Introduction to R: practical 1 solutions

This practical aims at introducing you to the R interface. By the end of this practical you should be able to load in data, calculate some summary statistics and construct some basic plots.

If you have brought your own data, then I would recommend that you quickly work through this practical first, then try to load in your own data.

1 Getting started

1. Open `Rstudio`
2. Now open a new R script:

   ```
   File -> New File -> R Script
   ```

3. In `Rstudio`, type a basic R command, say

   ```
   x = 5
   ```

4. Press `Ctrl + Enter` anywhere on the line with `x = 5`. This should send the command `x = 5` to the R console in the bottom left hand window.

   In Rstudio click on `Help` and `Keyboard short cuts` to see other short cuts.

5. In `Rstudio`, save the file you are currently working on. Rstudio will (correctly) add the file extension `.R`

Other `Rstudio` commands are:

- If the cursor is at the beginning of the line, pressing `Ctrl + Enter` will send that line to the R console.
- If you highlight a few lines of R code, pressing `Ctrl + Enter` will send that code to the R console.
- Pressing `Ctrl + Shift + S` sends the entire file to R console.

It's probably worth creating a directory to store any R files that you create.

1.1 Course R package

Installing the the course R package[1] is straightforward:

[1] A package is an *add-on* or a *module*. It provides with additional functions and data.

```
install.packages("nclRintroduction",
                 repos="http://R-Forge.R-project.org")
```

If this doesn't work, try

```
install.packages("nclRintroduction",
                 repos="http://R-Forge.R-project.org",
                 type="source")
```

This R package contains the copies of the practicals, solutions and data sets that we require. To load the package, use

```
library("nclRintroduction")
```

2 *The data set*

We are going to investigate the IMDB data set described in chapter 1. Movies were selected for inclusion if they had a known length, had been rated by at least one IMDB user and had an mpaa rating. This gives 4847 films, where each film has 24 associated variables. The first few rows are given in Table 1.1 in the notes. This is **only a subset** of the data, the actual data set contains information on over 50,000 movies.

2.1 *Retrieving the data*

To access the data into R, we use the following command:

You can enter the url into your web browser and view the data directly.

```
library(nclRintroduction)
data(movies)
d = movies
```

We can inspect the column names using:

```
## d is a data frame
colnames(d)
```

We can change the column names, for example,

```
colnames(d)[17]

## [1] "mpaa"

colnames(d)[17] = "MPAA"
```

We can select individual columns, using either their column name:

```
d$Year
```

or their column number:

```
d[,2]
```

When vectors or data frames are too large to manage, we use the function `head` to take a peek at the data:

```
head(d$Title, 5)

## [1] "A.k.a. Cassius Clay"
## [2] "AKA"
## [3] "AVP: Alien Vs. Predator"
## [4] "Abandon"
## [5] "Abendland"
```

- In the above function call, what does "5" specify? What happens if you omit it?

```
##5 specifies the number of rows to return
## For example, using 2 gives
head(d$Title, 2)

## [1] "A.k.a. Cassius Clay" "AKA"
```

- Another useful function is `tail`. What does this function do?

```
## tail gives the *last* few values/rows
```

Using the `dim` function, how many columns and rows does `d` have?

3 *Scatter plots (chapter 5.1 of the notes)*

Let's start with some simple scatter plots:

```
plot(d$Length, d$Rating)
```

which gives figure 1.
Now,

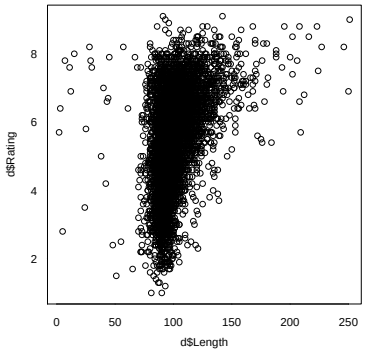

Figure 1: Scatter plot of movie length against rating.

- When you call `plot`, R guesses at suitable axis labels. Use the `xlab` and `ylab` arguments to specify better axis labels. For example,

```
plot(d$Length, d$Rating, xlab="Length", ylab="Rating")
```

gives figure 2.

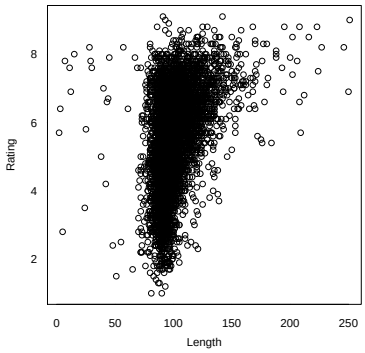

Figure 2: Scatter plot of movie length against rating with axis labels.

- Use the `ylim` argument to specify a y-axis range from 1 to 10.

```
plot(d$Length, d$Rating,
     xlab="Length", ylab="Rating",
     ylim=c(1, 10))
```

- Use the `col` argument to change the colour of the points.

```
plot(d$Length, d$Rating,
     xlab="Length", ylab="Rating",
     ylim=c(1, 10), col=2)
##Or
plot(d$Length, d$Rating,
     xlab="Length", ylab="Rating",
     ylim=c(1, 10), col="red")
```

- Use the `main` argument to give the plot a title.

```
plot(d$Length, d$Rating,
     xlab="Length", ylab="Rating",
     ylim=c(1, 10), col=2,
     main="Movie rating against length")
```

- I tend to alter the default plot command using:[2]

```
op = par(mar=c(3,3,2,1), mgp=c(2,.7,0),
         tck=-.01, las=1)
```

What do you think? Can you determine what `mar`, `mgp`, `tck` and `las` mean?

[2] An explantion of `op` and `par` are given at the end of this practical.

```
#  * mar: A numerical vector of the form c(bottom, left, top, right)
# which gives the number of lines of margin to be specified on the four
# sides of the plot. The default is c(5, 4, 4, 2) + 0.1
#
#  * mgp: The margin line for the axis title, axis labels and axis line.
# The first number affects title,  whereas the second and third values affect
# axis. The default is c(3, 1, 0).
#
#  * tck: Reduces the tick length.
#
#  * las: Changes the style the axis labels. las=1 rotates the y-axis label by 90 degrees.
```

To reset the plotting device, run the command

```
par(op)
```

4 *Summary Statistics*

Use the commands `mean`, `median`, and `sd` to calculate the summary statistics for the movie length.

Statistics	Value
Mean	
Median	
Std. dev	

Table 1: Summary statistics for the movie data set.

```
##Mean
mean(d$Length)

## [1] 100.9

##Mean
median(d$Length)

## [1] 97

##Mean
sd(d$Length)

## [1] 17.34
```

5 Histograms

We will now investigate the distribution of movie years using histograms. Use the `hist` function to plot a histogram of the movie years:

```
hist(d$Year)
```

which gives figure 3. The default method for determining the number of bins used in your histogram isn't that great. So you can use different rules

```
hist(d$Year, breaks="FD")
```

The arguments that we encountered when looking at scatter plots can be used with histograms.

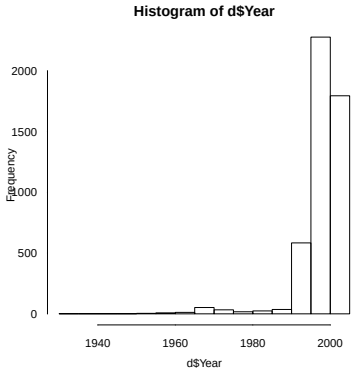

Figure 3: Histogram of movie year.

6 Boxplots

Let's generate a boxplot for the ratings data:

```
boxplot(d$Rating)
```

All the usual arguments, such as `xlab`, can be used here. Now lets, separate the movie data by whether it's a romantic movie:

```
boxplot(d$Rating ~ d$Romance)
```

Try generating a similar boxplot, but for other variables. What happens when you condition on more than one variable?

```
##Conditioning on other variables
boxplot(d$Rating ~ d$Romance + d$Action)
```

```
## Change axis labels
## Plot a boxplot but skip the labels
boxplot(d$Rating ~ d$Romance, axes=FALSE, frame.plot=TRUE,
        ylim=c(0,10))
## Y-axis. 0 to 10 in steps of 2.5
axis(2,at=seq(0,10,2.5))
## X-Axis, at points x=1 & x=2,
## but with labels "non R" and "Romantic"
axis(1,at=1:2, labels=c("Non R", "Romantic") )
```

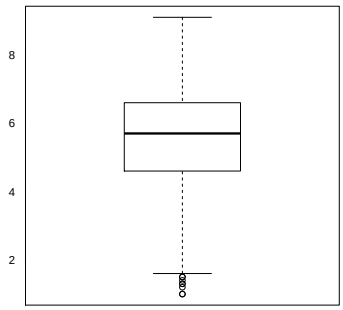

Figure 4: Boxplot of movie ratings.

Discussion of the `par` function

The `par` function is used to set or query graphical parameters. You can see a list of available parameters by looking at the help file `?par` or by just typing `par()` in the console. All of the available parameters have default settings (see the help page).

When we run the command

```
(op = par(cex.lab=0.9))

## $cex.lab
## [1] 0.95
```

The axis label size is reduced to 90% of the base size. The parameter `op` is set to the *previous* value of `cex.axis`, i.e.

```
op

## $cex.lab
## [1] 0.95
```

By setting `op` to the *previous* value, we can easily reset the graphical parameters via

```
par(op)
```

Solutions

Solutions are contained within this package:

```
library(nclRintroduction)
vignette("solutions1", package="nclRintroduction")
```