STAT COMPUTING
BANA 6043

Lecture 5

More about R (S-plus): R objects (cont'd), Plots and Regression

R object --- Data Frames

- Data frames are fundamental to the use of the R modeling and graphics functions.
- **A data frame is a generalization of a matrix**, in which different columns may have different modes.
- All elements of any column must however have the same mode, i.e. all numeric or all factor, or all character.

Making a data frame

```
> celsius <- 25:30
> fahrenheit <- 9/5*celsius+32
> degrees<-data.frame(celsius, fahrenheit)
> class(degrees)
> dim(degrees)
> degrees[,1]   # Extract the 1st column
> degrees$celsius # Extract the variable “celsius”
> degrees[1,] # Extract the 1st row
```

Tips and Tricks

- A *data frame* is a list with **class "data.frame".**
- The components must be vectors (numeric, character, or logical), factors, matrices, or other data frames.
- Vector structures appearing as variables of the data frame must all have the same length, and matrix structures must all have the same row size.
- A data frame can be regarded as a matrix with columns possibly of differing modes and attributes.
- A data frame is displayed in a matrix form, and its rows and columns can be extracted using matrix indexing conventions.

Exercise 5 – from SAS slides

The data set “CLINIC” consists of two variables, “TYPE” and “SCORE”. “TYPE” refers to what patients take. “SCORE” is a kind of health score of patients.

TYPE	SCORE	TYPE	SCORE
drug	8	drug	9
drug	10	placebo	7
placebo	5	placebo	6
drug	9	placebo	6

Step 1. Input the data set. Label “TYPE” and “SCORE” as “drug or placebo” and “health score” respectively.

Step 2. Calculate the means of health score for patients taking drug and for patients taking placebo respectively.

Code

```
> type<-
c(rep("drug",2),"placebo",rep("drug",2),rep("placobo",3))
> score<-c(8,10,5,9,9,7,6,6)
> clinic<-data.frame(type,score)
> clinic
     type score
1    drug     8
2    drug    10
3 placebo     5
4    drug     9
5    drug     9
6 placobo     7
7 placobo     6
8 placobo     6
> class(clinic)   # What is the output?
```

Try these…

```
> clinic2<-cbind(type,score)
> class(clinic2)
```

Compare the result to the previous one.

Result

```
> clinic2<-cbind(type,score)
> clinic2
     type      score
[1,] "drug"    "8"
[2,] "drug"    "10"
[3,] "placebo" "5"
[4,] "drug"    "9"
[5,] "drug"    "9"
[6,] "placobo" "7"
[7,] "placobo" "6"
[8,] "placobo" "6"
> class(clinic2)
[1] "matrix“  # It is a matrix!
```

Step 2. Calculate the means of health score for patients taking drug and for patients taking placebo respectively.

```
> mean(score[type=="drug"])
[1] 9
> mean(score[type=="placebo"])
[1] 5
```

Obtain subsets of a data frame

- Select columns

```
> clinic[,2]
[1]  8 10  5  9  9  7  6  6
> clinic[,"score"]
[1]  8 10  5  9  9  7  6  6
```

- Select rows

```
> clinic[type=="drug",]
  type score
1 drug     8
2 drug    10
4 drug     9
5 drug     9
> clinic[score>=9,]
  type score
2 drug    10
4 drug     9
5 drug     9
```

Tips and Tricks

Logical Operators

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

R object --- Lists

- An R *list* is an object consisting of an ordered collection of objects known as its *components*.

- There is no particular need for the components to be of the same mode or type.
  - for example, a list could consist of a numeric vector, a logical value, a matrix, a complex vector, a character array, a function, and so on.

- Here is a simple example of how to make a list:

```
> Lst <- list(name="Fred", wife="Mary”,
   no.children=3, child.ages=c(4,7,9))
```

Check these

class(Lst)

length(Lst)

class(Lst$name)

class(Lst$child.ages)

```
> class(Lst)
[1] "list“

> length(Lst)  # How many components Lst has?
[1] 4

> class(Lst$name)
[1] "character“

> class(Lst$child.ages)
[1] "numeric"
```

How to retrieve the elements in a list?

Try the following commands:

```
> Lst$name
> Lst[[1]]
> Lst$wife
> Lst[[2]]
> Lst$child.ages[1]
> Lst[[4]][1]
```

How to retrieve the elements in a list?

- Lst$name is the same as Lst[[1]] and is the string "Fred“.
- Lst$wife is the same as Lst[[2]] and is the string "Mary“.
- Lst$child.ages[1] is the same as Lst[[4]][1] and is the number 4.

R object --- Functions

- Example 1
  - Define a function that transform Fahrenheit degrees to Celsius degrees

```
> fahrenheit2celsius <- function(fahrenheit)
(fahrenheit-32)*5/9
> fahrenheit2celsius(32)
> fahrenheit2celsius(32:40)
```

Result

```
> fahrenheit2celsius(32)
[1] 0

> fahrenheit2celsius(32:40)
[1] 0.0000000 0.5555556 1.1111111 1.6666667 2.2222222
[6] 2.7777778 3.3333333 3.8888889 4.4444444
```

R object --- Functions

- Example 2
  - Define a function that calculate the mean and standard deviation of a sample

```
> mean.and.sd <- function(x){
 av <- mean(x)
 sd <- sqrt(var(x))
 return(c(mean=av, SD=sd))
 }
> x1<-rnorm(100)
> mean.and.sd(x1)
> x2<-rnorm(10000)
> mean.and.sd(x2)
```

Result

```
> x1<-rnorm(100)
> mean.and.sd(x1)
       mean           SD
-0.06109368  0.78416424


> x2<-rnorm(10000)
> mean.and.sd(x2)
         mean            SD
-0.0002776084  0.9828899765
```

Tips and Tricks

Syntax: **ff<-function(x) {arguments of x; return(y)}**

- A function is created using an assignment.
- On the right hand side, the parameters appear within round brackets. You can, if you wish, give a default.
- Following the closing “)” the function body appears. Except where the function body consists of just one statement, this is enclosed between curly braces ({ }).
- The return value usually appears on the final line of the function body. It is recommended to explicitly write a “return” function.

R Loops

Example 1

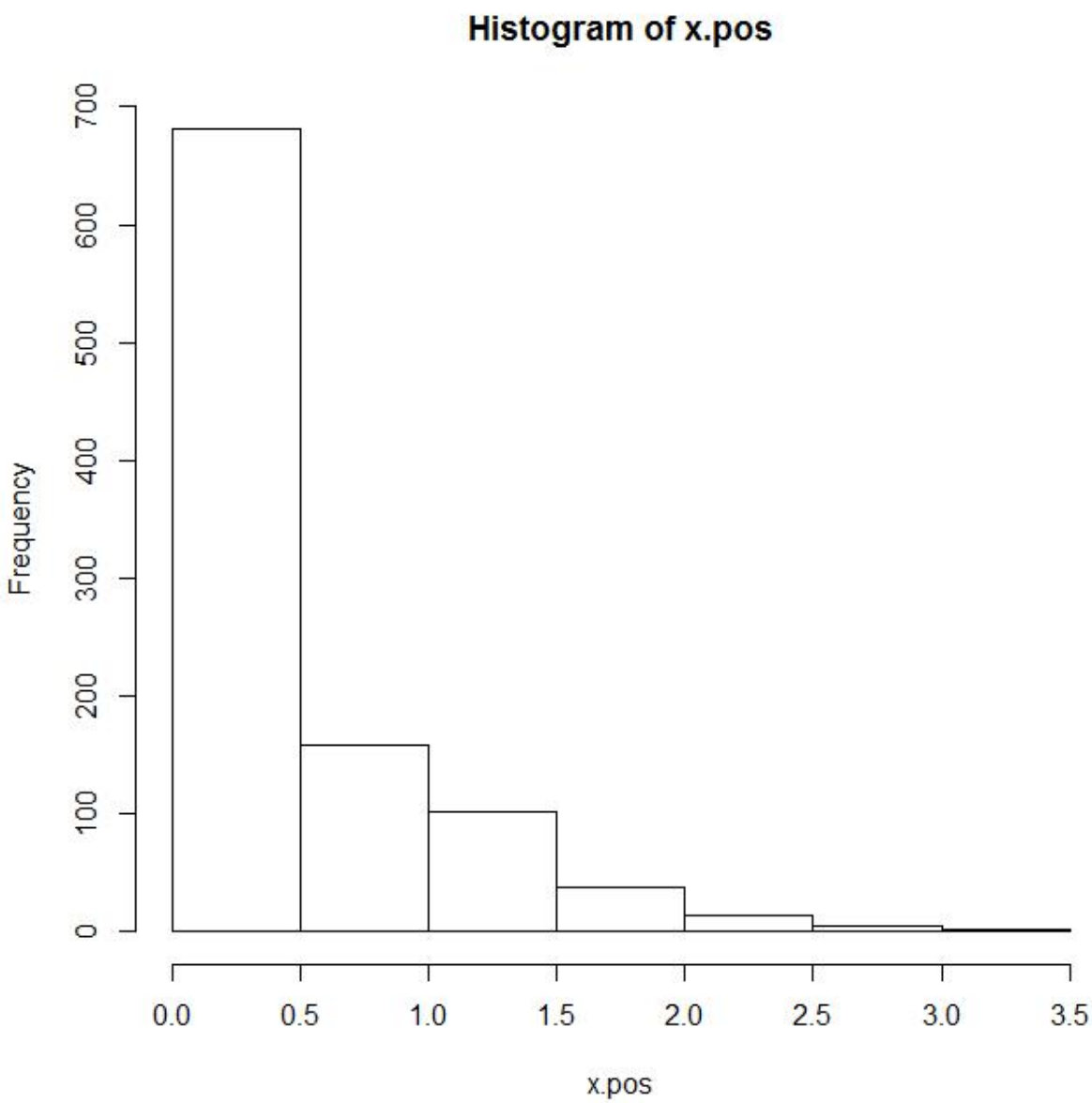
```
> for(i in 1:10) print(i)
```

Example 2

```
> for(celsius in 25:30)
> print(c(celsius, 9/5*celsius + 32))
```

Example 3

```
> x<-rnorm(1000); x.pos<-rep(NA, length(x))
> for(j in 1:length(x)) {
    if(x[j]<0) x.pos[j]<-0
      else x.pos[j]<-x[j]
    }
> hist(x)
> hist(x.pos)
```


Histogram of x.pos
Frequency
0
100
200
300
400
500
600
700
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
x.pos

Exercise 5

Suppose M is a matrix. You are asked to write your own function to realize the same functionality of the following command:

```
> apply(M,2,mean)
```

Note: You are only allowed to use one existing R function “dim()”. Do not call other functions.

Code

```
col.means<-function(M){ # Input: a matrix; Output: Column-wise means
        if(class(M)=="matrix"){
           n.row<-dim(M)[1]; n.col<-dim(M)[2]
           means<-rep(NA,n.col)
           for(i in 1:n.col){
              summation<-0
                   for(j in 1:n.row) summation<-summation+M[j,i]
                   means[i]<-summation/n.row
           }
           return(means)
        } else print("The input is not a matrix!")
}

M<-matrix(1:12,nrow=3)
col.means(M)
apply(M,2,mean)
col.means(2)
```

Conditional execution: IF statements

The R has available a conditional construction of the form

> **if (*expr_1*) *expr_2* else *expr_3***

where *expr_1* must evaluate to a single logical value.

Example:

```
> x<-rnorm(10)
> if(sum(x)>0) print("Positive") else print("negative")
```

Exercise 6 (Copied from SAS slides)

ID	name	sex	math	music
02	Mark	M	78	98
12	Bill	M	89	?
23	Cathy	F	93	79

To-do list:

1. Create a data set “student” in R.
2. Create a data set “stu.bio”without any score, using the data set you created in Step 1.
3. Create a data set “student.m” by selecting observations with male students in the data set “student”.
4. Create a new variable GOOD in such a way that if the math score of a student is greater or equal to 90, put YES and otherwise put NO. Append the new variable to the data set “student”.

code

```
ID<-c("02","12","23")
name<-c("Mark","Bill","Cathy")
sex<-c("M","M","F")
math<-c(78,89,93)
music<-c(98,NA,79)


student<-data.frame(ID,name,sex,math,music)
stu.bio<-student[,1:3]
student.m<-student[sex=="M",]
GOOD<-rep("YES",length(math))
GOOD[math<90]<-"NO"
student<-data.frame(student,GOOD)
```

Distribution-related functions

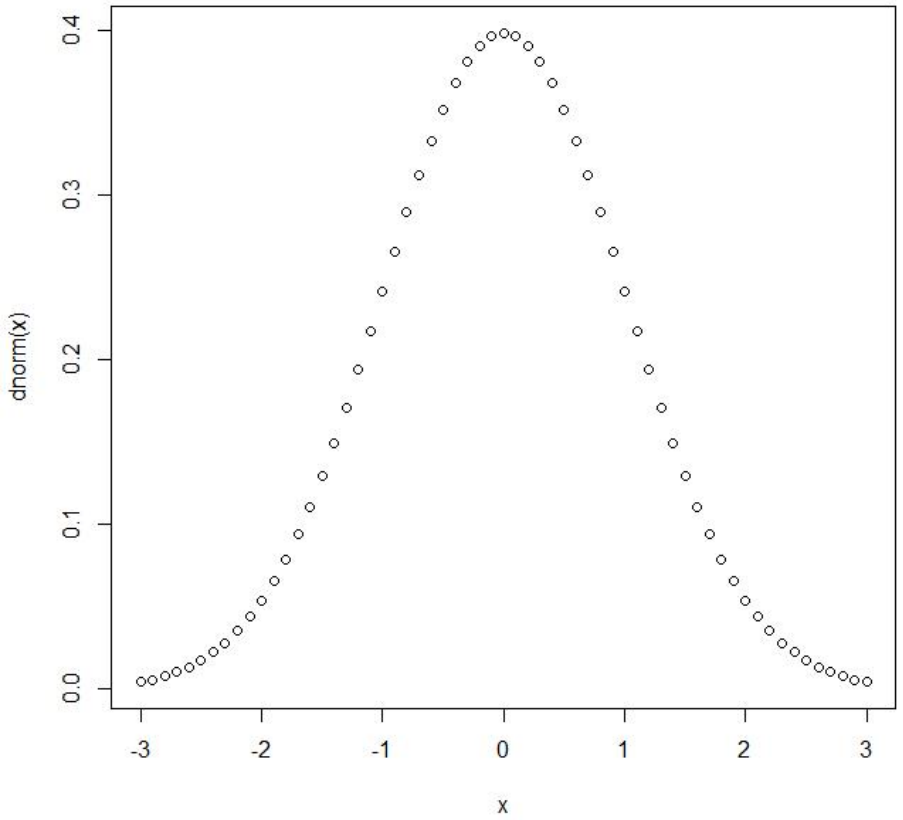
For normal distributions

- dnorm(x, mean = 0, sd = 1)
  - Returns density function (“d” -> density)
- pnorm(q, mean = 0, sd = 1)
  - Returns distribution function (“p” -> probability)
- qnorm(p, mean = 0, sd = 1)
  - Returns quantile function (“q” -> quantile)
- rnorm(n, mean = 0, sd = 1)
  - Returns random observations (“r”-> random)

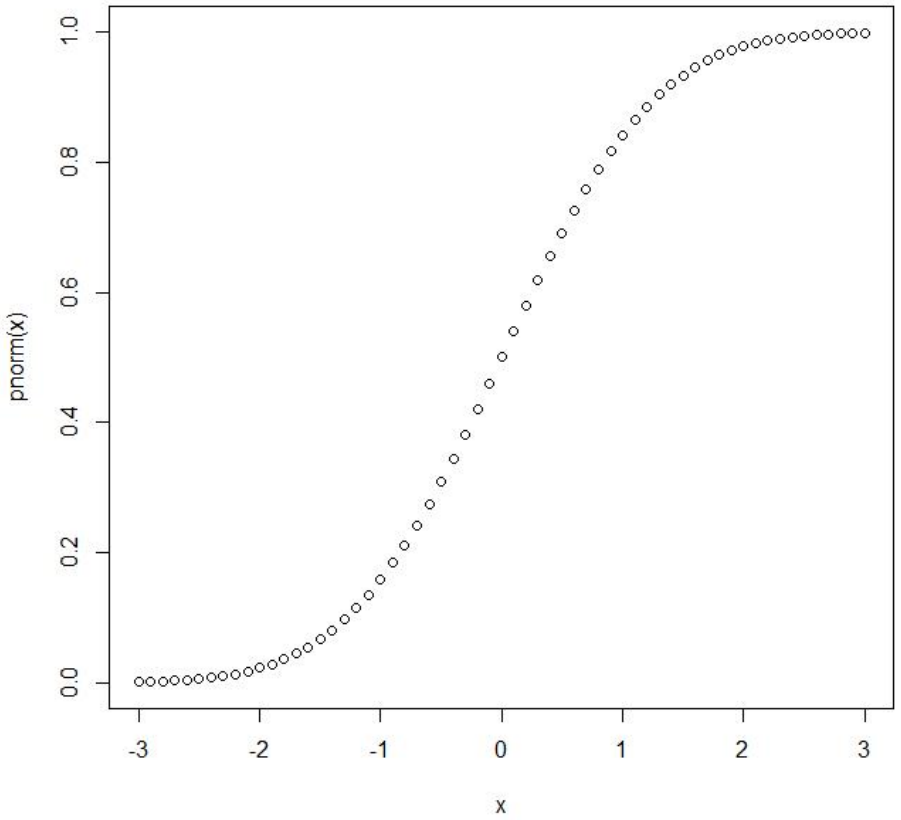
Similar: dbinom, pbinom, qbinom, rbinom

Examples

```
> x<-seq(-3,3,0.1)
> plot(x,dnorm(x))
```



> plot(x,pnorm(x))



You should know…

- The following R objects and their operations
  - Vectors
  - Matrices
  - Data Frames
  - Lists
  - Functions
- R loops (for loop)
- R conditional execution (IF ELSE statement)
- R distribution-related functions

Let us do the plots!

Example

Step 1. Create a dataset "simulation" by simulating 200 observations from the following linear model:

$$Y = \text{alpha} + \text{beta1} * X1 + \text{beta2} * X2 + \text{noise}$$

where

- alpha=1, beta1=2, beta2=-1.5
- X1 ~ N(1, 4), X2 ~ N(3,1), noise ~ N(0,1)

Step 2. Define a new binary variable Y_bin such that Y_bin=1 if Y>0 and Y_bin=0 otherwise.

Step 3. Make the final data contain only 4 variables: X1, X2, Y and Y_bin.

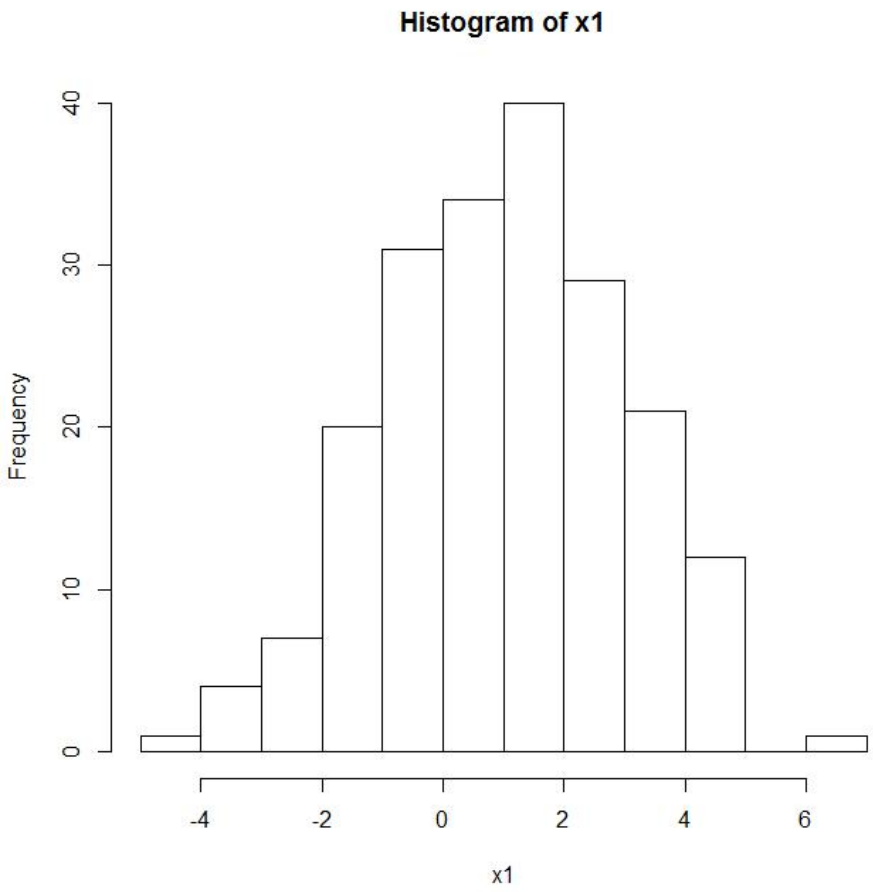
Code

```
> x1<-rnorm(200,mean=1,sd=2)
> x2<-rnorm(200,mean=3,sd=1)
> noise<-rnorm(200)
> y<-1+2*x1-1.5*x2+noise
> y_bin<-rep(0,200)
> y_bin[y>0]<-1
> simulation<-data.frame(x1,x2,y,y_bin)
```

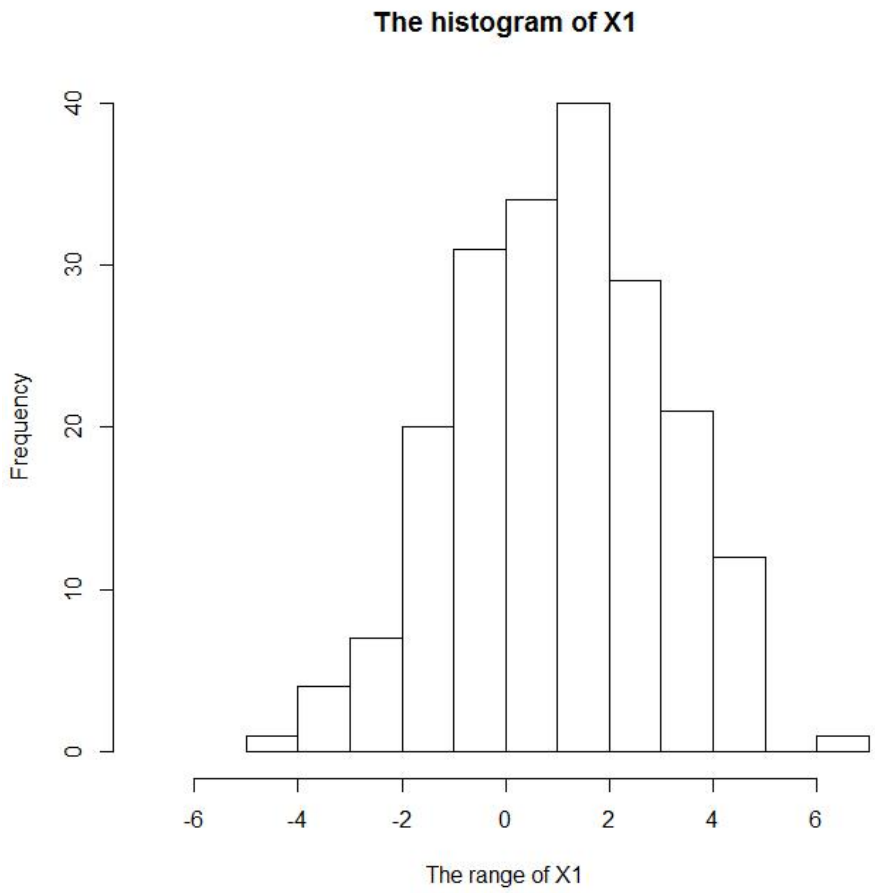
Plots that show the distribution of data values

- Histograms -- hist(…)
- Density plots – plot(density(…))
- Boxplots – boxplot(…)
- Normal probability plots – qqnorm(…)

> hist(x1)



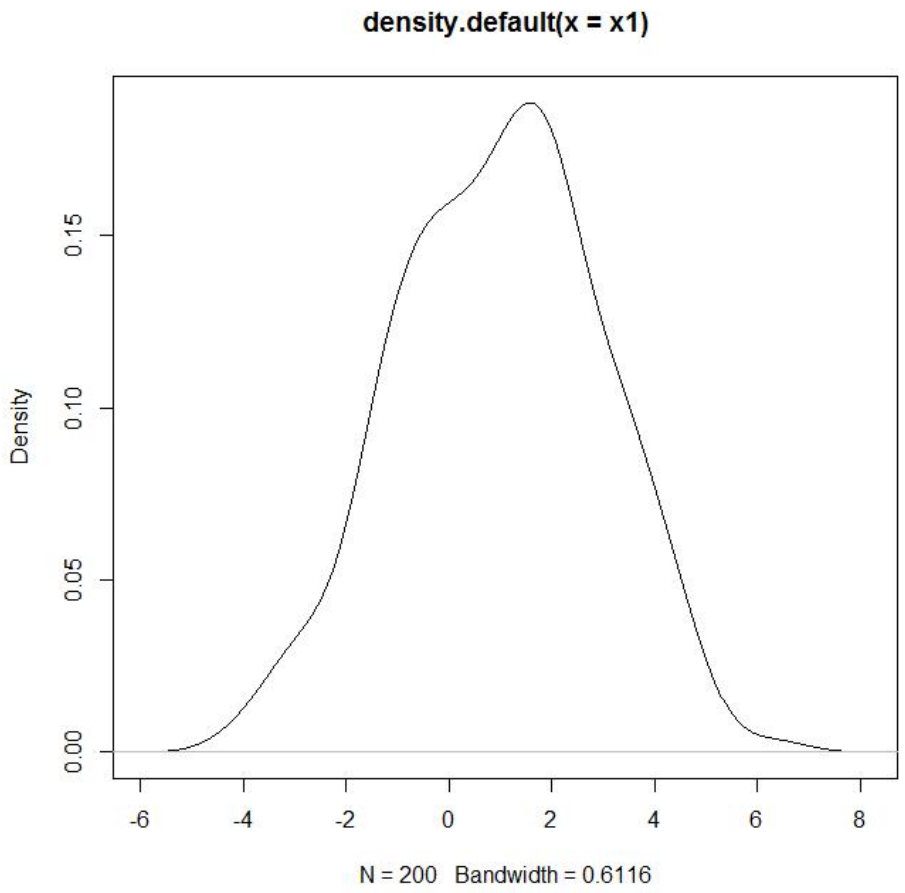
```
> hist(x1,xlim=c(-7,7),xlab="The range of X1",main="The histogram of X1")
```



Common options

- **xlim=c(a,b)** or ylim=c(a,b)
  - Specify the range of x (y) axis [a,b]
- **xlab="ZZZ"** or ylab="ZZZ"
  - Specify the label attached to the x (y) axis as ZZZ
- **main="ZZZ"**
  - Specify the title attached to the plot as ZZZ

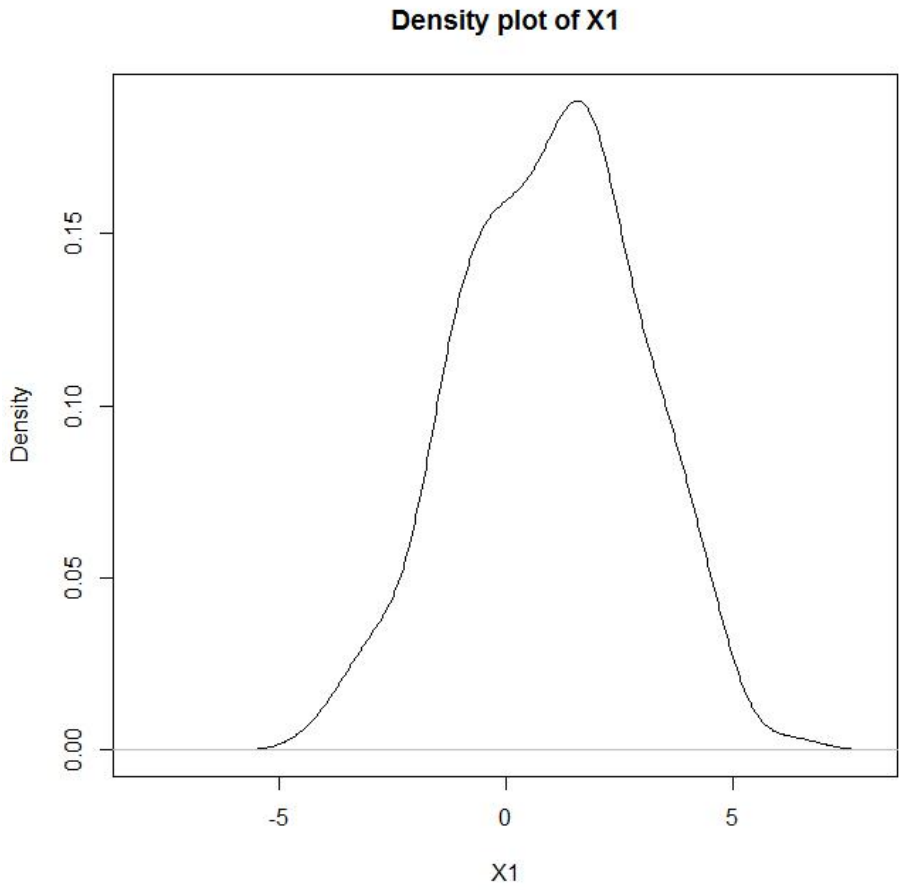
> plot(density(x1))



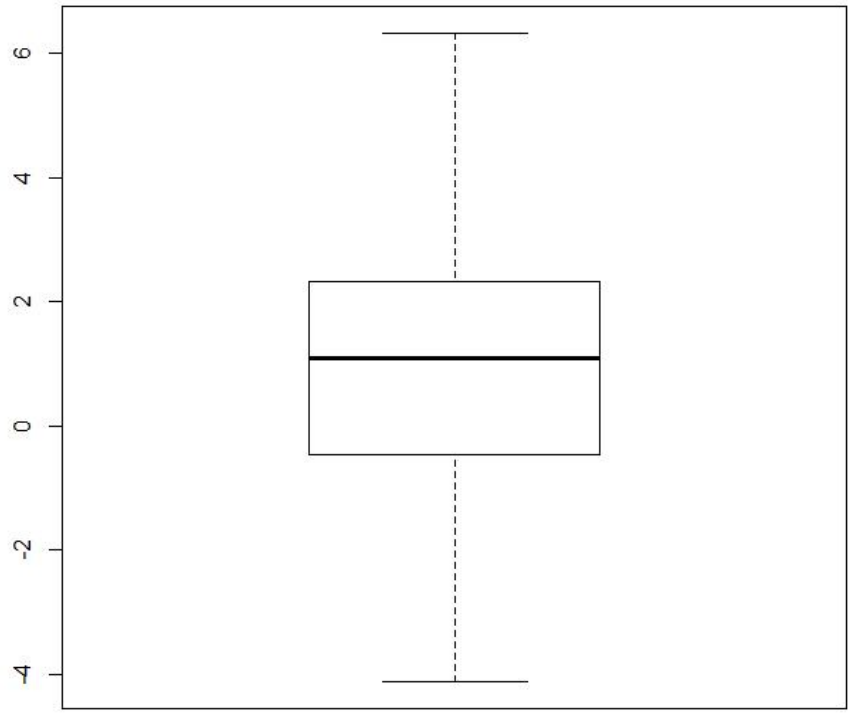
Do the following to the density plot

1. Change the range of x1 to [-8,8].
2. Set the title as “Density plot of X1”.
3. Set the x-label as “X1”.

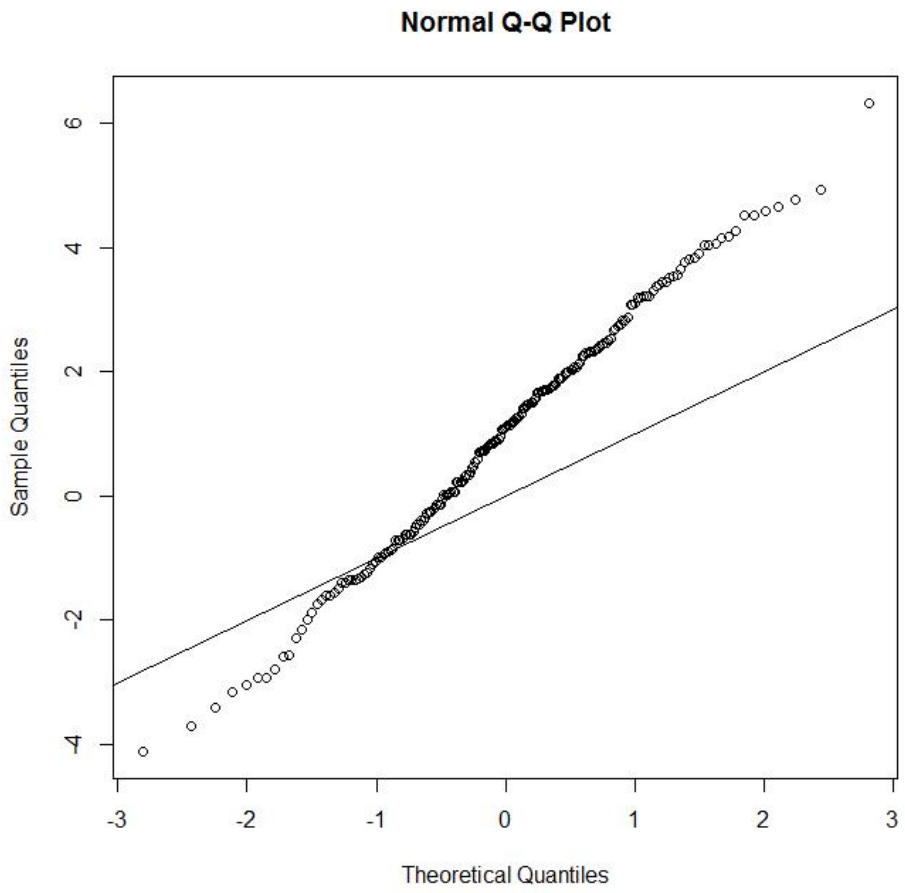
```
> plot(density(x1),xlim=c(-8,8),main="Density plot of
X1", xlab="X1")
```



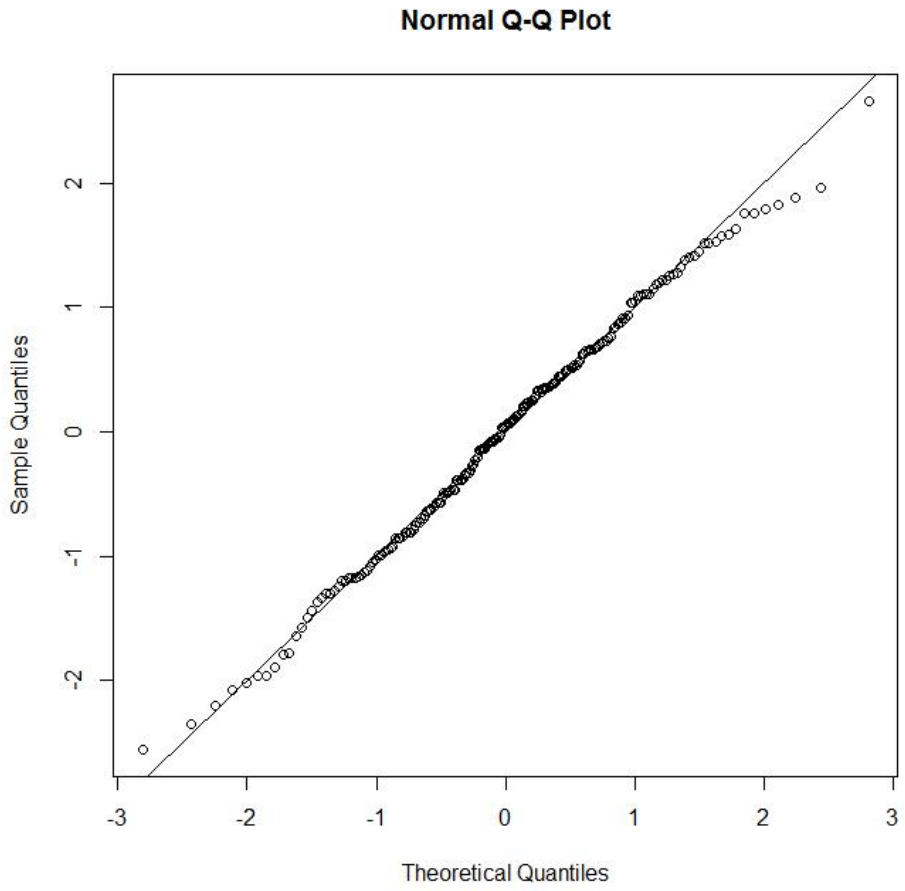
> boxplot(x1)



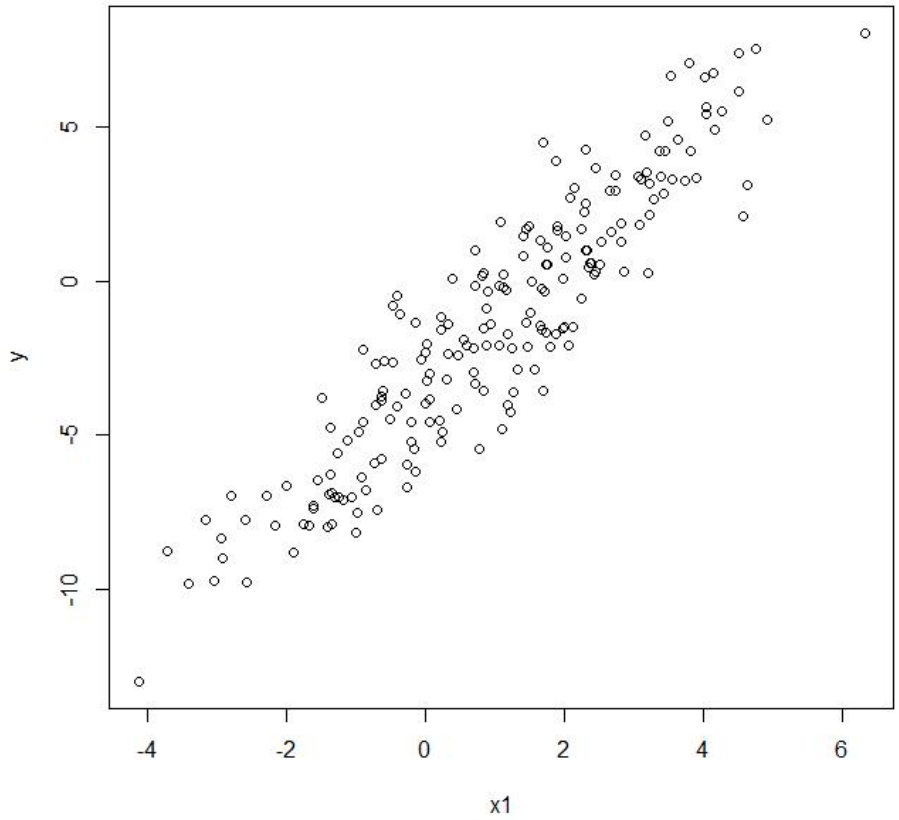
> qqnorm(x1)
> abline(0,1)



```
> qqnorm((x1-1)/2)
> abline(0,1)
```



X-Y plots {**plot(y~x1) or plot(x1,y)**}



Common options

> plot(y~x1,**pch**=2)

Specify the symbol for plotting points.

> plot(y~x1,**col**="red")

Specify the color for plotting points.

Try the follow commands:

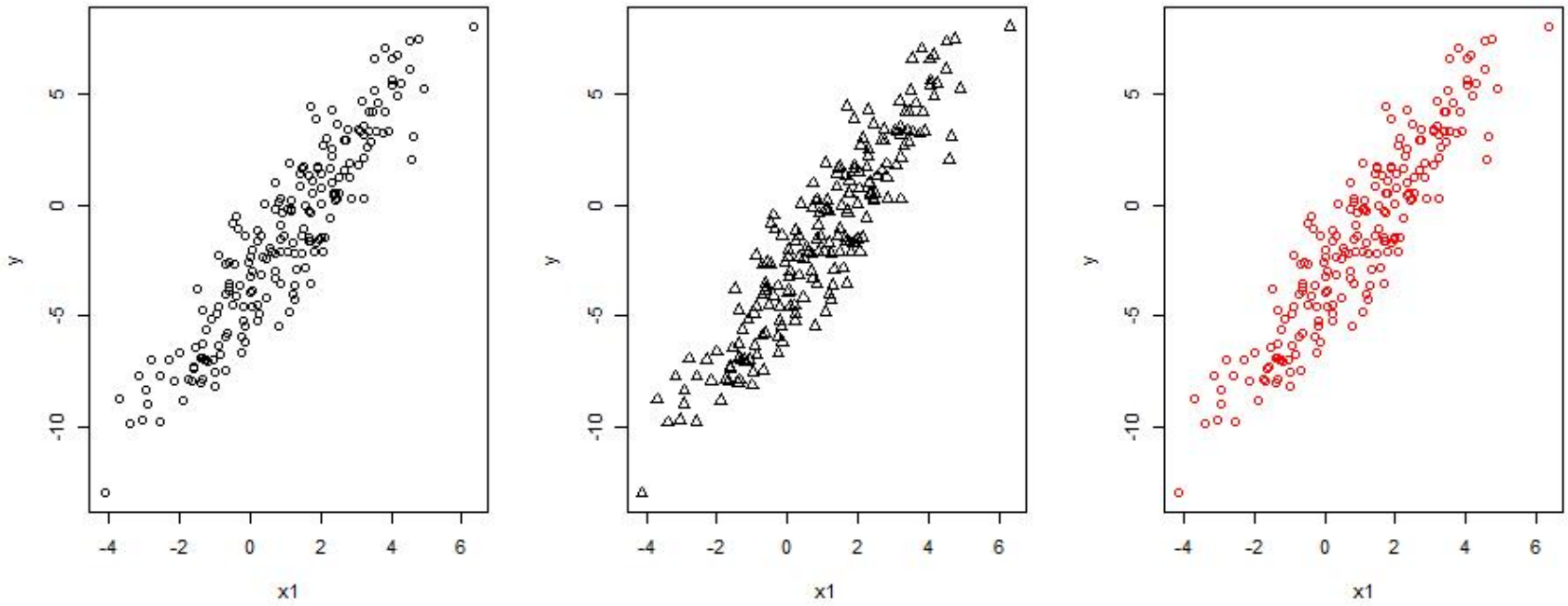
> **par(mfrow=c(1,3))**

> plot(y~x1)

> plot(y~x1,pch=2)

> plot(y~x1,col="red")

Multiple plots in one figure

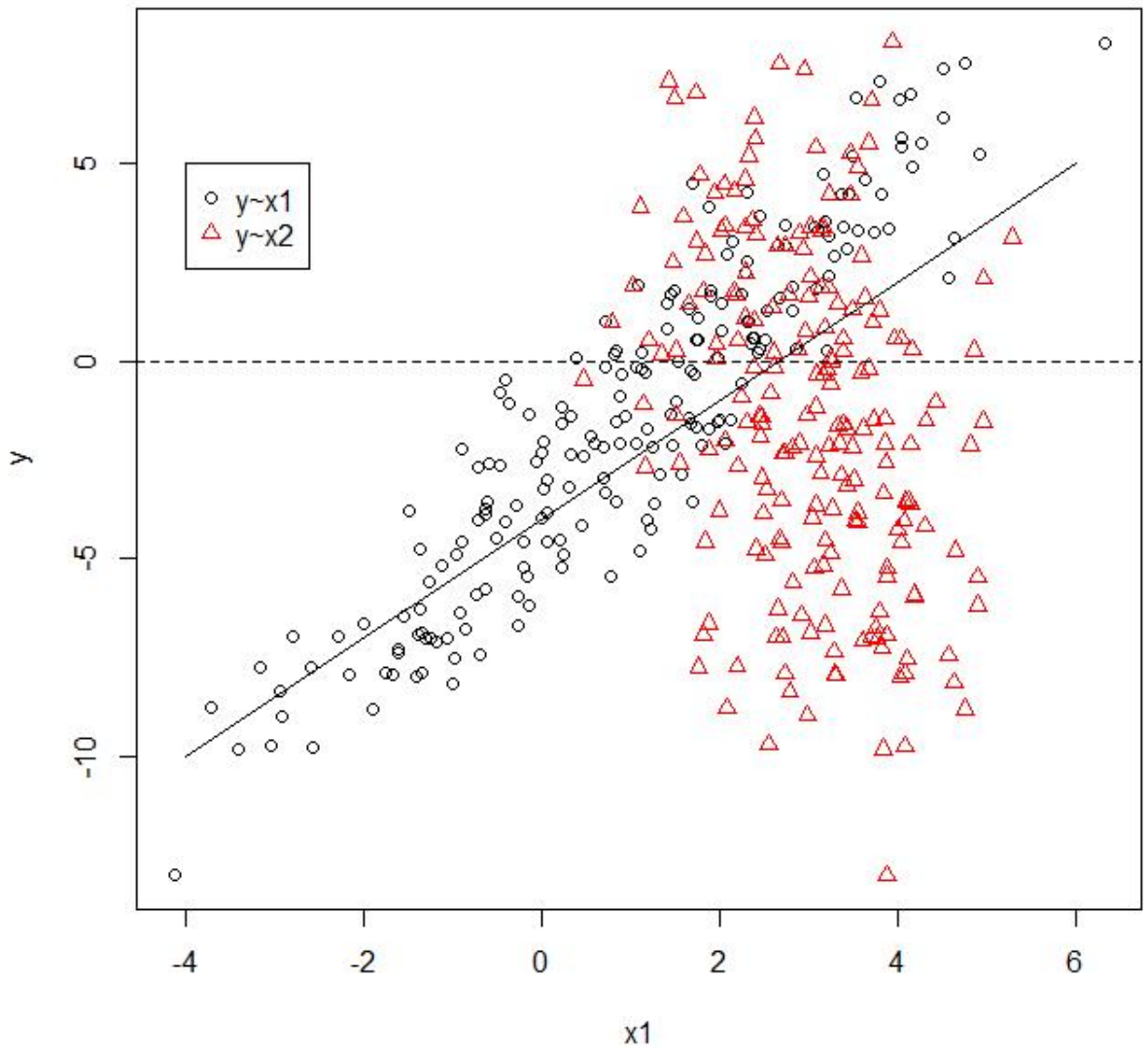


Add points or lines to an existing plot

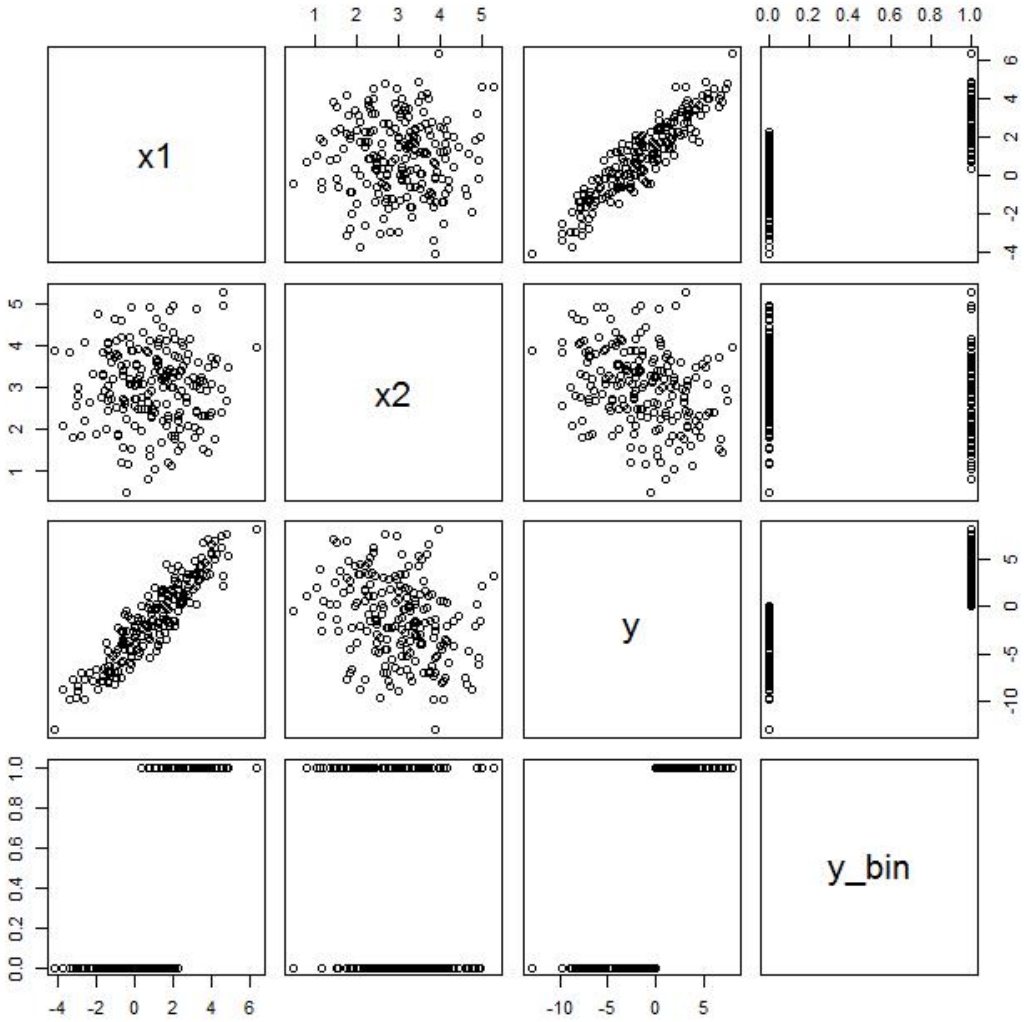
```
> plot(y~x1)
> points(y~x2,pch=2,col="red")
> abline(h=0,lty=2)
> lines(x=c(-4,6),y=c(-10,5))
> legend(-4,5,legend=c("y~x1","y~x2"),pch=c(1,2),col=c("black","red"))
```

Tips and tricks:

- The function points(…) adds points to an existing plot
- The functions abline(…) and lines(…) adds lines to an existing plot
- The function legend(…) attach a legend to an existing plot.


y~x1
y~x2
y
x1
5
0
-5
-10
-4
-2
0
2
4
6

Pairwise plots (pairs(simulation))



Comments

R provides useful functions for representing multivariate data.

If DATA is a numeric matrix or data frame, the command

> **pairs**(DATA)

produces a pairwise scatterplot matrix of the variables defined by the columns of DATA,

- that is, every column of DATA is plotted against every other column of DATA and the resulting *k(k-1)* plots are arranged in a matrix with plot scales constant over the rows and columns of the matrix.

Conditional plot

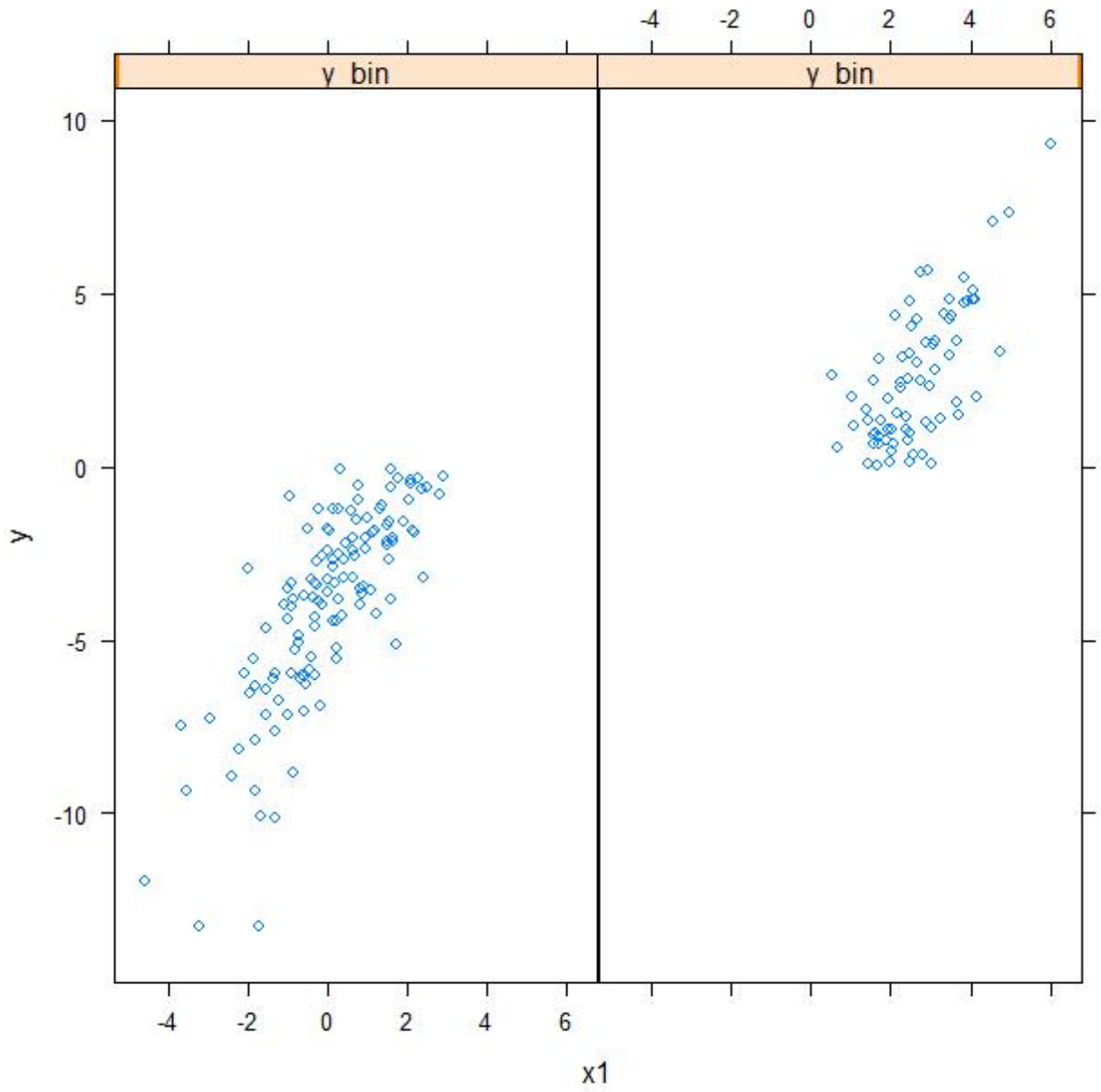
R can also produce X-Y plots conditional on some group variables.

> **xyplot(y~x1 | y_bin)**

The function xyplot(…) is in the package “lattice”

> install.packages("lattice")

> library("lattice")


y bin
y bin
-4
-2
0
2
4
6
10
5
0
-5
-10
y
x1

The correlation matrix

The function cor(…) gives the full correlation matrix for the variables in a data frame.

```
> cor(simulation)
                x1          x2          y      y_bin
x1     1.00000000  0.04205596  0.8903889  0.7062069
x2     0.04205596  1.00000000 -0.3359942 -0.3109189
y      0.89038895 -0.33599418  1.0000000  0.7860565
y_bin  0.70620694 -0.31091891  0.7860565  1.0000000
```

Linear regression

- General form

  lm(*response ~ var1+var2+…+vark*, data=…)

Note: “lm” stands for linear model

Example:

```
> model<-lm(y~x1+x2)
> class(model)
```

Check the results of…

```
> model
> summary(model)
> names(model)
> model$residuals
> model$fitted
> coef(model)
> vcov(model)
```

> summary(model)

```
Call:
lm(formula = y ~ x1 + x2)

Residuals:
    Min      1Q  Median      3Q     Max
-2.4374 -0.6016  0.0075  0.7182  3.2049

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.82360    0.22958   3.588 0.000421 ***
x1           2.00096    0.04091  48.907  < 2e-16 ***
x2          -1.43443    0.07104 -20.192  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.05 on 197 degrees of freedom
Multiple R-squared: 0.9325,     Adjusted R-squared: 0.9318
F-statistic:  1361 on 2 and 197 DF,  p-value: < 2.2e-16
```

> names(model)

```
> names(model)
 [1] "coefficients"  "residuals"     "effects"
 [4] "rank"          "fitted.values" "assign"
 [7] "qr"            "df.residual"   "xlevels"
[10] "call"          "terms"         "model“
```

Note:

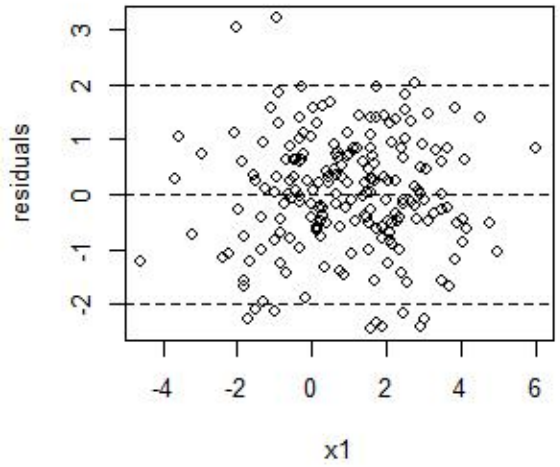
- “model” is an “lm” object.
- The command “names(model)” gives out the names of all elements in the “lm” object “model”.
- These elements can be called using their corresponding names. For example, model$residual, model$fitted

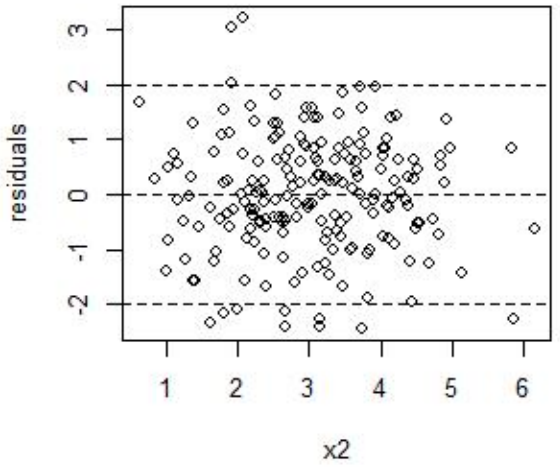
```
> coef(model)  # Extracts the coefficients
(Intercept)          x1          x2
  0.8236048   2.0009571  -1.4344263


> vcov(model) # Extracts the variance-covariance matrix
             (Intercept)            x1            x2
(Intercept)  0.052704697 -0.0012452737 -0.0151757639
x1          -0.001245274  0.0016738863 -0.0001222336
x2          -0.015175764 -0.0001222336  0.0050466167
```

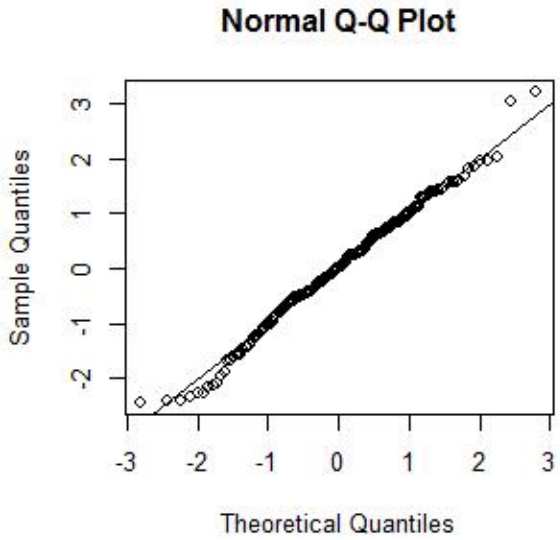
A check list for model diagnostics

1. Independent.
   - Plot the residuals versus X1 and X2 (PROC PLOT).
2. Normally distributed.
   - Plot the histogram of the residuals (PROC CHART)
   - Perform hypothesis testing (PROC UNIVARIATE).
3. Mean 0.
   - Check the residual plots.
   - Perform hypothesis testing (PROC TTEST)
4. Constant variance.
   - Check the residual plots

```
residuals<-model$res
par(mfrow=c(2,2))
plot(x1,residuals)
abline(h=c(-2,0,2),lty=2)
plot(x2,residuals)
abline(h=c(-2,0,2),lty=2)
qqnorm(residuals)
abline(0,1)
```


residuals
3
2
1
0
-1
-2
-4
-2
0
2
4
6
x1


residuals
3
2
1
0
-1
-2
1
2
3
4
5
6
x2


Normal Q-Q Plot
Sample Quantiles
3
2
1
0
-1
-2
-3
-2
-1
0
1
2
3
Theoretical Quantiles