Reproducible analysis in R

Stefano De Sabbata

This document is an small example of a reproducible analysis.

Data

This analysis document depends on *Gather_DfT_data_2015.R*, which retrieves some data from the Department for Transport and saves them in the *Data* folder, and it is assumed to have been executed before this document. The data file is relatively large and it is available in compressed *zip* format. It is thus necessary to download the file and process it locally before reading it in to R. The *download.file* and *unzip* functions can be used to read in data: *download.file* takes two arguments, the URL of the file to download, and the name of a local file into which it will be stored.

The function *unzip* takes at minimum one argument, which is the file to be unzipped. It is possible to specify the folder in which the files should be extracted to, and it is also possible to specify the files to be extracted (see *?unzip*). In this case, once the function has been executed, the files contained in the zip file are available in the working *Data* folder.

Read the *Gather_DfT_data_2015.R* code and then execute it by clicking on the *Source* button on the top-right of the code when the file is open on RStudio.

Libraries

As mentioned in earlier lectures, libraries are collections of functions. Libraries can be installed in R using the function *install.packages* and loaded using the function *library*, as shown below (note the use of quote for the first and lack thereof in the second). Once a library is installed on a computer you don't need to install it again, but every script needs to load all the library that it uses. Once a library is loaded all its functions can be used.

Before staring, let's load the *knitr* library that we need to use *kable* to create nice tables in R Markdown.

```
# Let's load some libraries
library(knitr)
```

Loading the data

R can import a number of different types of data files. One of the most commonly used format is the *.csv* (comma separated values). You can open the file using Notepad to see its text format, and using Microsoft Excel to see the tabular structure. The code below loads the data present in the file using the *read.csv* function into a *data.frame* variable named *accidents*. The function *head* is used to retrieve only the top rows to be displayed in this document. In this case, the code specifies to retrieve only the first five rows ($n = 5$). Those five row are the displayed using the *kable* function mentioned above, which create a nice and well-formatted table.

```
# Load data about accidents
accidents <- read.csv("../Data/Accidents_2015.csv")
```

```r
# Inspect the content in data frame
kable(head(accidents[, c("Date", "Time", "Local_Authority_.District.")], n = 5))
```

Date	Time	Local_Authority_.District.
12/01/2015	18:45	12
12/01/2015	07:50	12
12/01/2015	18:08	12
13/01/2015	07:40	12
09/01/2015	07:30	12

A simple analysis

The function *count* of the *dplyr* library can be used to count the number rows of a *data.frame*, groupped by a variable (the local authority district in the example below).

```r
# The warn.conflicts and quietly options can be used
# to suppress the message on loading
library(dplyr, warn.conflicts = FALSE, quietly = TRUE)
```

As you can see, the code below is formatted in a way similar to a code block, although it is not a code block. This is very common in R programming, especially when functiions have a lot of parameters, and it makes the code more readable.

```r
# Count number of accident by local authority
accidents_count <- count(
  accidents,
  Local_Authority_.District.
  )

# Inspect the content in data frame
kable(head(accidents_count, n=5))
```

Local_Authority_.District.	n
1	1578
2	936
3	844
4	857
5	1066

The function *summarise* of the same library *dplyr* can be used in combination with the function *group_by* to summarise the values of the rows of a *data.frame*. Rows can be groupped by a variable (in the example below, whether an area is urban or rural), and then aggregated based on a function over one or more columns (*mean* over the column *Accident_Severity* in the example).

```r
# Average severity per Urban and Rural areas
accidents_urb_rrl_severity <- summarise(
  group_by(
    accidents,
    Urban_or_Rural_Area
  ),
  avg_severity = mean(Accident_Severity)
```

```
)

# Inspect the content in data frame
kable(accidents_urb_rrl_severity)
```

Urban_or_Rural_Area	avg_severity
1	2.861045
2	2.781662

Exercise 2

Add another section of R code below, that creates a new column for the *accidents* dataset, which represents the hour of the accident – e.g., from *18:45* to *18*. That can be achieved using the function *substr* (to select a section of a character variable, see *?substr*) on the column *Time*.