Reproducible analysis in R

Stefano De Sabbata

This document illustrates how to download data in different ways from the internet.

There are a number of ways of getting data from the internet. The three approaches listed below are the most frequently used, and this notebook covers the first two.

- direct access to data
- downloading files
- working with APIs

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)
```

Direct Access to Data

In some situations you may be able to directly download from the internet. R can deal with this in a number of ways. If the data on the internet is simply a text file, then the URL can be sometimes be substituted for a file name with a number of commands. This works with *read.csv* and *read.table* for example. An simple demonstration is provided below. Data about UK mobile phone coverage are downloaded using a link from the Data.Gov.UK website and *read.csv* into a data frame called *coverage_data_original*.

```
# Download the data
coverage_data_original <- read.csv(
  "https://www.ofcom.org.uk/__data/assets/file/0034/94678/ofcom-uk-mobile-coverage-data-2012.csv",
  skip=1)
```

```
# A nice table, with only the first two columns
kable(head(coverage_data_original[, 1:2], n=3))
```

Local.Authority	X2G.Geographic..No.reliable.signal
Aberdeen City	0.10%
Aberdeenshire	15.90%
Abertawe - Swansea	1.00%

The *head* function is used to show the first three line, for the first two column. Note how *read.csv* created names for the columns adding an *X* in front of the names starting with a number (e.g., the second column). In the code above, the option *skip=1* is used to instruct R to ignore the first line of the csv file, which contains the date the dataset has last been updated. The data format is still problematic, as the percentages are interpreted as character variables. The code above transforms each percentage column to its numeric value. The function *gsub* is used to replace the character *%* with nothing, then the value is converted to numeric. The column names are then shortened by assigning a new vector of names to *colnames(coverage)*. Note the difference in the values and column names, compared to the original data above.

```r
# Make a copy of the data
coverage_data <- coverage_data_original

# For each column from the 2nd to the 19th
# assign the same column to the result of
# substituting the percentage symbol with nothing
# then convert to numeric
for(i in 2:19){
  coverage_data[,i] <- as.numeric( gsub("%", "", coverage_data[,i]) )
}

# Use simplier column names
colnames(coverage_data) <- c("LocalAuthority",
  "M2G.NoS","M2G.1op","M2G.2op","M2G.All",
  "M2G.NoS.Prem","M2G.1op.Prem","M2G.2op.Prem","M2G.All.Prem",
  "M3G.NoS","M3G.1op","M3G.2op","M3G.3op","M3G.All",
  "M3G.NoS.Prem","M3G.1op.Prem","M3G.2op.Prem","M3G.3op.Prem","M3G.All.Prem",
  "MonthMB.Prem")

# Print the table: showing percentage of areas with no 2G signal
kable(head(coverage_data[, c(1, 2)], n=3))
```

LocalAuthority	M2G.NoS
Aberdeen City	0.1
Aberdeenshire	15.9
Abertawe - Swansea	1.0

The table below shows local authorities with more than 20% of their area not covered by 2G network.

```r
kable(coverage_data[coverage_data$M2G.NoS>20, c("LocalAuthority", "M2G.NoS")])
```

	LocalAuthority	M2G.NoS
4	Angus	27.8
7	Argyll and Bute	38.9
69	Dumfries and Galloway	22.6
85	Gwynedd - Gwynedd	21.9
90	Highland	37.7
114	Moray	25.8
116	Na H-Eileanan an Iar	24.2
130	Northumberland	20.9
137	Perth and Kinross	27.5
139	Powys - Powys	21.6
149	Scottish Borders	24.5
152	Shetland Islands	22.9
155	Sir Ceredigion - Ceredigion	22.8
164	South Ayrshire	29.0
171	Stirling	29.2
174	Strabane	32.3

Exercise 1

Note: sometime it is difficult to understand the format and type of data downloaded from the internet. Try to solve this exercise without asking further clarification to the demonstrator. Refer to the original data and webpage linked above if necessary. If you are not familiar with the topic, the 2G mobile phone network is the "old" network, that can be used by mobile phones to make calls and send SMS. The 3G network is a more recent one, that mobile phones can use to access the internet. Search for further information on the internet if something is still not clear.

Add another section of R code below, that creates a table listing the geographic areas having no 3G network coverage on more than 50% of their area.

Downloading Files

The Direct Access approach works well provided the file referred to by the URL is a type of file directly readable by R. However, particularly with large files, it is quite common to use a binary format file (such as a zipped file). In some cases, it may still be possible to access the file directly (for example with *RData* files, via *load*) but it is often necessary to download the file and process it locally before reading it in to R. This is the case, for example, with zipped files. In this case, a combination of 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. For example, the following code downloads a zip file containing data about road accidents in the UK in 2015.

The function *unzip* takes at minimum one argument, which is the file to be unzipped – a place where to extract the information from the zipped file to can also be added, and it is also possible to specify the files to be extracted (see *?unzip*). The file contained in the zip file are now available in the working directory.

```
# Set the values: URL and file name
data_url <- "http://data.dft.gov.uk/road-accidents-safety-data/RoadSafetyData_2015.zip"
data_file_name <- "../Data/RoadSafetyData_2015.zip"
```

```
# Download the file
#download.file(data_url, data_file_name)

# Depending on the system these might also work:
# or download.file(data_url, data_file_name, method="auto")
# or download.file(data_url, data_file_name, method="curl")

# Unzip file in the Data folder
unzip(data_file_name, exdir = "../Data/")

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

# 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

The function *count* of the *dplyr* library can be used to count the number rows of a *data.frame* groupped by a given variable.

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

```
# 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

```
# 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*.