Introduction to R statistical software

BCAM - Basque Center for Applied Mathematics, Applied Statistics

Dae-Jin Lee < dlee@bcamath.org >

Azti - Tecnalia

Contents

This course is an introduction to statistical software R. The course will introduce the student to the basics of using R for statistical programming, computation, graphics, and basic statistical modeling.

Objectives:

- Use R for basic data analysis.
- Write functions in and use R in efficient way,
- Perform basic statistical analysis and basic statistical models.

Pre-requisites:

There are no formal prerequisites, but some basic knowledge of statistics is expected. The intended audience is anyone who needs a flexible statistical environment for their research. Any prior knowledge of R is not expected.

Compulsory:

- Bring your laptop with the latest version of R installed. Download it here
- Rstudio is a user-friendly interface. Download it here **Highly recommended!!!**
- Internet connection to download the material and install some packages.

---

1 Introduction to the R language

- R is a free software environment for statistical computing and graphics http://www.r-project.org
- R is a command-driven statistical package.
- The most important reasons to use R are:

  - R is free and multiplatform (Windows/Linux/MACos)
  - R allows you to do all the statistical tests/models/analysis you need :)
  - Excellent graphics and programming capabilities
  - Growing community of users and developers
  - Lots of online resources

- Statistical features:

  - Graphical Techniques (Exploratory Data Analysis)
  - Linear and non-linear modeling (linear regression, non-parametric regression, smoothing, etc . . .)
  - Classical statistical tests
  - Time-series analysis
  - Econometrics
  - Survival analysis
  - Classification and clustering (data mining, machine learning)
  - Optimization and Mathematical Programming
  - Bayesian inference etc . . . .

Visit http://cran.r-project.org/web/views or http://stackoverflow.com/questions/tagged/r

1.1 Start with R

- Get current working directory

```
getwd()
```

- list the objects in the current workspace

```
ls()
```

- Set working directory

```
setwd("/Users/dlee")
```

- work with your previous commands

```
history() # display last 25 commands
history(max.show=Inf) # display all previous commands
```

- save your command history

```
savehistory(file="myfile") # default is ".Rhistory"
```

- recall your command history

```
loadhistory(file="myfile") # default is ".Rhistory"
```

- save the workspace to the file `.RData`

```
save.image()
```

- save specific objects to a file if you don't specify the path, the cwd is assumed

```
save(<object list>,file="myfile.RData")
```

- load a workspace into the current session

```
load("myfile.RData")
```

- quit `R`. You will be prompted to save the workspace.

```
q()
```

1.2 Install and load an `R` library

```
install.packages("DAAG") # (Data Analysis And Graphics)
```

- Once installed the package, load it

```
library(DAAG) # or require(DAAG)
```

1.3 Reading data

The `R` console

```
x <- c(7.82,8.00,7.95) # c means "combine"
x
```

```
## [1] 7.82 8.00 7.95
```

A quicker way is to use `scan()`

```
x <- scan()  # enter a number followed by return and blank line to end
1: 7.82
2: 8.00
3: 7.95
4:
Read 3 items
```

To create a character vector use `""`

```
id <- c("John","Paul","George","Ringo")
```

To read a character vector

```
id <- scan(,"")
1: John
2: Paul
3: George
4: Ringo
5:
Read 4 items
```

```
id
```

```
## [1] "John"   "Paul"   "George" "Ringo"
```

1.4 Data Import

In most situations, we need to read data from a separate data file. There are several methods for doing this.

- `scan()` (see `?scan` for help)

```r
cat("Example:", "2 3 5 7", "11 13 17", file = "ex.txt", sep = "\n") # creates ex.txt
scan("ex.txt", skip = 1)
```

```
## [1]  2  3  5  7 11 13 17
```

```r
scan("ex.txt", skip = 1, nlines = 1) # only 1 line after the skipped one
```

```
## [1] 2 3 5 7
```

```r
unlink("ex.data") # tidy up
```

- Several formats are available (`.txt`, `.csv`, `.xls`, `.xlsx`, `SAS`, `Stata`, etc…)
- Some `R` libraries to import data are

```r
library(gdata)
library(foreign)
```

- Read data from a `.txt` or `.csv` files

```r
mydata1 = read.table("mydata.txt")
mydata2 = read.csv("mydata.csv")
```

- Other formats `.xls` and `.xlsx`

```r
# read in the worksheet named mysheet
mydata <- read.xlsx("myexcel.xlsx", sheetName = "mysheet")
```

- Minitab, SPSS, SAS or Stata

```r
library(foreign)
mydata = read.mtp("mydata.mtp")  # Minitab
mydata = read.spss("myfile", to.data.frame=TRUE) # SPSS
mydata = read.dta("mydata.dta") # Stata
```

- Or

```r
library(Hmisc)
mydata = spss.get("mydata.por", use.value.labels=TRUE)  # SPSS
```

1.5 Exporting data

- There are numerous methods for exporting `R` objects into other formats . For SPSS, SAS and Stata. you will need to load the `foreign` packages. For Excel, you will need the `xlsx` package.
- Tab delimited text file

```r
mtcars
?mtcars
write.table(mtcars, "mydata.txt", sep="\t")
```

- Excel spreadsheet

```r
library(xlsx)
write.xlsx(mydata, "mydata.xlsx")
```

1.6 Data vectors

- Download `R code` here
- Create a vector of weights and heights

```r
weight<-c(60,72,57,90,95,72)
class(weight)
```

```
## [1] "numeric"
```

```r
height<-c(1.75,1.80,1.65,1.90,1.74,1.91)
```

- calculate Body Mass Index

```
bmi<- weight/height^2
bmi
```

```
## [1] 19.59184 22.22222 20.93664 24.93075 31.37799 19.73630
```

1.7 Basic statistics

- mean, median, st dev, variance

```
mean(weight)
median(weight)
sd(weight)
var(weight)
```

- summarize data

```
summary(weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   57.00   63.00   72.00   74.33   85.50   95.00
```

- or

```
min(weight)
max(weight)
range(weight)
sum(weight)
length(weight)
```

- Quantiles and percentile

There are several quartiles of an observation variable. The first quartile, or lower quartile, is the value that cuts off the first 25% of the data when it is sorted in ascending order. The second quartile, or median, is the value that cuts off the first 50%. The third quartile, or upper quartile, is the value that cuts off the first 75%.

```r
quantile(weight)
```

```
##   0%  25%  50%  75% 100% 
## 57.0 63.0 72.0 85.5 95.0
```

The n^{th} percentile of an observation variable is the value that cuts off the first n percent of the data values when it is sorted in ascending order.

```r
quantile(weight,c(0.32,0.57,0.98))
```

```
##  32%  57%  98% 
## 67.2 72.0 94.5
```

- Covariance and correlation

The *covariance* of two variables x and y in a data sample measures how the two are linearly related. A positive covariance would indicates a positive linear relationship between the variables, and a negative covariance would indicate the opposite.

$$\mathrm{Cov(x,y)} = \frac{1}{\mathrm{n}}\sum_{\mathrm{i}=1}^{\mathrm{n}}(\mathrm{x_i}-\bar{\mathrm{x}})(\mathrm{y_i}-\bar{\mathrm{y}})$$

```r
cov(weight,height)
```

```
## [1] 0.6773333
```

The *correlation coefficient* of two variables in a data sample is their covariance divided by the product of their individual standard deviations. It is a normalized measurement of how the two are linearly related.

Formally, the sample correlation coefficient is defined by the following formula, where σ_x and σ_y are the sample standard deviations, and $\sigma_x y$ is the covariance.

$$\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \ \sigma_y}$$

```r
cor(weight,height)
```

```
## [1] 0.437934
```

1.8 Character vectors and factor variables

```
subject <- c("John","Peter","Chris","Tony","Mary","Jane")
sex <- c("MALE","MALE","MALE","MALE","FEMALE","FEMALE")
class(subject)
```

```
## [1] "character"
```

```
table(sex)
```

```
## sex
## FEMALE   MALE
##      2      4
```

1.9 Data frames

```
Dat <- data.frame(subject,sex,weight,height)
# add bmi to Dat
Dat$bmi <- bmi  # or Dat$bmi <- weight/height^2
class(Dat)
```

```
## [1] "data.frame"
```

```
str(Dat) # display object structure
```

```
## 'data.frame':    6 obs. of  5 variables:
##  $ subject: Factor w/ 6 levels "Chris","Jane",..: 3 5 1 6 4 2
##  $ sex    : Factor w/ 2 levels "FEMALE","MALE": 2 2 2 2 1 1
##  $ weight : num  60 72 57 90 95 72
##  $ height : num  1.75 1.8 1.65 1.9 1.74 1.91
##  $ bmi    : num  19.6 22.2 20.9 24.9 31.4 ...
```

```
# Change rownames
rownames(Dat)<-c("A","B","C","D","E","F")

# Access to data frame elements (similar to a matrix)
Dat[,1]      # 1st column
```

```
## [1] John  Peter Chris Tony  Mary  Jane 
## Levels: Chris Jane John Mary Peter Tony
```

```
Dat[,1:3]    # 1st to 3rd columns
```

```
##   subject    sex weight
## A    John   MALE     60
## B   Peter   MALE     72
## C   Chris   MALE     57
## D    Tony   MALE     90
## E    Mary FEMALE     95
## F    Jane FEMALE     72
```

```
Dat[1:2,]    # 1st to 2nd row
```

```
##   subject  sex weight height      bmi
## A    John MALE     60   1.75 19.59184
## B   Peter MALE     72   1.80 22.22222
```

1.10 Working with data frames

Example: Analyze data by groups

- Obtain the mean weight, heigth and bmi means by FEMALES and MALES:

1. Select each group and compute the mean

```
Dat[sex=="MALE",]
Dat[sex=="FEMALE",]

mean(Dat[sex=="MALE",3])  # weight average of MALEs
mean(Dat[sex=="MALE","weight"])
```

2. Use `apply` by columns

```
apply(Dat[sex=="FEMALE",3:5],2,mean)
apply(Dat[sex=="MALE",3:5],2,mean)

# we can use apply with our own function
apply(Dat[sex=="FEMALE",3:5],2,function(x){x+2})
```

3. `by` and `colMeans`

```
# 'by' splits your data by factors and do calculations on each subset.
by(Dat[,3:5],sex, colMeans)
```

4. aggregate

```
# another option
aggregate(Dat[,3:5], by=list(sex),mean)
```

1.11 Logical vectors

- Choose individuals with BMI>22

```
bmi
bmi>22
as.numeric(bmi>22) # convert a logical condition to a numeric value 0/1
which(bmi>22)  # gives the position of bmi for which bmi>22
```

- Which are between 20 and 25?

```
bmi > 20 & bmi < 25
which(bmi > 20 & bmi < 25)
```

1.12 Working with vectors

- Concatenate

```
x <- c(2, 3, 5, 2, 7, 1)
y <- c(10, 15, 12)
z <- c(x,y)  # concatenates x and y
```

- list two vectors

```
zz <- list(x,y) # create a list
unlist(zz) # unlist the list converting it to a concatenated vector
```

```
## [1]  2  3  5  2  7  1 10 15 12
```

- subset of vectors

```
x[c(1,3,4)]
```

```
## [1] 2 5 2
```

```
x[-c(2,6)] # negative subscripts omit the chosen elements
```

```
## [1] 2 5 2 7
```

- Sequences

```
seq(1,9) # or 1:9
```

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

```
seq(1,9,by=1)
```

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

```
seq(1,9,by=0.5)
```

```
##  [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0
```

```
seq(1,9,length=20)
```

```
##  [1] 1.000000 1.421053 1.842105 2.263158 2.684211 3.105263 3.526316
##  [8] 3.947368 4.368421 4.789474 5.210526 5.631579 6.052632 6.473684
## [15] 6.894737 7.315789 7.736842 8.157895 8.578947 9.000000
```

- Replicates

```
oops <- c(7,9,13)
rep(oops,3) # repeats the entire vector "oops" three times
rep(oops,1:3) # this function has the number 3 replaced
              #  by a vector with the three values (1,2,3)
              #  indicating that 7 should be repeated once, 9 twice and 13 three times.

rep(c(2,3,5), 4)
rep(1:2,c(10,15))

rep(c("MALE","FEMALE"),c(4,2)) # it also works with character vectors
c(rep("MALE",3), rep("FEMALE",2))
```

1.13 Matrices and arrays

```
x<- 1:12
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

```
dim(x)<-c(3,4)  # 3 rows and 4 columns

X <- matrix(1:12,nrow=3,byrow=TRUE)
X
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
```

```
X <- matrix(1:12,nrow=3,byrow=FALSE)
X
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
# rownames, colnames

rownames(X) <- c("A","B","C")
X
```

```
##   [,1] [,2] [,3] [,4]
## A    1    4    7   10
## B    2    5    8   11
## C    3    6    9   12
```

```
colnames(X) <- LETTERS[4:7]
X
```

```
##   D E F  G
## A 1 4 7 10
## B 2 5 8 11
## C 3 6 9 12
```

```
colnames(X) <- month.abb[4:7]
X
```

```
##   Apr May Jun Jul
## A   1   4   7  10
## B   2   5   8  11
## C   3   6   9  12
```

- Column/Row bind operations `cbind()`, `rbind()`

```
Y <- matrix(0.1*(1:12),3,4)

cbind(X,Y)  # bind column-wise
```

```
##   Apr May Jun Jul
## A   1   4   7  10 0.1 0.4 0.7 1.0
## B   2   5   8  11 0.2 0.5 0.8 1.1
## C   3   6   9  12 0.3 0.6 0.9 1.2
```

```
rbind(X,Y)  # bind row-wise
```

```
##   Apr May Jun  Jul
## A 1.0 4.0 7.0 10.0
## B 2.0 5.0 8.0 11.0
## C 3.0 6.0 9.0 12.0
##   0.1 0.4 0.7  1.0
##   0.2 0.5 0.8  1.1
##   0.3 0.6 0.9  1.2
```

1.14 Factors

```
gender<-c(rep("female",691),rep("male",692))
class(gender)
```

```
## [1] "character"
```

```
# change vector to factor (i.e. a category)
gender<- factor(gender)
levels(gender)
```

```
## [1] "female" "male"
```

```
summary(gender)
```

```
## female   male
##    691    692
```

```
table(gender)
```

```
## gender
## female   male
##    691    692
```

```
status<- c(0,3,2,1,4,5)     # This command creates a numerical vector pain,
                            #    encoding the pain level of five patients.
fstatus <- factor(status, levels=0:5)
levels(fstatus) <- c("student","engineer","unemployed","lawyer","economist","dentist")

Dat$status <- fstatus
Dat
```

```
##   subject    sex weight height      bmi     status
## A    John   MALE     60   1.75 19.59184    student
## B   Peter   MALE     72   1.80 22.22222     lawyer
## C   Chris   MALE     57   1.65 20.93664 unemployed
## D    Tony   MALE     90   1.90 24.93075   engineer
## E    Mary FEMALE     95   1.74 31.37799  economist
## F    Jane FEMALE     72   1.91 19.73630    dentist
```

1.15 Indexing vector with logicals

```
a <- c(1,2,3,4,5)
b <- c(TRUE,FALSE,FALSE,TRUE,FALSE)

max(a[b])
```

```
## [1] 4
```

```
sum(a[b])
```

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

1.16 Missing values (NA)

```
a <- c(1,2,3,4,NA)
sum(a)
```

```
## [1] NA
```

```
sum(a,na.rm=TRUE)
```

```
## [1] 10
```

```
a <- c(1,2,3,4,NA)
is.na(a)
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE
```

1.17 Working with data frames

- A data frame is used for storing data tables. It is a list of vectors of equal length.

```
mtcars
?mtcars        # or help(mtcars)
```

- look at the first rows

```
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
```

- Structure of the data frame

```
str(mtcars) # display the structure of the data frame
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

- Select a car model:

```
mtcars["Mazda RX4",] # using rows and columns names
```

```
##           mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4  21   6  160 110  3.9 2.62 16.46  0  1    4    4
```

```
mtcars[c("Datsun 710", "Camaro Z28"),]
```

```
##             mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Datsun 710 22.8   4  108  93 3.85 2.32 18.61  1  1    4    1
## Camaro Z28 13.3   8  350 245 3.73 3.84 15.41  0  0    3    4
```

- Or specific variables

```r
mtcars[,c("mpg","am")]
```

```
##                      mpg am
## Mazda RX4           21.0  1
## Mazda RX4 Wag       21.0  1
## Datsun 710          22.8  1
## Hornet 4 Drive      21.4  0
## Hornet Sportabout   18.7  0
## Valiant             18.1  0
## Duster 360          14.3  0
## Merc 240D           24.4  0
## Merc 230            22.8  0
## Merc 280            19.2  0
## Merc 280C           17.8  0
## Merc 450SE          16.4  0
## Merc 450SL          17.3  0
## Merc 450SLC         15.2  0
## Cadillac Fleetwood  10.4  0
## Lincoln Continental 10.4  0
## Chrysler Imperial   14.7  0
## Fiat 128            32.4  1
## Honda Civic         30.4  1
## Toyota Corolla      33.9  1
## Toyota Corona       21.5  0
## Dodge Challenger    15.5  0
## AMC Javelin         15.2  0
## Camaro Z28          13.3  0
## Pontiac Firebird    19.2  0
## Fiat X1-9           27.3  1
## Porsche 914-2       26.0  1
## Lotus Europa        30.4  1
## Ford Pantera L      15.8  1
## Ferrari Dino        19.7  1
## Maserati Bora       15.0  1
## Volvo 142E          21.4  1
```

```r
library(psych)
describe(mtcars)
```

```
##      vars  n   mean     sd median trimmed    mad   min    max  range  skew
## mpg     1 32  20.09   6.03  19.20   19.70   5.41 10.40  33.90  23.50  0.61
## cyl     2 32   6.19   1.79   6.00    6.23   2.97  4.00   8.00   4.00 -0.17
## disp    3 32 230.72 123.94 196.30  222.52 140.48 71.10 472.00 400.90  0.38
## hp      4 32 146.69  68.56 123.00  141.19  77.10 52.00 335.00 283.00  0.73
## drat    5 32   3.60   0.53   3.70    3.58   0.70  2.76   4.93   2.17   0.27
## wt      6 32   3.22   0.98   3.33    3.15   0.77  1.51   5.42   3.91   0.42
## qsec    7 32  17.85   1.79  17.71   17.83   1.42 14.50  22.90   8.40   0.37
## vs      8 32   0.44   0.50   0.00    0.42   0.00  0.00   1.00   1.00   0.24
## am      9 32   0.41   0.50   0.00    0.38   0.00  0.00   1.00   1.00   0.36
## gear   10 32   3.69   0.74   4.00    3.62   1.48  3.00   5.00   2.00   0.53
## carb   11 32   2.81   1.62   2.00    2.65   1.48  1.00   8.00   7.00   1.05
##      kurtosis    se
## mpg     -0.37  1.07
## cyl     -1.76  0.32
## disp    -1.21 21.91
## hp      -0.14 12.12
## drat    -0.71  0.09
## wt      -0.02  0.17
## qsec     0.34  0.32
## vs      -2.00  0.09
## am      -1.92  0.09
## gear    -1.07  0.13
## carb     1.26  0.29
```

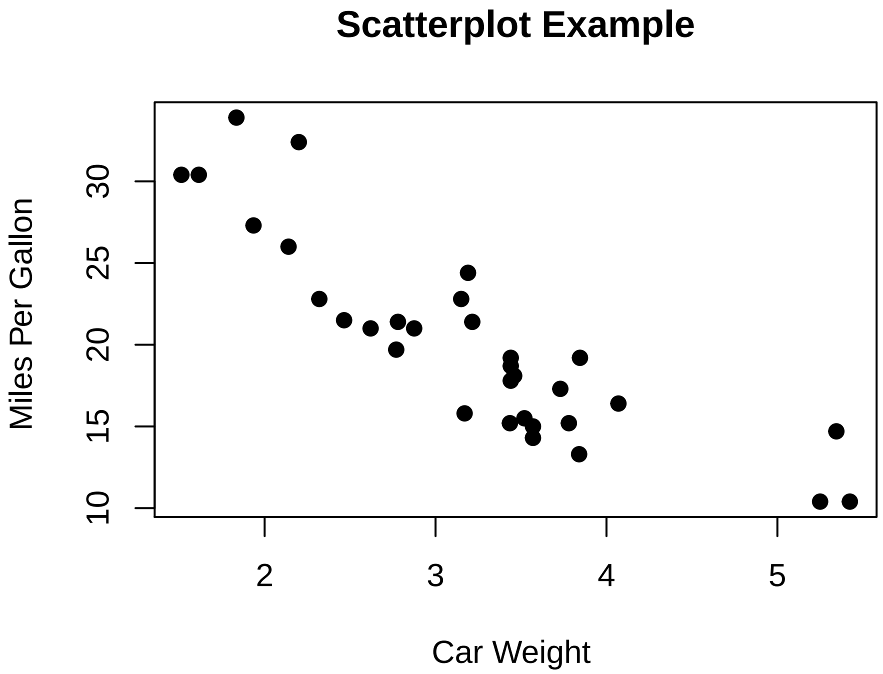
---

2 Basic data analysis in R

2.1 Basic plotting

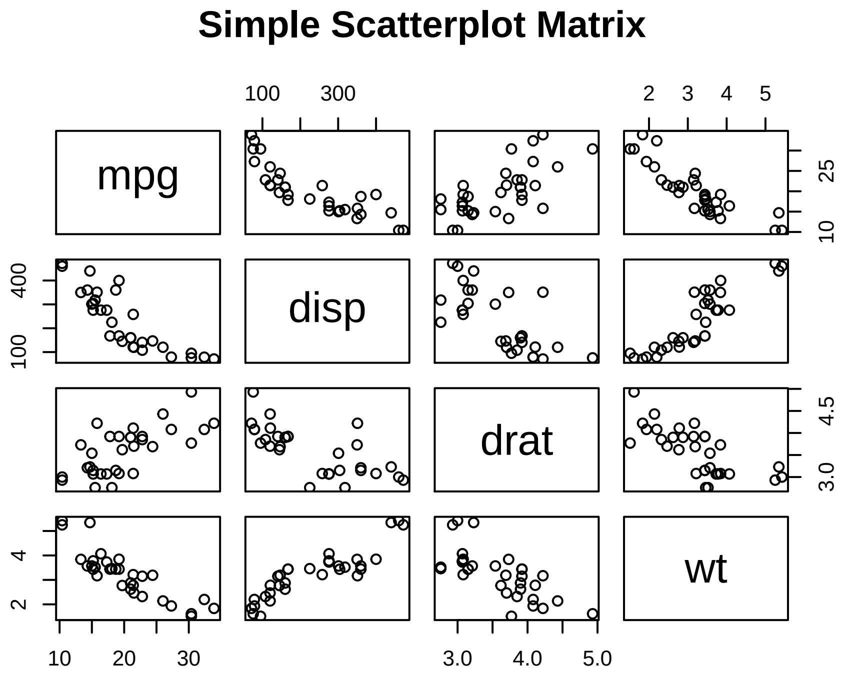
- Scatterplot

```
attach(mtcars)
plot(wt, mpg, main="Scatterplot Example",
   xlab="Car Weight ", ylab="Miles Per Gallon ", pch=19)
```



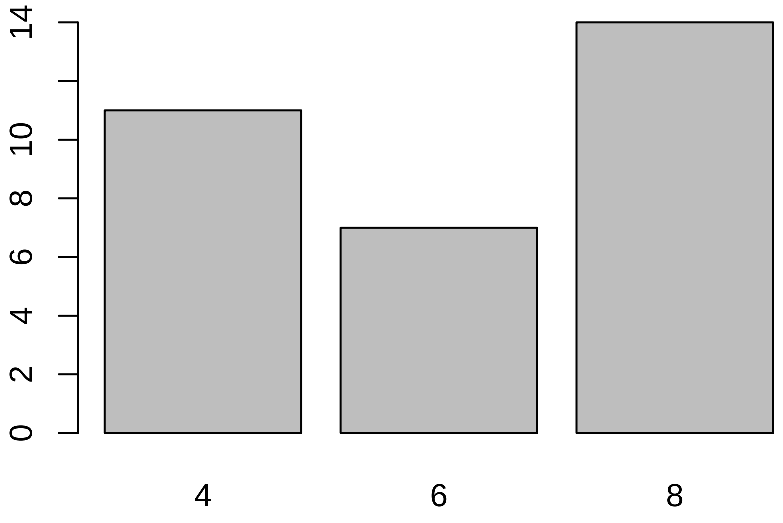
- Basic Scatterplot Matrix

```
pairs(~mpg+disp+drat+wt,data=mtcars,
   main="Simple Scatterplot Matrix")
```



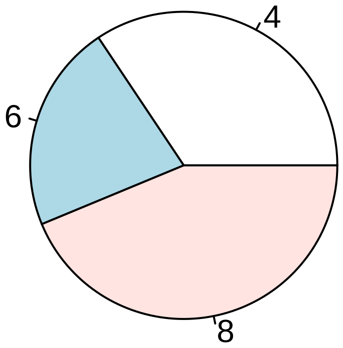
- Barplot

```
tab <- table(mtcars[,c("cyl")])
barplot(tab)
```



- Piechart

```
pie(tab)
```



Exercises:

1. The data.frame `VADeaths` contains the death rates per 1000 in Virginia (US) in 1940

- The death rates are measured per 1000 population per year. They are cross-classified by age group (rows) and population group (columns). The age groups are: 50-54, 55-59, 60-64, 65-69, 70-74 and the population groups are Rural/Male, Rural/Female, Urban/Male and Urban/Female.

```
data(VADeaths)
VADeaths
```

```
##       Rural Male Rural Female Urban Male Urban Female
## 50-54       11.7          8.7       15.4          8.4
## 55-59       18.1         11.7       24.3         13.6
## 60-64       26.9         20.3       37.0         19.3
## 65-69       41.0         30.9       54.6         35.1
## 70-74       66.0         54.3       71.1         50.0
```

- Compute the mean for each age group.

    - **Result:**

```
##  50-54  55-59  60-64  65-69  70-74
## 11.050 16.925 25.875 40.400 60.350
```

- Compute the mean for each population group.
  - **Result:**

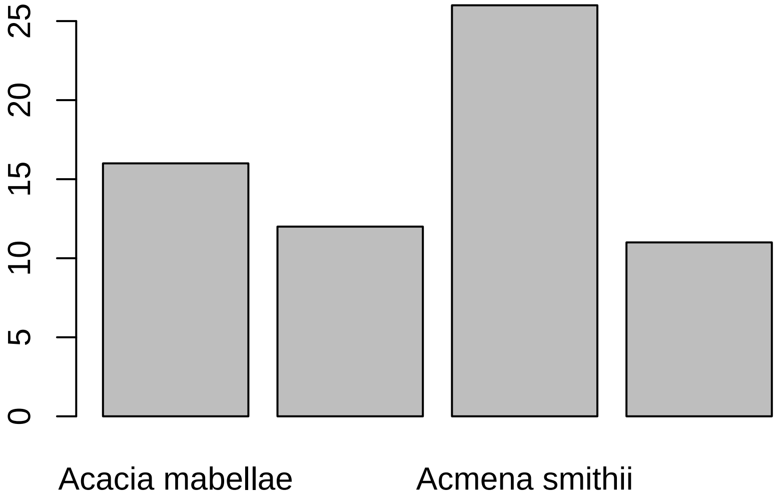
```
##   Rural Male Rural Female   Urban Male Urban Female 
##        32.74        25.18        40.48        25.28
```

2. The `data.frame` `rainforest` contains several variables from different `species`

```r
library(DAAG)
rainforest
```

- Create a table of counts for each `species` and make a graphic with the results.
  - **Result:**

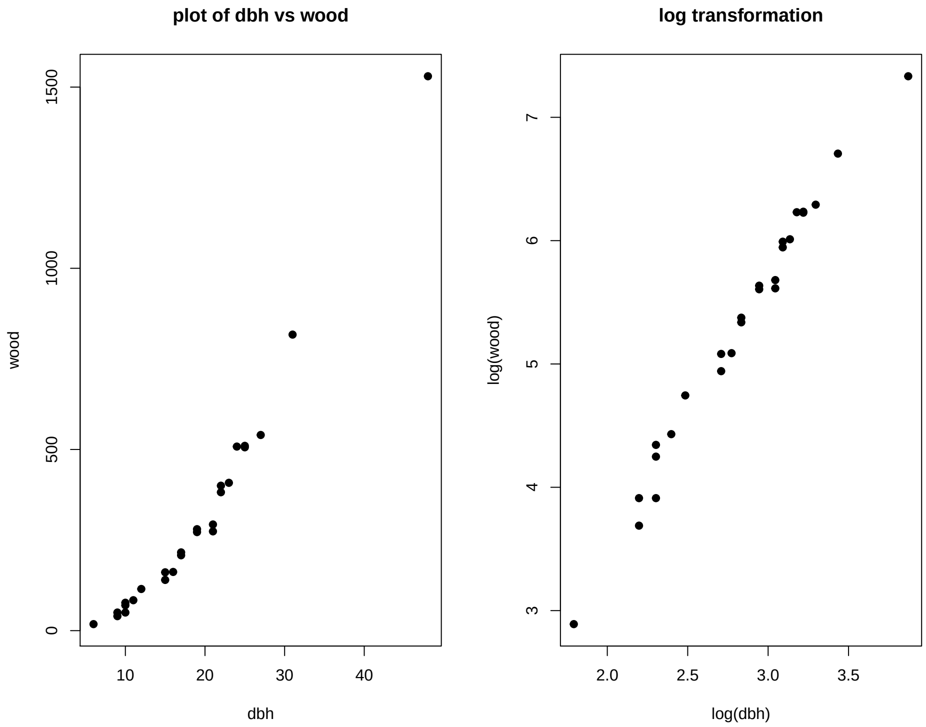
```
## 
## Acacia mabellae      C. fraseri  Acmena smithii   B. myrtifolia 
##              16              12              26              11
```



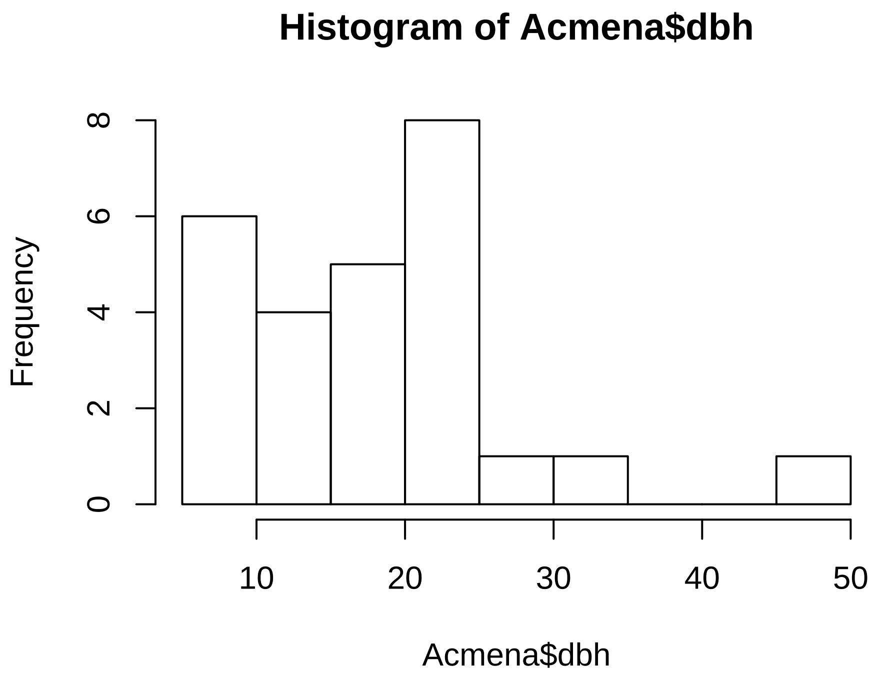
3. The `Acmena` `data.frame` is created from `rainforest` using the function `subset`.

- Plot the relationship between the wood biomass (`wood`) and the diameter of the breast height (`dbh`). Use also a logarithm scale.

```
Acmena <- subset(rainforest, species == "Acmena smithii")
```



- Compute a histogram of variable `dbh` using function `hist`



4. Create a vector of the positive odd integers less than 100 and remove the values greater than 60 and less than 80.

    - **Result:**

```
##  [1] 61 63 65 67 69 71 73 75 77 79
```

- Solutions here

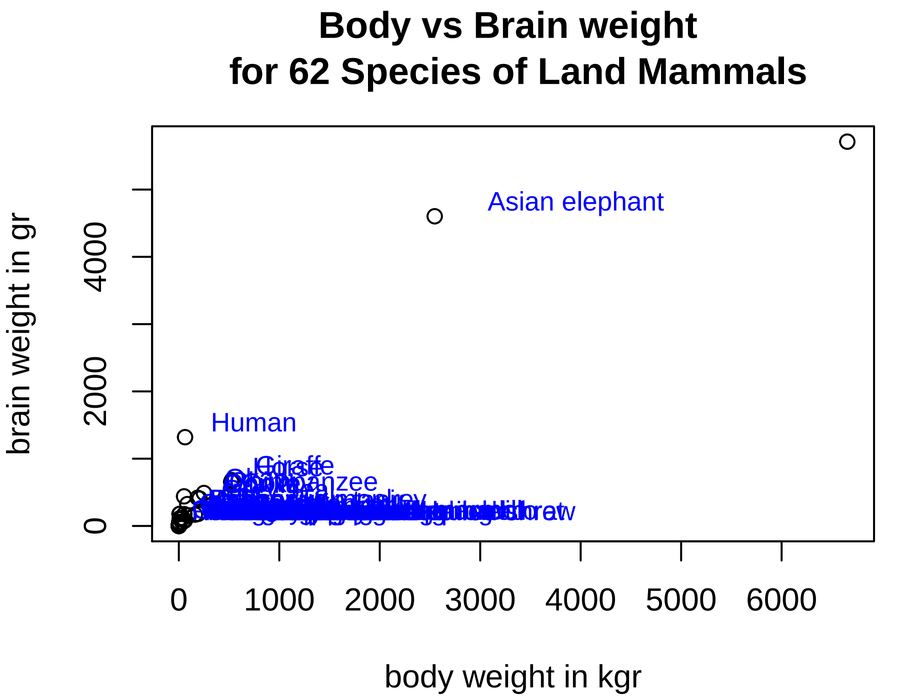
2.2 Scatterplots

```
library(MASS)
data("mammals")
?mammals
head(mammals)
```

```
##                            body brain
## Arctic fox                3.385  44.5
## Owl monkey         0.480  15.5
## Mountain beaver    1.350   8.1
## Cow              465.000 423.0
## Grey wolf         36.330 119.5
## Goat              27.660 115.0
```

```
attach(mammals)
species <- row.names(mammals)
x <- body
y <- brain
```

```
library(calibrate)
# scatterplot
plot(x,y, xlab = "body weight in kgr", ylab = "brain weight in gr",
     main="Body vs Brain weight \n for 62 Species of Land Mammals")
textxy(x,y,labs=species,col = "blue",cex=0.85)
```

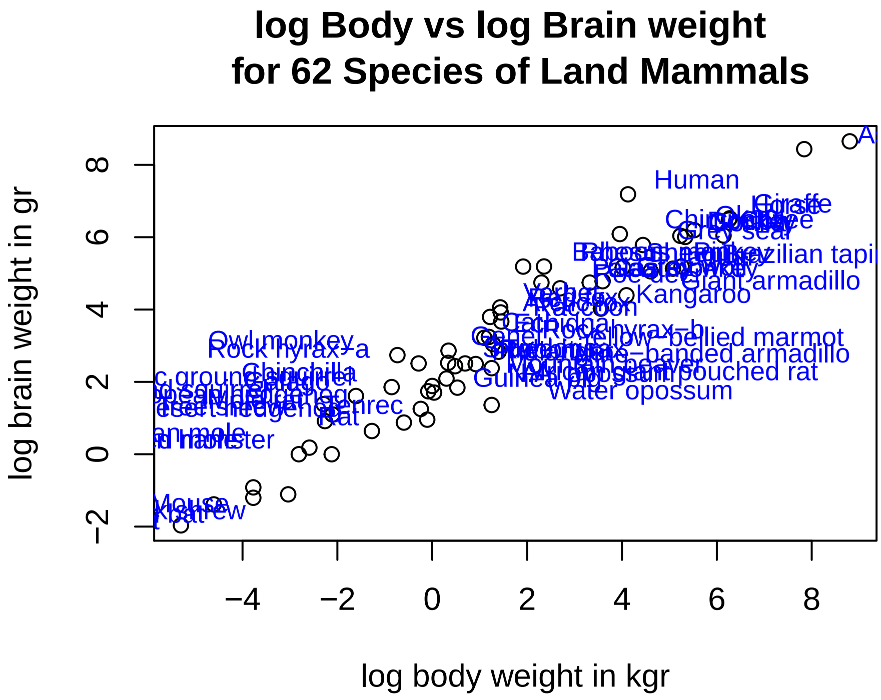


Identify a point in the scatterplot

```
identify(x,y,species)
```

Plot in the log scale

```r
plot(log(x),log(y), xlab = "log body weight in kgr", ylab = "log brain weight in gr",
     main="log Body vs log Brain weight \n for 62 Species of Land Mammals")
textxy(log(x),log(y),labs=species,col = "blue",cex=0.85)
```



Identify a point in the log scale scatterplot

```r
identify(log(x),log(y),species)
```

2.3 More plotting options

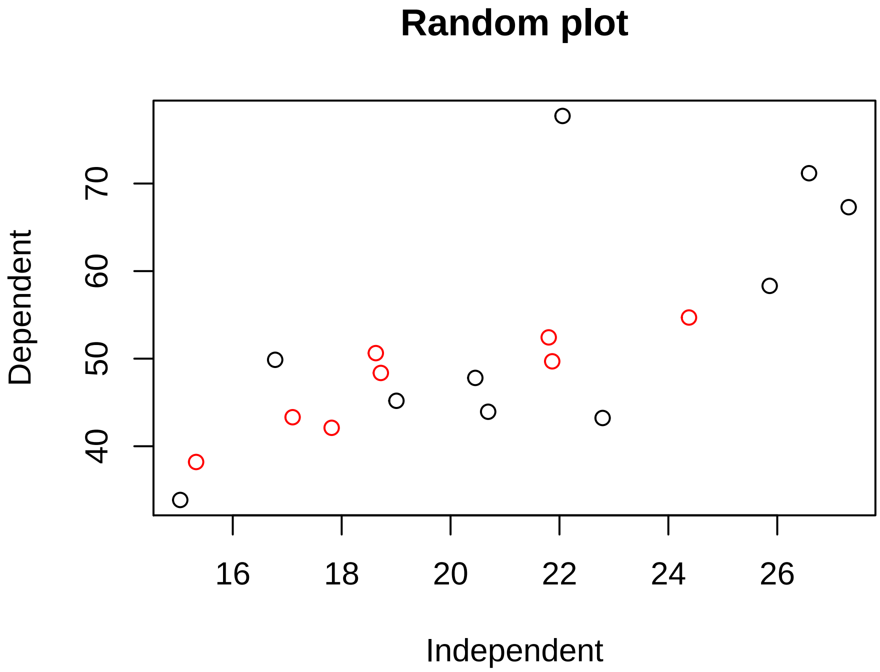
Multiple Data Sets on One Plot

One common task is to plot multiple data sets on the same plot. In many situations the way to do this is to create the initial plot and then add additional information to the plot. For example, to plot bivariate data the `plot` command is used to initialize and create the plot. The `points` command can then be used to add additional data sets to the plot.

```r
 x <- rnorm(10,sd=5,mean=20)
 y <- 2.5*x - 1.0 + rnorm(10,sd=9,mean=0)
 cor(x,y)
```

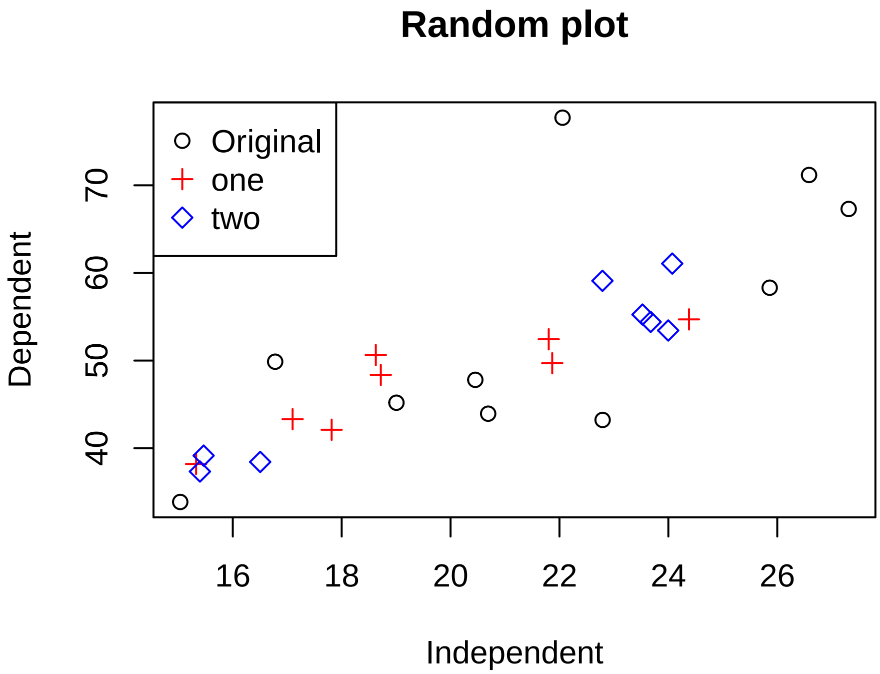
```
## [1] 0.7032196
```

```
plot(x,y,xlab="Independent",ylab="Dependent",main="Random plot")
x1 <- runif(8,15,25)
y1 <- 2.5*x1 - 1.0 + runif(8,-6,6)
points(x1,y1,col=2)
```



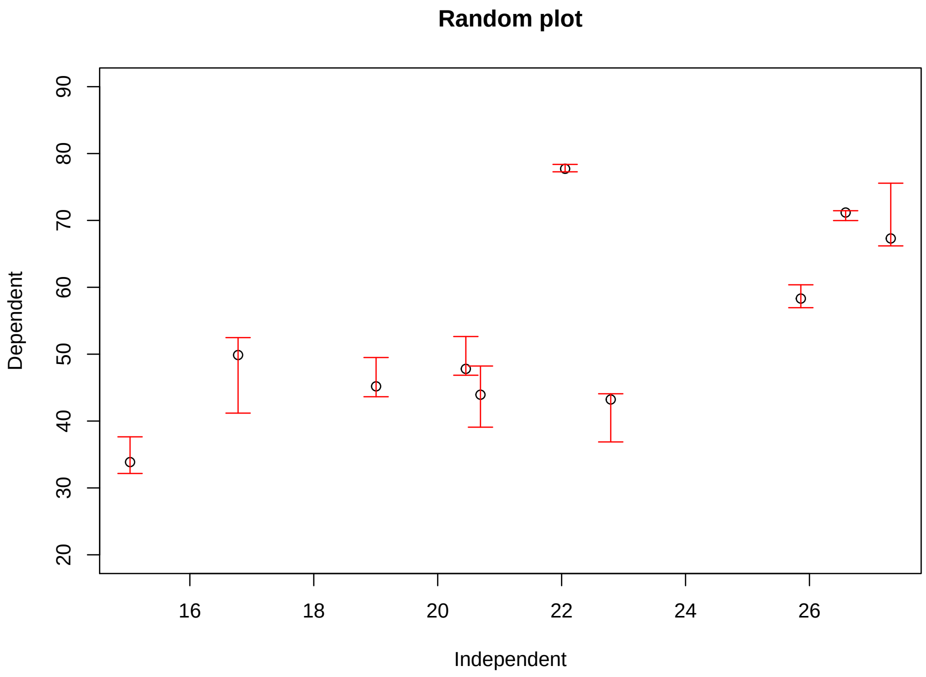
with legend and (x_2, y_2) points:

```
x2 <- runif(8,15,25)
y2 <- 2.5*x2 - 1.0 + runif(8,-6,6)
 plot(x,y,xlab="Independent",ylab="Dependent",main="Random plot")
 points(x1,y1,col=2,pch=3)
 points(x2,y2,col=4,pch=5)
 legend("topleft",c("Original","one","two"),col=c(1,2,4),pch=c(1,3,5))
```

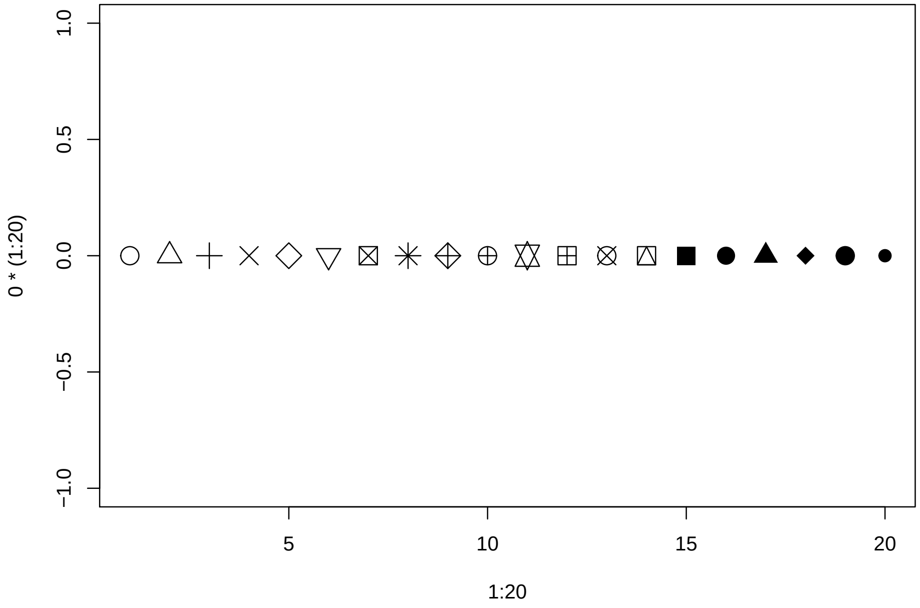


Errors bars:

```
plot(x,y,xlab="Independent",ylab="Dependent",main="Random plot",ylim=c(20,90))
xHigh <- x
yHigh <- y + abs(rnorm(10,sd=3.5))
xLow <- x
yLow <- y - abs(rnorm(10,sd=3.1))
arrows(xHigh,yHigh,xLow,yLow,col=2,angle=90,length=0.1,code=3)
```

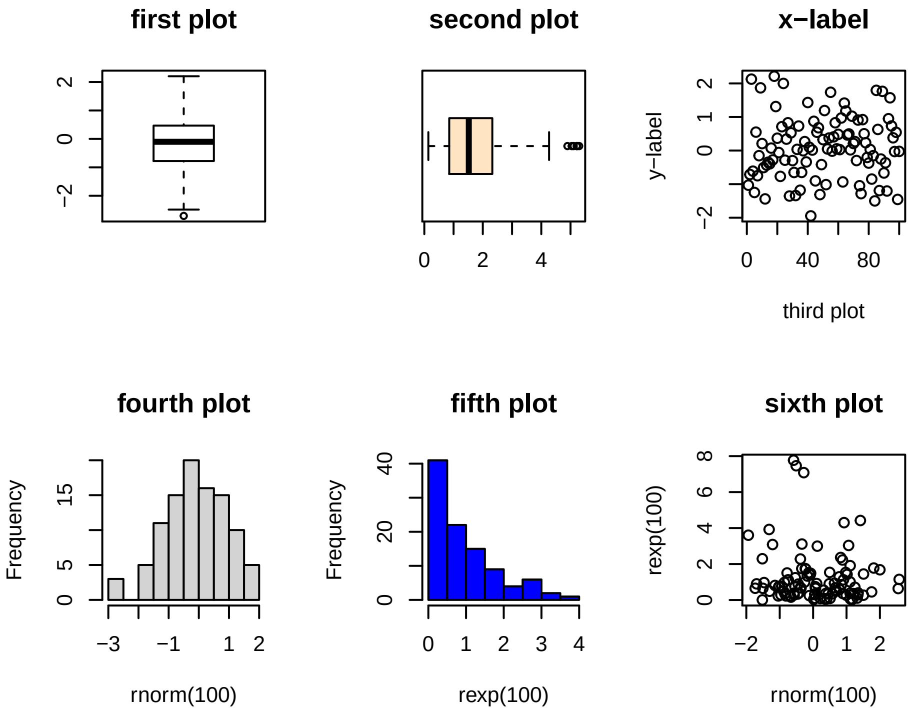


```
plot(1:20,0*(1:20),pch=1:20,cex=2)
```



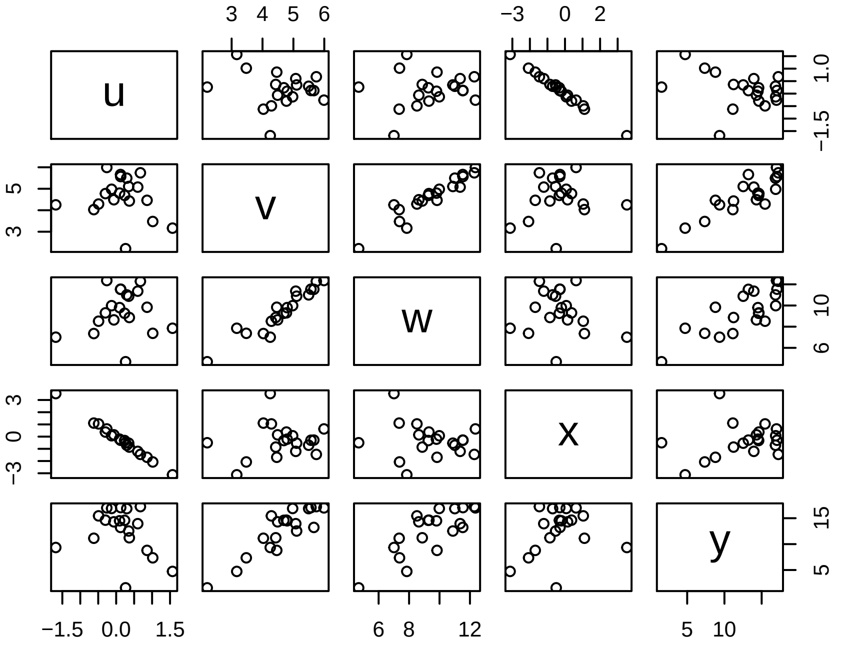
Multiple Graphs on One Image:

```
par(mfrow=c(2,3))
boxplot(rnorm(100),main="first plot")
boxplot(rgamma(100,2),main="second plot", horizontal=TRUE,col="bisque")
plot(rnorm(100),xlab="third plot",
     ylab="y-label",main="x-label")
hist(rnorm(100),main="fourth plot",col="lightgrey")
hist(rexp(100),main="fifth plot",col="blue")
plot(rnorm(100),rexp(100),main="sixth plot")
```



Pairwise relationships

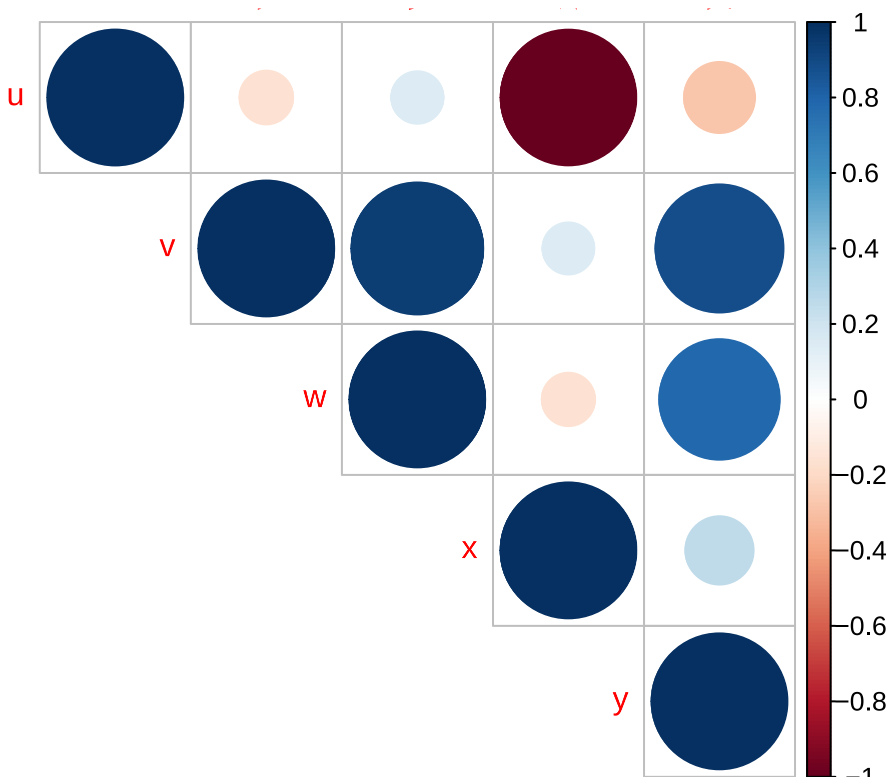
```
uData <- rnorm(20)
vData <- rnorm(20,mean=5)
wData <- uData + 2*vData + rnorm(20,sd=0.5)
xData <- -2*uData+rnorm(20,sd=0.1)
yData <- 3*vData+rnorm(20,sd=2.5)
d <- data.frame(u=uData,v=vData,w=wData,x=xData,y=yData)
pairs(d)
```



Plotting correlations

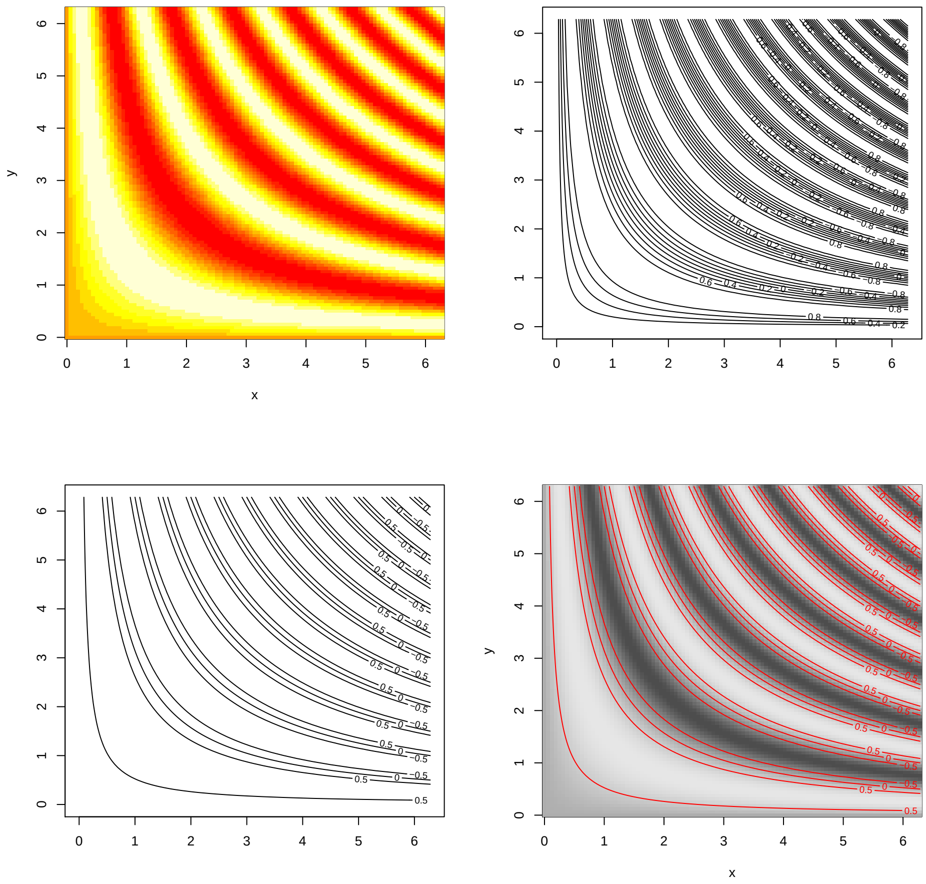
The function `corrplot` in the `library(corrplot)` visualizes a correlation matrix calculate with function `cor`

```
library(corrplot)
M <- cor(d)
corrplot(M, method="circle",type="upper")
```



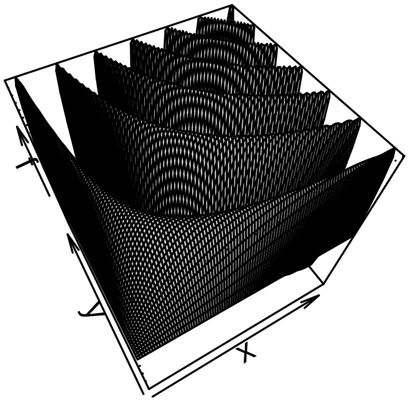
Plotting surfaces: `image`, `contour` and `persp` plots

```
x <- seq(0,2*pi,by=pi/50)
y <- x
xg <- (x*0+1) %*% t(y)
yg <- (x) %*% t(y*0+1)
f <- sin(xg*yg)

par(mfrow=c(2,2))
image(x,y,f)
contour(x,y,f)
contour(x,y,f,nlevels=4)
image(x,y,f,col=grey.colors(100))
contour(x,y,f,nlevels=4,add=TRUE,col="red")
```



Similarly, one can use **`persp`** plot

```
persp(x,y,f,theta=-30,phi=55,col="lightgrey",shade=.01)
```

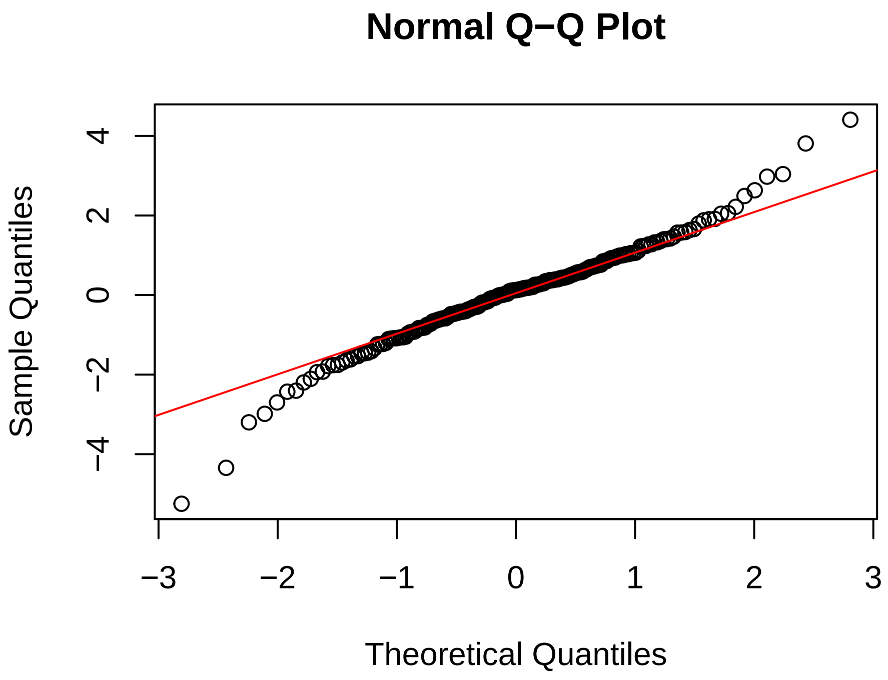


2.4 QQ-plot

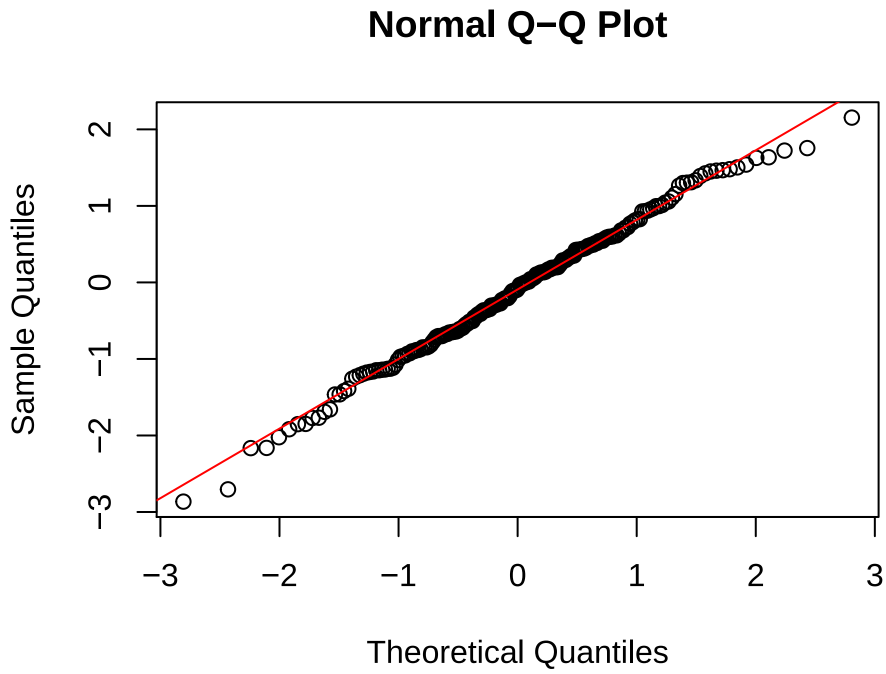
A Q-Q plot is a plot of the quantiles of two distributions against each other, or a plot based on estimates of the quantiles. The pattern of points in the plot is used to compare the two distributions.

qqnorm is used to determine if your data is close to being normally distributed. You cannot be sure that the data is normally distributed, but you can rule out if it is not normally distributed.

```
set.seed(1234)

require(graphics)
y <- rt(200, df = 5)
qqnorm(y);
qqline(y, col = 2)
```

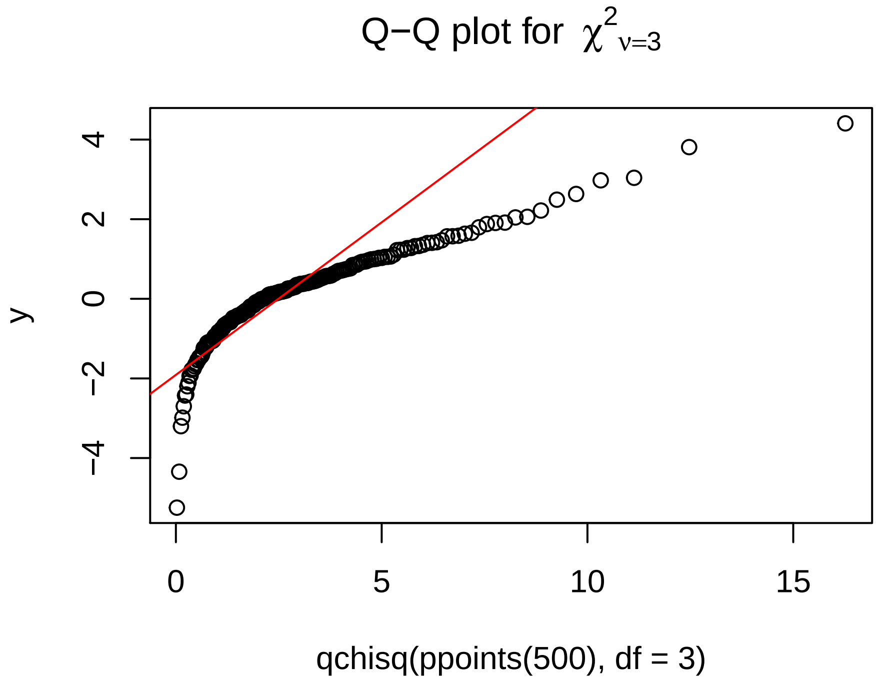


```
z <- rnorm(200,mean=0,sd=1)
qqnorm(z)
qqline(z, col = 2)
```



To compare two distributions

```
set.seed(1234)
qqplot(qchisq(ppoints(500), df = 3), y,
       main = expression("Q-Q plot for" ~~ {chi^2}[nu == 3]))
qqline(y, distribution = function(p) qchisq(p, df = 3),
       prob = c(0.1, 0.6), col = 2)
```



If the two distributions being compared are identical, the Q-Q plot follows the 45Âž line y = x.

2.5 Tables and Cross-classification

```
library(MASS)
data(quine)
?quine
attach(quine)
table(Sex)
```

```
## Sex
##  F  M
## 80 66
```

```
table(Sex,Age)
```

```
##    Age
## Sex F0 F1 F2 F3
##   F 10 32 19 19
##   M 17 14 21 14
```

```
# or xtabs
xtabs(~Sex+Age,data=quine)
```

```
##    Age
## Sex F0 F1 F2 F3
##   F 10 32 19 19
##   M 17 14 21 14
```

```
xtabs(~Sex+Age+Eth,data=quine)
```

```
## , , Eth = A
##
##    Age
## Sex F0 F1 F2 F3
##   F  5 15  9  9
##   M  8  5 11  7
##
## , , Eth = N
##
##    Age
## Sex F0 F1 F2 F3
##   F  5 17 10 10
##   M  9  9 10  7
```

2.6 Calculation of cross-classifications

```
tapply(Days,Age,mean)
```

```
##       F0       F1       F2       F3
## 14.85185 11.15217 21.05000 19.60606
```

```
tapply(Days,list(Sex,Age),mean)
```

```
##         F0       F1       F2       F3
## F 18.70000 12.96875 18.42105 14.00000
## M 12.58824  7.00000 23.42857 27.21429
```

```
tapply(Days,list(Sex,Age),function(x) sqrt(var(x)/length(x)))
```

```
##         F0       F1       F2       F3
## F 4.208589 2.329892 5.299959 2.940939
## M 3.768151 1.418093 3.766122 4.569582
```

2.7 Qualitative data

A data sample is called qualitative, also known as categorical, if its values belong to a collection of known defined non-overlapping classes. Common examples include student letter grade (A, B, C, D or F), commercial bond rating (AAA, AAB, . . .) and consumer clothing shoe sizes (1, 2, 3, . . .).

Let us consider some artificial data consisting of the `treatment` and `improvement` of patients with rheumatoid arthritis.

```r
treatment <- factor(rep(c(1, 2), c(43, 41)), levels = c(1, 2),
                    labels = c("placebo", "treated"))
improved <- factor(rep(c(1, 2, 3, 1, 2, 3), c(29, 7, 7, 13, 7, 21)),
                   levels = c(1, 2, 3),
                   labels = c("none", "some", "marked"))
```

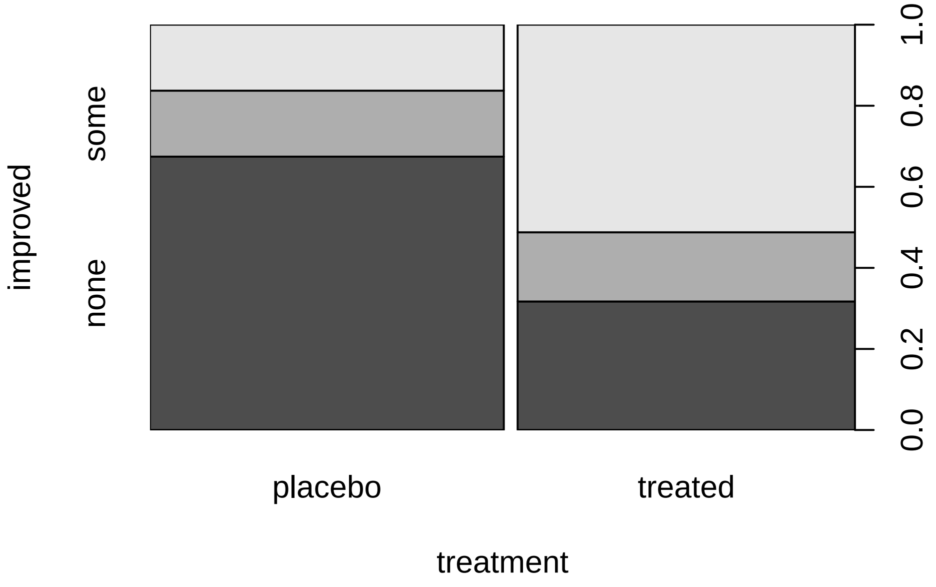
We can compute a cross-classification table

```r
xtabs(~treatment+improved)
```

```
##          improved
## treatment none some marked
##   placebo   29    7      7
##   treated   13    7     21
```

Graphically,

```r
spineplot(improved ~ treatment)
```



The R dataset UCBAdmissions contains aggregated data on applicants to graduate school at Berkeley for the six largest departments in 1973 classified by admission and sex.

```
data("UCBAdmissions")
?UCBAdmissions
apply(UCBAdmissions, c(2,1), sum)
```

```
##         Admit
## Gender   Admitted Rejected
##   Male       1198     1493
##   Female      557     1278
```

```
prop.table(apply(UCBAdmissions, c(2,1), sum))
```

```
##         Admit
## Gender    Admitted  Rejected
##   Male   0.2646929 0.3298719
##   Female 0.1230667 0.2823685
```

```
ftable(UCBAdmissions)
```

```
##                 Dept   A   B   C   D   E   F
## Admit    Gender
## Admitted Male        512 353 120 138  53  22
##          Female       89  17 202 131  94  24
## Rejected Male        313 207 205 279 138 351
##          Female       19   8 391 244 299 317
```

The same but with a more readable format can be obtained using ftable

```
ftable(round(prop.table(UCBAdmissions), 3),
       row.vars="Dept", col.vars = c("Gender", "Admit"))
```

```
##      Gender     Male              Female
##      Admit  Admitted Rejected Admitted Rejected
## Dept
## A              0.113    0.069    0.020    0.004
## B              0.078    0.046    0.004    0.002
## C              0.027    0.045    0.045    0.086
## D              0.030    0.062    0.029    0.054
## E              0.012    0.030    0.021    0.066
## F              0.005    0.078    0.005    0.070
```

More interesting are the proportions admitted for each `Gender` by `Dept` combination (dimensions 2 and 3 of the array). Notice that `male` and `female` admission rates are about the same in all departments, except "A", where female admission rates are higher.

```
# prop.table(UCBAdmissions, c(2,3))
ftable(round(prop.table(UCBAdmissions, c(2,3)), 2),
       row.vars="Dept", col.vars = c("Gender", "Admit"))
```

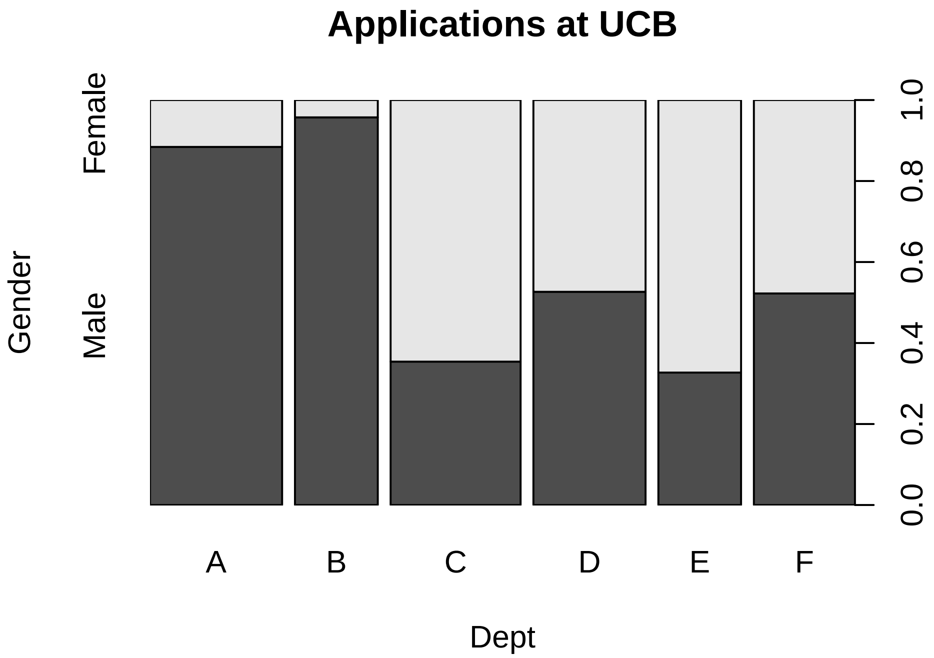
```
##      Gender     Male              Female         
##      Admit  Admitted Rejected Admitted Rejected
## Dept                                           
## A               0.62     0.38     0.82     0.18
## B               0.63     0.37     0.68     0.32
## C               0.37     0.63     0.34     0.66
## D               0.33     0.67     0.35     0.65
## E               0.28     0.72     0.24     0.76
## F               0.06     0.94     0.07     0.93
```

```
## Data aggregated over departments
apply(UCBAdmissions, c(1, 2), sum)
```

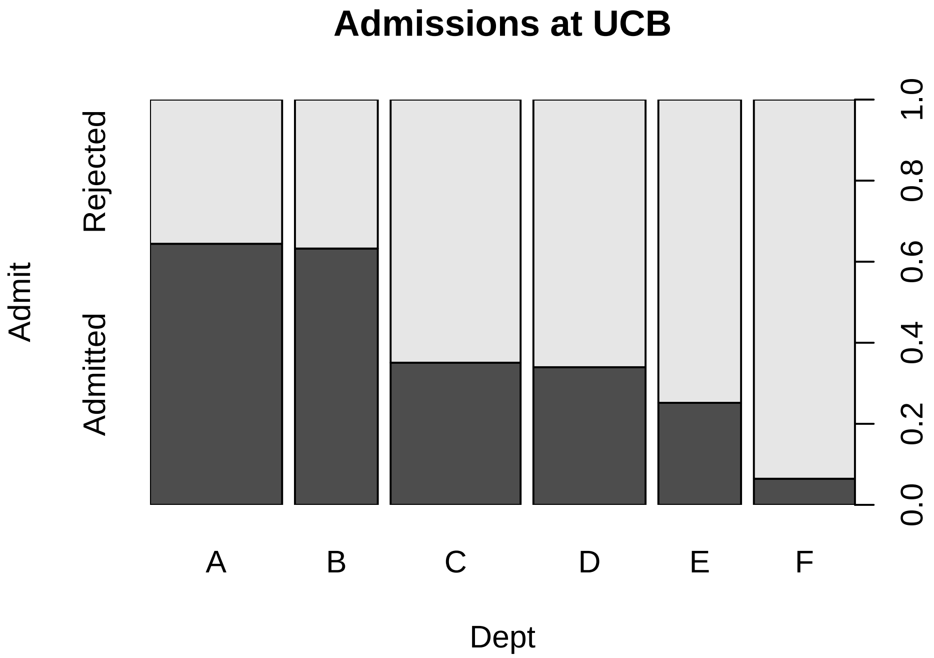
```
##           Gender
## Admit      Male Female
##   Admitted 1198    557
##   Rejected 1493   1278
```

Applications and admissions by department at UC Berkeley can be viewed graphically

```
spineplot(margin.table(UCBAdmissions, c(3, 2)),
          main = "Applications at UCB")
```



```r
spineplot(margin.table(UCBAdmissions, c(3, 1)),
          main = "Admissions at UCB")
```



This data set is frequently used for illustrating *Simpson's paradox*. At issue is whether the data show evidence of sex bias in admission practices. There were 2691 male applicants, of whom 1198 (44.5%) were admitted, compared with 1835 female applicants of whom 557 (30.4%) were admitted. Men were much more successful in admissions than women. Wikipedia: Gender Bias UC Berkeley.

2.8 Quantitative data

Quantitative data, also known as continuous data, consists of numeric data that support arithmetic operations. This is in contrast with qualitative data, whose values belong to pre-defined classes with no arithmetic operation allowed. We will explain how to apply some of the R tools for quantitative data analysis with examples.

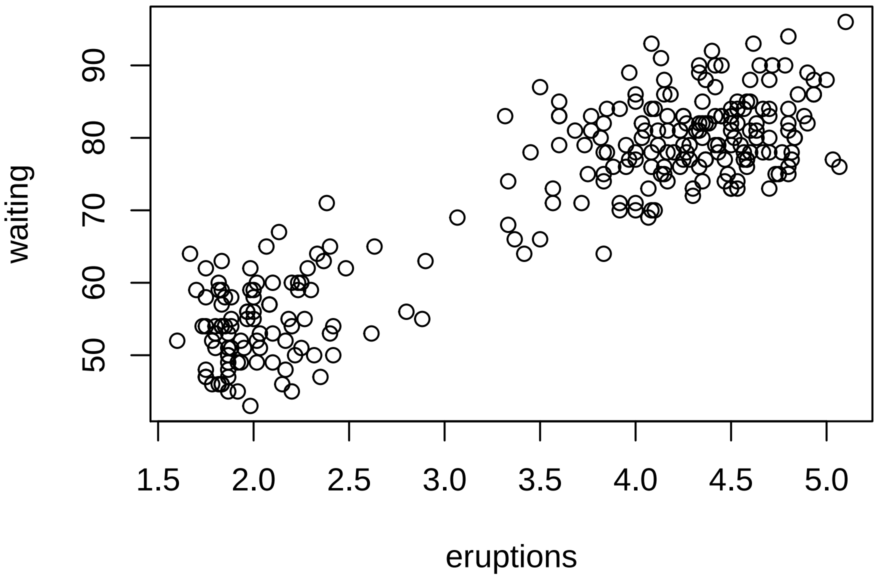
```r
head(faithful)
```

```
##   eruptions waiting
## 1     3.600      79
## 2     1.800      54
## 3     3.333      74
## 4     2.283      62
## 5     4.533      85
## 6     2.883      55
```

It consists of a collection of observations of the Old Faithful geyser in the USA Yellowstone National Park.

There are two observation variables in the data set. The first one, called `eruptions`, is the duration of the geyser eruptions. The second one, called `waiting`, is the length of waiting period until the next eruption. It turns out there is a correlation between the two variables.

```r
plot(faithful)
```



2.8.1 Frequency distribution of quantitative data

The frequency distribution of a data variable is a summary of the data occurrence in a collection of non-overlapping categories.

Let us find the frequency distribution of the eruption duration in `faithful` data set.

```
duration <- faithful$eruptions
range(duration)
```

```
## [1] 1.6 5.1
```

Now we create the range of non-overlapping sub-intervals by defining a sequence of equal distance break points. If we round the endpoints of the interval [1.6, 5.1] to the closest half-integers, we come up with the interval [1.5, 5.5]. Hence we set the break points to be the half-integer sequence { 1.5, 2.0, 2.5, . . . }.

```
breaks <- seq(1.5,5.5,by=0.5)
breaks
```

```
## [1] 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
```

Classify the eruption durations according to the half-unit-length sub-intervals with cut. As the intervals are to be closed on the left, and open on the right, we set the right argument as `FALSE`.

```
duration.cut = cut(duration, breaks, right=FALSE)
```

Compute the frequency of eruptions in each sub-interval with the table function.

```
duration.freq = table(duration.cut)
duration.freq
```

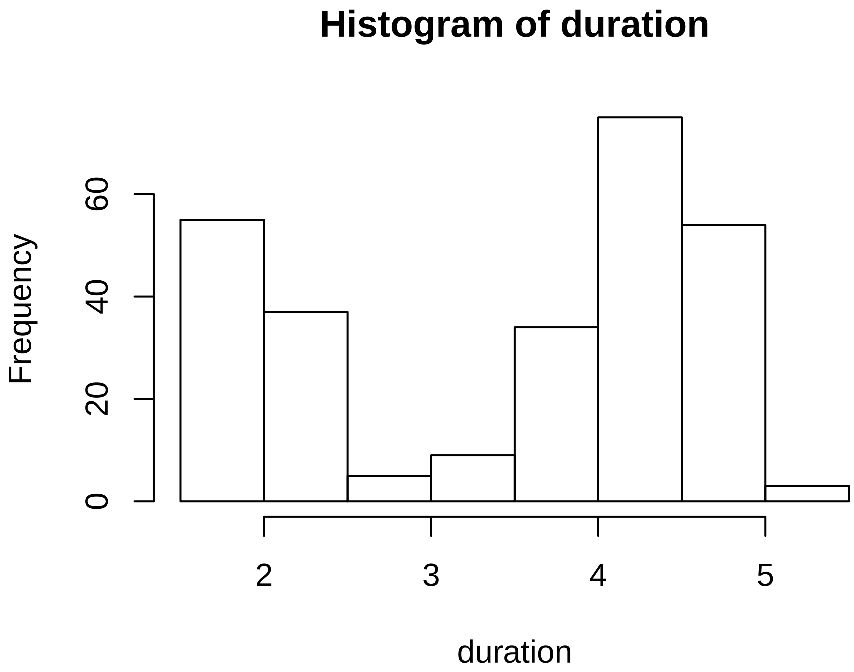
```
## duration.cut
## [1.5,2) [2,2.5) [2.5,3) [3,3.5) [3.5,4) [4,4.5) [4.5,5) [5,5.5)
##      51      41       5       7      30      73      61       4
```

`hist` function does all the computaions to find the frequency distribution:

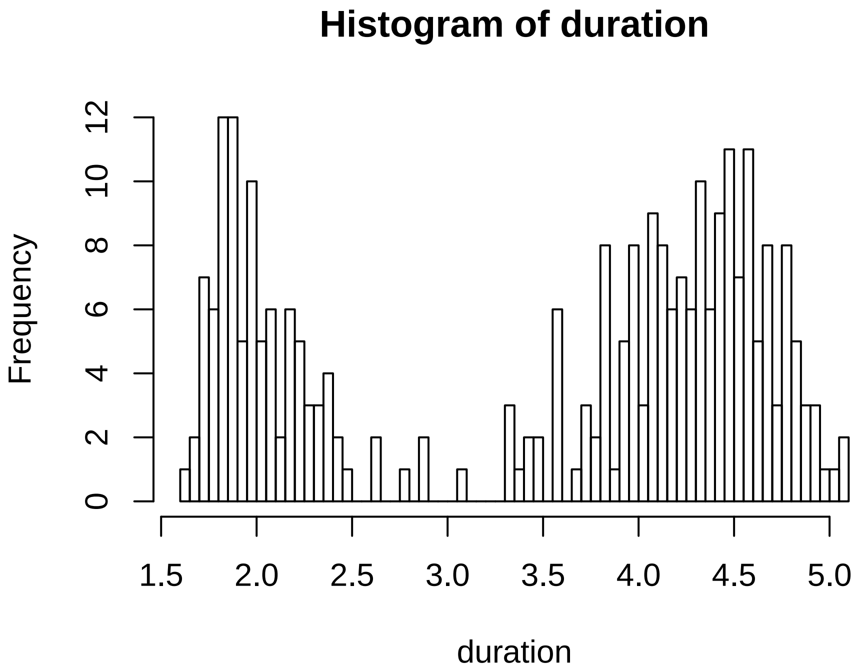
```
freq <- hist(duration)
freq
```

```
## $breaks
## [1] 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
##
## $counts
## [1] 55 37  5  9 34 75 54  3
##
## $density
## [1] 0.40441176 0.27205882 0.03676471 0.06617647 0.25000000 0.55147059
## [7] 0.39705882 0.02205882
##
## $mids
## [1] 1.75 2.25 2.75 3.25 3.75 4.25 4.75 5.25
##
## $xname
## [1] "duration"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
freq <- hist(duration,breaks = breaks)
```



```r
hist(duration,50)
```

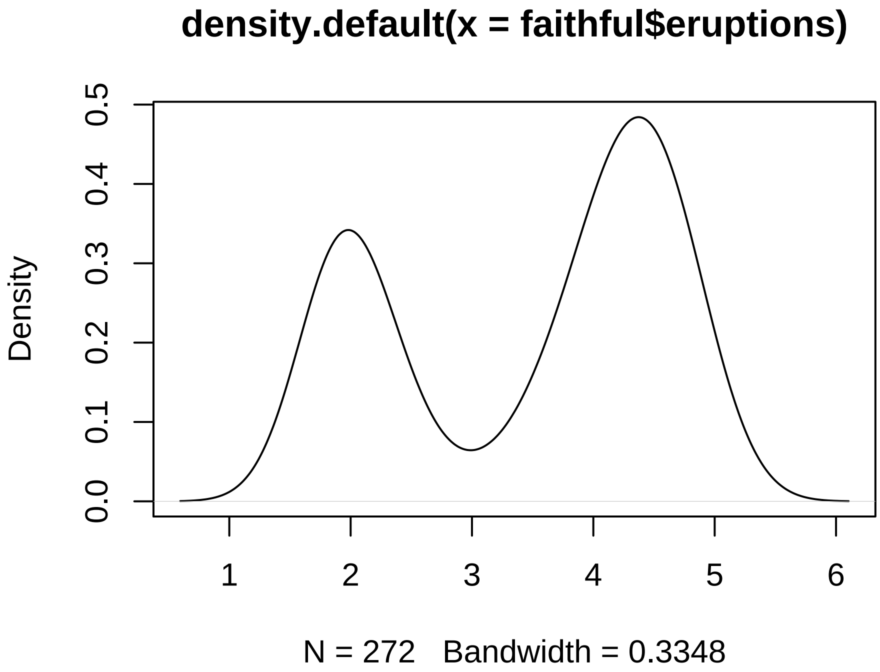


Density estimation builds an estimate of some underlying probability density function using an observed data sample.

```r
require(graphics)
d <- density(faithful$eruptions)
d
```

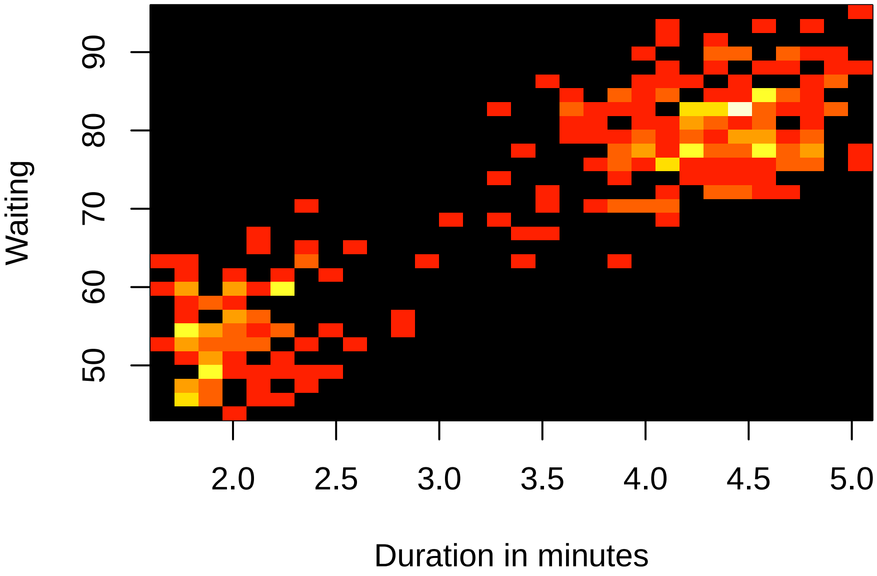
```
##
## Call:
##  density.default(x = faithful$eruptions)
##
## Data: faithful$eruptions (272 obs.); Bandwidth 'bw' = 0.3348
##
##        x                 y
##  Min.   :0.5957   Min.   :0.0002262
##  1st Qu.:1.9728   1st Qu.:0.0514171
##  Median :3.3500   Median :0.1447010
##  Mean   :3.3500   Mean   :0.1813462
##  3rd Qu.:4.7272   3rd Qu.:0.3086071
##  Max.   :6.1043   Max.   :0.4842095
```

```r
plot(d)
```



Two dimension histogram:

```
library(gplots)
h2 <- hist2d(faithful, nbins=30,xlab="Duration in minutes",ylab="Waiting")
```



```
h2
```

```
##
## ----------------------------
## 2-D Histogram Object
## ----------------------------
##
## Call: hist2d(x = faithful, nbins = 30, xlab = "Duration in minutes",
##     ylab = "Waiting")
##
## Number of data points:  272
## Number of grid bins:  30 x 30
## X range: ( 1.6 , 5.1 )
## Y range: ( 43 , 96 )
```

```
names(h2)
```

```
## [1] "counts"   "x.breaks" "y.breaks" "x"        "y"        "nobs"
## [7] "call"
```

Relative frequencies

```
duration.relfreq <- duration.freq / nrow(faithful)
tab <- cbind(duration.freq, duration.relfreq)
apply(tab,2,sum)
```

```
##    duration.freq duration.relfreq
##              272                1
```

Cumulative frequency distribution

```
cumsum(duration.freq)
```

```
## [1.5,2) [2,2.5) [2.5,3) [3,3.5) [3.5,4) [4,4.5) [4.5,5) [5,5.5)
##      51      92      97     104     134     207     268     272
```

```
cumsum(duration.relfreq)
```

```
##   [1.5,2)   [2,2.5)   [2.5,3)   [3,3.5)   [3.5,4)   [4,4.5)   [4.5,5)
## 0.1875000 0.3382353 0.3566176 0.3823529 0.4926471 0.7610294 0.9852941
##   [5,5.5)
## 1.0000000
```

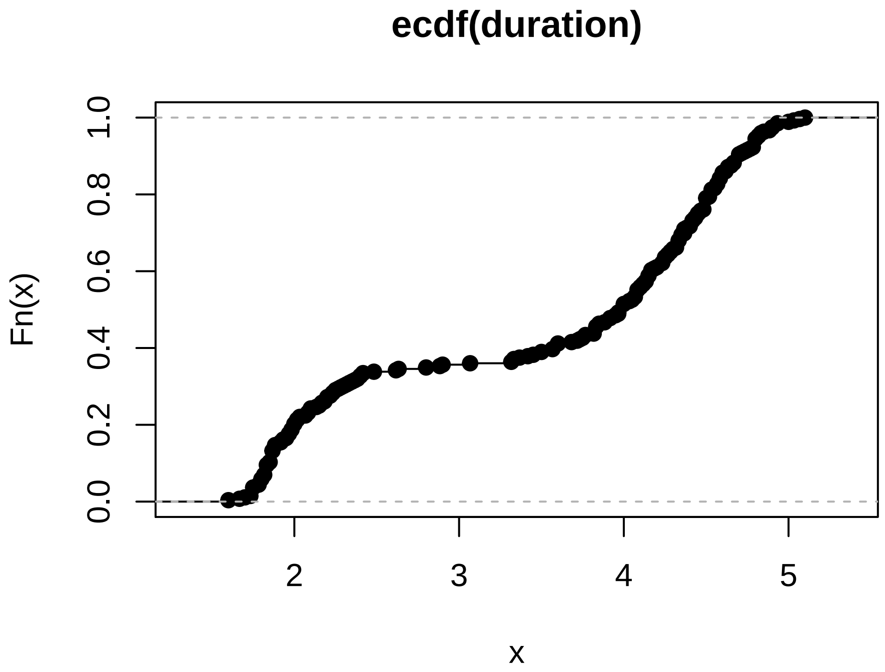
We can plot the cumulative relative frequency graph of a quantitative variable, which is a curve graphically showing the cumulative relative frequency distribution. The e.c.d.f. (empirical cumulative distribution function) F_n is a step function with jumps i/n at observation values, where i is the number of tied observations at that value. Missing values are ignored.

For observations $x = (x_1, x_2, ...x_n)$, F_n is the fraction of observations less or equal to t, i.e.,

$$F_n(t) = \#x_i <= t/n = 1/n \sum_{i=1}^{n} I(x_i \leq t).$$

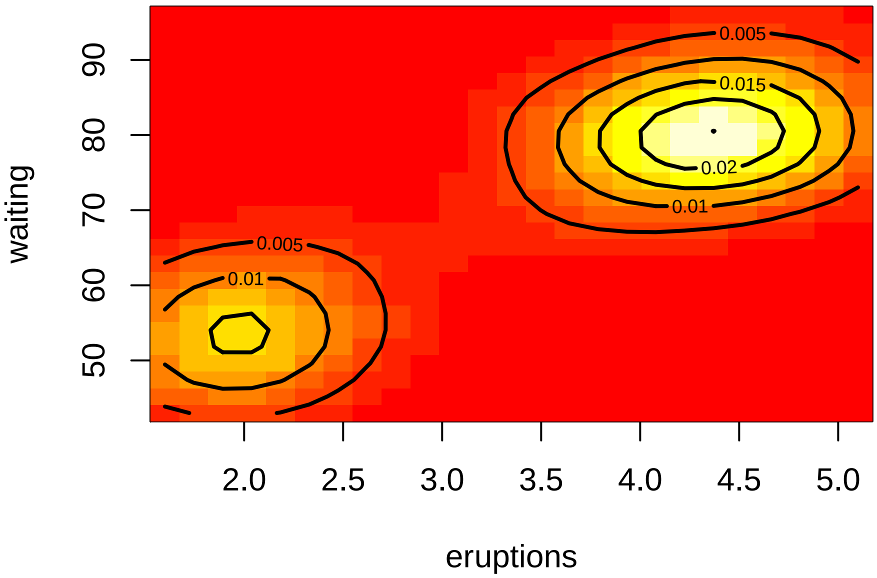
where I is an indication function.

```
plot(ecdf(duration))
```



Bivariante Density estimation:

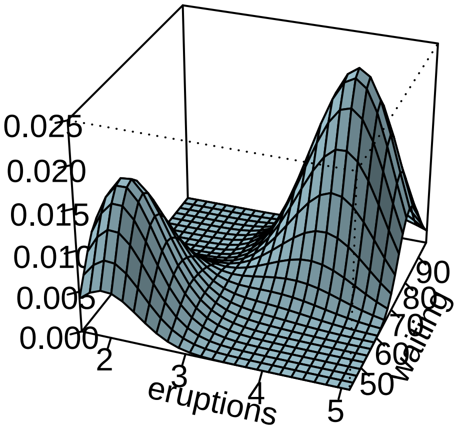
```
data("faithful")
attach(faithful)
Dens2d<-kde2d(eruptions,waiting)
image(Dens2d,xlab="eruptions",ylab="waiting")
contour(Dens2d,add=TRUE,col="black",lwd=2,nlevels=5)
```



```
detach("faithful")
```

Perspective plot:

```
persp(Dens2d,phi=30,theta=20,d=5,xlab="eruptions",ylab="waiting",zlab="",shade=.2,col="light
```



3 Introduction to basic programming in R

3.1 Control Structures

3.1.1 Conditional Executions

3.1.1.1 Comparison Operators

- equal: ==

```
"hola" == "hola"
```

```
## [1] TRUE
```

```
"hola" == "Hola"
```

```
## [1] FALSE
```

```
1 == 2-1
```

```
## [1] TRUE
```

- not equal: !=

```
a <- c(1,2,4,5)
b <- c(1,2,3,5)
a == b
```

```
## [1]  TRUE  TRUE FALSE  TRUE
```

```
a != b
```

```
## [1] FALSE FALSE  TRUE FALSE
```

- greater/less than: > <

```
set.seed(1)
a <- rnorm(10,3,1)
b <- rnorm(10,4,2)
a<b
```

```
##  [1]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

- greater/less than or equal: >= <=

```
set.seed(1)
a <- rpois(10,1)
b <- rbinom(10,2,.56)
a >= b
```

```
##  [1] FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
```

- which

```
set.seed(1)
which(a>b)
```

```
## [1] 4 6 7 8
```

```
LETTERS
```

```
##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q"
## [18] "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
```

```
which(LETTERS=="R")
```

```
## [1] 18
```

- which.min or which.max

```
set.seed(1)
a <- rnorm(10,2,1)
a
```

```
##  [1] 1.373546 2.183643 1.164371 3.595281 2.329508 1.179532 2.487429
##  [8] 2.738325 2.575781 1.694612
```

```
which.min(a)
```

```
## [1] 3
```

```
which.max(a)
```

```
## [1] 4
```

- is.na

```
 a[2] <- NA
is.na(a)
```

```
##  [1] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
which(is.na(a))
```

```
## [1] 2
```

3.1.2 Logical Operators

- and: `&`

```
z = 1:6
which(2 < z & z > 3)
```

```
## [1] 4 5 6
```

- or: `|`

```
z = 1:6
(z > 2) & (z < 5)
```

```
## [1] FALSE FALSE  TRUE  TRUE FALSE FALSE
```

```
which((z > 2) & (z < 5))
```

```
## [1] 3 4
```

- not: `!`

```
x <- c(TRUE,FALSE,0,6)
y <- c(FALSE,TRUE,FALSE,TRUE)

!x
```

```
## [1] FALSE  TRUE  TRUE FALSE
```

Operators `&` and `|` perform element-wise operation producing result having length of the longer operand. But `&&` and `||` examines only the first element of the operands resulting into a single length logical vector. Zero is considered `FALSE` and non-zero numbers are taken as `TRUE`. **Example:**

- `&&` vs `&`

```
x&y
```

```
## [1] FALSE FALSE FALSE  TRUE
```

```
x&&y
```

```
## [1] FALSE
```

- `||` vs `|`

```
x||y
```

```
## [1] TRUE
```

```
x|y
```

```
## [1]  TRUE  TRUE FALSE  TRUE
```

3.2 if statements

`if(cond1=true) { cmd1 } else { cmd2 }`

```
if(1==0) {
    print(1)
} else {
    print(2)
}
```

```
## [1] 2
```

3.3 ifelse statement

`ifelse(test, true_value, false_value)`

```
x <- 1:10 # Creates sample data
ifelse(x<5 | x>8, x, 0)
```

```
##  [1]  1  2  3  4  0  0  0  0  9 10
```

3.4 `while` statement

3.5 Loops

The most commonly used loop structures in `R` are `for`, `while` and `apply` loops. Less common are `repeat` loops. The `break` function is used to break out of loops, and next halts the processing of the current iteration and advances the looping index.

3.5.1 `for`

For loops are controlled by a looping vector. In every iteration of the loop one value in the looping vector is assigned to a variable that can be used in the statements of the body of the loop. Usually, the number of loop iterations is defined by the number of values stored in the looping vector and they are processed in the same order as they are stored in the looping vector.

Syntax

```
for(variable in sequence) {
    statements
}
```

```
for (j in 1:5)
{
  print(j^2)
}
```

```
## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
```

Repeat the loop saving the resuls in a vector `x`.

```
n = 5
x = NULL  # creates a NULL object
for (j in 1:n)
{
  x[j] = j^2
}
x
```

```
## [1]  1  4  9 16 25
```

Let's use a for loop to estimate the average of squaring the result of a roll of a dice.

```
nsides = 6
ntrials = 1000
trials = NULL
for (j in 1:ntrials)
{
  trials[j] = sample(1:nsides,1)  # We get one sample at a time
}
mean(trials^2)
```

```
## [1] 15.122
```

Example: stop on condition and print error message

```
x <- 1:10
z <- NULL
for(i in seq(along=x)) {
    if (x[i]<5) {
        z <- c(z,x[i]-1)
    } else {
        stop("values need to be <5")
    }
}
## Error: values need to be <5
z
## [1] 0 1 2 3
```

3.6 `while`

Similar to `for` loop, but the iterations are controlled by a conditional statement.

```
z <- 0
while(z < 5) {
    z <- z + 2
    print(z)
}
```

```
## [1] 2
## [1] 4
## [1] 6
```

3.7 apply loop family

For Two-Dimensional Data Sets: apply

Syntax:

```
apply(X, MARGIN, FUN, ARGs)
```

`X`: `array`, `matrix` or `data.frame`; `MARGIN`: 1 for rows, 2 for columns, `c(1,2)` for both; `FUN`: one or more functions; `ARGs`: possible arguments for function.

```
## Example for applying predefined mean function
apply(mtcars[,1:3], 1, mean)

## With custom function
x <- 1:10
test <- function(x) { # Defines some custom function
    if(x < 5) {
        x-1
    } else {
        x / x
    }
}

apply(as.matrix(x), 1, test)

## Same as above but with a single line of code
apply(as.matrix(x), 1, function(x) { if (x<5) { x-1 } else { x/x } })
```

For Ragged Arrays: `tapply`

Apply a function to each cell of a ragged array, that is to each (non-empty) group of values given by a unique combination of the levels of certain factors.

```
## Computes mean values of vector agregates defined by factor
tapply(as.vector(mtcars$mpg), factor(mtcars$cyl), mean)
```

```
##        4        6        8
## 26.66364 19.74286 15.10000
```

```
## The aggregate function provides related utilities
aggregate(mtcars[,c(1,3,4)], list(mtcars$cyl), mean)
```

```
##   Group.1      mpg     disp        hp
## 1       4 26.66364 105.1364  82.63636
## 2       6 19.74286 183.3143 122.28571
## 3       8 15.10000 353.1000 209.21429
```

For Vectors and Lists: `lapply` and `sapply`

Both apply a function to vector or list objects. The function `lapply` returns a list, while `sapply` attempts to return the simplest data object, such as `vector` or `matrix` instead of `list`.

Syntax

```
lapply(X,FUN)
sapply(X,FUN)
```

```
## Creates a sample list
mylist <- as.list(mtcars[,c(1,4,6)])
mylist
```

```
## $mpg
##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2
## [15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4
## [29] 15.8 19.7 15.0 21.4
##
## $hp
##  [1] 110 110  93 110 175 105 245  62  95 123 123 180 180 180 205 215 230
## [18]  66  52  65  97 150 150 245 175  66  91 113 264 175 335 109
##
## $wt
##  [1] 2.620 2.875 2.320 3.215 3.440 3.460 3.570 3.190 3.150 3.440 3.440
## [12] 4.070 3.730 3.780 5.250 5.424 5.345 2.200 1.615 1.835 2.465 3.520
## [23] 3.435 3.840 3.845 1.935 2.140 1.513 3.170 2.770 3.570 2.780
```

Compute sum of each list component and return result as list

```
lapply(mylist, sum)
```

```
## $mpg
## [1] 642.9
##
## $hp
## [1] 4694
##
## $wt
## [1] 102.952
```

Compute sum of each list component and return result as vector

```
sapply(mylist, sum)
```

```
##      mpg       hp       wt 
##  642.900 4694.000  102.952
```

3.8 Other Loops

Repeat Loop

Syntax

`repeat` statements

Loop is repeated until a break is specified. This means there needs to be a second statement to test whether or not to break from the loop.

Example:

```
z <- 0
repeat {
    z <- z + 1
    print(z)
    if(z > 100) break()
}
```

3.9 Improving Speed Performance of Loops

Looping over very large data sets can become slow in `R`. However, this limitation can be overcome by eliminating certain operations in loops or avoiding loops over the data intensive dimension in an object altogether. The latter can be achieved by performing mainly vector-to-vector or matrix-to-matrix computations which run often over 100 times faster than the corresponding `for()` or `apply()` loops in `R`. For this purpose, one can make use of the existing speed-optimized R functions (e.g.: `rowSums`, `rowMeans`, `table`, `tabulate`) or one can design custom functions that avoid expensive `R` loops by using vector- or matrix-based approaches. Alternatively, one can write programs that will perform all time consuming computations on the C-level.

1. Speed comparison of `for` loops with an append versus and inject step

```
N <- 1e6
myMA <- matrix(rnorm(N), N, 10, dimnames=list(1:N, paste("C", 1:10, sep="")))
results <- NULL
system.time(for(i in seq(along=myMA[,1]))
              results <- c(results, mean(myMA[i,])))

results <- numeric(length(myMA[,1]))
system.time(for(i in seq(along=myMA[,1]))
              results[i] <- mean(myMA[i,]))
```

The inject approach is 20-50 times faster than the append version.

2. Speed comparison of `apply` loop versus `rowMeans` for computing the mean for each row in a large matrix:

```
system.time(myMAmean <- apply(myMA, 1, mean))
system.time(myMAmean <- rowMeans(myMA))
```

The `rowMeans` approach is over 200 times faster than the `apply` loop.

4 Probability distributions

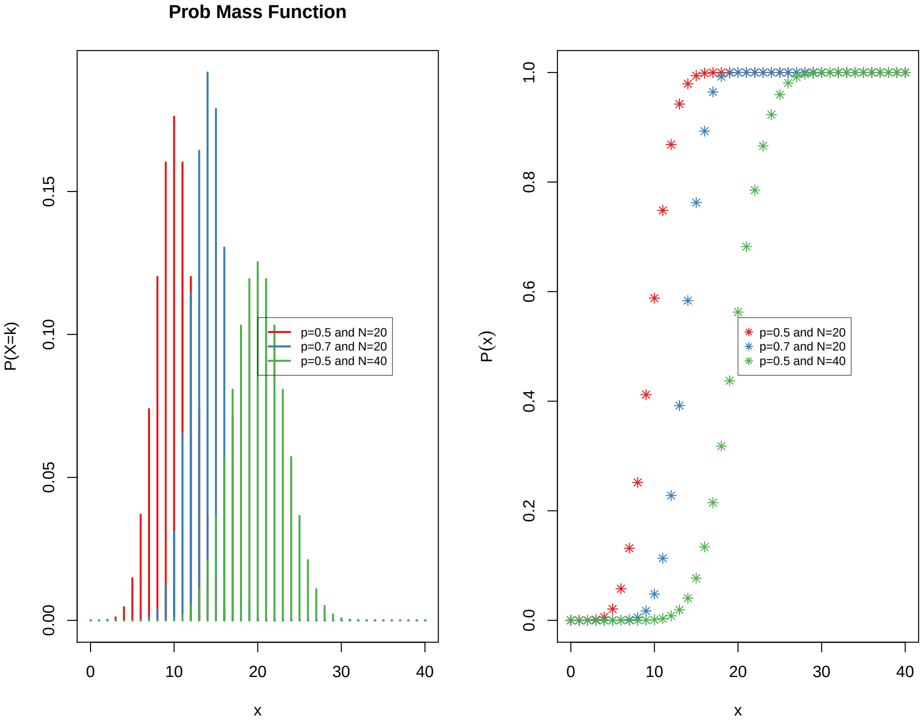
4.1 Binomial distribution $Bin(n, p)$

The binomial distribution is a discrete probability distribution. It describes the outcome of n independent trials in an experiment. Each trial is assumed to have only two outcomes, either success or failure. If the probability of a successful trial is p, then the probability of having k successful outcomes in an experiment of n independent trials is given by the **probability mass function**:.

$$f(k, n, p) = \Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k = 0, 1, 2, ..., n$$

The **cumulative distribution function** can be expressed as:

$$F(k; n, p) = \Pr(X \leq k) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}$$



with *mean* np and *variance* $np(1-p)$.

Question:

Suppose there are twelve multiple choice questions in an Maths class quiz. Each question has five possible answers, and only one of them is correct. Find the probability of having four or less correct answers if a student attempts to answer every question at random.

What is the probability of 2 or 3 questions answered correctly?

Question:

Suppose company **A** manufactures a product **B** which have probability 0.005 of being defective. Suppose product B is shipped in cartons containing 25 B items. What is the probability that a randomly chosen carton contains exactly one defective product? What is the probability that a randomly chosen carton contains no more than one defective widgit?

Solutions here

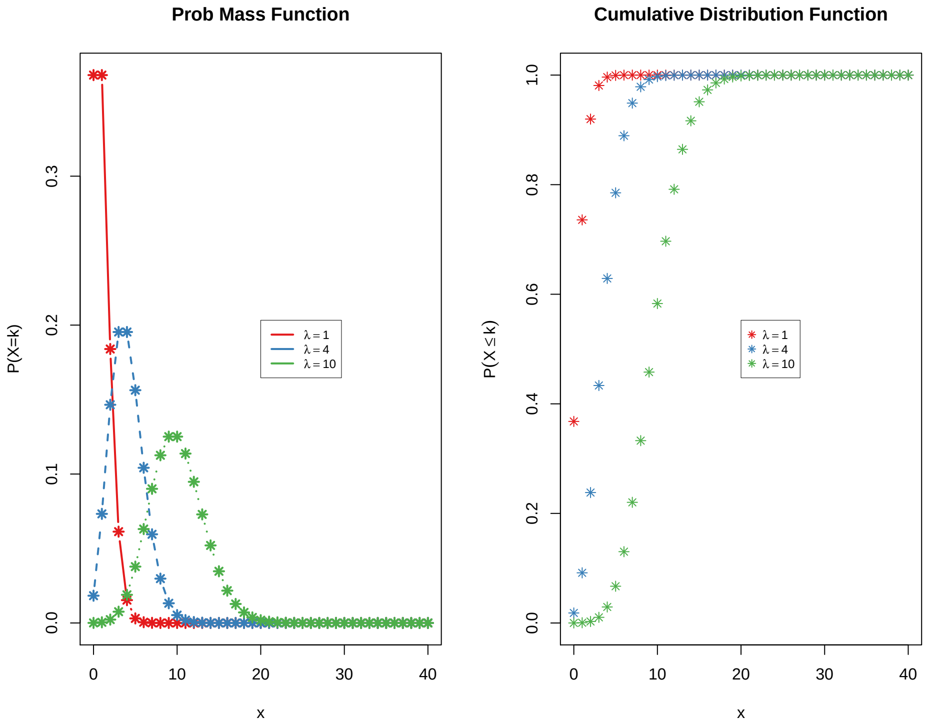
4.2 Poisson distribution $Pois(\lambda)$

The Poisson distribution is the probability distribution of independent event occurrences in an interval. If λ is the mean occurrence per interval, then the probability of having k occurrences within a given interval is the probability mass function given by:

$$\Pr(k \text{ events in interval}) = \frac{\lambda^k e^{-\lambda}}{k!}$$

The **cumulative density function** for the Poisson cumulative probability function is

$$P(X \leq x \mid \lambda) = \frac{e^{-\lambda}\lambda^x}{x!} \quad \text{for } x = 0, 1, 2, \ldots$$



Question:

Suppose the number of individual plants of a given species we expect to find in a one meter square quadrat follows the Poisson distribution with mean $\lambda = 10$. Find the probability of finding exactly 12 individuals.

Question:

If there are twelve cars crossing a bridge per minute on average, find the probability of having seventeen or more cars crossing the bridge in a particular minute.

Solutions here

4.3 Aproximation of Binomial as Poisson

Example

Five percent (5%) of Christmas tree light bulbs manufactured by a company are defective. The company's Quality Control Manager is quite concerned and therefore randomly samples 100 bulbs coming off of the assembly line. Let X denote the number in the sample that are defective. What is the probability that the sample contains at most three defective bulbs?

```
p = 0.05
k = 3
n = 100
pbinom(k,size=n,prob=p)
```

```
## [1] 0.2578387
```

It can be demonstrated that the Binomial distribution can be approximated with the Poisson probability mass function when n is large. Using the Poisson distribution, the mean $\lambda = np$

```
lambda <- n*p
sum(dpois(0:3,lambda))
```

```
## [1] 0.2650259
```

It is important to keep in mind that the Poisson approximation to the binomial distribution works well only when n is large and p is small. In general, the approximation works well if $n \geq 20$ and $p \leq 0.05$, or if $n \geq 100$ and $p \leq 0.10$.

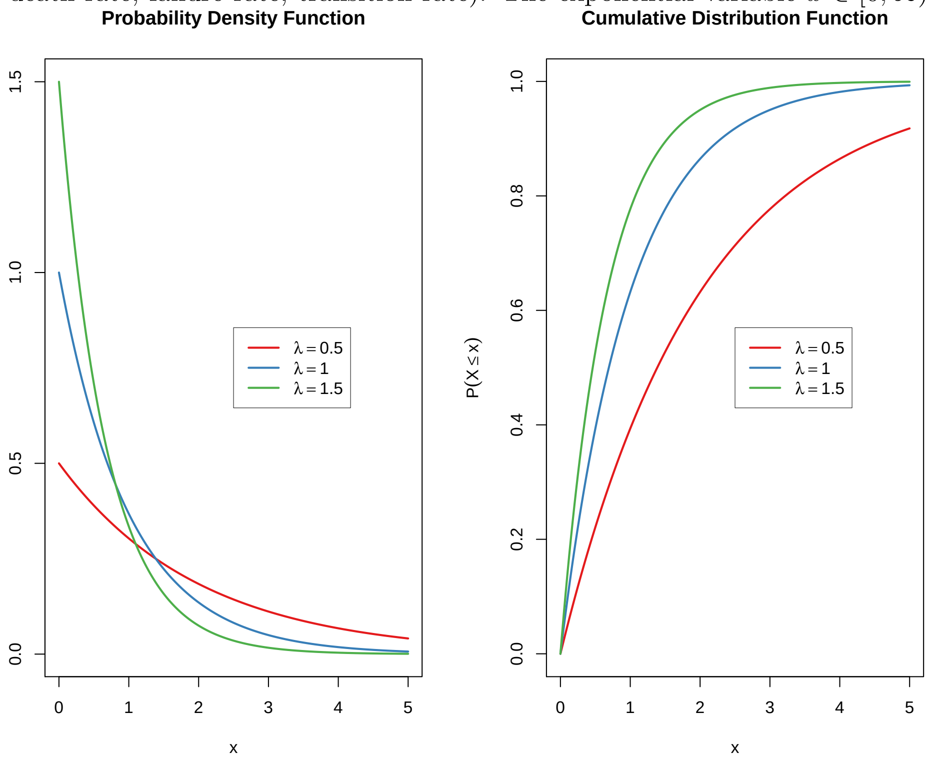
4.4 Exponential distribution $Exp(\lambda)$

The exponential distribution is the probability distribution that describes the time between events in a Poisson process, i.e. a process in which events occur continuously and independently at a constant average rate. It is a particular case of the gamma distribution. It is the continuous analogue of the geometric distribution, and it has the key property of being memoryless. In addition to being used for the analysis of Poisson processes, it is found in various other contexts.

The probability density function (pdf) of an exponential distribution as

$$f(x; \lambda) = \lambda \exp(-\lambda x)$$

where $\lambda > 0$ is the event rate (also known as rate parameter, arrival rate, death rate, failure rate, transition rate). The exponential variable $x \in [0, \infty)$



Cumulative distribution function of the exponential distribution is

$$F(x) = \Pr(X \leq x) = \begin{cases} 1 - e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Mean $\mathbb{E}(X) = 1/\lambda$, and $\mathbb{V}ar(X) = 1/\lambda^2$.

Question:

Suppose that the amount of time one spends in a bank is exponentially distributed with mean 10 minutes, $\lambda = 1/10$.

- What is the probability that a customer will spend more than 15 minutes in the bank?
- What is the probability that a customer will spend more than 15 minutes in the bank given that he is still in the bank after 10 minutes?

Solutions here

4.5 The Normal distribution $\mathcal{N}(\mu, \sigma^2)$

The probability density function of the Normal distribution is:

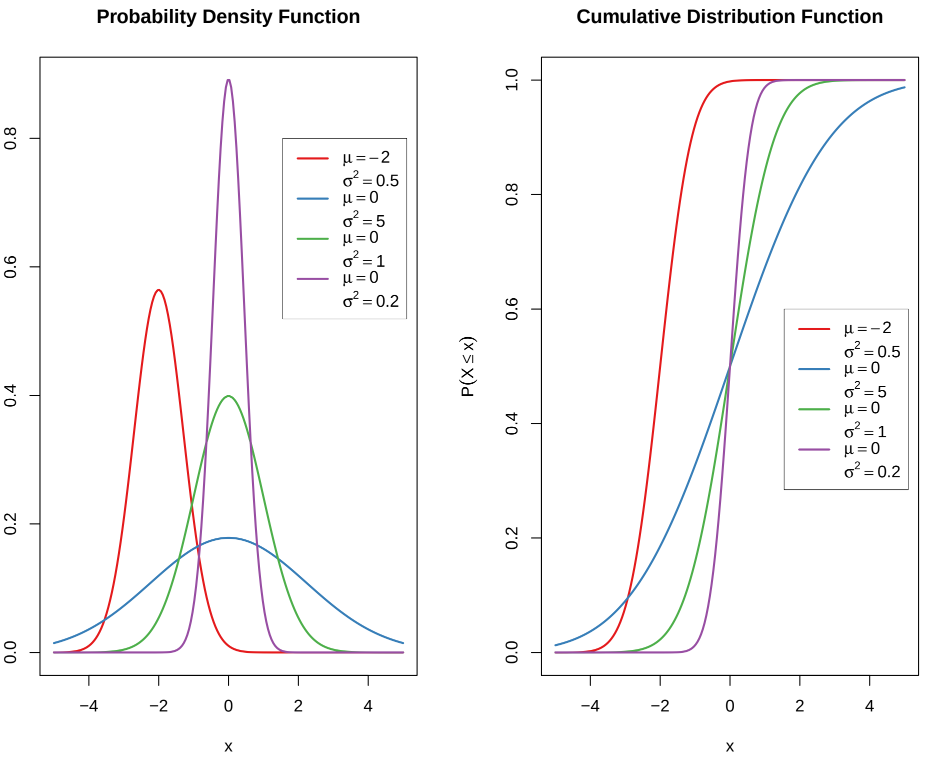
$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

where

- μ is the mean of the distribution (also the median and the mode).
- σ is the standard deviation ($\sigma > 0$).
- σ^2 is the variance.

The process to standardized Normal distribution consists in transforming the Normal variable $N(\mu,\sigma)$ to $N(0,1)$, i.e.

$$Z = \frac{X-\mu}{\sigma} \sim N(0,1)$$



Question:

X is a normally distributed variable with mean $\mu = 30$ and standard deviation $\sigma = 4$. Find

a) $P(x < 40)$

b) $P(x > 21)$

c) $P(30 < x < 35)$

Question:

Entry to a certain University is determined by a national test. The scores on this test are normally distributed with a mean of 500 and a standard deviation of 100. Tom wants to be admitted to this university and he knows that he must score better than at least 70% of the students who took the test. Tom takes the test and scores 585. Will he be admitted to this university?

Solutions here

4.6 Exercises

1. A dice is thrown at random. What is the expectation of number on it? (Or) If x denotes the number of points on a dice, find the expectation and the variance of x.

```r
x = 1:6
prob <- 1/6

 E  <- sum(x*rep(prob,length(x)))

 E2 <- E^2
 Ex2 <- sum(x^2*rep(prob,length(x)))
Var <- Ex2-E2
```

2. If a person gains or loses an amount equal to the number appearing when a balanced die is rolled once, according to whether the number is even or odd, how much money can be expect per game in the long run?

Let x indicate the amount that the person wins

The amount gained by the person would be

- `- Rs.` 1 if the dice shows up ONE
- `+ Rs.` 2 if the dice shows up TWO
- `- Rs.` 3 if the dice shows up THREE
- `+ Rs.` 4 if the dice shows up FOUR

- `- Rs. 5` if the dice shows up FIVE
- `+ Rs. 6` if the dice shows up SIX

```
x = c(1:6)*c(-1,1,-1,1,-1,1)
pr <- rep(1/6,6)
 Ex <- sum(x*pr)
Ex2 <- sum(x^2*pr)
Var <- Ex2-Ex^2
Ex
```

```
## [1] 0.5
```

```
Var
```

```
## [1] 14.91667
```

5 Statistical Inference

Statistical inference means drawing conclusions based on data. There are many ways of performing inference including statistical modeling, data oriented strategies and explicit use of designs and randomization in analyses. Furthermore, there are broad theories (frequentists, Bayesian, likelihood, design based, . . .) and numerous complexities (missing data, observed and unobserved confounding, biases) for performing inference. This section presents the fundamentals of inference in a practical approach for getting things done in order to use statistical inference for making informed choices in analyzing data.

5.1 Interval estimation

It is a common requirement to efficiently estimate population parameters based on simple random sample data.

```
library(MASS)
?survey
head(survey)
```

```
##      Sex Wr.Hnd NW.Hnd W.Hnd    Fold Pulse    Clap Exer Smoke Height
## 1 Female   18.5   18.0 Right  R on L    92    Left Some Never 173.00
## 2   Male   19.5   20.5  Left  R on L   104    Left None Regul 177.80
## 3   Male   18.0   13.3 Right  L on R    87 Neither None Occas     NA
## 4   Male   18.8   18.9 Right  R on L    NA Neither None Never 160.00
## 5   Male   20.0   20.0 Right Neither    35   Right Some Never 165.00
## 6 Female   18.0   17.7 Right  L on R    64   Right Some Never 172.72
##        M.I    Age
## 1   Metric 18.250
## 2 Imperial 17.583
## 3     <NA> 16.917
## 4   Metric 20.333
## 5   Metric 23.667
## 6 Imperial 21.000
```

Point estimate of population mean

For any particular random sample, we can always compute its sample mean. Although most often it is not the actual population mean, it does serve as a good point estimate. For example, in the data set survey, the survey is performed on a sample of the student population. We can compute the sample mean and use it as an estimate of the corresponding population parameter.

Problem: Find a point estimate of mean university student height with the sample data from survey.

```
mean(survey$Height)
```

```
## [1] NA
```

```
mean(survey$Height,na.rm=TRUE)
```

```
## [1] 172.3809
```

```
# or
mean(na.omit(survey$Height))
```

```
## [1] 172.3809
```

After we found a point estimate of the population mean, we would need a way to quantify its accuracy.

5.1.1 Interval Estimate of Population Mean (with known variance)

Let us denote the 100 $(1-\alpha/2)$ percentile of the standard normal distribution as $z_{\alpha/2}$. For random sample of sufficiently large size, the end points of the interval estimate at $(1-\alpha/2)$ confidence level is given as follows:

$$\bar{x} \pm \frac{\sigma}{\sqrt{n}}$$

Problem: assume the population standard deviation σ of the student height in survey is 9.48. Find the margin of error and interval estimate at 95% confidence level.

```
x <- na.omit(survey$Height)
n <- length(x)
sigma <- 9.48
sem = sigma/sqrt(n)  # Standard error of the mean
sem
```

```
## [1] 0.6557453
```

Since there are two tails of the normal distribution, the 95% confidence level would imply the 97.5th percentile of the normal distribution at the upper tail. Therefore, $z_{\alpha/2}$ is given by `qnorm(.975)`. We multiply it with the standard error of the mean `sem` and get the margin of error.

```
E <- qnorm(0.975)*sem
E  # margin of error
```

```
## [1] 1.285237
```

```
xbar <- mean(x,na.rm = TRUE)
xbar + c(-E,E)
```

```
## [1] 171.0956 173.6661
```

Alternatively, we can use `z.test`

```
library(TeachingDemos)
z.test(x,sd=sigma)
```

```
##
##  One Sample z-test
##
## data:  x
## z = 262.88, n = 209.00000, Std. Dev. = 9.48000, Std. Dev. of the
## sample mean = 0.65575, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  171.0956 173.6661
## sample estimates:
## mean of x
##  172.3809
```

Assuming the population standard deviation σ being 9.48, the margin of error for the student height survey at 95% confidence level is 1.2852372 centimeters. The confidence interval is between 171.095624 and 173.6660984 centimeters.

5.1.2 Interval Estimate of Population Mean (with unknown variance)

```
s = sd(x)
SE <- s/sqrt(n)
E = qt(.975, df=n-1)*SE; E
```

```
## [1] 1.342878
```

```
xbar + c(-E, E)
```

```
## [1] 171.0380 173.7237
```

```
# or

t.test(x)
```

```
##
##  One Sample t-test
##
## data:  x
## t = 253.07, df = 208, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  171.0380 173.7237
## sample estimates:
## mean of x
##  172.3809
```

Without assumption on the population standard deviation, the margin of error for the student height survey at 95% confidence level is 1.3429 centimeters. The confidence interval is between 171.04 and 173.72 centimeters.

5.1.2.1 Sampling Size of Population Mean The quality of a sample survey can be improved by increasing the sample size. The formula below provide the sample size needed under the requirement of population mean interval estimate at $(1-\alpha)$ confidence level, margin of error E, and population variance σ^2. Here, $z_{\alpha-2}$ is the $100(1-\alpha/2)$ percentile of the standard normal distribution.

$$n = \frac{(z_{\alpha/2})^2\sigma^2}{E^2}$$

Problem:

Assume the population standard deviation σ of the student height in `survey` is 9.48. Find the sample size needed to achieve a 1.2 centimeters margin of error at 95% confidence level.

```
zstar = qnorm(.975)
sigma = 9.48
E = 1.2
zstar^2 *sigma^2/ E^2
```

```
## [1] 239.7454
```

Based on the assumption of population standard deviation being 9.48, it needs a sample size of 240 to achieve a 1.2 centimeters margin of error at 95% confidence level.

5.1.3 Interval estimates of population proportion

Point estimate of population proportion

Multiple choice questionnaires in a survey are often used to determine the the proportion of a population with certain characteristic. For example, we can estimate the proportion of female students in the university based on the result in the sample data set `survey`.

Problem: Find a point estimate of the female student proportion from survey.

```
library(MASS)
gender.response <- na.omit(survey$Sex) # filter out NA's
n <- length(gender.response)
table(gender.response)
```

```
## gender.response
## Female    Male
##    118    118
```

To find out the number of female students, we compare `gender.response`

```r
k <- sum(gender.response == "Female")
# or
k <- table(gender.response)[1]

pbar <- k/n; pbar
```

```
## Female
##    0.5
```

The point estimate of the female student proportion in survey is 50%.

In order to compute the confidence interval of the estimated proportion, let us denote the $100(1-\alpha/2)$ percentile of the standard normal distribution as $z_{\alpha/2}$. If the samples size n and population proportion p satisfy the condition that $np \geq 5$ and $n(1-p) \geq 5$, than the end points of the interval estimate at $(1-\alpha)$ confidence level is defined in terms of the sample proportion as follows.

$$\bar{p} = \pm z_{\alpha/2}\sqrt{\frac{\bar{p}(1-\bar{p})}{n}}$$

We use the `prop.test` function for direct calculation

```r
prop.test(k,n,conf.level = 0.95)
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  k out of n, null probability 0.5
## X-squared = 0, df = 1, p-value = 1
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.4367215 0.5632785
## sample estimates:
##   p
## 0.5
```

At 95% confidence level, between 43.6% and 56.3% of the university students are female, and the margin of error is 6.4%.

5.1.3.1 Sampling Size of Population proportion The quality of a sample survey can be improved by increasing the sample size. The formula below provide the sample size needed under the requirement of population proportion interval estimate at $(1-\alpha)$ confidence level, margin of error E, and planned proportion estimate p. Here, $z_{\alpha/2}$ is the $100(1-\alpha/2)$ percentile of the standard normal distribution.

$$n = \frac{(z_{\alpha/2})^2 p(1-p)}{E^2}$$

```
zstar = qnorm(.975)
p = 0.5
E = 0.05
zstar^2 * p * (1-p) / E^2
```

```
## [1] 384.1459
```

With a planned proportion estimate of 50% at 95% confidence level, it needs a sample size of 385 to achieve a 5% margin of error for the survey of female student proportion.

5.2 Inference about two populations

It is often necessary to draw conclusion on the difference between two populations by their data samples. In the following tutorials, we discuss how to estimate the difference of means and proportions between two normally distributed data populations.

5.2.1 Population mean between two matches samples

Two data samples are matched if they come from repeated observations of the same subject. Here, we assume that the data populations follow the normal distribution. Using the paired t-test, we can obtain an interval estimate of the difference of the population means.

```
library(MASS)
data(immer)
?immer
head(immer)
```

```
##   Loc Var    Y1    Y2
## 1  UF   M  81.0  80.7
## 2  UF   S 105.4  82.3
## 3  UF   V 119.7  80.4
## 4  UF   T 109.7  87.2
## 5  UF   P  98.3  84.2
## 6   W   M 146.6 100.4
```

Problem:

Assuming that the data in `immer` follows the normal distribution, find the 95% confidence interval estimate of the difference between the mean barley yields between years 1931 and 1932.

```
t.test(immer$Y1, immer$Y2, paired=TRUE)
```

```
##
##  Paired t-test
##
## data:  immer$Y1 and immer$Y2
## t = 3.324, df = 29, p-value = 0.002413
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   6.121954 25.704713
## sample estimates:
## mean of the differences
##                15.91333
```

Answer:

Between years 1931 and 1932 in the data set `immer`, the 95% confidence interval of the difference in means of the barley yields is the interval between 6.122 and 25.705.

5.3 Population Mean Between Two Independent Samples

Two data samples are independent if they come from unrelated populations and the samples does not affect each other. Here, we assume that the data populations follow the normal distribution. Using the unpaired t-test, we can obtain an interval estimate of the difference between two population means.

Example:

In the data frame column mpg of the data set `mtcars`, there are gas mileage data of various 1974 U.S. automobiles. `am` variable indicates the transmission type of the automobile model (0 = automatic, 1 = manual).

In particular, the gas mileage for manual and automatic transmissions are two independent data populations.

Problem:

Assuming that the data in `mtcars` follows the normal distribution, find the 95% confidence interval estimate of the difference between the mean gas mileage of manual and automatic transmissions.

```
t.test(mpg ~ am, data=mtcars)
```

```
##
##  Welch Two Sample t-test
##
## data:  mpg by am
## t = -3.7671, df = 18.332, p-value = 0.001374
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -11.280194  -3.209684
## sample estimates:
## mean in group 0 mean in group 1
##        17.14737        24.39231
```

In `mtcars`, the mean mileage of automatic transmission is 17.147 `mpg` and the manual transmission is 24.392 `mpg`. The 95% confidence interval of the difference in mean gas mileage is between 3.2097 and 11.2802 `mpg`.

5.4 Comparison of Two Population Proportions

A survey conducted in two distinct populations will produce different results. It is often necessary to compare the survey response proportion between the two populations. Here, we assume that the data populations follow the normal distribution.

Example:

In the built-in data set named quine, children from an Australian town is classified by ethnic background, gender, age, learning status and the number of days absent from school.

```
library(MASS)
data(quine)
?quine
head(quine)
```

```
##   Eth Sex Age Lrn Days
## 1   A   M  F0  SL    2
## 2   A   M  F0  SL   11
## 3   A   M  F0  SL   14
## 4   A   M  F0  AL    5
## 5   A   M  F0  AL    5
## 6   A   M  F0  AL   13
```

Column `Eth` indicates whether the student is Aboriginal or Not (“A” or “N”), and the column `Sex` indicates Male or Female (“M” or “F”).

```
table(quine$Eth,quine$Sex)
```

```
##
##      F  M
##   A 38 31
##   N 42 35
```

Problem:

Assuming that the data in quine follows the normal distribution, find the 95% confidence interval estimate of the difference between the female proportion of Aboriginal students and the female proportion of Non-Aboriginal students, each within their own ethnic group.

Solution

We apply the ‘prop.test function to compute the difference in female proportions. The Yates’s continuity correction is disabled for pedagogical reasons.

```
prop.test(table(quine$Eth, quine$Sex), correct=FALSE)
```

```
##
##  2-sample test for equality of proportions without continuity
##  correction
##
## data:  table(quine$Eth, quine$Sex)
## X-squared = 0.0040803, df = 1, p-value = 0.9491
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.1564218  0.1669620
## sample estimates:
##    prop 1    prop 2
## 0.5507246 0.5454545
```

The 95% confidence interval estimate of the difference between the female proportion of Aboriginal students and the female proportion of Non-Aboriginal students is between -15.6% and 16.7%.

5.5 Goodness-of-Fit

Many statistical quantities derived from data samples are found to follow the Chi-squared distribution. Hence we can use it to test whether a population fits a particular theoretical probability distribution.

5.6 Chi-squared Test of Independence

Two random variables x and y are called independent if the probability distribution of one variable is not affected by the presence of another. Assume f_{ij} is the observed frequency count of events belonging to both i-th category of x and j-th category of y. Also assume e_{ij} to be the corresponding expected count if x and y are independent. The null hypothesis of the independence assumption is to be rejected if the p-value of the following Chi-squared test statistics is less than a given significance level α.

$$\chi^2 = \sum_{i,j} \frac{(f_i - e_i)^2}{e_i}$$

Example:

In the built-in data set `survey`, the `Smoke` column records the students smoking habit, while the `Exer` column records their exercise level. The allowed values in Smoke are “Heavy”, “Regul” (regularly), “Occas” (occasionally) and “Never”. As for Exer, they are “Freq” (frequently), “Some” and “None”.

```
library(MASS)
data(survey)
tbl <- table(survey$Smoke,survey$Exer)
tbl
```

```
##
##         Freq None Some
##   Heavy    7    1    3
##   Never   87   18   84
##   Occas   12    3    4
##   Regul    9    1    7
```

Problem:

Test the hypothesis whether the students smoking habit is independent of their exercise level at 0.05 significance level.

```
chi2test <- chisq.test(tbl)
chi2test
```

```
##
##  Pearson's Chi-squared test
##
## data:  tbl
## X-squared = 5.4885, df = 6, p-value = 0.4828
```

As the p-value is 0.4828422 greater than the 0.05 significance level, we do not reject the null hypothesis that the smoking habit is independent of the exercise level of the students.

The warning message found in the solution above is due to the small cell values in the contingency table. To avoid such warning, we combine the second and third columns of `tbl`, and save it in a new table named `ctbl`. Then we apply the `chisq.test` function against `ctbl` instead.

```
ctbl <- cbind(tbl[,"Freq"], tbl[,"None"] + tbl[,"Some"])
ctbl
```

```
##       [,1] [,2]
## Heavy    7    4
## Never   87  102
## Occas   12    7
## Regul    9    8
```

```
chi2test <- chisq.test(ctbl)
chi2test
```

```
##
##  Pearson's Chi-squared test
##
## data:  ctbl
## X-squared = 3.2328, df = 3, p-value = 0.3571
```

The p-value is 0.3571031 also greater than the 0.05 significance level.

5.7 Mann-Whitney U test

Also known as Mann-Whitney-Wilcoxon (MWW), Wilcoxon rank-sum test, or Wilcoxon-Mann-Whitney test is a non-parametric test of the null hypothesis that two samples come from the same sample population against an alternative hypothesis, especially that a particular population tends to have larger values than the other. Is the alternative test to the independent sample t-test. As a non-parametric test, it does not assume any assumptions related to the distribution. Hence, we can decide whether the population distributions are identical without assuming them to follow the normal distribution.

There are, however, some assumptions that are assumed:

1. The sample drawn from the population is random.
2. Independence within the samples and mutual independence is assumed.

3. Ordinal measurement scale is assumed.

Mann-Whitney U test is used for every field, but in frequently used in psychology, medical/nursing and business. For example, in psychology, it is used to compare attitude or behavior, etc. In medicine, it is used to know the effect of two medicines and whether they are equal or not. It is also used to know whether or not a particular medicine cures the ailment or not. In business, it can be used to know the preferences of different people and it can be used to see if that changes depending on location.

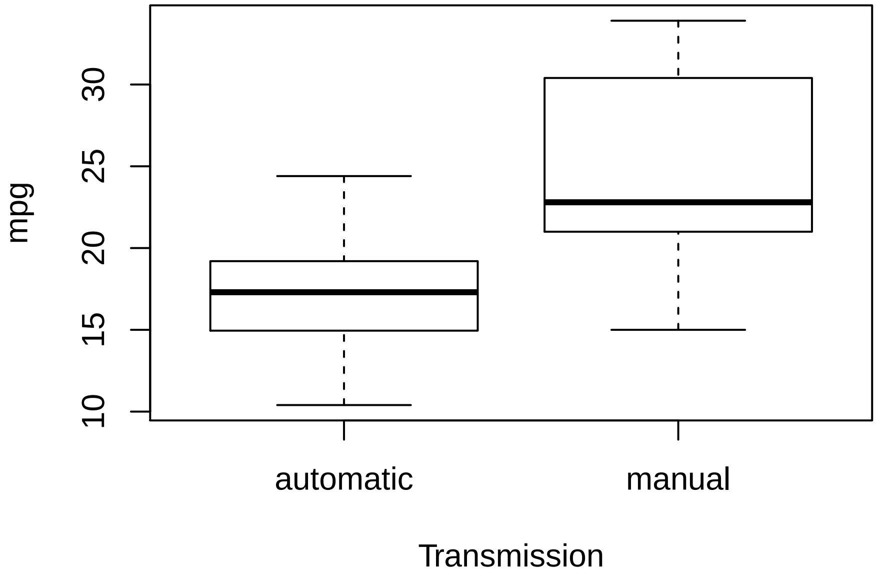
Example Consider the `mtcars` data set. We are aim to find out whether automatic transmission cars are less fuel efficient compared to manual transmission cars. Since 1950, most cars sold in North America have been available with an automatic transmission.

The `mtcars` dataset is from the 1974 “Motor Trend US” magazine, consisting of fuel consumption measurement (mpg) and 10 different aspects of automoblie design and performance for 32 automobiles (1973-74 models).

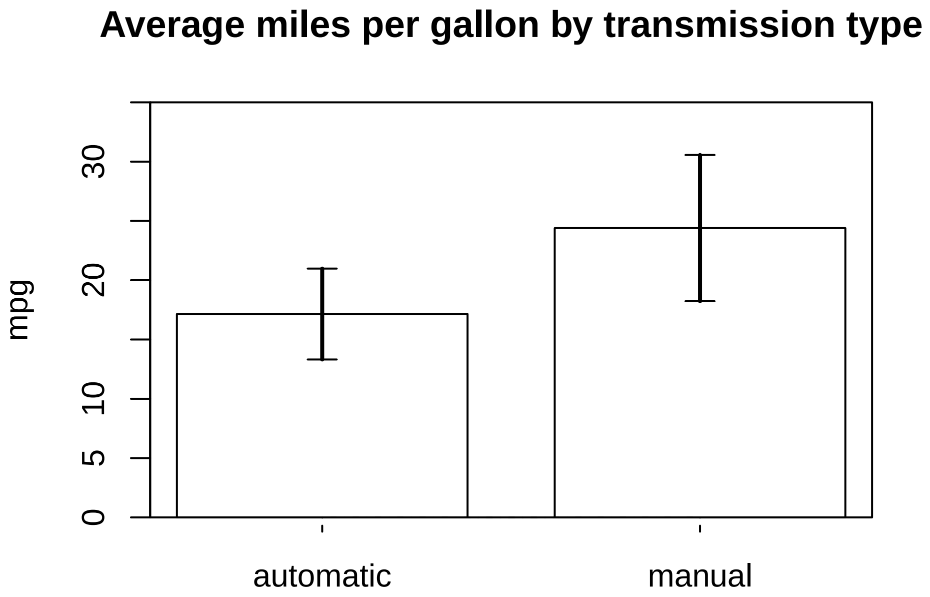
We first plot the data (mpg vs transmission)

either with a boxplot

```
boxplot(mpg~am,data=mtcars,names=c("automatic","manual"),ylab="mpg",xlab="Transmission")
```



or with a bar chart (with error bars)



We run some statistical tests to compare mpg between automatic- and manual-transmitted cars.

The first one is a t-test, assuming that the mileage data have a normal distribution. The test result clearly shows that the manual transmission cars are more gas-efficient than automatic transmission cars (miles per gallon: 24.39 versus 17.15).

```
t.test(mpg ~ factor(am), data = mtcars)
```

```
##
##  Welch Two Sample t-test
##
## data:  mpg by factor(am)
## t = -3.7671, df = 18.332, p-value = 0.001374
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -11.280194  -3.209684
## sample estimates:
## mean in group 0 mean in group 1
##        17.14737        24.39231
```

However, the normality assumption could be quite strong, given the fact that we do not know the true underlying distributions of `mpg` data. Moreover, the number of data points are not large enough to apply the central limit theorem. Therefore, a more conservative test would be the Wilcoxon test, with the null hypothesis that the gas mileage data of manual and automatic transmissions are from identical populations.

```r
wilcox.test(mpg ~ am, data = mtcars)
```

```
## Warning in wilcox.test.default(x = c(21.4, 18.7, 18.1, 14.3, 24.4, 22.8, :
## cannot compute exact p-value with ties
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  mpg by am
## W = 42, p-value = 0.001871
## alternative hypothesis: true location shift is not equal to 0
```

A non-parametric, Wilcoxon test also rejects the null hypothesis that the mileage data of the manual and automatic transmissions are from the same population (indicating a difference).

6 Linear models in R

6.1 Simple linear regression

- Regression is a statistical method used to predict the value of a response variable based on the values of a set of explanatory variables.
- One very general form for the model would be

$$y = f(x_1, x_2, ..., x_p) + \epsilon,$$

where f is some unknown function and ϵ is the error in this representation. Since we usually don't have enough data to try to estimate f directly (*inverse problem*), we usually have to assume that it has some restricted form.

- Any statistical model attempts to approximate the response variable or dependent variable y as a mathematical function of the explanatory variables or regressors X (also called covariates or independent variables).
- The simplest and most common form is the **linear model (LM)**

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_z + \epsilon,$$

where β_i $i = 0, 1, 2$ are *unknown* parameters. β_0 is called the intercept term. Hence, the problem is reduced to the estimation of four values rather than the complicated infinite dimensional f.

- A simple linear model with a single exploratory variable is defined as:

$$\hat{y} = \beta_0 + \beta_1 x$$

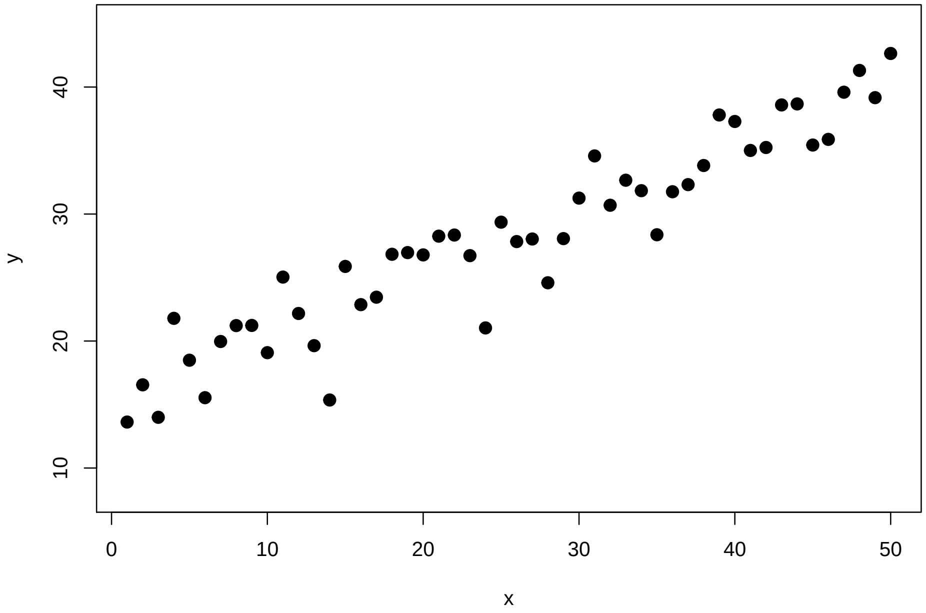
where $\hat{y}$ is the fitted values for β_0 (intercept) and β_1 (slope). Then for a given x_i we obtain a $\hat{y}_i$ that approximates y_i

Let us create a toy example (with $p = 1$):

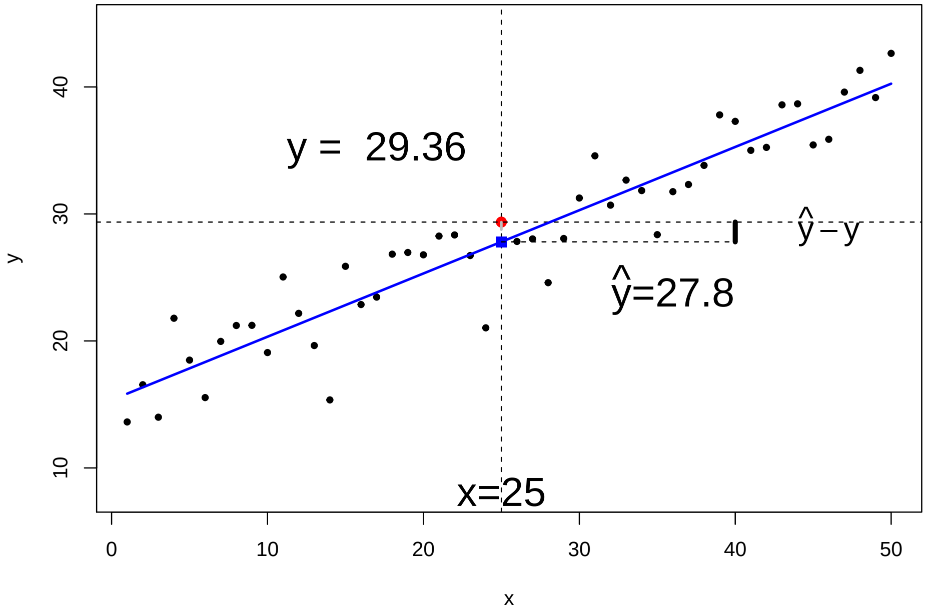
```
set.seed(1)
n <- 50

x <- seq(1,n)
 beta0 <- 15
 beta1 <- 0.5

sigma <- 3 # standar deviation of the errors
eps <- rnorm(n,mean=0,sd=3) # generate gaussian random errors

# Generate random data
 y <- beta0 + beta1*x  +  eps
```

Plot the data

```
plot(x,y,ylim = c(8,45), cex=1.3, xlab = "x", ylab="y",pch=19)
```

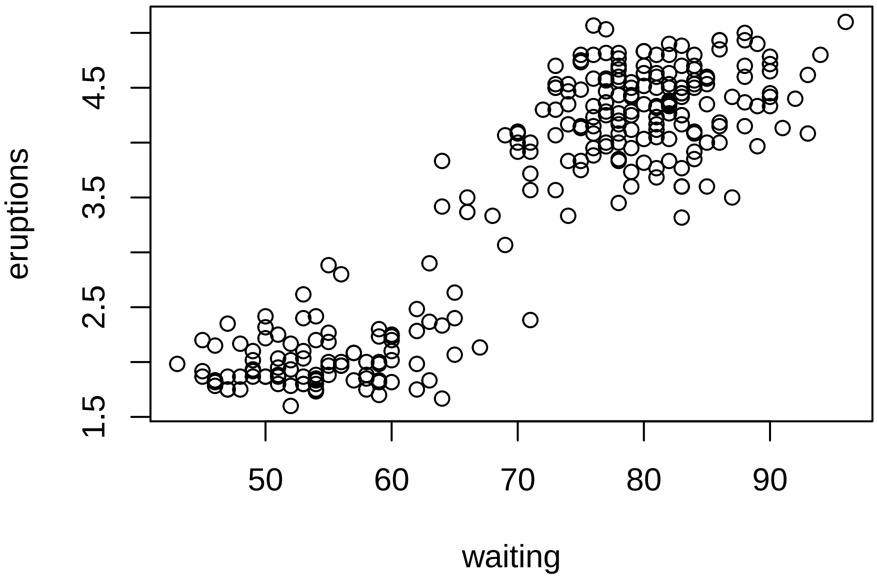


A mathematical procedure for finding the best-fitting curve to a given set of points by minimizing the sum of the squares of the residuals of the points from the fitted line. Illustration of the least squares fit



For example, in the data set `faithful`, it contains sample data of two random variables named `waiting` and `eruptions`. The `waiting` variable denotes the waiting time until the next eruptions, and `eruptions` denotes the duration.

```
plot(eruptions~waiting,data=faithful)
```



Its linear regression model can be expressed as:

$$Eruptions = \beta_0 + \beta_1 * Waiting + \epsilon$$

If we choose the parameters β_0 and β_1 in the simple linear regression model so as to minimize the sum of squares of the error term ϵ. Suppose for the data set `faithful`, we aim to estimate the next eruption duration if the waiting time since the last eruption has been 80 minutes.

We apply the `lm` function to a formula that describes the variable eruptions by the variable waiting, and save the linear regression model in a new variable `eruption.lm`.

```
data("faithful")
eruption.lm <- lm(eruptions~waiting,data=faithful)
```

Then we extract the parameters of the estimated regression equation with the coefficients function.

```
coeffs <- coefficients(eruption.lm); coeffs
```

```
## (Intercept)     waiting
## -1.87401599  0.07562795
```

We now fit the eruption duration using the estimated regression equation.

```
waiting = 80            # the waiting time
duration = coeffs[1] + coeffs[2]*waiting
duration
```

```
## (Intercept)
##     4.17622
```

Based on the simple linear regression model, if the waiting time since the last eruption has been 80 minutes, we expect the next one to last 4.1762198 minutes.

We wrap the waiting parameter value inside a new data frame named newdata.

```
newdata = data.frame(waiting=80) # wrap the parameter
```

Then we apply the `predict` function to `eruption.lm` along with `newdata`.

```
predict(eruption.lm, newdata)    # apply predict
```

```
##       1
## 4.17622
```

We can directly calculate quantities of interest, i.e. the ordinary least squares solution consists of:

$$\min_{\beta_0,\beta_1} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Then $\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$ and $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x}$

In matrix form, with $X = [1 : x_1 : ... : x_p]$

$$\hat{\beta} = (X'X)^{-1}X'$$

where $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1)$

6.2 Defining models in `R`

To complete a linear regression using `R` it is first necessary to understand the syntax for defining models.

Syntax	Model	Comment
`y ~ x`	$y = \beta_0 + \beta_1 x$	Straight-line with an im
`y ~ -1 + x`	$y = \beta_1 x$	Straight-line with no intercept; that
`y ~ x + I(x^2)`	$y = \beta_0 + \beta_1 x + \beta_2 x^2$	Polynomial model; `I()` allows fo
`y ~ x + z`	$y = \beta_0 + \beta_1 x + \beta_2 z$	Multiple regressio
`y ~ x:z`	$y = \beta_0 + \beta_1 xz$	Model with interaction b
`y ~ x*z`	$y = \beta_0 + \beta_1 x + \beta_2 z + \beta_3 xz$	Equivalent to `y~`

In R function model.matrix helps us to create the X matrix.

```
x <- model.matrix(~waiting, data = faithful)
y <- faithful$eruptions
xtxi <- solve(t(x) %*% x)
betas <- xtxi %*% t(x) %*% y
betas
```

```
##                    [,1]
## (Intercept) -1.87401599
## waiting      0.07562795
```

or

```
solve(crossprod(x, x), crossprod(x, y))
```

```
##                    [,1]
## (Intercept) -1.87401599
## waiting      0.07562795
```

Of course, it is not necessary here because the lm() function does the job but it is very useful when the statistic you want is not part of the pre-packaged functions.

It is possible to get $(X'X)^{-1}$ as

```
summary(eruption.lm)$cov.unscaled
```

```
##               (Intercept)       waiting
## (Intercept)  0.104029479 -1.415475e-03
## waiting     -0.001415475  1.996521e-05
```

The names() command is the way to see the components of an R object

```r
names(eruption.lm)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

To access for example

- Fitted (or predictd) values ($\hat{y}$):

```r
eruption.lm$fitted.values
```

- Residuals ($y - \hat{y}$)

```r
eruption.lm$residuals
```

We can estimate σ as $\sigma = \frac{(y_i - \hat{y}_i)^2}{n-p}$ in `R`

```r
eruption.lm.sum <- summary(eruption.lm)
names(eruption.lm.sum)
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

```r
sqrt(deviance(eruption.lm)/df.residual(eruption.lm))
```

```
## [1] 0.4965129
```

```r
# is obtained directly as
eruption.lm.sum$sigma
```

```
## [1] 0.4965129
```

We may also obtain the standard errors for the coefficients. Also `diag()` returns the diagonal of a matrix:

```r
xtxi
```

```
##              (Intercept)       waiting
## (Intercept)  0.104029479 -1.415475e-03
## waiting     -0.001415475  1.996521e-05
```

```
sqrt(diag(xtxi)) * eruption.lm.sum$sigma
```

```
## (Intercept)     waiting
## 0.160143302 0.002218541
```

```
eruption.lm.sum$coef[, 2]
```

```
## (Intercept)     waiting
## 0.160143302 0.002218541
```

6.3 Coefficient of Determination

The **coefficient of determination** of a linear regression model is the quotient of the variances of the fitted values and observed values of the dependent variable. If we denote y_i as the observed values of the dependent variable, $\bar{y}$ as its mean, and $\bar{y}_i$ as the fitted value, then the coefficient of determination is:

$$R^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{(y_i - \bar{y})^2}$$

```
summary(eruption.lm)$r.squared
```

```
## [1] 0.8114608
```

or

```
1-sum(eruption.lm$res^2)/sum((y-mean(y))^2)
```

```
## [1] 0.8114608
```

More options:

- `fitted.values()` or `fitted()` Fitted values of the model
- `predict()`: to predict $\hat{y}_*$ for new values of x_*
- `confint()`: confidence intervals for model parameters
- `resid()`: residuals of the model
- `anova()`: ANOVA table for residuals
- `deviance()`: deviance of the fitted model, in the LM case is $\sum_i^n (\hat{y}_i - y_i)^2$

See Faraway's (2002) book (Chapters 1-7)

6.4 Significance Test for Linear Regression

Assume that the error term ?? in the linear regression model is independent of x, and is normally distributed, with zero mean and constant variance. We can decide whether there is any significant relationship between x and y by testing the null hypothesis that $\beta_1 = 0$.

we print out the F-statistics of the significance test with the summary function.

```
summary(eruption.lm)
```

```
##
## Call:
## lm(formula = eruptions ~ waiting, data = faithful)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.29917 -0.37689  0.03508  0.34909  1.19329
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.874016   0.160143  -11.70   <2e-16 ***
## waiting      0.075628   0.002219   34.09   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4965 on 270 degrees of freedom
## Multiple R-squared:  0.8115, Adjusted R-squared:  0.8108
## F-statistic:  1162 on 1 and 270 DF,  p-value: < 2.2e-16
```

6.5 Confidence Interval for Linear Regression

Assume that the error term ϵ in the linear regression model is independent of x, and is normally distributed, with zero mean and constant variance. For a given value of x, the interval estimate for the mean of the dependent variable, $\bar{y}$, is called the confidence interval.

A 95% confidence interval of the mean eruption duration for the waiting time of 80 minutes is given by

```
predict(eruption.lm, newdata, interval="confidence")
```

```
##       fit      lwr      upr
## 1 4.17622 4.104848 4.247592
```

The 95% confidence interval of the mean eruption duration for the waiting time of 80 minutes is between 4.1048 and 4.2476 minutes.

6.6 Prediction Interval for Linear Regression

For a given value of x, the interval estimate of the dependent variable y is called the prediction interval.

```r
predict(eruption.lm, newdata, interval="predict")
```

```
##       fit      lwr      upr
## 1 4.17622 3.196089 5.156351
```

The 95% prediction interval of the eruption duration for the waiting time of 80 minutes is between 3.1961 and 5.1564 minutes.

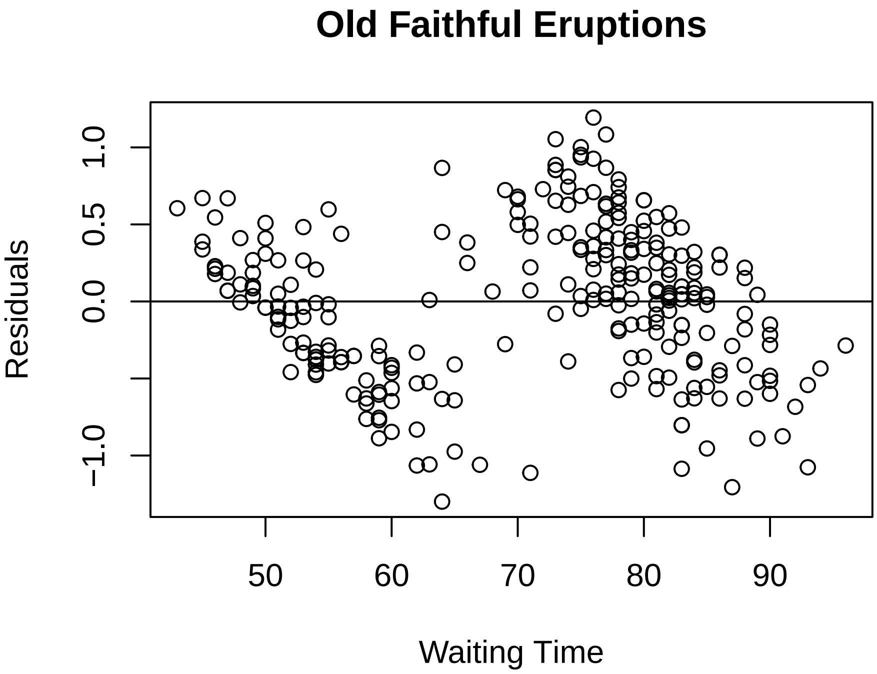
6.7 Residual Plot

The residual data of the simple linear regression model is the difference between the observed data of the dependent variable y and the fitted values $\hat{y}$.

$$Residual = y - \hat{y}$$

```r
eruption.res = resid(eruption.lm)
```

```r
plot(faithful$waiting,
    eruption.res,
    ylab="Residuals", xlab="Waiting Time",
    main="Old Faithful Eruptions")
abline(0, 0)
```



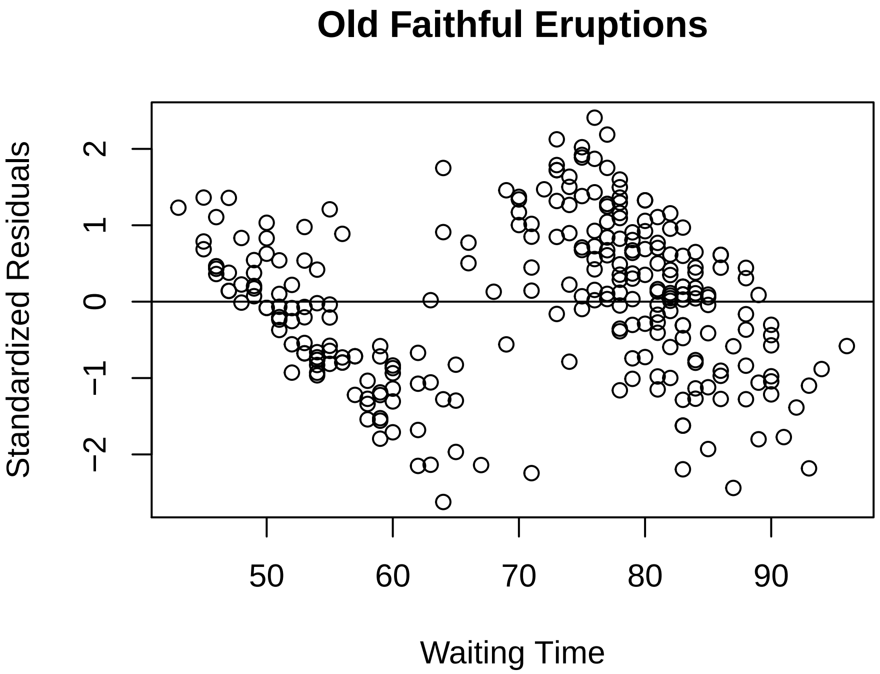
6.8 Standardized Residual

The standardized residual is the residual divided by its standard deviation.

$$\text{Standardized residual}_i = \frac{Residual_i}{SD.of.Residual_i}$$

```
eruption.stdres <- rstandard(eruption.lm)
```

```
plot(faithful$waiting, eruption.stdres,
      ylab="Standardized Residuals",
      xlab="Waiting Time",
      main="Old Faithful Eruptions")
abline(0, 0)
```



As the p-value is much less than 0.05, we reject the null hypothesis that $\beta_1 = 0$. Hence there is a significant relationship between the variables in the linear regression model of the data set faithful.

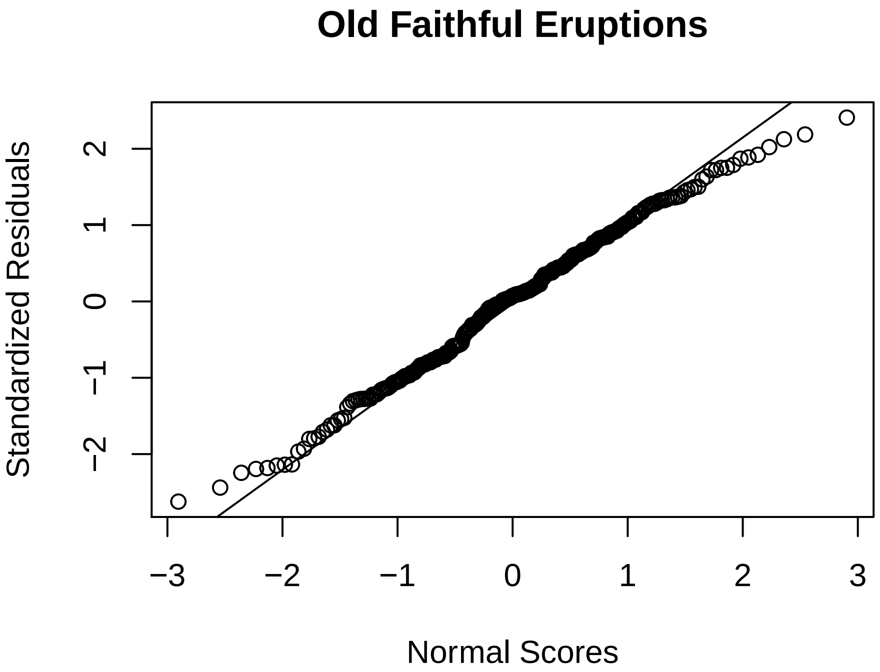
6.9 Normal Probability Plot of Residuals

The normal probability plot is a graphical tool for comparing a data set with the normal distribution. We can use it with the standardized residual of the linear regression model and see if the error term ϵ is actually normally distributed.

```
eruption.lm = lm(eruptions ~ waiting, data=faithful)
eruption.stdres = rstandard(eruption.lm)
```

We now create the normal probability plot with the `qqnorm` function, and add the `qqline` for further comparison.

```
qqnorm(eruption.stdres,
    ylab="Standardized Residuals",
    xlab="Normal Scores",
    main="Old Faithful Eruptions")
qqline(eruption.stdres)
```



6.10 Multiple linear regression

A multiple linear regression (MLR) model that describes a dependent variable y by independent variables $x_1, x_2, ..., x_p$ $(p > 1)$ is expressed by the equation:

$$y = \beta_0 + \sum_{k}^{p} \beta_k + \epsilon$$

where the numbers β_0 and β_k $(k = 1, 2, ..., p)$ are the parameters, and ϵ is the error term.

Example:

Consider the data `stackloss` from observations of a chemical plant operation, if we assign `stackloss` as the dependent variable, and assign `Air.Flow` (cooling air flow), `Water.Temp` (inlet water temperature) and `Acid.Conc.` (acid concentration) as independent variables, the multiple linear regression model is:

$$stack.loss = \beta_0 + \beta_1 * Air.Flow + \beta_2 * Water.Temp + \beta_3 * Acid.Conc + \epsilon$$

```
data("stackloss")
?stackloss
head(stackloss)
```

```
##   Air.Flow Water.Temp Acid.Conc. stack.loss
## 1       80         27         89         42
## 2       80         27         88         37
## 3       75         25         90         37
## 4       62         24         87         28
## 5       62         22         87         18
## 6       62         23         87         18
```

Fit the multiple linear regression in R

```
stackloss.lm = lm(stack.loss ~ Air.Flow + Water.Temp + Acid.Conc., data=stackloss)
stackloss.lm
```

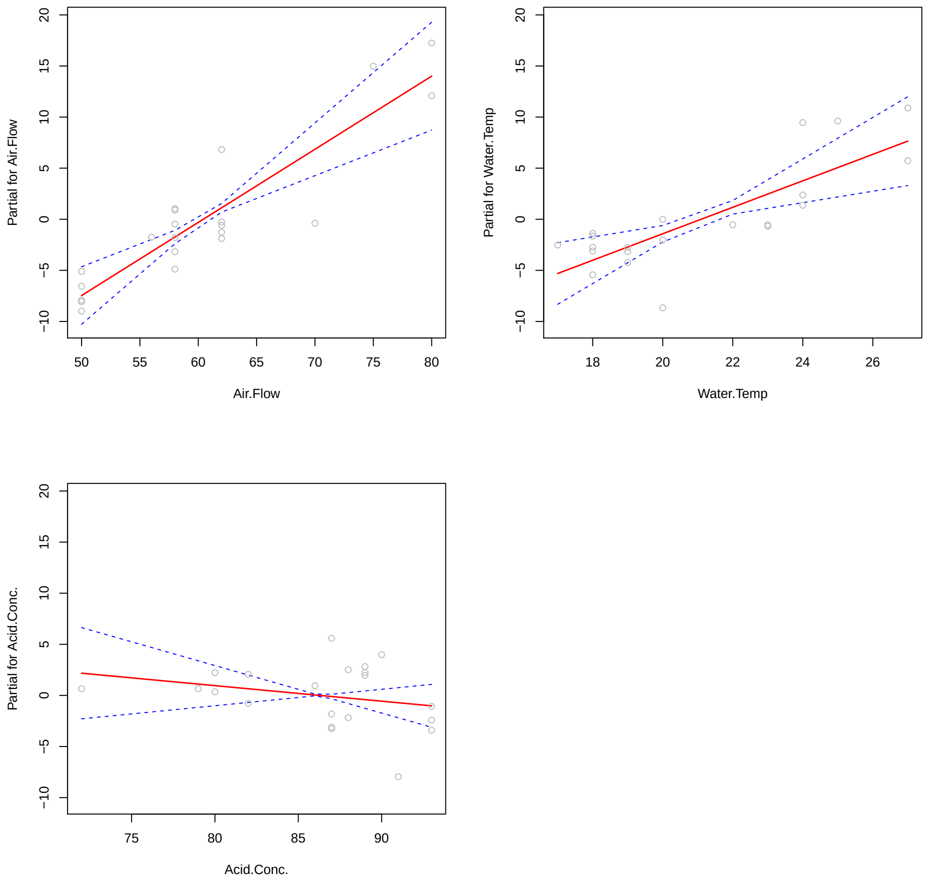
```
##
## Call:
## lm(formula = stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.,
##     data = stackloss)
##
## Coefficients:
## (Intercept)     Air.Flow   Water.Temp   Acid.Conc.
##    -39.9197       0.7156       1.2953      -0.1521
```

```
summary(stackloss.lm)
```

```
##
## Call:
## lm(formula = stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.,
##     data = stackloss)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2377 -1.7117 -0.4551  2.3614  5.6978
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -39.9197    11.8960  -3.356  0.00375 **
## Air.Flow      0.7156     0.1349   5.307  5.8e-05 ***
## Water.Temp    1.2953     0.3680   3.520  0.00263 **
## Acid.Conc.   -0.1521     0.1563  -0.973  0.34405
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.243 on 17 degrees of freedom
## Multiple R-squared:  0.9136, Adjusted R-squared:  0.8983
## F-statistic:  59.9 on 3 and 17 DF,  p-value: 3.016e-09
```

Function `termplot` plots regression terms against their predictors:

```
?termplot
par(mfrow=c(2,2))
termplot(stackloss.lm, partial.resid = TRUE, se=TRUE,col.se = "blue")
```



What is the stack loss if the air flow is 72, water temperature is 20 and acid concentration is 85?

Create a new data frame:

```
newdata <- data.frame(Air.Flow=72,Water.Temp=20,Acid.Conc.=85)
```

Use `predict`

```
predict(stackloss.lm, newdata)
```

```
##        1
## 24.58173
```

Based on the multiple linear regression model and the given parameters, the predicted stack loss is 24.5817284.

To obtain the multiple coefficient of determination

```
summary(stackloss.lm)$r.squared
```

```
## [1] 0.9135769
```

6.10.1 Adjusted coefficient of determination

The adjusted coefficient of determination of a multiple linear regression model is defined in terms of the coefficient of determination as follows, where n is the number of observations in the data set, and p is the number of independent variables.

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

```
summary(stackloss.lm)$adj.r.squared
```

```
## [1] 0.8983258
```

6.10.2 Significant tests and confidence/prediction intervals

```
summary(stackloss.lm)
```

```
##
## Call:
## lm(formula = stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.,
##     data = stackloss)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2377 -1.7117 -0.4551  2.3614  5.6978
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -39.9197    11.8960  -3.356  0.00375 **
## Air.Flow      0.7156     0.1349   5.307  5.8e-05 ***
## Water.Temp    1.2953     0.3680   3.520  0.00263 **
## Acid.Conc.   -0.1521     0.1563  -0.973  0.34405
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.243 on 17 degrees of freedom
## Multiple R-squared:  0.9136, Adjusted R-squared:  0.8983
## F-statistic:  59.9 on 3 and 17 DF,  p-value: 3.016e-09
```

As the p-values of `Air.Flow` and `Water.Temp` are less than 0.05, they are both statistically significant in the multiple linear regression model of `stackloss`.

95% Confidence intervals of the stack loss if the air flow is 72, water temperature is 20 and acid concentration is 85.

```r
predict(stackloss.lm, newdata, interval="confidence")
```

```
##        fit      lwr    upr
## 1 24.58173 20.21846 28.945
```

95% Prediction intervals are

```r
predict(stackloss.lm, newdata, interval="prediction")
```

```
##        fit     lwr      upr
## 1 24.58173 16.4661 32.69736
```

6.11 Linear regression with factor variables

Let us consider the `mtcars` we analyzed previously. In Section.

```r
data(mtcars)
t.test(mpg ~ am, data=mtcars)
```

```
##
##  Welch Two Sample t-test
##
## data:  mpg by am
## t = -3.7671, df = 18.332, p-value = 0.001374
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -11.280194  -3.209684
## sample estimates:
## mean in group 0 mean in group 1 
##        17.14737        24.39231
```

The results from the statistical tests focus on `mpg` and `am` only, without controlling for influences from other variables.

The benefit of regressional analysis serve the purpose. If we apply a multivariate regression to control for certain available design and performance variables, the marginal impact of automatic- or manual- transmission cars does not turn out to be significant. The confounding variables include displacement (disp), rear axle ratio (drat) and car weight(wt). Take car weight for example. The regression suggests that, holding other variables constant (ceteris paribus), maual tramsmitted cars consume on avarage -0.024 more gallons of gas per mile, and the results are no longer statistically significant. Similar analysis work similarly for the other two variables: drat and wt.

```
fit0 <- lm(mpg ~ factor(am), data = mtcars)
summary(fit0)
```

```
## 
## Call:
## lm(formula = mpg ~ factor(am), data = mtcars)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -9.3923 -3.0923 -0.2974  3.2439  9.5077 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)   17.147      1.125  15.247 1.13e-15 ***
## factor(am)1    7.245      1.764   4.106 0.000285 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 4.902 on 30 degrees of freedom
## Multiple R-squared:  0.3598, Adjusted R-squared:  0.3385 
## F-statistic: 16.86 on 1 and 30 DF,  p-value: 0.000285
```

If we apply a multivariate regression to control for certain available design and performance variables, the marginal impact of automatic- or manual- transmission cars does not turn out to be significant. The confounding variables include displacement (`disp`), rear axle ratio (`drat`) and car weight(`wt`). Take car weight for example:

```
fit1 <- lm(mpg ~ factor(am) + wt, data = mtcars)
summary(fit1)
```

```
##
## Call:
## lm(formula = mpg ~ factor(am) + wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5295 -2.3619 -0.1317  1.4025  6.8782
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.32155    3.05464  12.218 5.84e-13 ***
## factor(am)1 -0.02362    1.54565  -0.015    0.988
## wt          -5.35281    0.78824  -6.791 1.87e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.098 on 29 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7358
## F-statistic: 44.17 on 2 and 29 DF,  p-value: 1.579e-09
```

The regression suggests that, holding other variables constant, manual tramsmitted cars consume on average -0.024 more gallons of gas per mile, and the results are no longer statistically significant. Similar analysis work similarly for the other two variables: `drat` and `wt`.

```
fit2 <- lm(mpg ~ factor(am) + drat, data = mtcars)
summary(fit2)
```

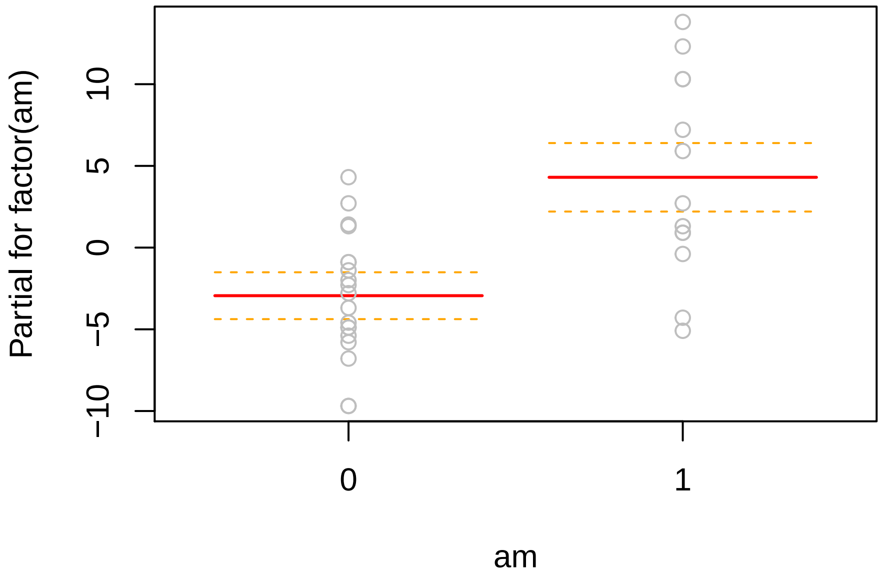
```
##
## Call:
## lm(formula = mpg ~ factor(am) + drat, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5802 -2.5206 -0.5153  2.4419  8.5198
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.950      7.073  -0.276   0.7848
## factor(am)1    2.807      2.282   1.230   0.2286
## drat           5.811      2.130   2.728   0.0107 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.448 on 29 degrees of freedom
## Multiple R-squared:  0.4906, Adjusted R-squared:  0.4554
## F-statistic: 13.96 on 2 and 29 DF,  p-value: 5.659e-05
```

```
fit3 <- lm(mpg ~ factor(am) + disp, data = mtcars)
summary(fit3)
```

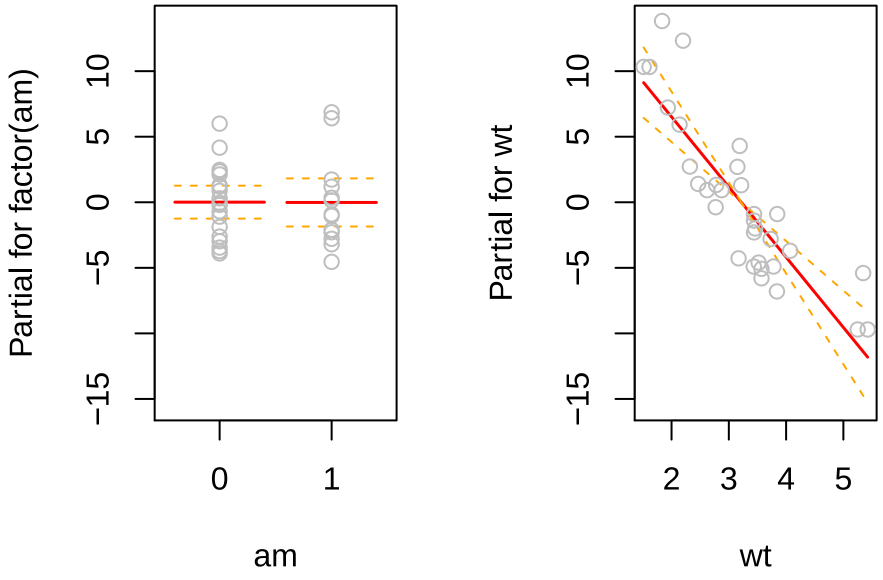
```
##
## Call:
## lm(formula = mpg ~ factor(am) + disp, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6382 -2.4751 -0.5631  2.2333  6.8386
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 27.848081   1.834071  15.184 2.45e-15 ***
## factor(am)1  1.833458   1.436100   1.277    0.212
## disp        -0.036851   0.005782  -6.373 5.75e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.218 on 29 degrees of freedom
## Multiple R-squared:  0.7333, Adjusted R-squared:  0.7149
## F-statistic: 39.87 on 2 and 29 DF,  p-value: 4.749e-09
```

We can use `termplot`

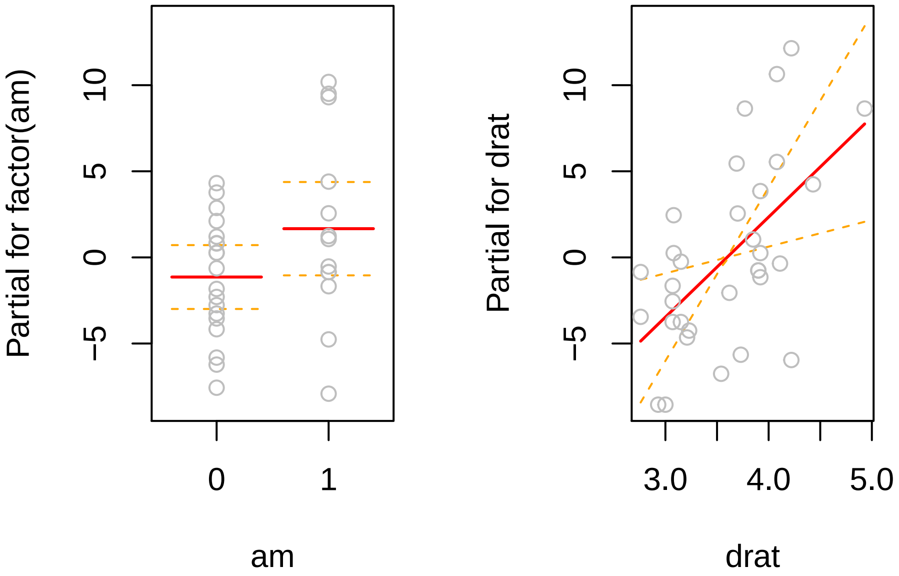
```
termplot(fit0,partial.resid = TRUE,se=TRUE)
```



```
par(mfrow=c(1,2))
termplot(fit1,partial.resid = TRUE,se=TRUE)
```



```r
termplot(fit2,partial.resid = TRUE,se=TRUE)
```



6.12 Inference

Hypothesis test to compare models: the Likelihood ratio test

A likelihood ratio test is a statistical test used to compare the goodness of fit of two models, one of which (the null model) is a special case of the other (the alternative model). The test is based on the likelihood ratio, which expresses how many times more likely the data are under one model than the other.

Comparing two models fit to the same data can be set up as a hypothesis testing problem. Let M_0 and M_1 denote the models. Consider as the null hypothesis "M_1 is not a significant improvement on M_0", and the alternative the negation. This hypothesis can often be formulated so that a statistic can be generated from the two models.

Normally, the models are nested in that the variables in M_0 are a subset of those in M_1. The statistic often involves the RSS (residual sum of squares) values for both models, adjusted by the number of parameters used. In linear regression this becomes an `anova` test (comparing variances).

More robust is a likelihood ratio test for nested models. When models are sufficiently specific to define a probability distribution for y, the model will report the log-likelihood, $\hat{L}$. Under some mild assumptions, $-2(\hat{L}_0 - \hat{L}_1)$ follows a Chi-squared distribution with degrees of freedom = difference in number of parameters on the two models.

The utility of a single model M_1 is often assessed by comparing it with the null model, that reflects no dependence of y on the explanatory variables. The model formula for the null model is `y~1`, i.e. we use a constant to approximate `y` (e.g.: the mean of `y`). The likelihood ratio test is implemented in the function `anova`:

```
M0 <- lm(mpg ~ 1, data = mtcars)
M1 <- lm(mpg ~ factor(am), data = mtcars)
summary(M0)
```

```
##
## Call:
## lm(formula = mpg ~ 1, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.6906 -4.6656 -0.8906  2.7094 13.8094
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   20.091      1.065   18.86   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.027 on 31 degrees of freedom
```

```
summary(M1)
```

```
##
## Call:
## lm(formula = mpg ~ factor(am), data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.3923 -3.0923 -0.2974  3.2439  9.5077
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   17.147      1.125  15.247 1.13e-15 ***
## factor(am)1    7.245      1.764   4.106 0.000285 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.902 on 30 degrees of freedom
## Multiple R-squared:  0.3598, Adjusted R-squared:  0.3385
## F-statistic: 16.86 on 1 and 30 DF,  p-value: 0.000285
```

```
anova(M0,M1)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ 1
## Model 2: mpg ~ factor(am)
##   Res.Df    RSS Df Sum of Sq     F   Pr(>F)
## 1     31 1126.0
## 2     30  720.9  1    405.15 16.86 0.000285 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The likelihood ratio test can also test the significance of predictors. Hence, we can compare the model `fit0` (where `am` is significant) with `fit1`, `fit2` or `fit3`, i.e.:

```
anova(fit0, fit1)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ factor(am)
## Model 2: mpg ~ factor(am) + wt
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     30 720.90
## 2     29 278.32  1    442.58 46.115 1.867e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(fit0, fit2)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ factor(am)
## Model 2: mpg ~ factor(am) + drat
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     30 720.90
## 2     29 573.64  1    147.26 7.4444 0.0107 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(fit0, fit3)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ factor(am)
## Model 2: mpg ~ factor(am) + disp
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     30 720.90
## 2     29 300.28  1    420.62 40.621 5.748e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

However, the likelihood ratio tests suggest that it is important to consider these dimensions (i.e., displacement, rear axle ratio and weight) since these variables increase model fit.

6.13 ANALYSIS-OF-VARIANCE (ANOVA)

In an experiment study, various treatments are applied to test subjects and the response data is gathered for analysis. A critical tool for carrying out the analysis is the Analysis of Variance (ANOVA). It enables a researcher to differentiate treatment results based on easily computed statistical quantities from the treatment outcome.

The statistical process is derived from estimates of the population variances via two separate approaches. The first approach is based on the variance of the sample means, and the second one is based on the mean of the sample variances. Under the ANOVA assumptions as stated below, the ratio of the two statistical estimates follows the F distribution. Hence we can test the null hypothesis on the equality of various response data from different treatments via estimates of critical regions.

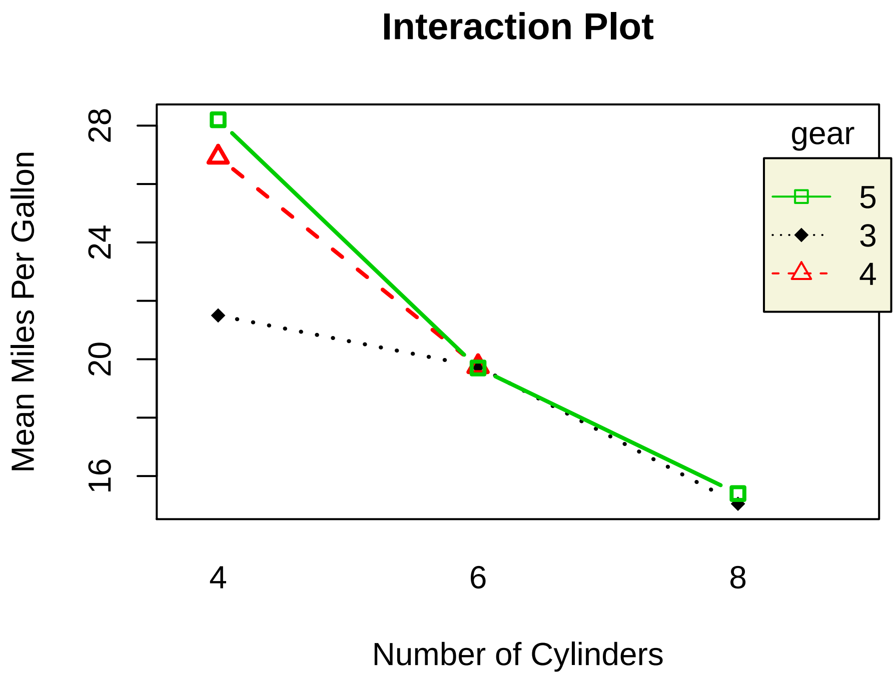
```
The treatment responses are independent of each other.
The response data follow the normal distribution.
The variances of the response data are identical.
```

In the following tutorials, we demonstrate how to perform ANOVA on a few basic experimental designs.

```r
# Two-way Interaction Plot
data(mtcars)
attach(mtcars)
```

```
## The following objects are masked from mtcars (pos = 11):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt
```

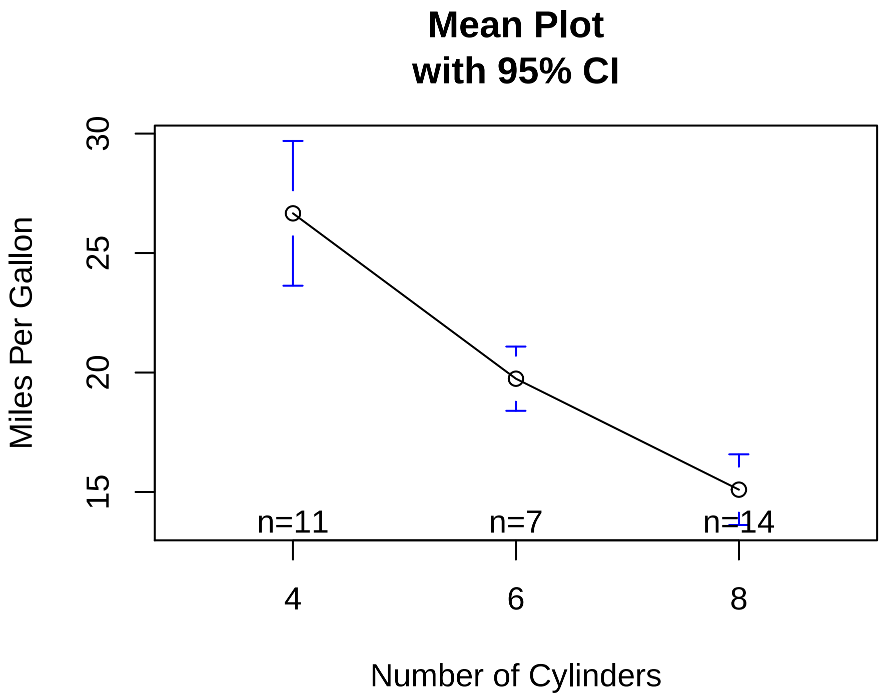
```r
gear <- factor(gear)
cyl <- factor(cyl)
interaction.plot(cyl, gear, mpg, type="b", col=c(1:3),
   leg.bty="o", leg.bg="beige", lwd=2, pch=c(18,24,22),
   xlab="Number of Cylinders",
   ylab="Mean Miles Per Gallon",
   main="Interaction Plot")
```



```r
# Plot Means with Error Bars
library(gplots)
attach(mtcars)
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     cyl, gear

## The following objects are masked from mtcars (pos = 3):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt

## The following objects are masked from mtcars (pos = 12):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt
```

```
cyl <- factor(cyl)
plotmeans(mpg~cyl,xlab="Number of Cylinders",
  ylab="Miles Per Gallon", main="Mean Plot\nwith 95% CI")
```



7 Logistic regression

A logistic regression is typically used when there is one dichotomous outcome variable (such as winning or losing), and a continuous predictor variable which is related to the probability or odds of the outcome variable. It can also be used with categorical predictors, and with multiple predictors.

7.1 ESR and Plasma Proteins

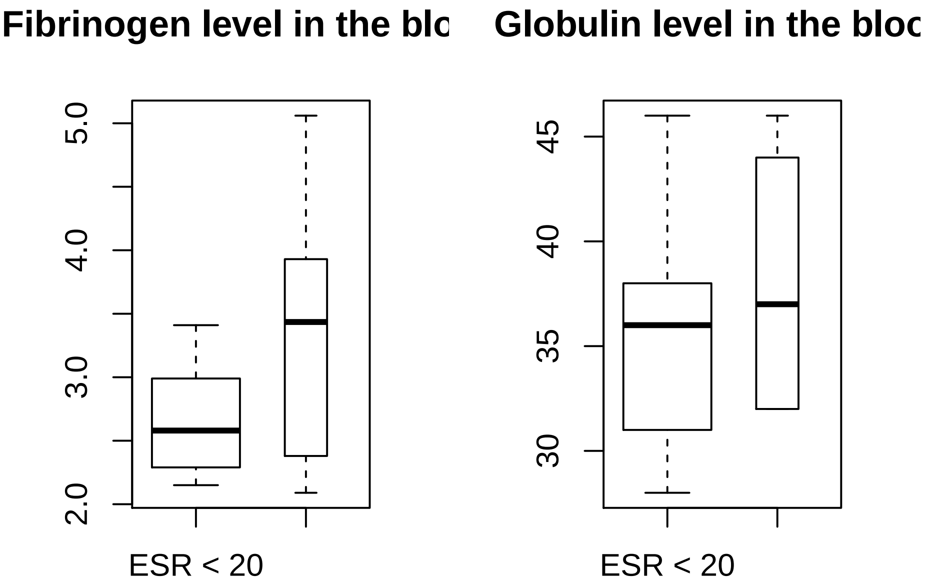
The erythrocyte sedimentation rate (ESR) is the rate at which red blood cells (erythrocytes) settle out of suspension in blood plasma, when measured under standard conditions. If the ESR increases when the level of certain proteins in the blood plasma rise in association with conditions such as rheumatic diseases, chronic infections and malignant diseases, its determination might be useful in screening blood samples taken from people suspected of suffering from one of the conditions mentioned. The absolute value of the ESR is not of great importance; rather, less than 20mm/hr indicates a “healthy” individual. To assess whether the ESR is a useful diagnostic tool, the question of interest is whether there is any association between the probability of an ESR reading greater than 20mm/hr and the levels of the two plasma proteins. If there is not then the determination of ESR would not be useful for diagnostic purposes. A data frame with 32 observations on the following 3 variables.

- `fibrinogen` the fibrinogen level in the blood.
- `globulin` the globulin level in the blood.
- `ESR` the erythrocyte sedimentation rate, either less or greater 20 mm / hour.

```
data("plasma", package = "HSAUR")
head(plasma)
```

```
##   fibrinogen globulin      ESR
## 1       2.52       38 ESR < 20
## 2       2.56       31 ESR < 20
## 3       2.19       33 ESR < 20
## 4       2.18       31 ESR < 20
## 5       3.41       37 ESR < 20
## 6       2.46       36 ESR < 20
```

```
layout(matrix(1:2, ncol = 2))
boxplot(fibrinogen ~ ESR, data = plasma, varwidth = TRUE, main="Fibrinogen level in the bloo
boxplot(globulin ~ ESR, data = plasma, varwidth = TRUE, main="Globulin level in the blood")
```



The question of interest is whether there is any association between the probability of an ESR reading greater than 20mm/hr and the levels of the two plasma proteins. If there is not then the determination of ESR would not be useful for diagnostic purposes.

Since the response variable is binary, a multiple regression model is not suitable for a regression analysis.

We may write

$$\mathbb{P}\text{r}(y_i = 1) = \pi_i \qquad \mathbb{P}\text{r}(y_i = 0) = 1 - \pi_i$$

Normally, we will have a set of covariates $X = (x_1, ..., x_p)$ associated with each individual, and our goal will be to investigate the relationship between the response probability $\pi = \pi(X)$ and the explanatory variables.

So instead of modelling the expected value of the response directly as a linear function of explanatory variables, a suitable transformation is modelled. In this case the most suitable transformation is the logistic or logit function of π leading to the model

$$\text{logit}(\pi) = \text{logit}\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$$

The logit of a probability is simply the log of the odds of the response taking the value one or logit transformation, of p: $logit(p) = \log(p/1-p)$. Logit is sometimes called "log odds." Because of the properties of odds given in the list above, the logit has these properties:

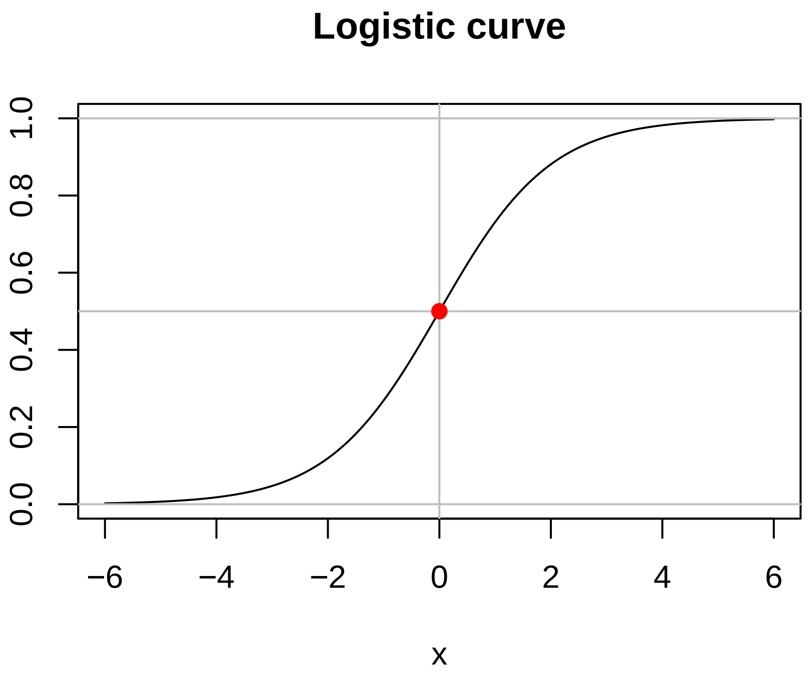
- If `odds(y=1s) = 1`, then `logit(p) = 0`.

- If `odds(y=1) < 1`, then `logit(p) < 0`.
- If `odds(y=1) > 1`, then `logit(p) > 0`.

The logit transform fails if $p = 0$‘.

When the response is a binary (dichotomous) variable, and x is numeric, logistic regression fits a logistic curve to the relationship between x and y. Hence, logistic regression is linear regression on the logit transform of y, where y is the proportion (or probability) of success at each value of x. However, you should avoid the temptation to do a traditional least-squares regression at this point, as neither the normality nor the homoscedasticity assumption will be met.

```r
x <- seq(-6,6,0.01)
logistic <- exp(x)/(1+exp(x))
plot(x,logistic,t='l',main="Logistic curve",ylab="")
abline(h=c(0,0.5,1),v=0,col="grey")
points(0,0.5,pch=19,col=2)
```



Logistic regression model in `R` can be fitted using the function `glm`. First, we start with a model that includes only a single predictor `fibrinogen`

```r
plasma_glm_1 <- glm(ESR ~ fibrinogen, data = plasma,family = binomial())
summary(plasma_glm_1)
```

```
##
## Call:
## glm(formula = ESR ~ fibrinogen, family = binomial(), data = plasma)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9298  -0.5399  -0.4382  -0.3356   2.4794
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -6.8451     2.7703  -2.471   0.0135 *
## fibrinogen    1.8271     0.9009   2.028   0.0425 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 30.885  on 31  degrees of freedom
## Residual deviance: 24.840  on 30  degrees of freedom
## AIC: 28.84
##
## Number of Fisher Scoring iterations: 5
```

We see that the regression coefficient for `fibrinogen` is significant at the 5% level. An increase of one unit in this variable increases the log-odds in favour of an ESR value greater than 20 by an estimated 1.83 with 95% confidence interval.

```
confint(plasma_glm_1,parm="fibrinogen")
```

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

```
##     2.5 %    97.5 %
## 0.3387619 3.9984921
```

These values are more helpful if converted to the corresponding values for the odds themselves by exponentiating the estimate

```
exp(coef(plasma_glm_1)["fibrinogen"])
```

```
## fibrinogen
##   6.215715
```

and the confidence interval

```
exp(confint(plasma_glm_1, parm = "fibrinogen"))
```

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

##      2.5 %    97.5 %
##  1.403209 54.515884
```

The confidence interval is very wide because there are few observations overall and very few where the ESR value is greater than 20. Nevertheless it seems likely that increased values of fibrinogen lead to a greater probability of an ESR value greater than 20. We can now fit a logistic regression model that includes both explanatory variables using the code

```
plasma_glm_2 <- glm(ESR ~ fibrinogen + globulin, data = plasma,family = binomial())
summary(plasma_glm_2)
```

```
##
## Call:
## glm(formula = ESR ~ fibrinogen + globulin, family = binomial(),
##     data = plasma)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9683  -0.6122  -0.3458  -0.2116   2.2636
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -12.7921     5.7963  -2.207   0.0273 *
## fibrinogen    1.9104     0.9710   1.967   0.0491 *
## globulin      0.1558     0.1195   1.303   0.1925
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 30.885  on 31  degrees of freedom
## Residual deviance: 22.971  on 29  degrees of freedom
## AIC: 28.971
##
## Number of Fisher Scoring iterations: 5
```

The coefficient for gamma globulin is not significantly different from zero.

Both nested models can be compared using a likelihood ratio test with `anova` function

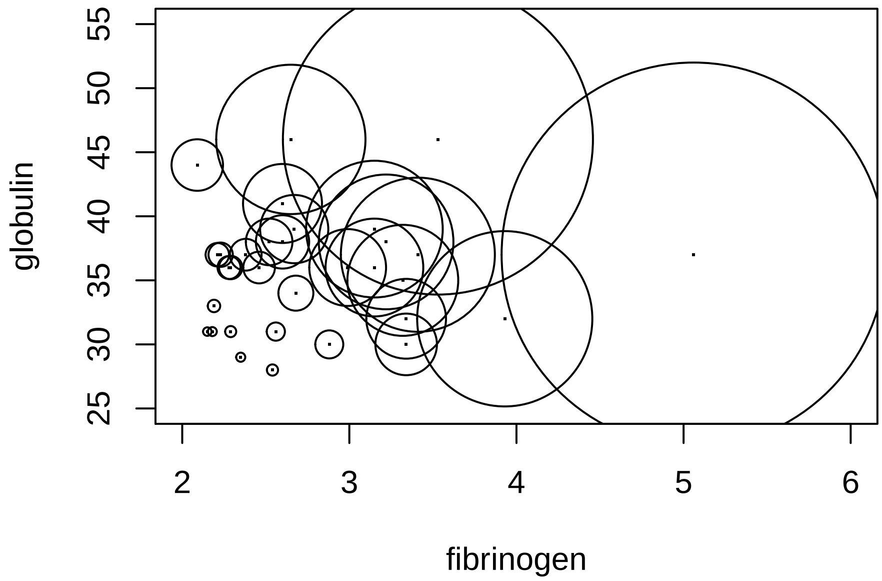
```
anova(plasma_glm_1, plasma_glm_2, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: ESR ~ fibrinogen
## Model 2: ESR ~ fibrinogen + globulin
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1        30     24.840
## 2        29     22.971  1   1.8692   0.1716
```

So we conclude that gamma globulin is not associated with ESR level.

The plot clearly shows the increasing probability of an ESR value above 20 (larger circles) as the values of fibrinogen, and to a lesser extent, gamma globulin, increase.

```
prob <- predict(plasma_glm_2,type="response")
plot(globulin ~ fibrinogen, data = plasma, xlim = c(2, 6),ylim = c(25, 55), pch = ".")
symbols(plasma$fibrinogen, plasma$globulin, circles = prob,add = TRUE)
```



8 Advanced graphics in R

8.1 lattice

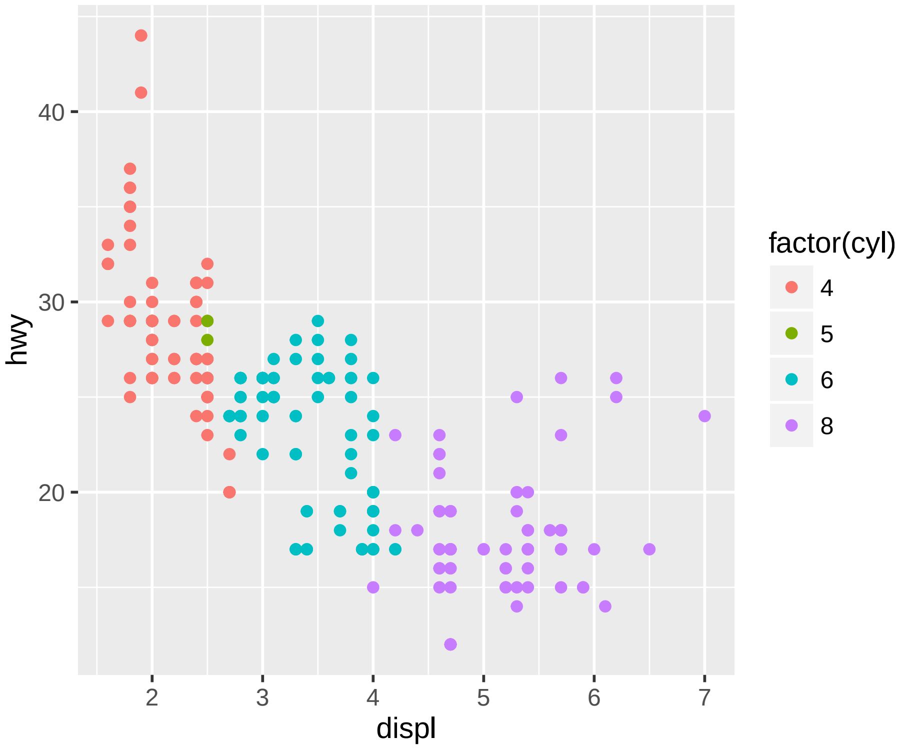
8.2 ggplot2

Why ggplot2?

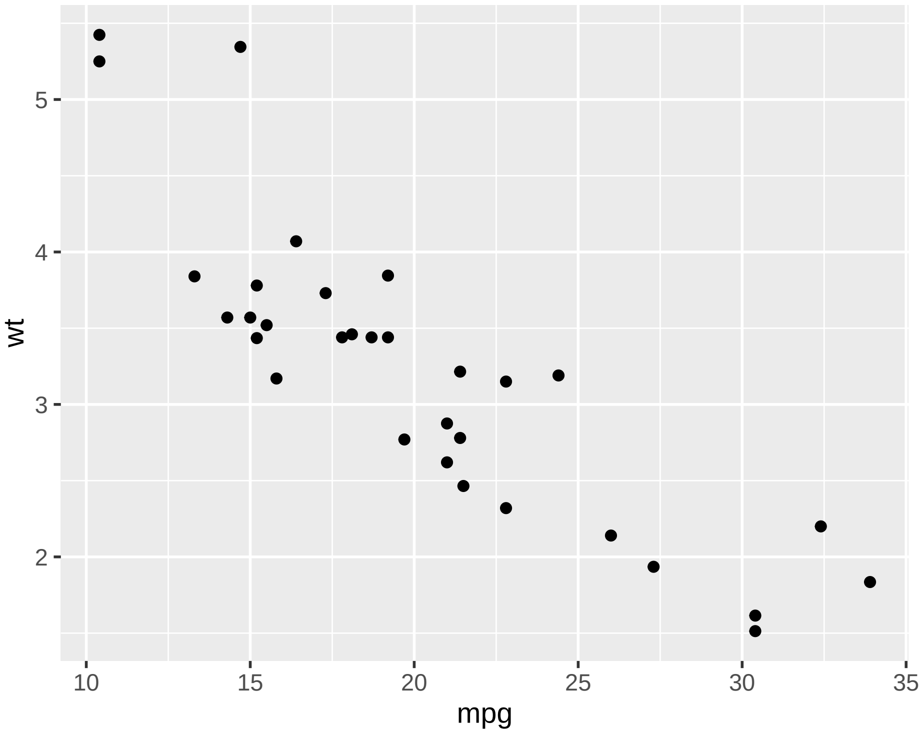
Advantages of ggplot2

- consistent underlying grammar of graphics (Wilkinson, 2005)
- plot specification at a high level of abstraction
- very flexible
- theme system for polishing plot appearance
- mature and complete graphics system
- many users, active mailing list

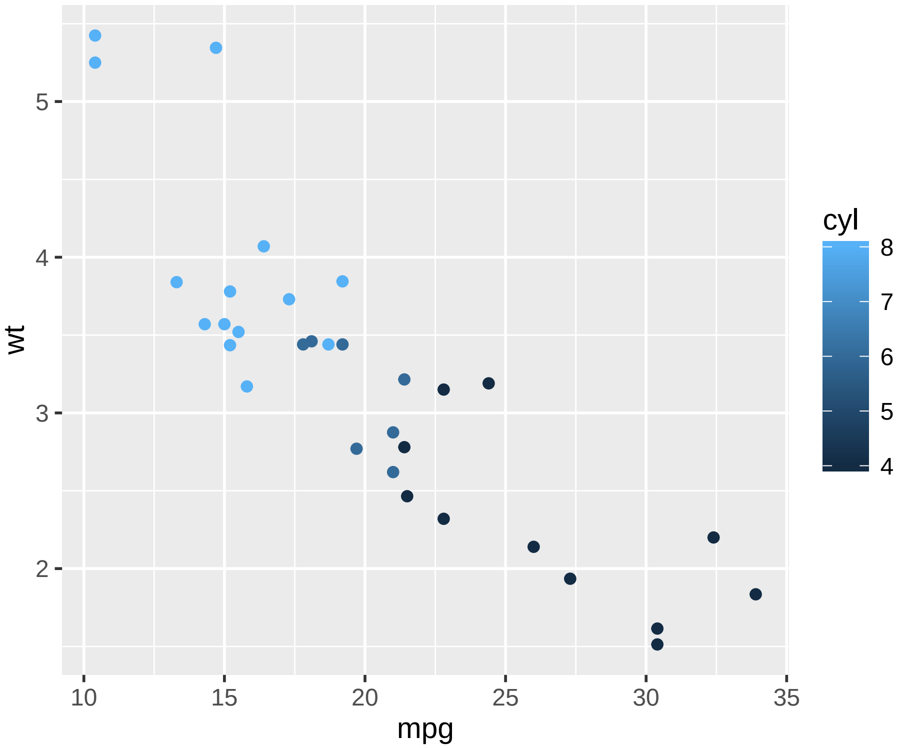
```r
library(ggplot2)
?qplot
qplot(displ, hwy, data = mpg, colour = factor(cyl))
```



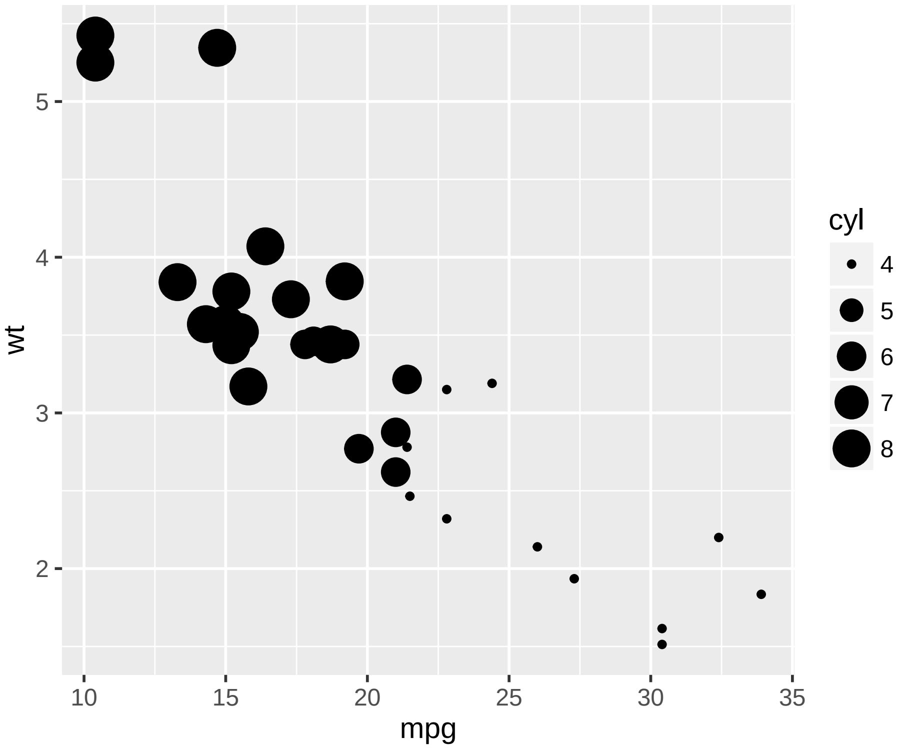
```r
qplot(mpg, wt, data = mtcars)
```



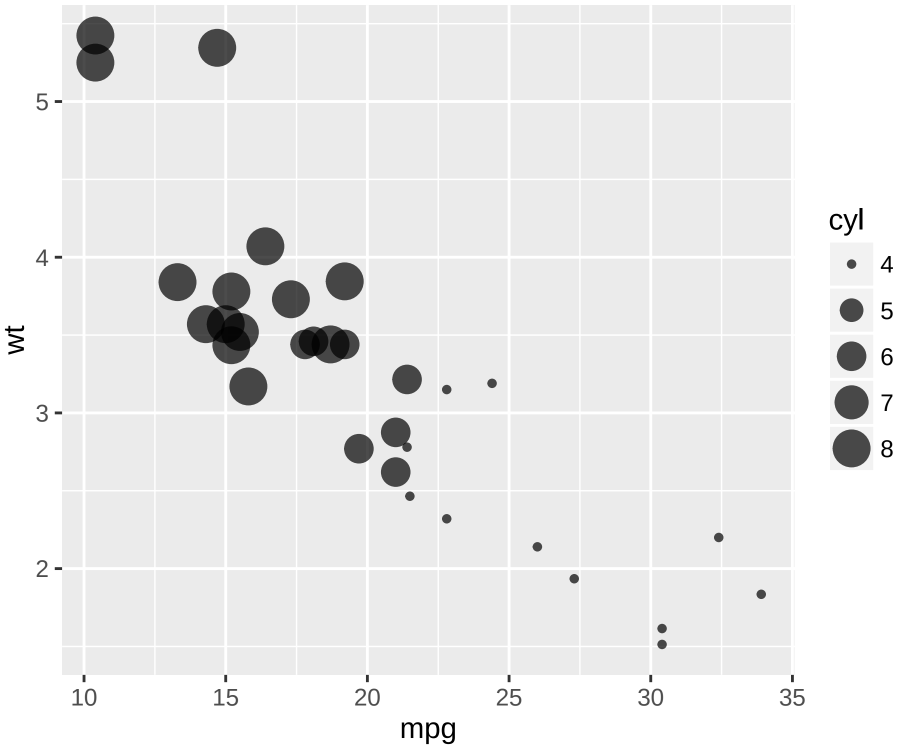
```
qplot(mpg, wt, data = mtcars, colour = cyl)
```



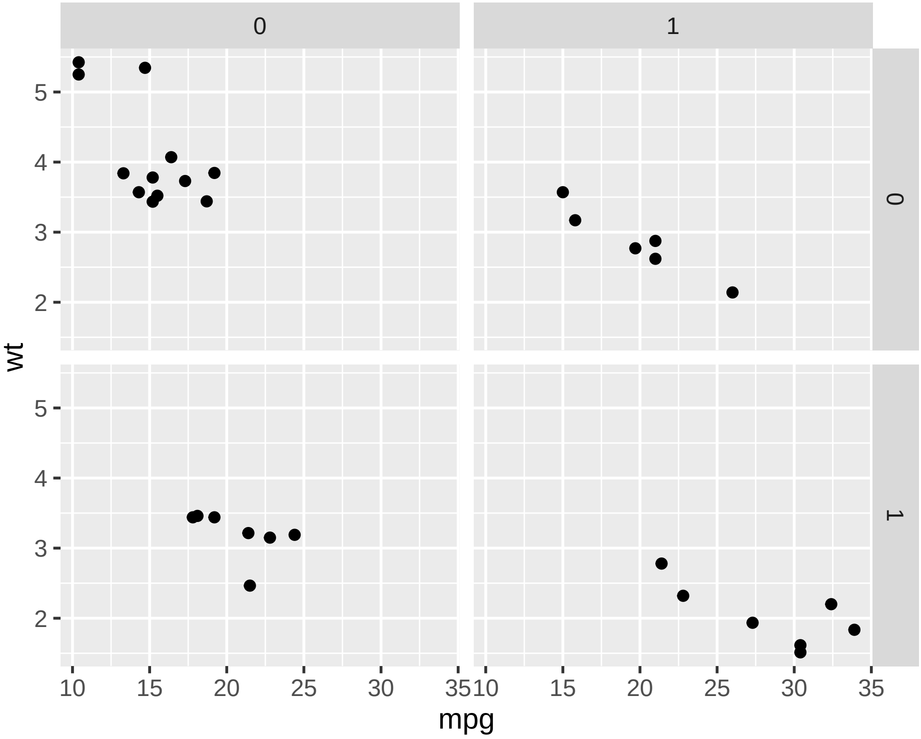
```
qplot(mpg, wt, data = mtcars, size = cyl)
```



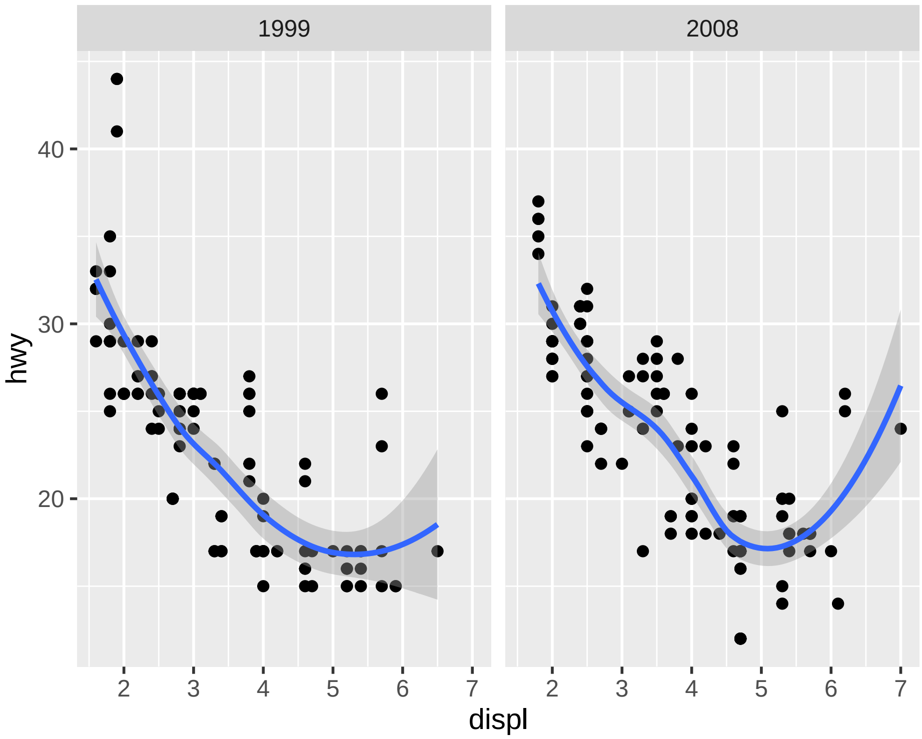
```r
qplot(mpg, wt, data = mtcars, size = cyl, alpha = I(0.7))
```



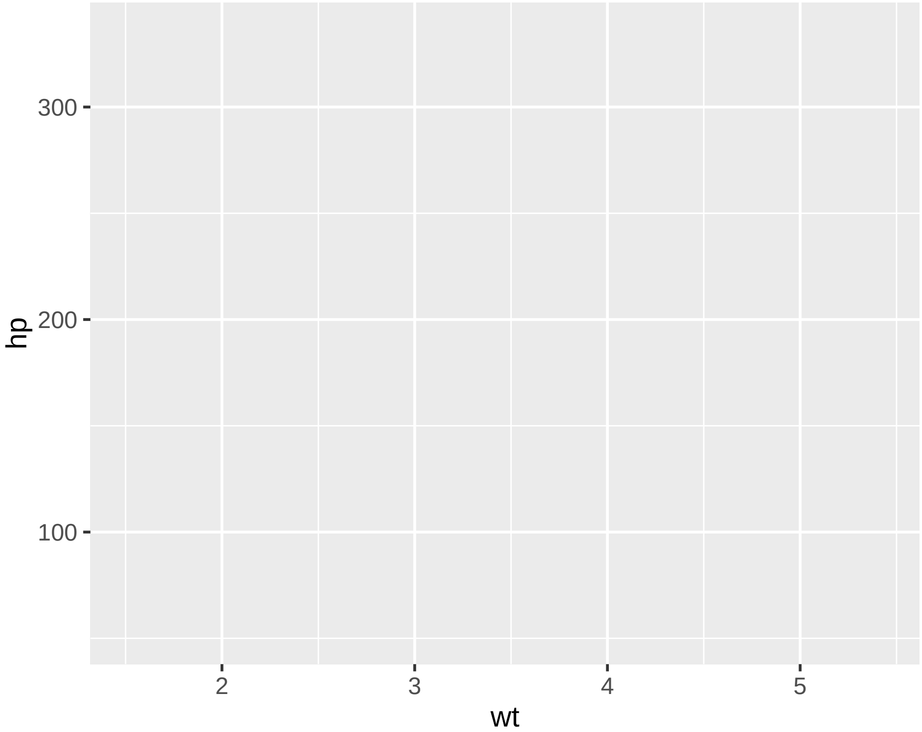
```
qplot(mpg, wt, data = mtcars, facets = vs ~ am)
```



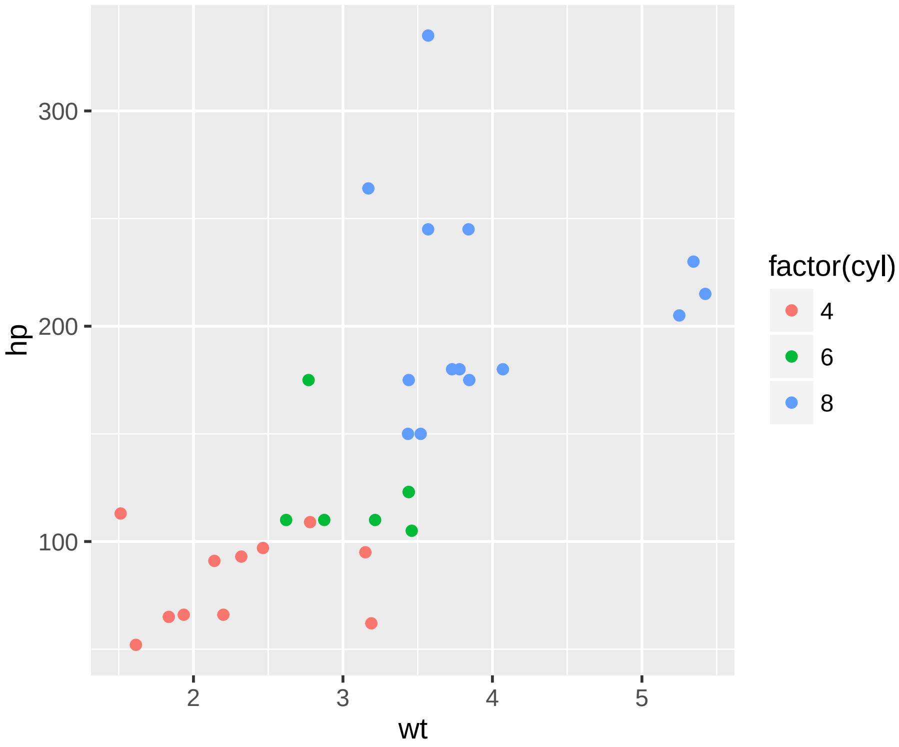
```
qplot(displ, hwy, data=mpg, facets = . ~ year) + geom_smooth()
```



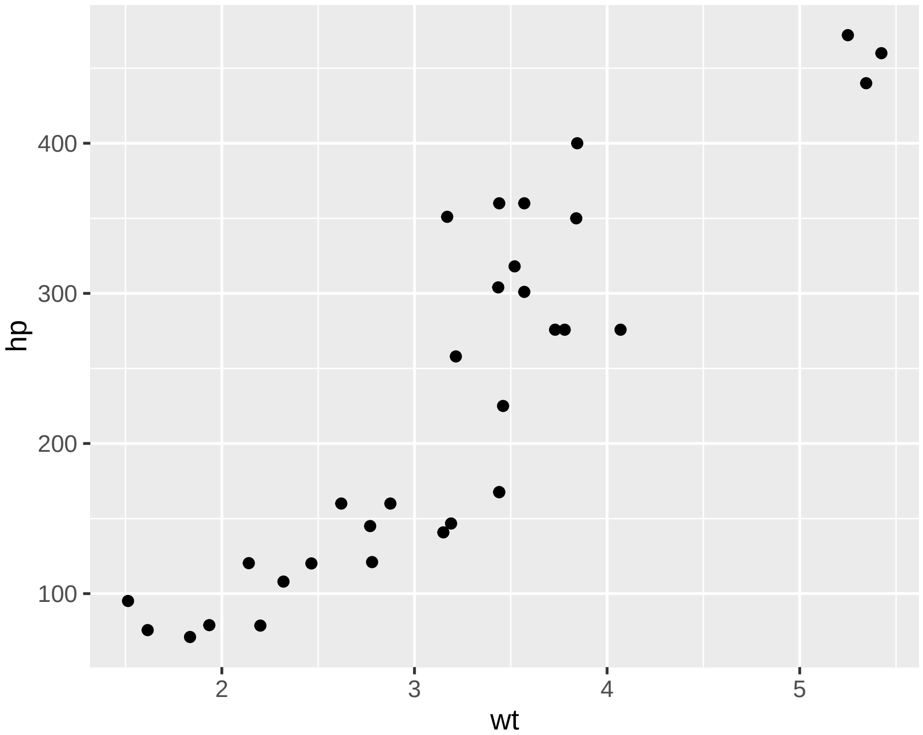
```
p <- ggplot(mtcars)
p <- p + aes(wt, hp)
p
```



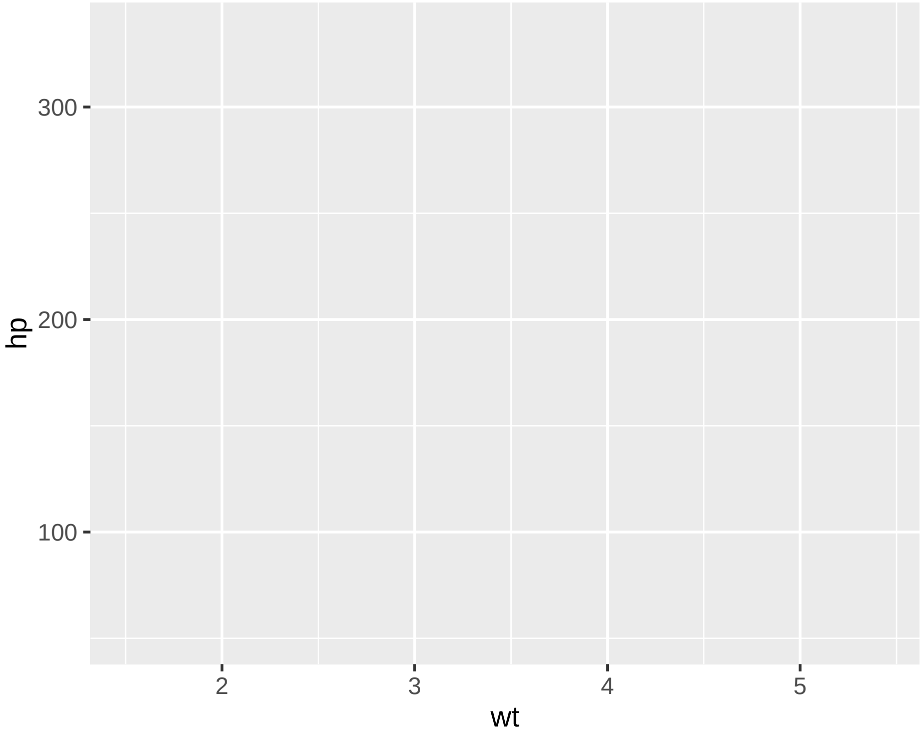
```
p + geom_point(aes(colour = factor(cyl)))
```



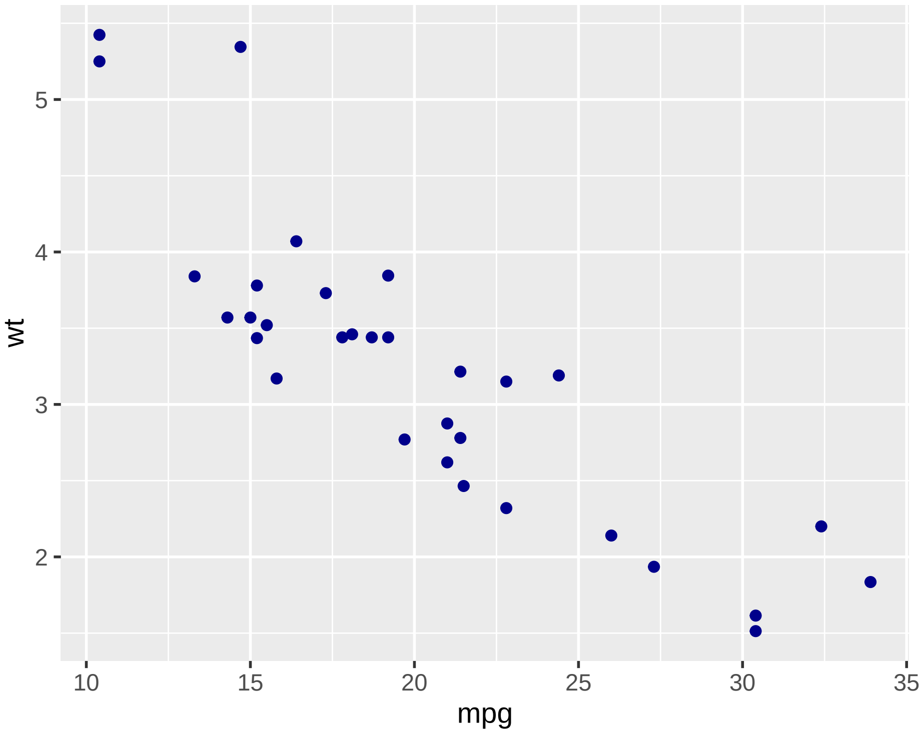
```
p + geom_point(aes(y = disp))
```



```
p
```



```
p <- ggplot(mtcars, aes(mpg, wt))
p + geom_point(colour = "darkblue")
```

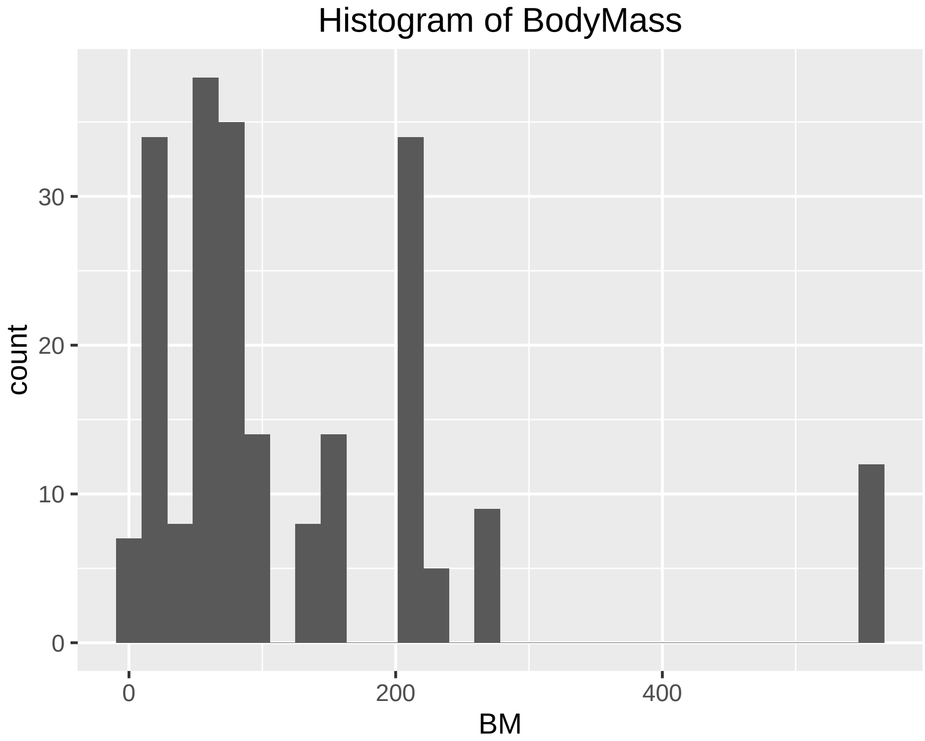


```
filepath <- "http://idaejin.github.io/bcam-courses/azti-2016/introR/data/ggplot2_data.txt"

myData<-read.table(file=url(filepath),header=TRUE,sep="\t")

str(myData)
```

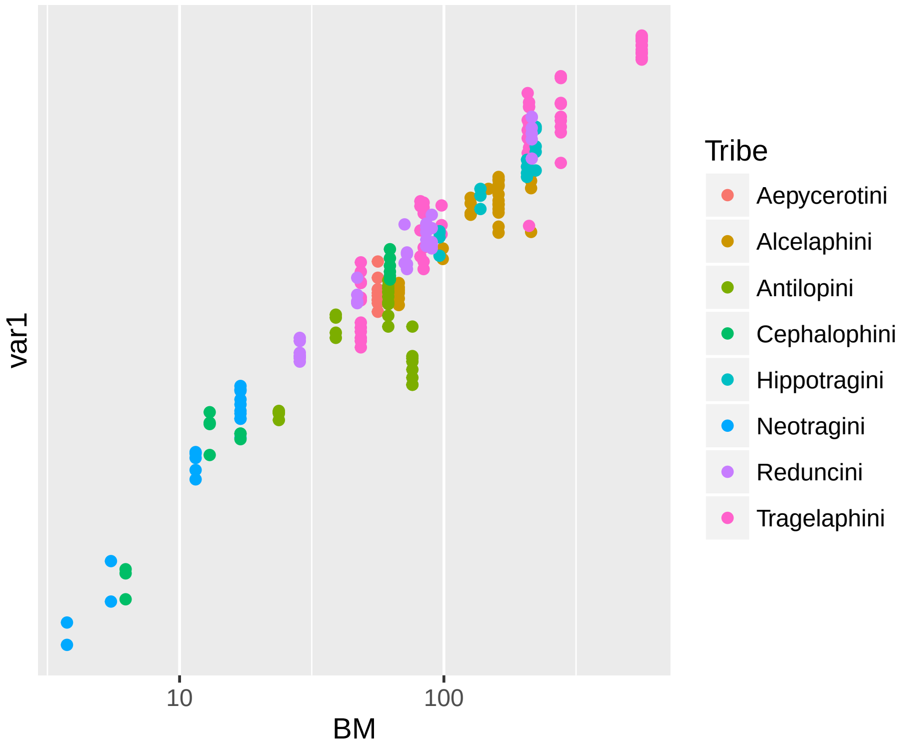
```
## 'data.frame':    218 obs. of  4 variables:
##  $ Tribe: Factor w/ 8 levels "Aepycerotini",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Hab  : Factor w/ 4 levels "F","H","L","O": 3 3 3 3 3 3 3 3 3 3 ...
##  $ BM   : num  56.2 56.2 56.2 56.2 56.2 ...
##  $ var1 : num  36.5 40.9 37 36.2 36.6 37.7 37.3 39 37.7 35.3 ...
```

```
qplot(data=myData,x=BM,main="Histogram of BodyMass")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
qplot(data=myData,x=BM,y=var1,log="xy",color=Tribe)
```



8.3 Maps

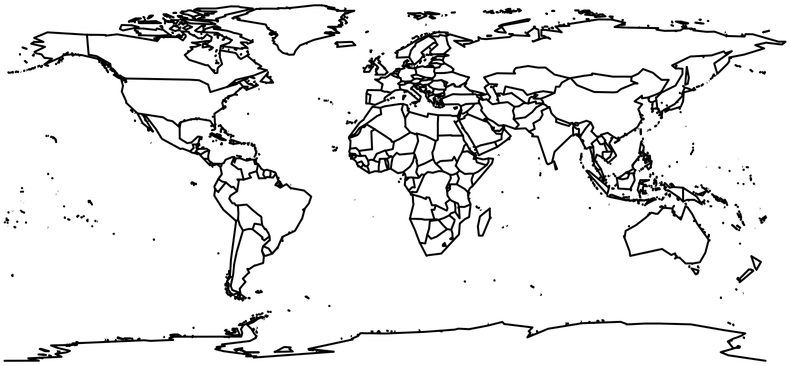
Packages for Spatial Regression / Geostatistics / Spatial Point Pattern methods

- `sp`, `maptools`, `spatstat`
- `maps`

```
library(maps)
```

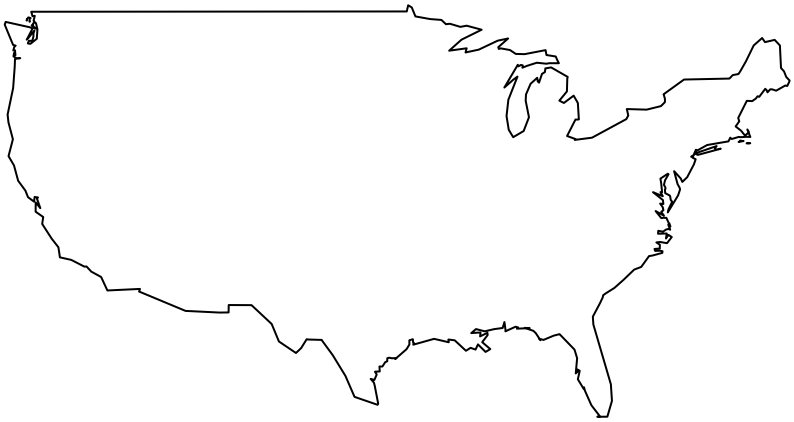
Basic syntax

```
map(database = "world",regions=".")
```

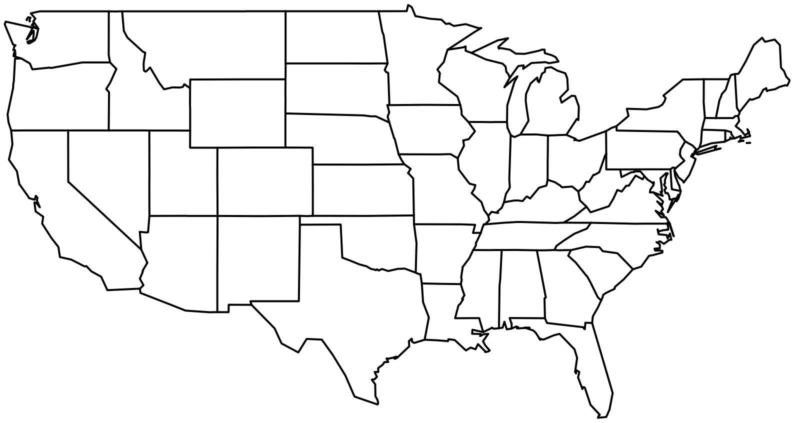


Databases are available for US, France, Italy and New Zealand. For other countries, you need to import a database with the corresponding map.

```
map(database = "usa")
```

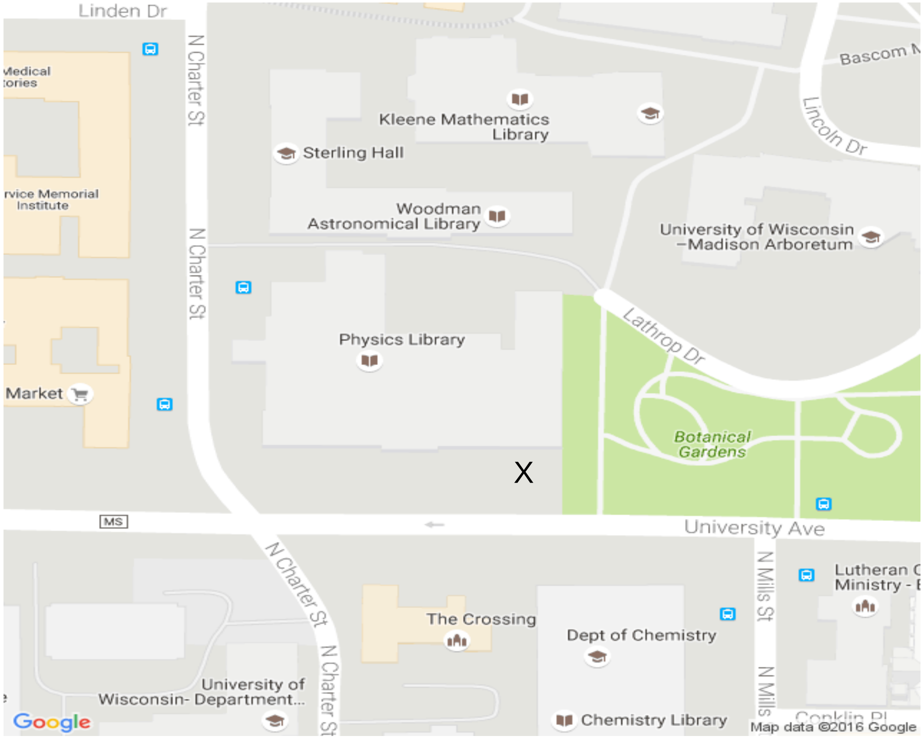


```
map("state")
```



With the package `RgoogleMaps`, you can draw a background from Google Maps!

```r
require(RgoogleMaps)
lat <- 43.073888
lon <- -89.405236
center <- c(lat, lon)
zoom <- 18
MyMap <- GetMap(center=center, zoom=zoom)
PlotOnStaticMap(MyMap)
text(lat,lon, "X") # I missed the class!
```



`ggmap` offers plotting capabilities like `ggplot2`

```r
require(ggmap)
geocode("Union South, Madison, WI")
```

```
##         lon      lat
## 1 -89.29914 42.77606
```

9 Case studies

9.1 The Forbes 2000 Ranking of the World's Biggest Companies (Year 2004)

The data handling and manipulation techniques explained will be illustrated by means of a data set of 2000 world leading companies, the Forbes 2000 list for the year 2004 collected by Forbes Magazine. This list is originally available from `www.forbes.com`

Here we show a subset of the data set:

```
library("HSAUR2")
data("Forbes2000")
```

rank	name	country	category	sales	profits	assets	marketv
1	Citigroup	United States	Banking	94.71	17.85	1264.03	25
2	General Electric	United States	Conglomerates	134.19	15.59	626.93	32
3	American Intl Group	United States	Insurance	76.66	6.46	647.66	19
4	ExxonMobil	United States	Oil & gas operations	222.88	20.96	166.99	27
5	BP	United Kingdom	Oil & gas operations	232.57	10.27	177.57	17
6	Bank of America	United States	Banking	49.01	10.81	736.45	11

The data consists of 2000 observations on the following 8 variables.

- `rank`: the ranking of the company.
- `name`: the name of the company.
- `country`: a factor giving the country the company is situated in.
- `category`: a factor describing the products the company produces.
- `sales`: the amount of sales of the company in billion USD.
- `profits`: the profit of the company in billion USD.
- `assets`: the assets of the company in billion USD.
- `marketvalue`: the market value of the company in billion USD.

Types of variables

`R` command

```
str(Forbes2000)
```

```
## 'data.frame':    2000 obs. of  8 variables:
##  $ rank       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ name       : chr  "Citigroup" "General Electric" "American Intl Group" "ExxonMobil" ..
##  $ country    : Factor w/ 61 levels "Africa","Australia",..: 60 60 60 60 56 60 56 28 60 6
##  $ category   : Factor w/ 27 levels "Aerospace & defense",..: 2 6 16 19 19 2 2 8 9 20 ...
##  $ sales      : num  94.7 134.2 76.7 222.9 232.6 ...
##  $ profits    : num  17.85 15.59 6.46 20.96 10.27 ...
##  $ assets     : num  1264 627 648 167 178 ...
##  $ marketvalue: num  255 329 195 277 174 ...
```

Factor levels

Nominal measurements are represented by factor variables in R, such as the country of the company or the category of the business segment.

A factor in R is divided into levels

How many countries are on the top 2000 ranking?

R command

```
nlevels(Forbes2000[,"country"])
```

```
## [1] 61
```

Which countries?

R command

```
levels(Forbes2000[,"country"])
```

```
##  [1] "Africa"                         "Australia"
##  [3] "Australia/ United Kingdom"      "Austria"
##  [5] "Bahamas"                        "Belgium"
##  [7] "Bermuda"                        "Brazil"
##  [9] "Canada"                         "Cayman Islands"
## [11] "Chile"                          "China"
## [13] "Czech Republic"                 "Denmark"
## [15] "Finland"                        "France"
## [17] "France/ United Kingdom"         "Germany"
## [19] "Greece"                         "Hong Kong/China"
## [21] "Hungary"                        "India"
## [23] "Indonesia"                      "Ireland"
## [25] "Islands"                      "Israel"
## [27] "Italy"                        "Japan"
## [29] "Jordan"                       "Kong/China"
## [31] "Korea"                        "Liberia"
## [33] "Luxembourg"                   "Malaysia"
## [35] "Mexico"                       "Netherlands"
## [37] "Netherlands/ United Kingdom"  "New Zealand"
## [39] "Norway"                       "Pakistan"
## [41] "Panama/ United Kingdom"       "Peru"
## [43] "Philippines"                  "Poland"
## [45] "Portugal"                     "Russia"
## [47] "Singapore"                    "South Africa"
## [49] "South Korea"                  "Spain"
## [51] "Sweden"                       "Switzerland"
## [53] "Taiwan"                       "Thailand"
## [55] "Turkey"                       "United Kingdom"
## [57] "United Kingdom/ Australia"    "United Kingdom/ Netherlands"
## [59] "United Kingdom/ South Africa" "United States"
## [61] "Venezuela"
```

And in the top 20?

R commands

```
top20 <- droplevels(subset(Forbes2000,rank<=20))
levels(top20[,"country"])
```

```
## [1] "France"                      "Japan"
## [3] "Netherlands"                 "Netherlands/ United Kingdom"
## [5] "Switzerland"                 "United Kingdom"
## [7] "United States"
```

As a simple summary statistic, the frequencies of the levels of such a factor variable can be found from

```
table(top20[,"country"])
```

```
##
##                       France                       Japan
##                            2                           1
##                  Netherlands Netherlands/ United Kingdom
##                            1                           1
##                  Switzerland              United Kingdom
##                            1                           3
##                United States
##                           11
```

Which type of companies?

```
levels(Forbes2000[,"category"])
```

```
##  [1] "Aerospace & defense"              "Banking"
##  [3] "Business services & supplies"     "Capital goods"
##  [5] "Chemicals"                        "Conglomerates"
##  [7] "Construction"                     "Consumer durables"
##  [9] "Diversified financials"           "Drugs & biotechnology"
## [11] "Food drink & tobacco"             "Food markets"
## [13] "Health care equipment & services" "Hotels restaurants & leisure"
## [15] "Household & personal products"    "Insurance"
## [17] "Materials"                        "Media"
## [19] "Oil & gas operations"             "Retailing"
## [21] "Semiconductors"                   "Software & services"
## [23] "Technology hardware & equipment"  "Telecommunications services"
## [25] "Trading companies"                "Transportation"
## [27] "Utilities"
```

How many of each category?

```
table(Forbes2000[,"category"])
```

```
##
##              Aerospace & defense                          Banking
##                               19                              313
##     Business services & supplies                    Capital goods
##                               70                               53
##                        Chemicals                    Conglomerates
##                               50                               31
##                     Construction                Consumer durables
##                               79                               74
##           Diversified financials            Drugs & biotechnology
##                              158                               45
##             Food drink & tobacco                     Food markets
##                               83                               33
## Health care equipment & services     Hotels restaurants & leisure
##                               65                               37
##    Household & personal products                        Insurance
##                               44                              112
##                        Materials                            Media
##                               97                               61
##             Oil & gas operations                        Retailing
##                               90                               88
##                   Semiconductors              Software & services
##                               26                               31
##  Technology hardware & equipment      Telecommunications services
##                               59                               67
##                Trading companies                   Transportation
##                               25                               80
##                        Utilities
##                              110
```

A simple summary statistics such as the mean, median, quantiles and range can be found from continuous variables such as `sales`

`R` command

```
summary(Forbes2000[,"sales"])
```

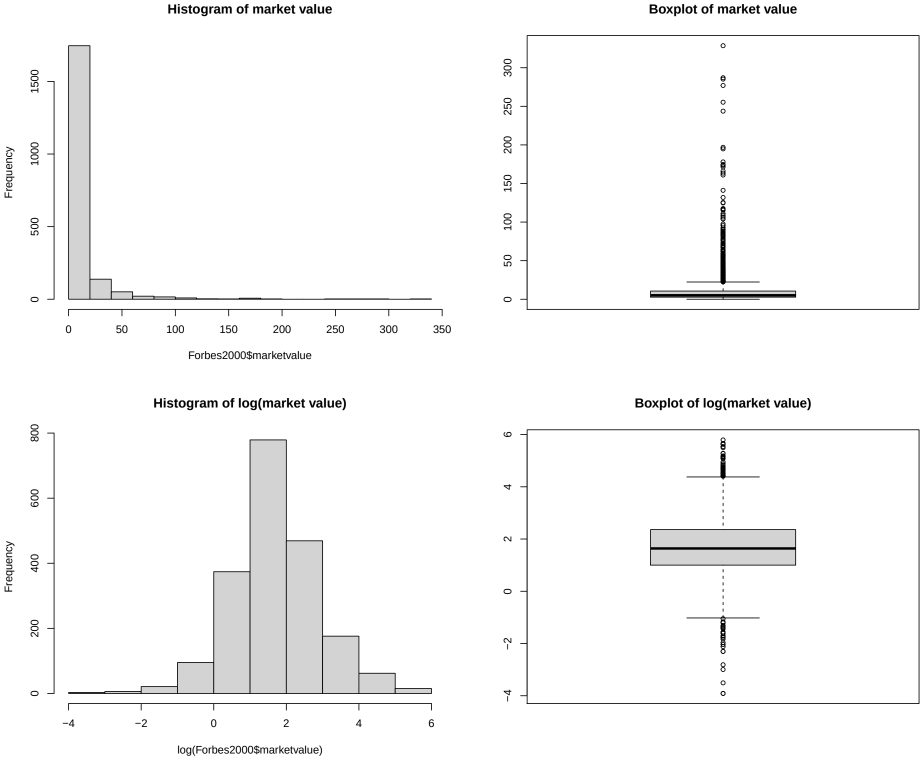
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.010   2.018   4.365   9.697   9.548 256.300
```

Simple Graphics

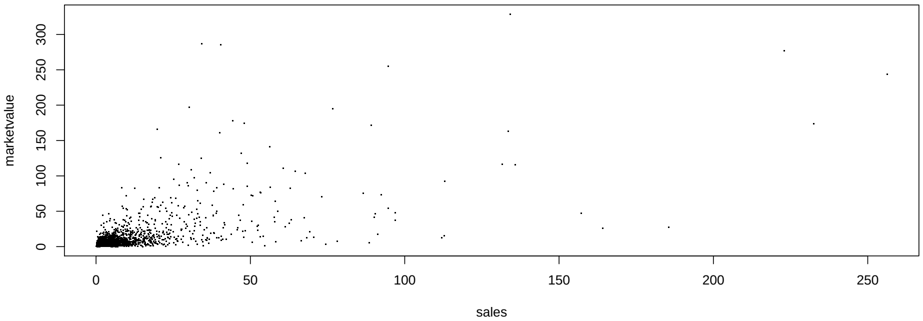
Chambers et al. (1983), “there is no statistical tool that is as powerful as a well chosen graph”

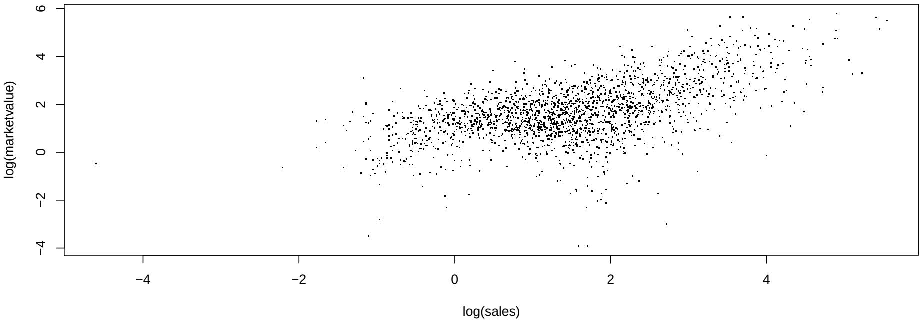
Histograms and boxplots

```
layout(matrix(1:4, nrow = 2,ncol=2))
hist(Forbes2000$marketvalue, col="lightgrey",main="Histogram of market value")
hist(log(Forbes2000$marketvalue),col="lightgrey",main="Histogram of log(market value)")
boxplot(Forbes2000$marketvalue, col="lightgrey",main="Boxplot of market value")
boxplot(log(Forbes2000$marketvalue),col="lightgrey",main="Boxplot of log(market value)")
```



Scatterplots to visualize the relationship betwen variables

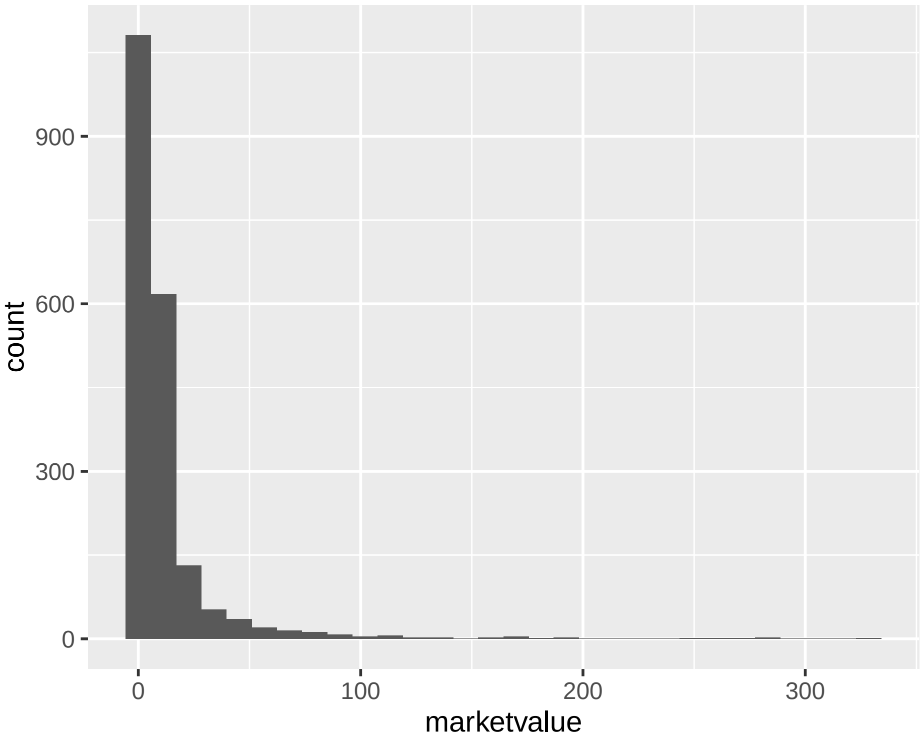




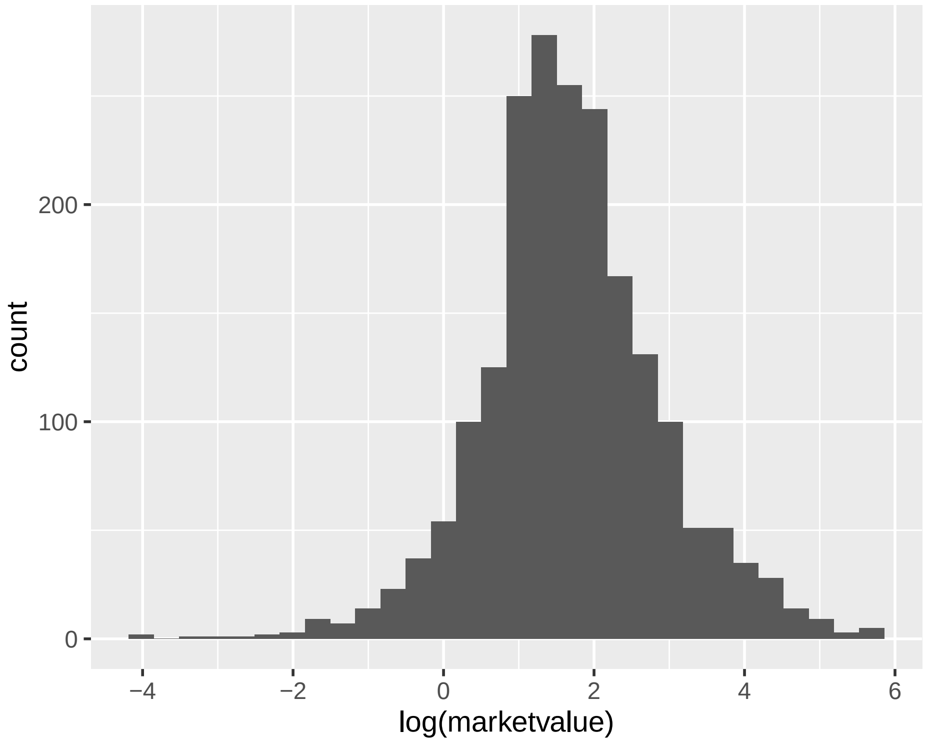
Cool Graphics

Using the `ggplot2` library

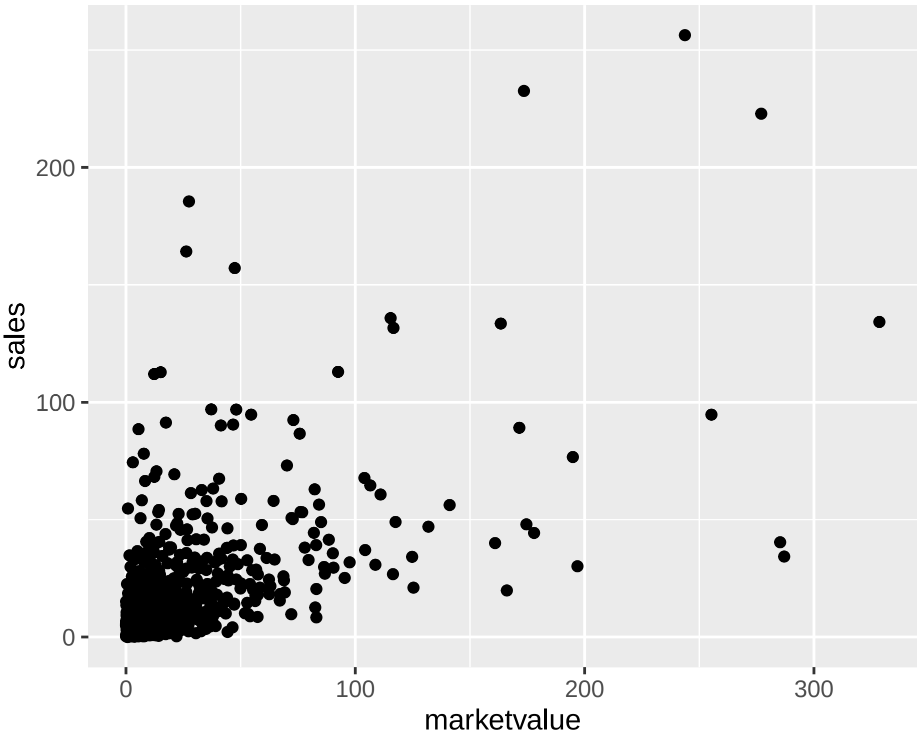
```
library(ggplot2)
#?qplot
qplot(marketvalue,data = Forbes2000)
```



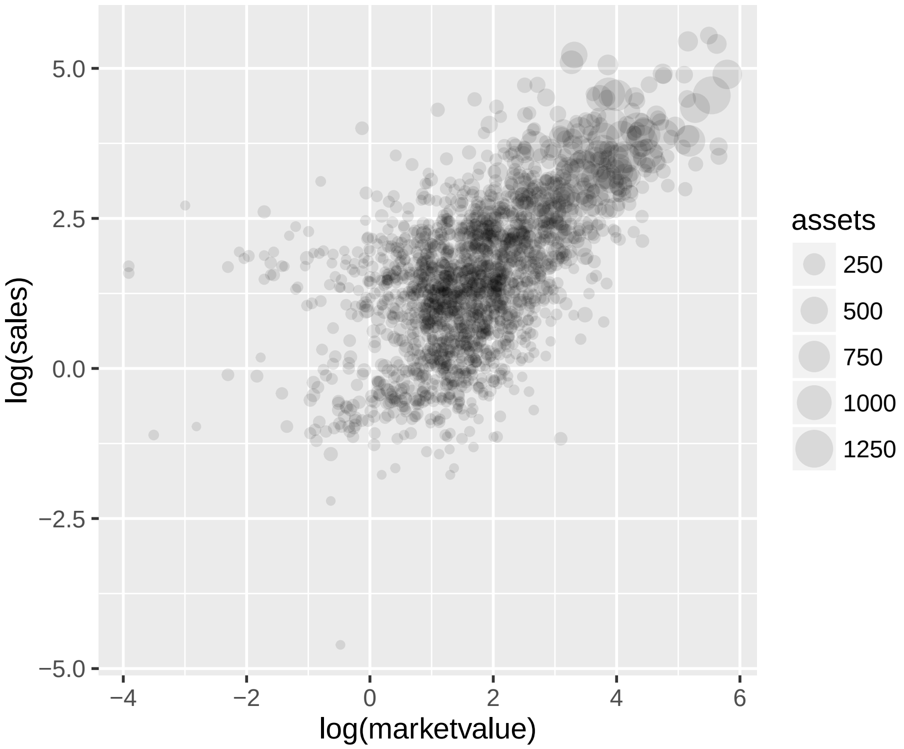
```
qplot(log(marketvalue),  data = Forbes2000)
```



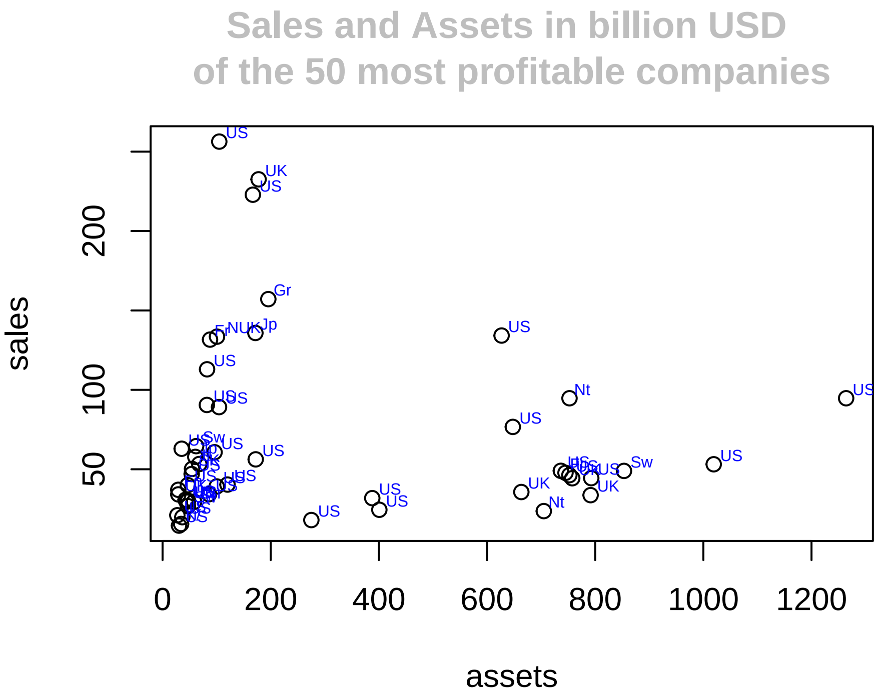
```
qplot(marketvalue,sales, data=Forbes2000)
```



```r
qplot(log(marketvalue),log(sales),size=assets,alpha = I(0.1),data=Forbes2000)
```



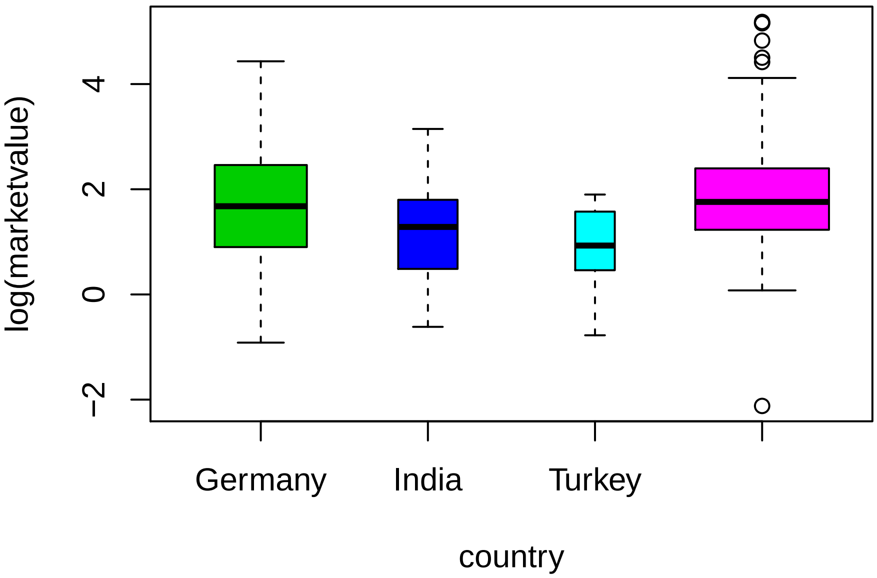
```
library(calibrate)
profits_all = na.omit(Forbes2000$profits)  # all_profts without No data
order_profits = order(profits_all)      # index of the profitable companies in decreasing ord
top_50 = rev(order_profits)[1:50]       # top 50 profitable companies

sales = Forbes2000$sales[top_50]        # sales of the 50 top profitable companies
assets = Forbes2000$assets[top_50]      # assets of the 50 top profitable companies
countries = Forbes2000$country[top_50] # countries where the 50 top profitable companies are

plot(assets,sales,pch =1)
textxy(assets,sales, abbreviate(countries,2),col = "blue",cex=0.5)  # used to put the countr
title(main = "Sales and Assets in billion USD \n of the 50 most profitable companies ", col
```



Graphics by factor

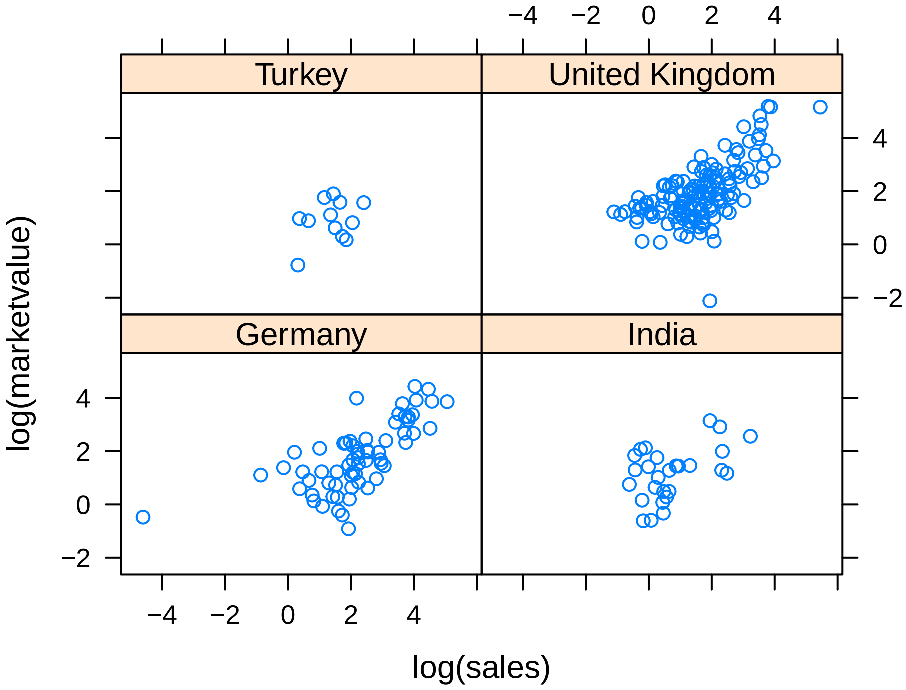
Boxplots of the logarithms of the market value for four selected countries, the width of the boxes is proportional to the square roots of the number of companies.

```
tmp <- subset(Forbes2000,
        country %in% c("United Kingdom", "Germany",
                       "India", "Turkey"))
tmp$country <- tmp$country[,drop = TRUE]
plot(log(marketvalue) ~ country, data = tmp, col = 3:6,
       ylab = "log(marketvalue)", varwidth = TRUE)
```



Scatterplots by country

```r
library(lattice)
xyplot(log(marketvalue)~log(sales)|country,data=tmp)
```



9.2 Malignant Melanoma in the USA

Fisher and Belle (1993) report mortality rates due to malignant melanoma of the skin for white males during the period 1950-1969, for each state on the US mainland.

```r
data("USmelanoma",package="HSAUR2")
```

A data consists of 48 observations on the following 5 variables.

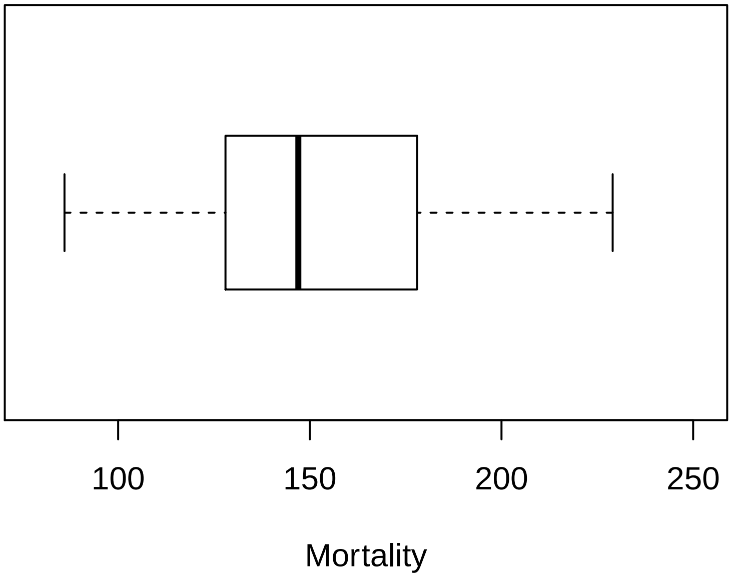
- `mortality`: number of white males died due to malignant melanoma 1950-1969 per one million inhabitants.
- `latitude`: latitude of the geographic centre of the state.
- `longitude`: longitude of the geographic centre of each state.
- `ocean`: a binary variable indicating contiguity to an ocean at levels `no` or `yes`.

Plotting mortality rates

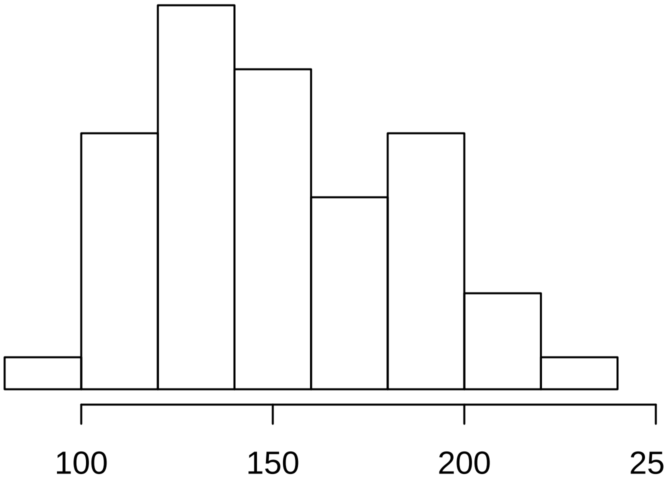
```r
xr <- range(USmelanoma$mortality) * c(0.9, 1.1)
```

Let us plot mortality rates in

```r
#layout(matrix(1:2, nrow = 2))
boxplot(USmelanoma$mortality, ylim = xr, horizontal = TRUE,xlab = "Mortality")
```

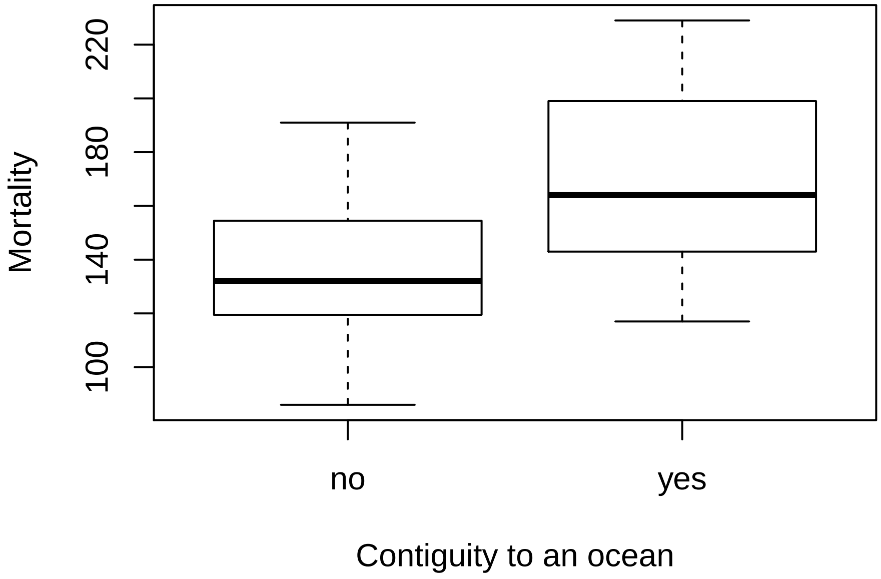


```
hist(USmelanoma$mortality, xlim = xr, xlab = "", main = "",axes = FALSE, ylab = "")
axis(1)
```



Malignant melanoma mortality rates by contiguity to an ocean

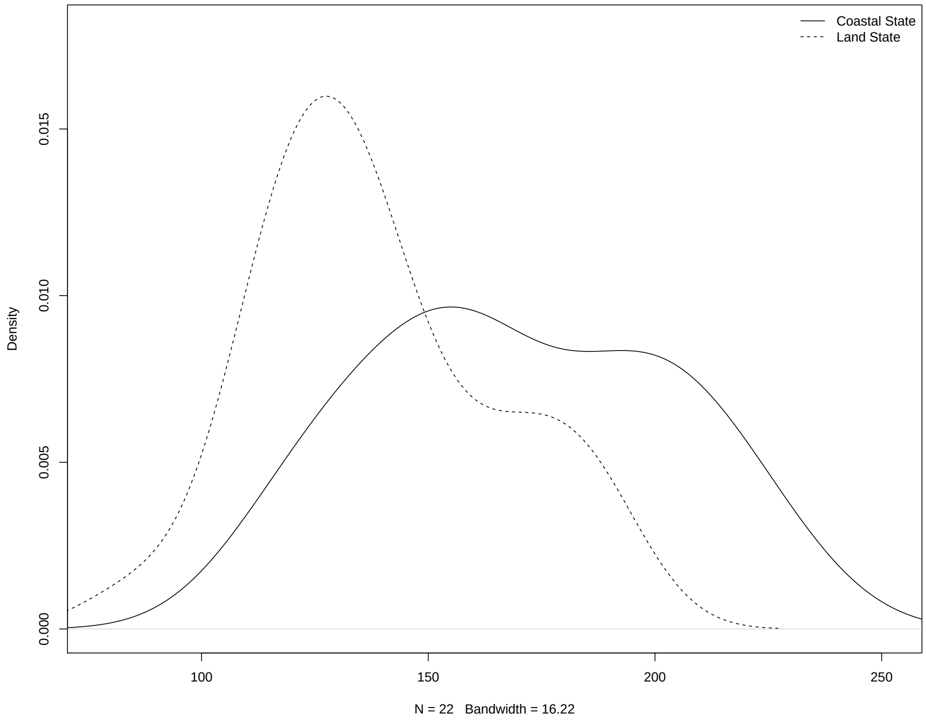
```
plot(mortality ~ ocean, data = USmelanoma, xlab = "Contiguity to an ocean", ylab = "Mortalit
```



Histograms can often be misleading for displaying distributions because of their dependence on the number of classes chosen. An alternative is to formally estimate the density function of a variable and then plot the resulting estimate.

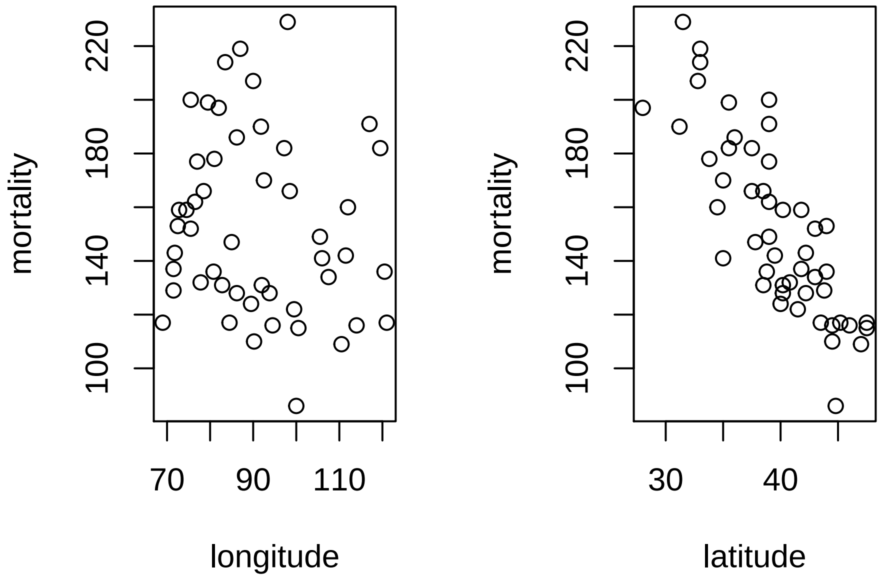
The estimated densities of malignant melanoma mortality rates by contiguity to an ocean looks like this:

```r
dyes <- with(USmelanoma, density(mortality[ocean == "yes"]))
dno <- with(USmelanoma, density(mortality[ocean == "no"]))
plot(dyes, lty = 1, xlim = xr, main = "", ylim = c(0, 0.018))
lines(dno, lty = 2)
legend("topright", lty = 1:2, legend = c("Coastal State","Land State"), bty = "n")
```



Now we might move on to look at how mortality rates are related to the geographic location of a state as represented by the latitude and longitude of the centre of the state.

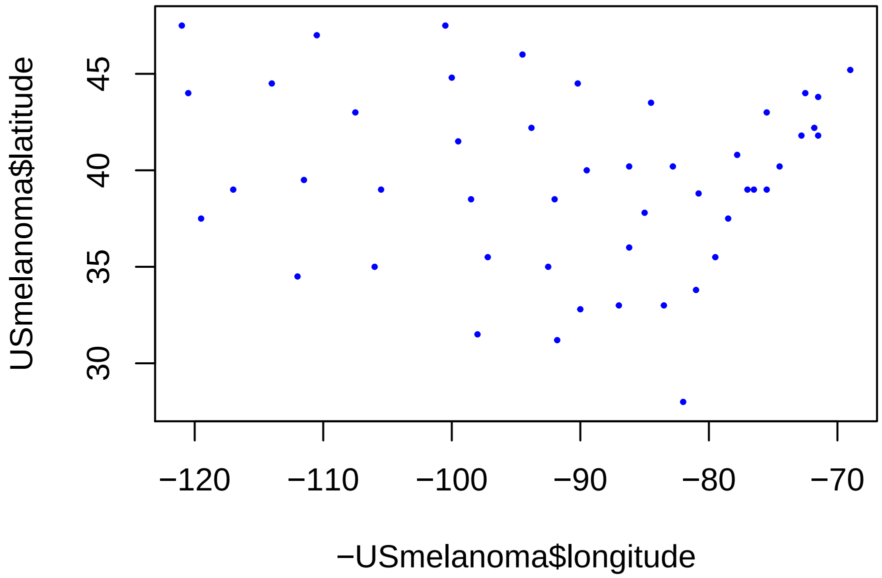
```
layout(matrix(1:2, ncol = 2))
plot(mortality ~ -longitude, data = USmelanoma)
plot(mortality ~ latitude, data = USmelanoma)
```



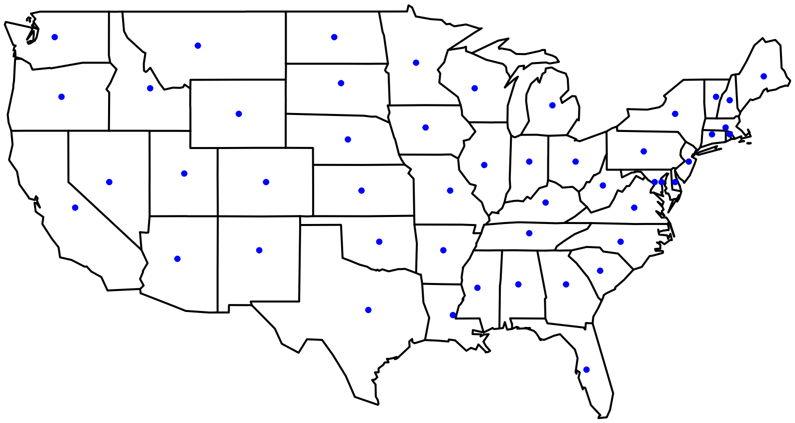
9.3 Mapping mortality rates

The data contains the longitude and latitude of the centroids

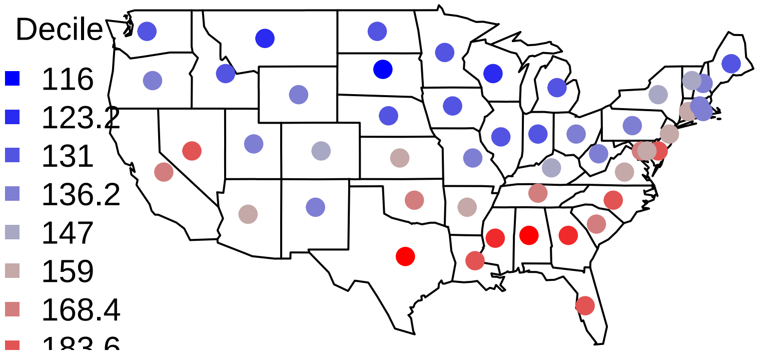
```
plot(-USmelanoma$longitude,USmelanoma$latitude,asp=1.5,cex=.3,pch=19,col="blue")
```



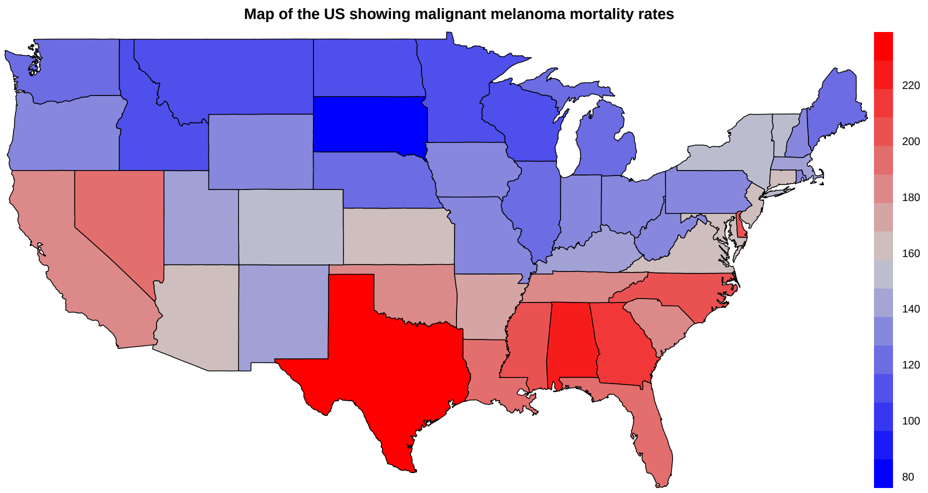
```r
library("sp")
library("maps")
library("maptools")
library("RColorBrewer")
map("state")
points(-USmelanoma$longitude,USmelanoma$latitude,asp=1.5,cex=.3,pch=19,col="blue")
```



```r
#Create a function to generate a continuous color palette
rbPal <- colorRampPalette(c('blue','grey','red'))
#This adds a column of color values
# based on the y values
USmelanoma$Col <- (rbPal(10)[as.numeric(cut(USmelanoma$mortality,breaks = 10))])
map("state",xlim=c(-135,-65))
points(-USmelanoma$longitude,USmelanoma$latitude,col=USmelanoma$Col,asp=1.5,pch=19,cex=1.2)
legend("topleft",title="Decile",legend=quantile(USmelanoma$mortality,seq(0.1,1,l=10)),col =r
```



```
states <- map("state", plot = FALSE, fill = TRUE)
IDs <- sapply(strsplit(states$names, ":"), function(x) x[1])
rownames(USmelanoma) <- tolower(rownames(USmelanoma))

us1 <- map2SpatialPolygons(states, IDs=IDs,proj4string = CRS("+proj=longlat +datum=WGS84"))
us2 <- SpatialPolygonsDataFrame(us1, USmelanoma)

col <- colorRampPalette(c('blue', 'gray80','red'))

spplot(us2, "mortality", col.regions = col(200),par.settings = list(axis.line = list(col =
```


Map of the US showing malignant melanoma mortality rates
220
200
180
160
140
120
100
80