Introduction to Statistical Modelling in R

BCAM - Basque Center for Applied Mathematics, Applied Statistics

Dae-Jin Lee < dlee@bcamath.org >

CHAPTER 1. Introduction to R language

Contents

1 Introduction to the R language **2**

1.1 Installing R . . . 2
1.2 Start with R . . . 2
1.3 Install and load an R library . . . 3
1.4 Reading data . . . 4
1.5 Data Import . . . 4
1.6 Exporting data . . . 6
1.7 Data vectors . . . 6
1.8 Basic statistics . . . 7
1.9 Creating your own functions in R . . . 9
1.10 Character vectors and factor variables . . . 10
1.11 Data frames . . . 10
1.12 Working with data frames . . . 11
1.13 Logical vectors . . . 12
1.14 Working with vectors . . . 13
1.15 Matrices and arrays . . . 14
1.16 Factors . . . 16
1.17 Indexing vector with logicals . . . 17

1.18 Missing values 17
1.19 Working with data frames 18
1.20 The Forbes 2000 Ranking of the World's Biggest Companies (Year 2004) 20
1.21 Questions 33

1 Introduction to the R language

1.1 Installing R

- Latest version of R. Download it here

1.1.1 RStudio environment

- Rstudio is a user-friendly interface. Download it here **Highly recommended!!!**

1.2 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()
```

1.3 Install and load an `R` library

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

or several packages

```
install.packages(c("DAAG","HSAUR2","Hmisc","psych","foreign","xlsx"))
```

In Rstudio (go to `package` and click `Install`)

Once installed the package, load it

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

1.4 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"
```

1.5 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)

```
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
```

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

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

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

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

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

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

Create a folder, name it `data` and download `cars` data (cardata.zip)

```
mydata1 = read.table("data/cardata.txt")
mydata2 = read.csv("data/cardata.csv")
```

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

```
library(gdata)
mydata3 = read.xls("data/cardata.xls", sheet = 1, header = TRUE)

library(xlsx)
mydata4 = read.xlsx("data/cardata.xlsx", sheetIndex = 1, header = TRUE,colClasses=NA)
```

- Minitab, SPSS, SAS or Stata

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

- Or

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

1.6 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

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

- Excel spreadsheet

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

1.7 Data vectors

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

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

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

```
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
```

1.8 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%.

```
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.

```
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 indicate a positive linear relationship between the variables, and a negative covariance would indicate the opposite.

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

```
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 standard deviations. It is a normalised 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.

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

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

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

1.9 Creating your own functions in R

One of the great strengths of R is the user's ability to add functions. In fact, many of the functions in R are actually functions of functions. The structure of a function is given below.

```
myfunction <- function(arg1, arg2, ... ){
  statements
return(object)
}
```

```
f <- function(x){
  x^2
    }
f
```

```
## function(x){
##   x^2
##     }
```

Example:

```
# Given a number
f(2)
```

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

```
# Given a vector
x <- c(1,2,-4,7)
f(x)
```

```
## [1]  1  4 16 49
```

Let us create a function that returns a set of summary statistics given a numeric vector:

```
mysummary <- function(x){
  mean <- sum(x)/length(x)
   var <- var(x)
    sd <- sd(x)
 range <- range(x)
 result <- list(mean=mean,var=var,sd=sd,range=range)
 return(result)
}
```

Then

```
set.seed(1234)
x <- rnorm(10)
stats <- mysummary(x)
stats
```

```
## $mean
## [1] -0.3831574
##
## $var
## [1] 0.9915928
##
## $sd
## [1] 0.9957875
##
## $range
## [1] -2.345698  1.084441
```

1.10 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
```

1.11 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
```

1.12 Working with data frames

Example: Analyze data by groups

- Obtain the mean weight, height 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)
```

1.13 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)
```

1.14 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))
```

1.15 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
```

1.16 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
```

1.17 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
```

1.18 Missing values

In R, missing values are represented by the symbol NA (not available) . Impossible values (e.g., dividing by zero) are represented by the symbol NaN (not a number).

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

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

Excluding missing values from functions

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

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

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

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

The function `complete.cases()` returns a logical vector indicating which cases are complete.

```
complete.cases(a)
```

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

The function `na.omit()` returns the object with listwise deletion of missing values.

```
na.omit(a)
```

```
## [1] 1 2 3 4
## attr(,"na.action")
## [1] 5
## attr(,"class")
## [1] "omit"
```

`NA` in data frames:

```
require(graphics)
?airquality
pairs(airquality, panel = panel.smooth, main = "airquality data")
ok <- complete.cases(airquality)
airquality[ok,]
```

1.19 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
mtcars[c("Datsun 710", "Camaro Z28"),]
```

- Or specific variables

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

There are some packages that include particular functions to summarize data frames, for instance the library `psych` has the function `describe`

```
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
```

1.20 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

rank	name	country	category	sales	profits	assets	marketv
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"])
```

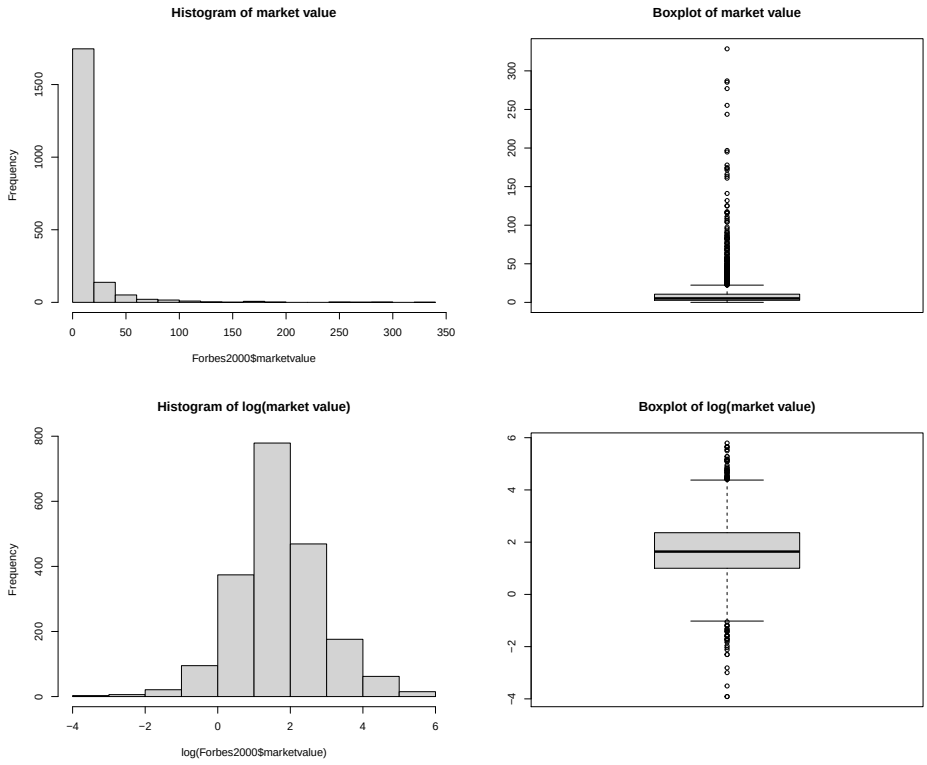
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.010   2.018   4.365   9.697   9.548 256.300
```

Simple Graphics

Chambers et al. (1983), "there is no statistical tool that is as powerful as a well chosen graph"

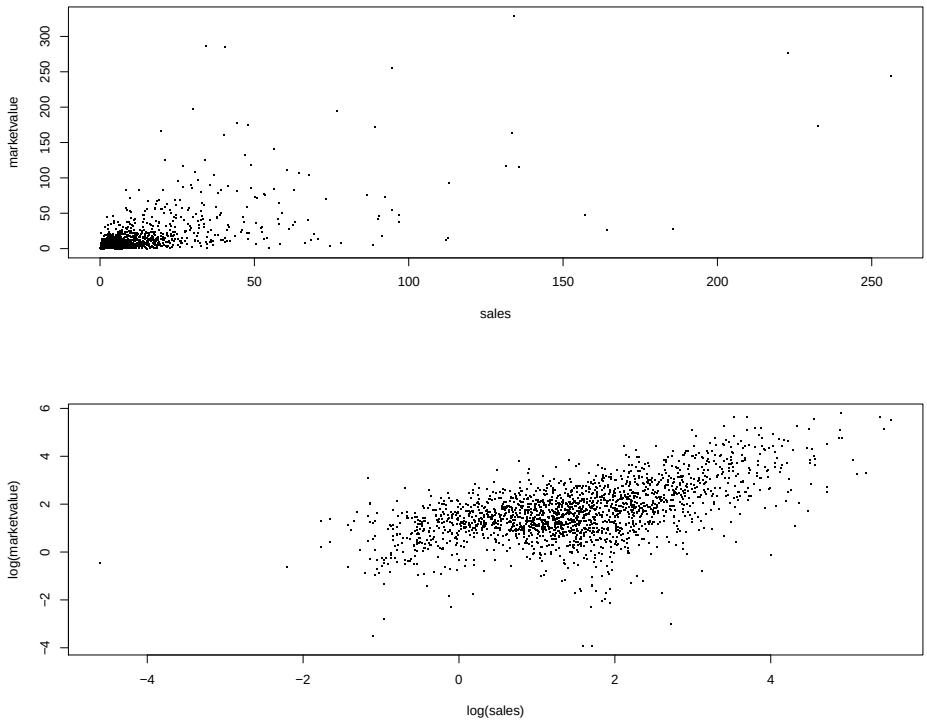
Histograms and boxplots

```
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

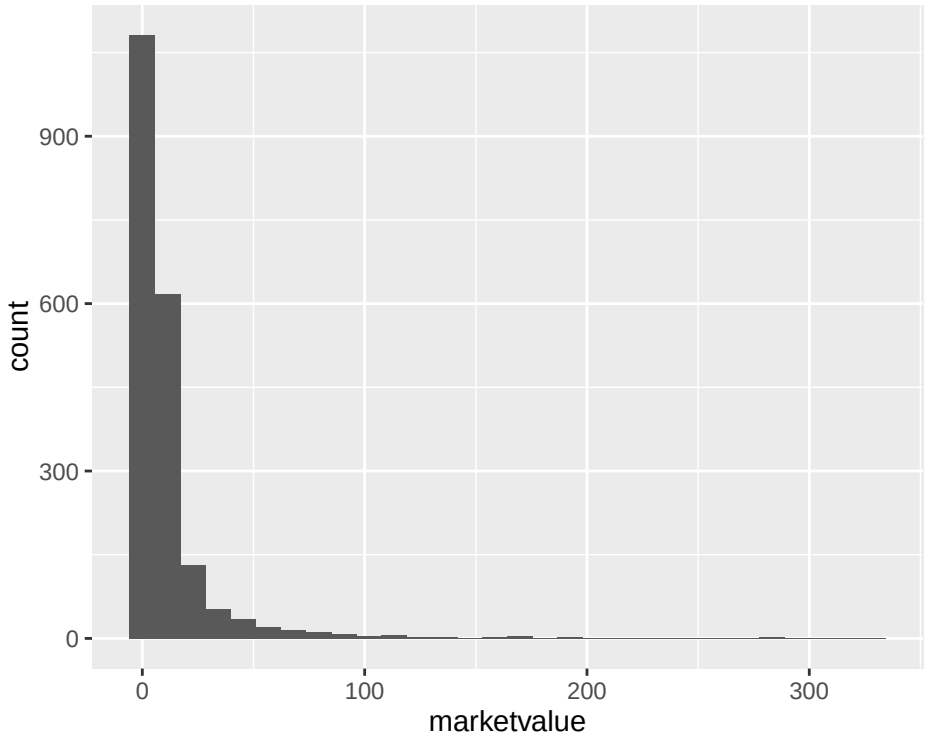
```
layout(matrix(1:2, nrow = 2))
plot(marketvalue ~ sales, data = Forbes2000, pch = ".")
plot(log(marketvalue) ~ log(sales), data = Forbes2000, pch = ".")
```



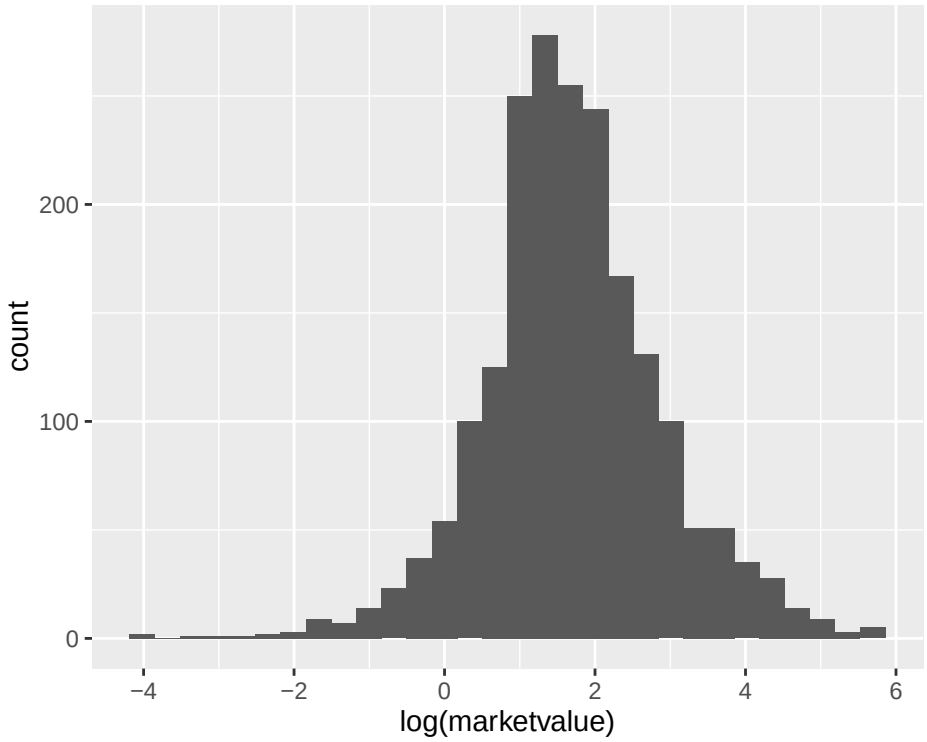
Cool Graphics

Using the `ggplot2` library

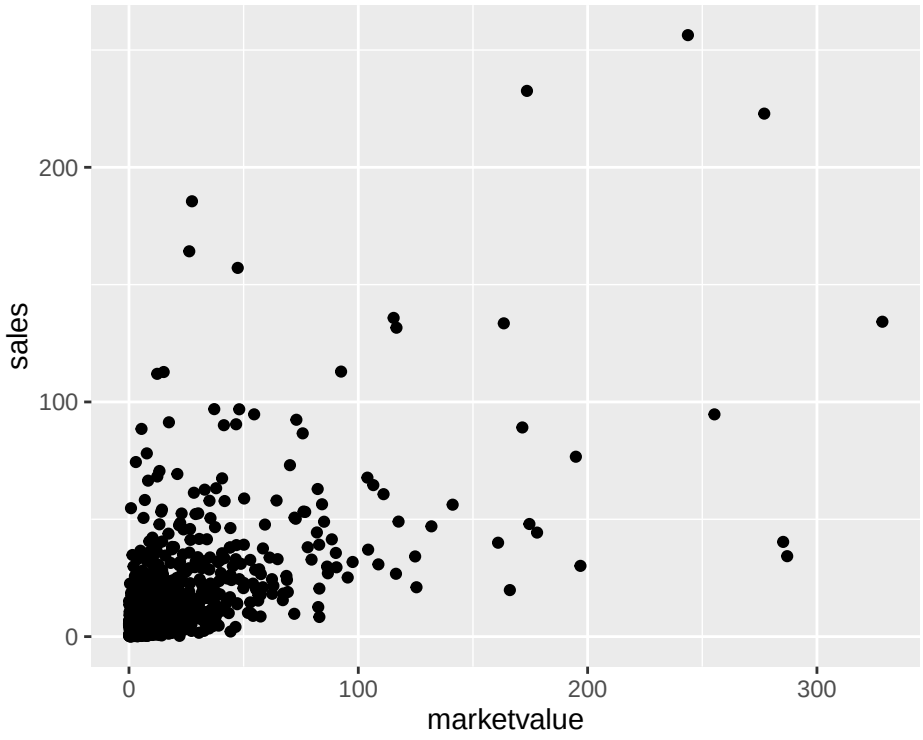
```
library(ggplot2)
#?qplot
qplot(marketvalue,data = Forbes2000)
```



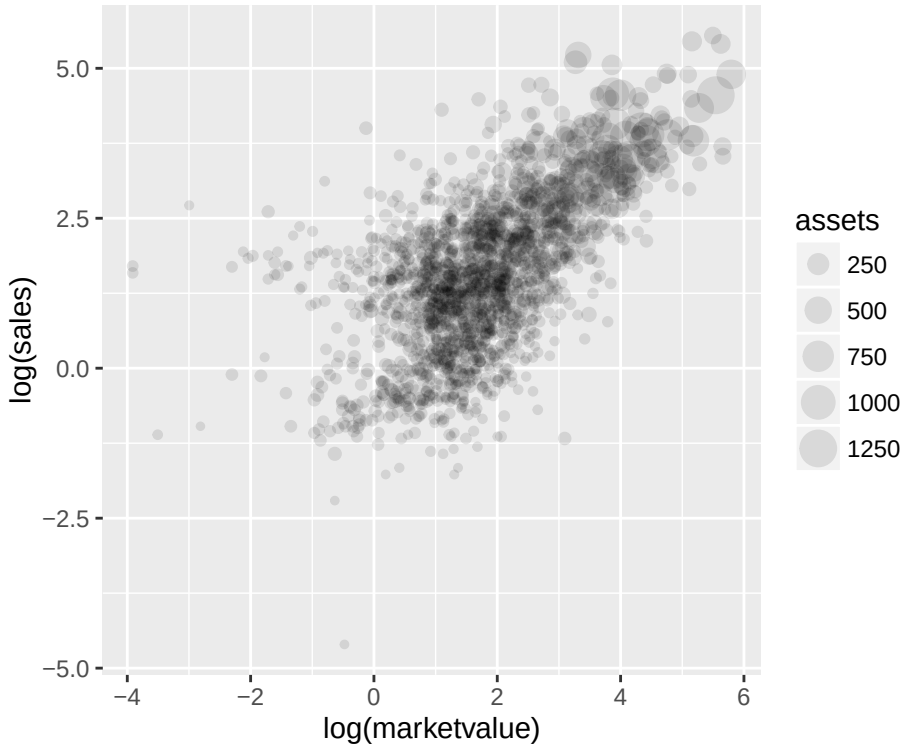

```
qplot(log(marketvalue),  data = Forbes2000)
```



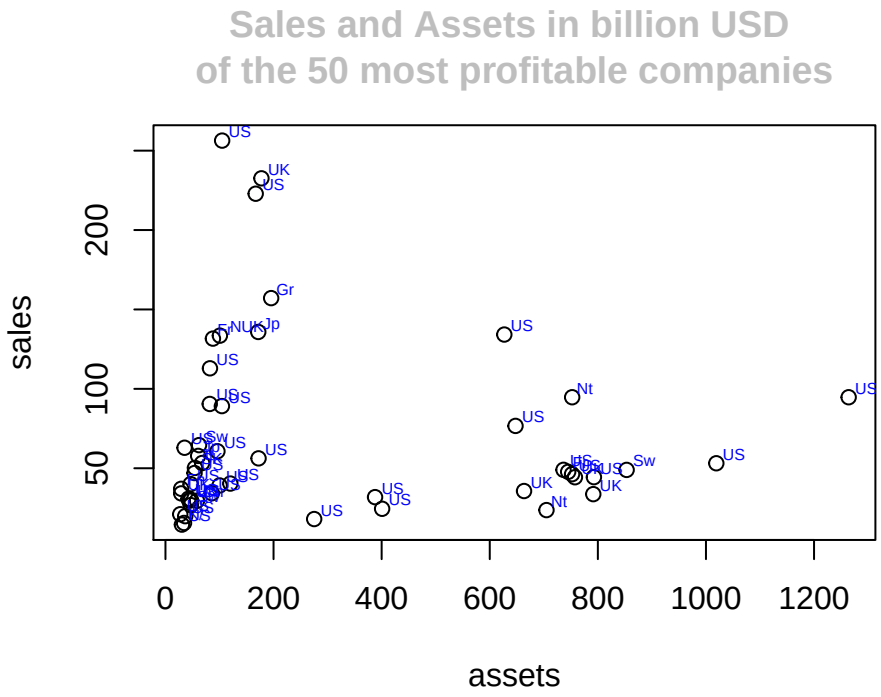

```
qplot(marketvalue,sales, data=Forbes2000)
```




```
qplot(log(marketvalue),log(sales),size=assets,alpha = I(0.1),data=Forbes2000)
```



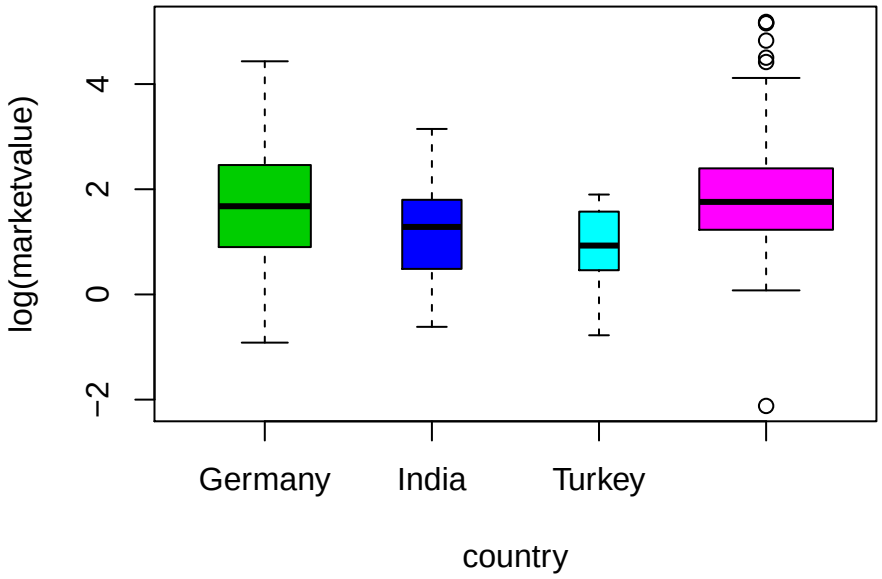

```
library(calibrate) # to use ?textxy
profits_all = na.omit(Forbes2000$profits)  # all_profts without No data
order_profits = order(profits_all)     # index of the profitable companies in decreasing or
top_50 = rev(order_profits)[1:50]      # top 50 profitable companies

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 ar

plot(assets,sales,pch =1)
textxy(assets,sales, abbreviate(countries,2),col = "blue",cex=0.5)  # used to put the count
title(main = "Sales and Assets in billion USD \n of the 50 most profitable companies ", col
```



Graphics by factor

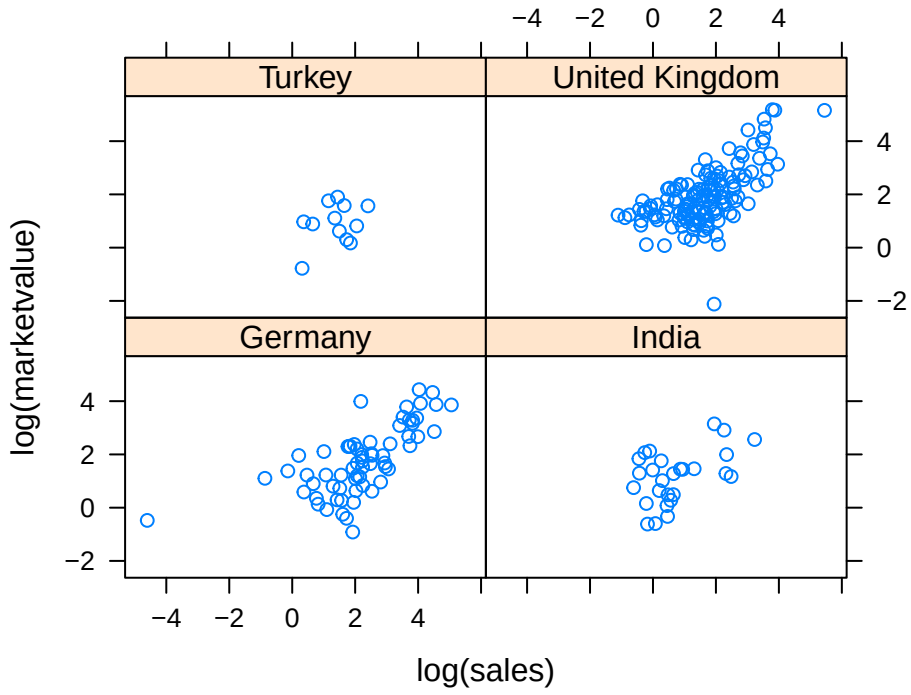
Boxplots of the logarithms of the market value for four selected countries, the width of the boxes is proportional to the square roots of the number of companies.

```
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

```
library(lattice)
xyplot(log(marketvalue)~log(sales)|country,data=tmp)
```



1.21 Questions

1. Calculate the median profit for the companies in the US and the median profit for the companies in the UK, France and Germany.

2. Find all German companies with negative profit.

3. To which business category do most of the Bermuda island companies belong?

4. For the 50 companies in the Forbes data set with the highest profits, plot sales against assets (or some suitable transformation of each variable), labelling each point with the appropriate country name which may need to be abbreviated (using abbreviate) to avoid making the plot look too "messy".

5. Find the average value of sales for the companies in each country in the Forbes data set, and find the number of companies in each country with profits above 5 billion US dollars.