medicine contain, on average, the same amount of acetaminophen, twelve different tablets from each of the two brands were randomly selected and tested for the amount of acetaminophen each contains. The results (in milligrams) follow. A test is wanted at the significance level 0.01.

Brand A				Brand B			
517,	495,	503,	491	493,	508,	513,	521
503,	493,	505,	495	541,	533,	500,	515
498,	481,	499,	494	536,	498,	515,	515

State and perform an appropriate hypothesis test. Assume that the samples were selected independently and randomly.

- **Answer:** Let μ_1= the mean amount of acetaminophen in cold tablet brand A and μ_2= the mean amount of acetaminophen in cold tablet brand B. Steps of hypothesis test:

1. Null hypothesis: $H_0 : \mu_1 = \mu_2$ and alternative hypothesis: $H_0 : \mu_1 \neq \mu_2$ (two tailed test).
2. The significance level $\alpha = 0.01$.

Two sample t-test

```
> # Two sample Test
> rm(list=ls())
> xx=c(517,495,503,491,503,493,505,495,498,481,499,494)
> yy=c(493,508,513,521,541,533,500,515,536,498,515,515)
>
> stat=t.test(xx, yy, alternative = "two.sided",
+ mu = 0, paired = FALSE, conf.level = 0.99)
> stat
        Welch Two Sample t-test

data:  xx and yy
t = -3.524, df = 17.705, p-value = 0.002474
alternative hypothesis: true difference in means
is not equal to 0 99 percent confidence interval:
 -32.427938  -3.238729
sample estimates:
mean of x mean of y
 497.8333  515.6667
```

Two sample t-test

```
> # Two sample t-test output
>
> Estimate1=stat$estimate # estimated mean
> Estimate1
mean of x mean of y
 497.8333  515.6667
> statistic1= stat$statistic # statistic value
> statistic1
        t
-3.524031
> pvalue1=stat$p.value # p-value
> pvalue1
[1] 0.002474401
> CI1=stat$conf.int # 99 percent confidence interval
> CI1
[1] -32.427938  -3.238729
attr(,"conf.level")
[1] 0.99
```

Analysis of Variance

—Example

We are to investigate the formulation of a new synthetic fibre that will be used to make cloth for shirts. The cotton content varies from $10 - 40$ percent by weight and the experimenter chooses 5 levels of this factor: 15, 20, 25, 30, and 35 percent. The response variable is Y = tensile strength. There are 5 replicates (complete repetitions of the experiment). In a replicate five shirts, each with a different cotton content, are randomly chosen from the five populations of shirts. The 25 tensile strengths are measured, in random order. This is then a completely randomized design (CRD). The following are the tensile strength data:

	Replicates					Total	Averages
Level	1	2	3	4	5	$y_{i\cdot}$	$\bar{y}_{i\cdot}$
A	7	7	15	11	9	49	9.8
B	12	17	12	18	18	77	15.4
C	14	18	18	19	19	88	17.6
D	19	25	22	19	23	108	21.6
E	7	10	11	15	11	54	10.8
						$y_{\cdot\cdot} = 376$	$\bar{y}_{\cdot\cdot} = 15.04$

Analysis of Variance

Example is continued...

Questions: Does changing the cotton content (level) change the mean strength? That is, is the variation between the means at the 5 levels large enough, relative to the variation within the means at the 5 levels, that we can conclude that it is not arising purely by chance?

```
> rm(list=ls()) # clear memory
> # tensile strength data
> A = c(7, 7, 15, 11, 9)
> B = c(12, 17, 12, 18, 18)
> C = c(14, 18, 18, 19, 19)
> D = c(19, 25, 22, 19, 23)
> E = c(7, 10, 11, 15, 11)
>
> data = rbind(A, B, C, D, E)
> #The sample mean for each treatment
> AveR=apply(data, 1, mean)
> AveR
   A    B    C    D    E
 9.8 15.4 17.6 21.6 10.8
> #The sample mean for each replicate
> apply(data, 2, mean)
[1] 11.8 15.4 15.6 16.4 16.0
```

Analysis of Variance

Example is continued...

```
> # The overall mean
> mean(data)
[1] 15.04
> # Draw a scatterplot of 'data' as a data frame
> # with treatments as columns
> stripchart(data.frame(t(data)), vertical = TRUE)
>
> #Arrange responses by column
> strength = c(data)
> strength
 [1]  7 12 14 19  7  7 17 18 25 10 15 12 18 22 11 11
  18 19 19 15  9 18 19 23 11
> # Total SS:
> (25-1)*var(strength)
[1] 636.96
```

Analysis of Variance

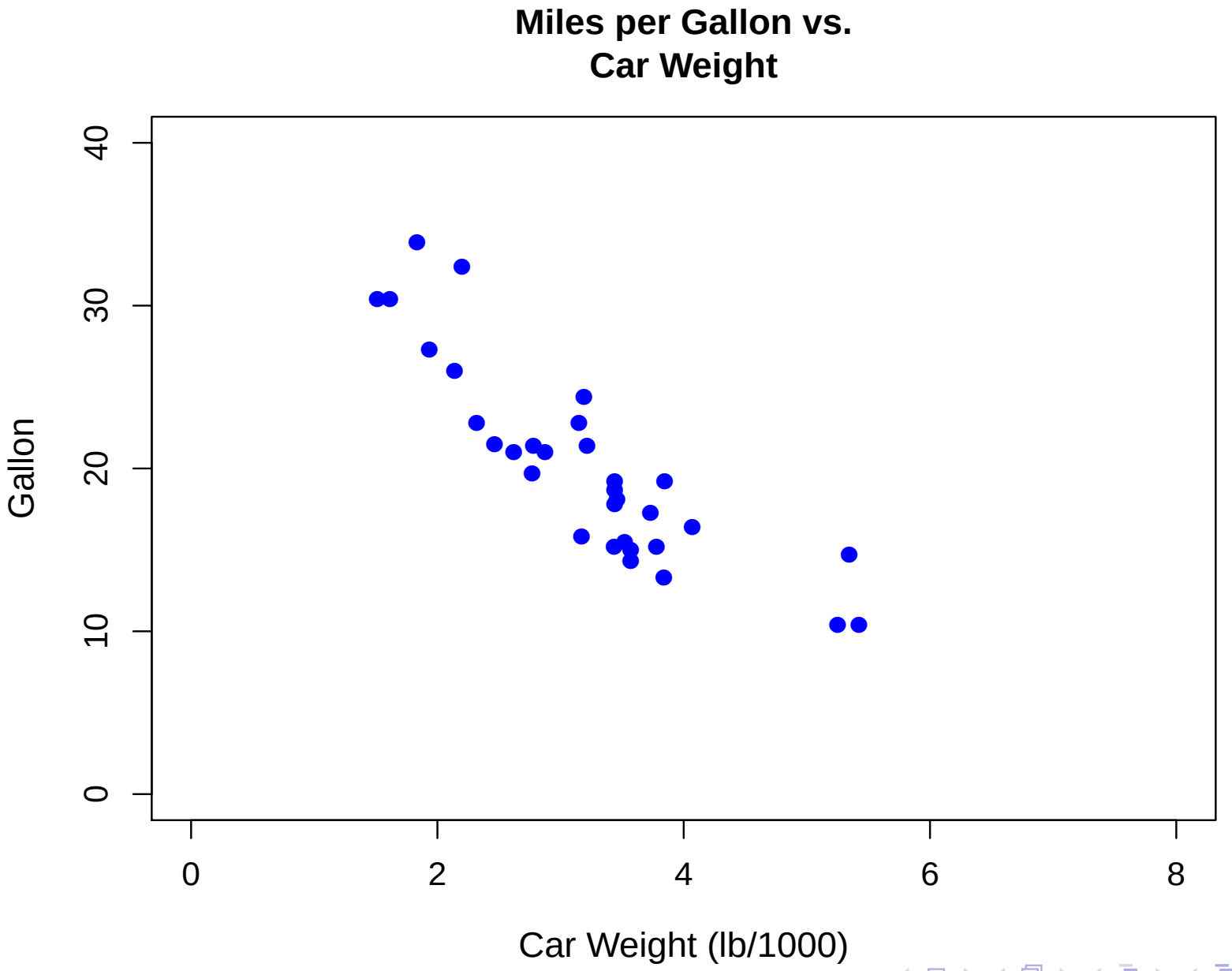
Example is continued...

```
> #Set the factor at 5 levels
>
> d = rep(c("A", "B", "C", "D", "E"), times=5)
> content = as.factor(d)
> content
 [1] A B C D E A B C D E A B C D E A B C D E A B C D E
Levels: A B C D E

> #Perform one-way ANOVA (lm stands for 'linear model').
> g = lm(strength~content)
> anova(g)
Analysis of Variance Table
Response: strength
          Df Sum Sq Mean Sq F value    Pr(>F)
content    4 475.76  118.94  14.757 9.128e-06
Residuals 20 161.20    8.06
> names(g)  # Here are the components of g;
 [1] "coefficients"  "residuals"     "effects"         "rank"
 [5] "fitted.values" "assign"        "qr"              "df.residual"
 [9] "contrasts"     "xlevels"       "call"            "terms"
[13] "model"
```

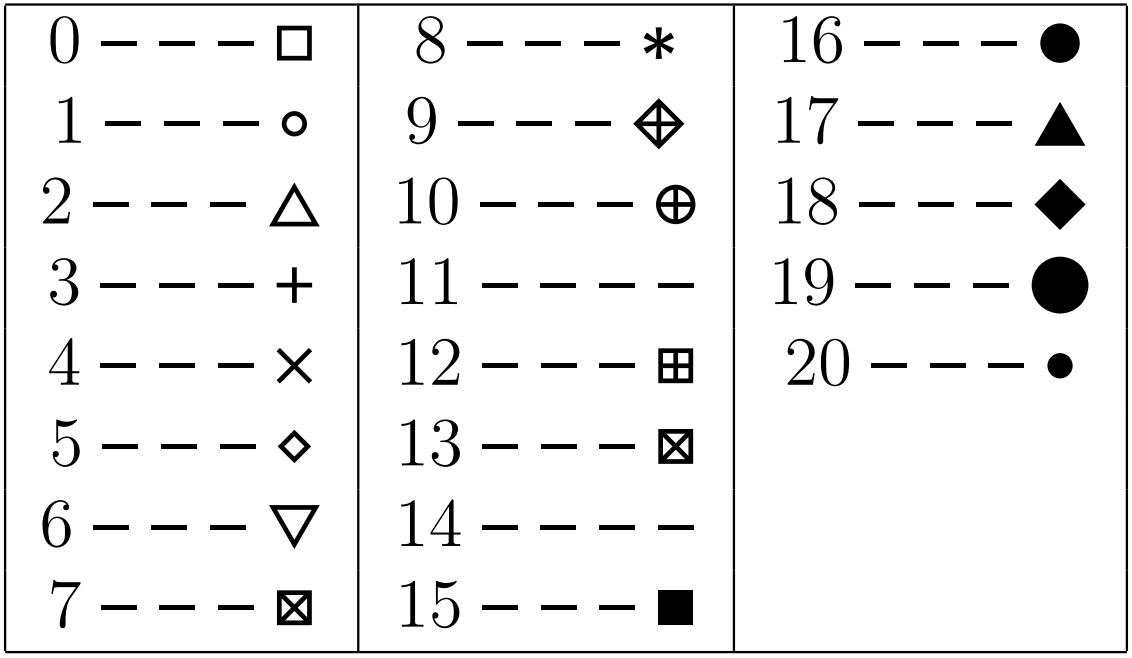
```
library(MASS)
names(mtcars) # to see the name of variables
plot(mtcars$mpg ~ mtcars$wt,
main="Miles per Gallon vs.
Car Weight", xlab="Car Weight (lb/1000)",
ylab="Miles Per Gallon", xlim=c(0,8),
ylim=c(0,30), col="blue", pch=16, cex.axis=1.1,
cex.lab=1.2, cex.main=1.2, cex=1.2);

# pch=plotting symbol number
# cex=changing text size, axis labels,
# title, symbol size
```

There are many colors available to use. The easiest is to use them by name. To see colors with names, use the following R-function:
colors() # name of colors

Basic graphs —motor trend car road tests data (mtcars)
Scatter plot



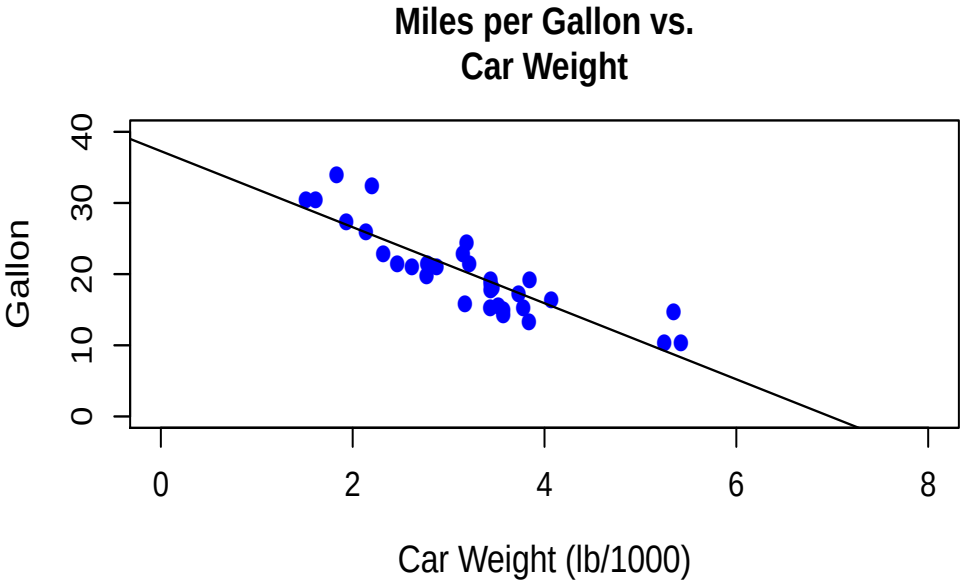
Basic graphs —plotting symbols (pch)

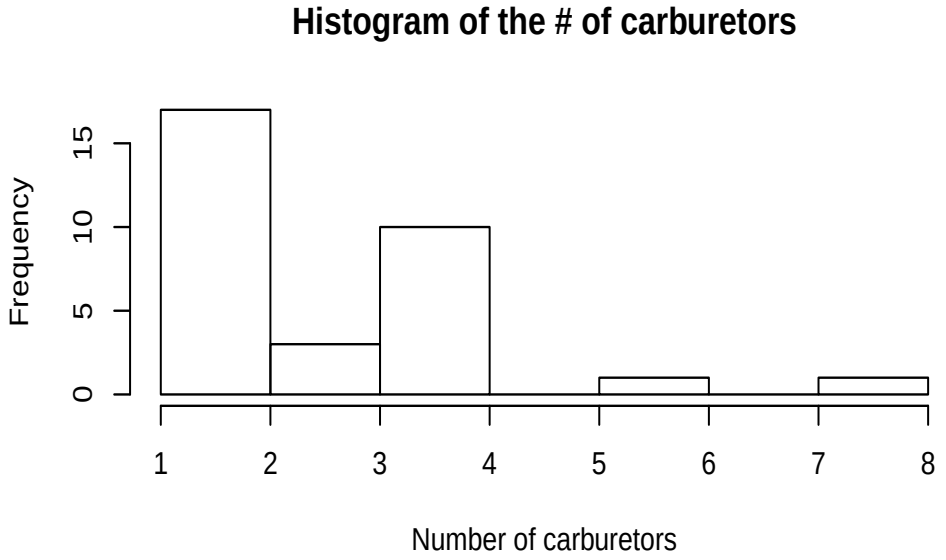


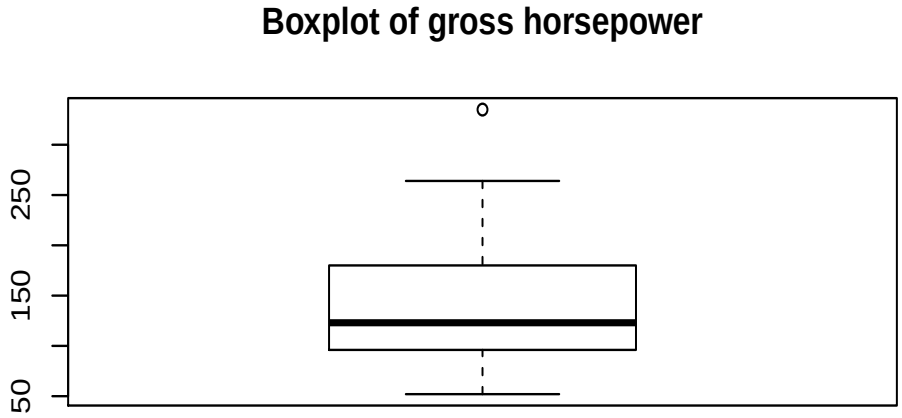
Basic graphs —motor trend car road tests data and chick's weight data

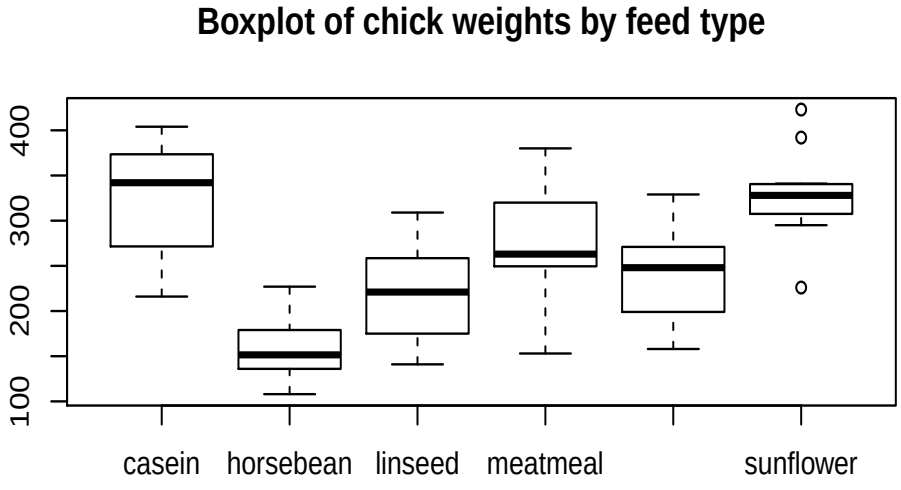
```
par(mfrow=c(2,2))
plot(mtcars$mpg ~ mtcars$wt,
main="Miles per gallon vs.
car weight", xlab="Car weight (lb/1000)",
ylab="Miles per gallon", xlim=c(0,8), ylim=c(0,40),
col="blue", pch=16, cex.axis=1.1, cex.lab=1.2,
cex.main=1.2, cex=1.2)
abline(lm(mtcars$mpg ~ mtcars$wt))

hist (mtcars$carb, xlab='Number of carburetors',
main='Histogram of the # of carburetors')
boxplot(mtcars$hp,main='Boxplot of gross horsepower')

# Box plot of chick's weight data by feed type
# There are six types of feed in the chickwts dataset
boxplot(chickwts$weight ~ chickwts$feed,
main="Boxplot of chick weights by feed type")
```

Basic graphs —motor trend car road tests data and chick's weights data









Basic graphs —pie chart of chick's weight data

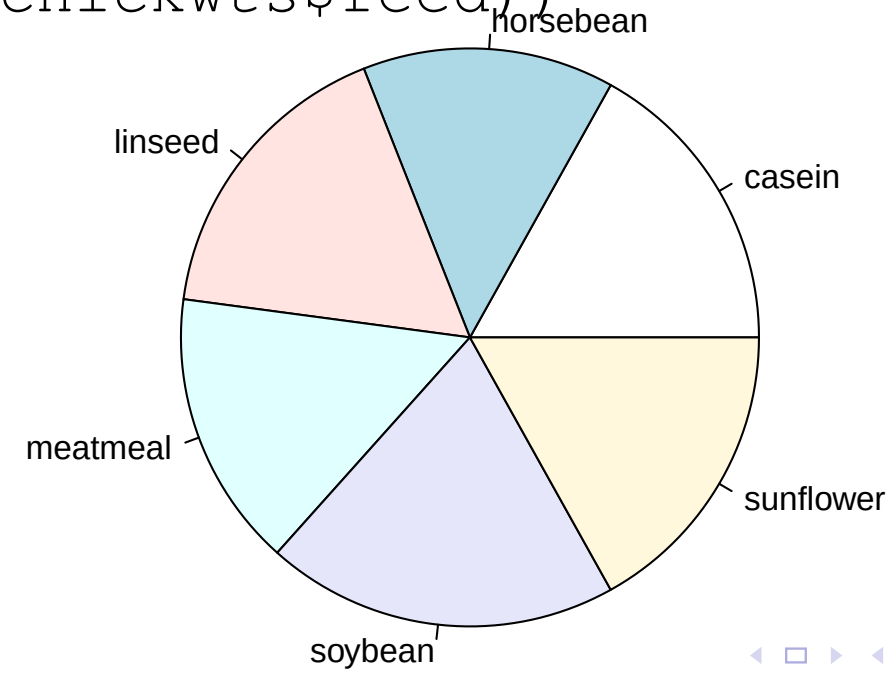
```
> table(chickwts$feed)
casein horsebean linseed  meatmeal  soybean sunflower
12         10         12         11         14         12
```

A barplot of how many chickens are in each feed category:

```
> barplot(table(chickwts$feed))
```

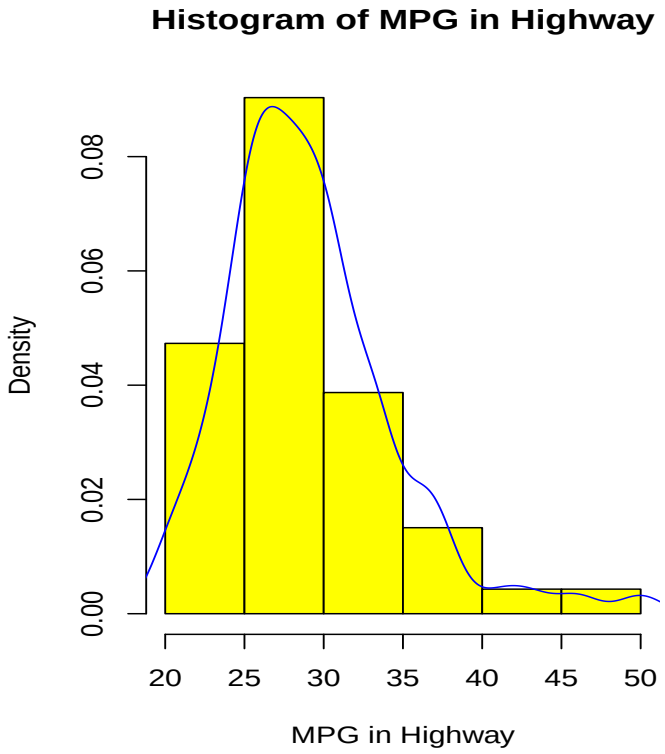
A pie chart showing how many chickens in each category:

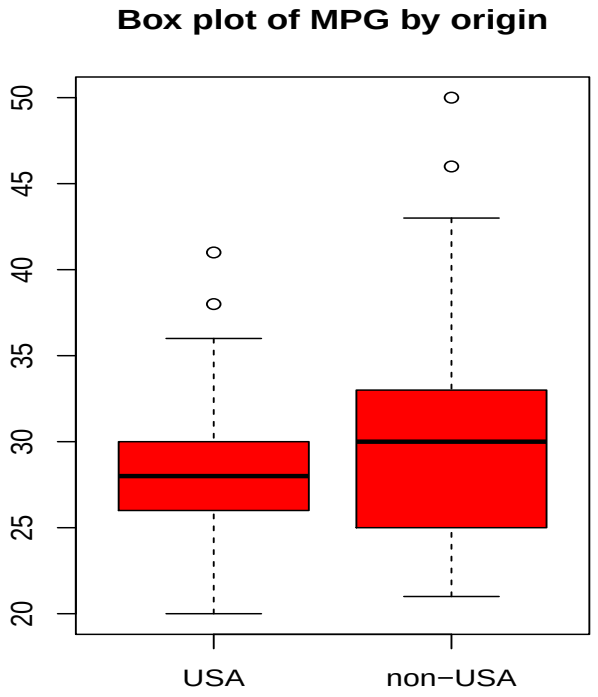
```
> pie(table(chickwts$feed))
```



Graphics —Cars93 data Example

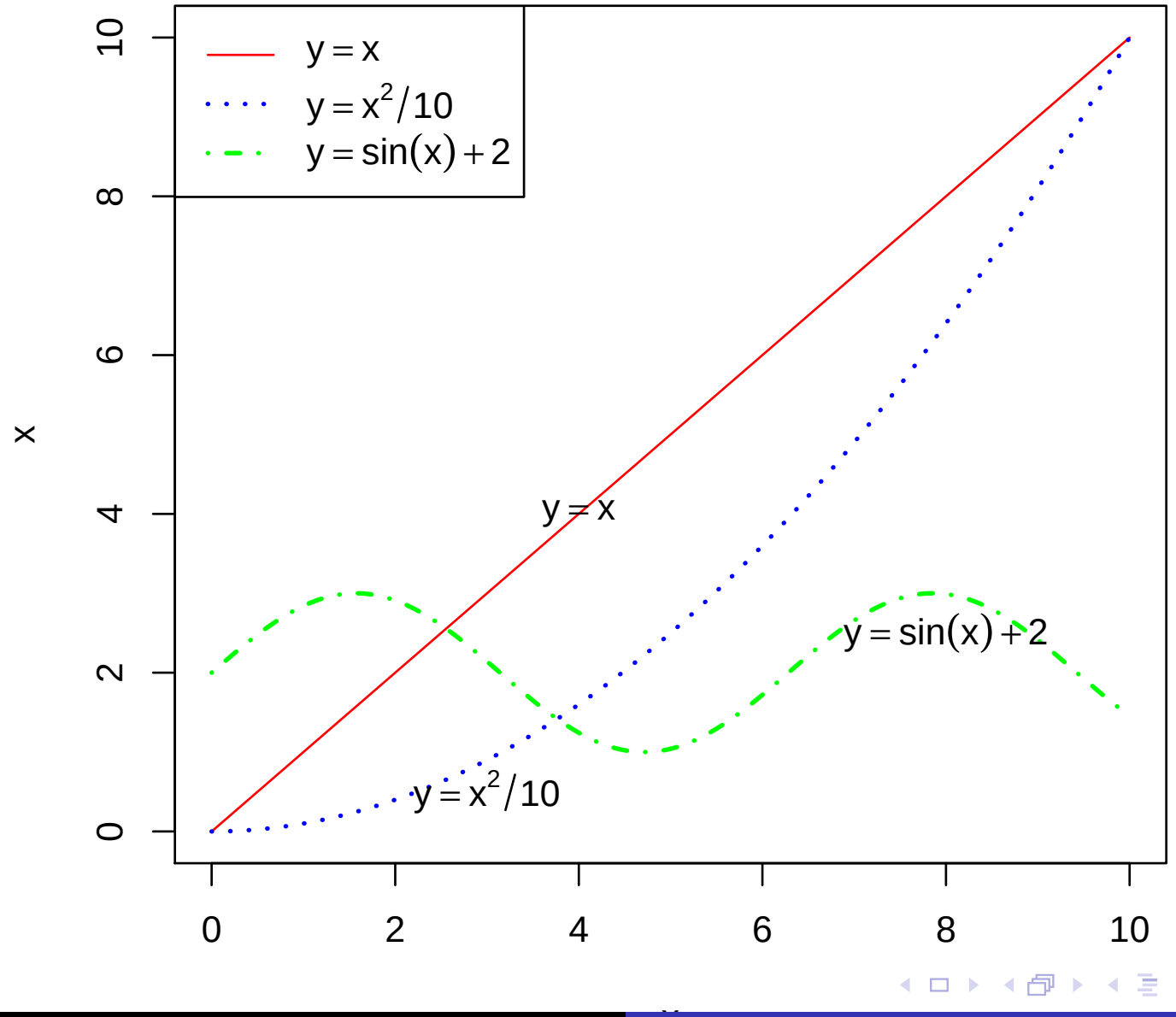
```
rm(list=ls())
library(MASS)
par(mfrow = c(1, 2))
hist(Cars93$MPG.highway, col = "yellow", prob = T,
xlab='MPG in Highway',main='Histogram of MPG in Highway')
lines(density(Cars93$MPG.highway), col = "blue")
boxplot(MPG.highway ~ Origin, col = "red", data = Cars93,
main='Box plot of MPG by origin')
```





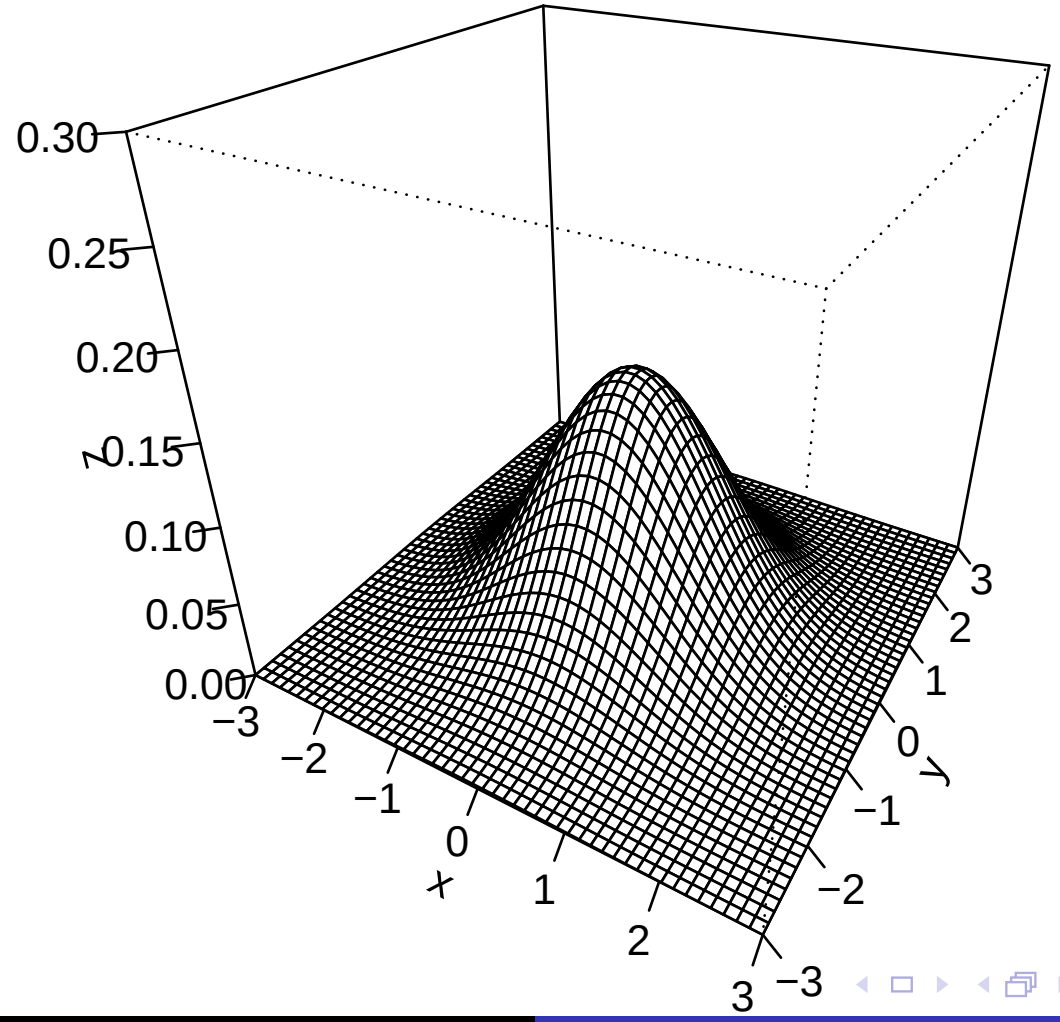
```
x=seq(0,10,length=250)
y1=x^2/10
y2=sin(x) +2

plot(x,x,type="l", col='red', lwd=1)
text(4,4,expression(y==x))

lines(x,y1, lwd=2, lty=3, col='blue')
text(3,0.5,expression(y==x^2/10))

lines(x,y2,lwd=2,lty=4, col='green')
text(8,2.5,expression(y==sin(x)+2))


legend(locator(1), c(expression(paste(y==x)),
expression(paste(y==x^2/10)), expression(paste(y==sin(x)+2))),
col=c('red','blue','green'),lwd=c(1,2,2),
lty=c(1,3,4))
```

Graphics



Graphics —perspective plot

```
x=seq(-3,3,length=50)
y=seq(-3,3,length=50)
f=function(x,y){
return(dnorm(x)*dnorm(y))}
z=outer(x,y,f)
persp(x,y,z, zlim=c(0,0.3), theta = 30, phi = 30,ticktype='detailed')
```



Graphics —three dimensional bar plot

```
library(scatterplot3d)

my.mat = matrix(runif(25), nrow = 5)
dimnames(my.mat) = list(LETTERS[1:5], letters[11:15])
s3d.dat = data.frame(columns = c(col(my.mat)),
rows = c(row(my.mat)), value = c(my.mat))

scatterplot3d(s3d.dat, type = "h", lwd = 3, pch = 16,
x.ticklabs = colnames(my.mat), y.ticklabs = rownames(my.mat),
color = 'red', main = "3D barplot")
```