LAB 1

Michael Kummer

Introduction to R

What is R? "R is a language and environment for statistical computing and graphics." (http://www.r-project.org/about.html)

- Download R: http://cran.r-project.org/
- Install.

What is Rstudio? "RStudio is a free and open source integrated development environment (IDE) for R, a programming language for statistical computing and graphics." (http://en.wikipedia.org/wiki/RStudio)

- Download Rstudio: http://www.rstudio.com/
- Install.
- Open Rstudio (not R): If you have a Windows PC and Rstudio does not show on your desktop you have to search for it on your start menu.

During this course we will always work on Rstudio as it is more user-friendly.

Rstudio screen

The Rstudio screen is divided in four panels, some of which are subdivided into tabs. The main panels are:

1. **Console**
   - Here you can type commands and see their output.
2. **Environment and history**
   - Environment: This tab stores the things you create during your R session.
   - History: This is a record of your activity, you can save it for later use.
3. **Files, plots, packages and help**
   - Files: This tab shows you the files and folders in your workspace (or work folder).
   - Plots: This tab will show you all the plots/graphics you produce.
   - Packages: This tab will show you a list of the add-ons you have available and indicate whether they are on/off.
   - Help: This tab gives you a search box where you can search for additional information.
4. **The R script(s) and data view**
   - The R script is where you keep a record of your work.

Packages

Think of the R packages as smartphone applications. There is a large number of available packages that serve different purposes. In order to install a package you should type:

install.packages("write.name.of.package.here")

You need to be connected to the internet in order to install packages. When installing a package do not worry about the messages in red showing on your console unless they explicitly say "error".

You only need to install each package once. Once you have installed a package, you can turn it on by typing:

library(write.name.of.package.here)

The package tab will show you which packages you have available. If the package is checked that means it is active, if it is not checked you will need select it in order to use it.

Working Directory

At any moment, if you want to check what is your current working directory, you can type:

getwd()

During and R session, if you want to work from a different folder than your default folder you can type:

setwd("write/path/to/folder/here")

You can check what files you have in your working directory by running the following command:

list.files()

Getting Started

Using the Console as a calculator

You can use the Console just like a calculator (a very sophisticated calculator.) Try performing some calculations:

```
1+1
```

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

```
3*4
```

```
## [1] 12
```

```
24/6
```

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

```
(2*10) - (3*4)
```

```
## [1] 8
```

```
2^3
```

```
## [1] 8
```

```
8^(1/3)
```

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

R has built in mathematical functions, for example:

```
sqrt(25) # square root
```

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

```
log(1)   # natural log
```

```
## [1] 0
```

You can store the results of your calculations in your work environment by giving them names. In order to name any kind of object you use "<-". Lets try:

```
my.sum <- 10 + 10  # save result
my.sum             # display result
```

```
## [1] 20
```

Once you store an object in your environment you can interact with it directly by using its name:

```
my.sum/10
```

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

R scripts

When working with R, you should keep a record of your code so that you can keep track of what you have done and re-use it later. In this course we will always work on Rscripts and **not** on the Console.

Create new Rscript

- You can open a new Rscript by clicking on the blank sheet with a green plus sign on the top left corner of your screen and selecting the option "R script".
- You can then save your script, for instance - "Lab1-notes.R".

Working on an Rscript

- In order to send a command line from the Rscript to the Console just put the cursor on the selected command line and click ctrl+enter (PC users) or cmd+enter (MAC users).
- You can also select several command lines and enter them all at once.
- You can write comments on your Rscript (lines that the program will ignore) by using the symbol "#" on the beginning of a word or phrase, for instance:

```
# This is a comment, it will be ignored by the program.
```

- Every time you want to re-run a command you have already typed in your script, just put the cursor on top of it and enter it, you do not need to retype it.
- At the end, don't forget to save your script before you close Rstudio.
- R is **case sensitive** and **spelling sensitive**.

Loading Data into Rstudio

On the course's page you will find an example dataset. Please do the following:

- Download the dataset to your computer.
- Either save the file directly in your TopicsDig folder or copy the file from your downloads folder to your TopicsDig folder.
- Type the following in your Rscript:

```
setwd("C:/TopicsDig/Labs")
# change the file's path to your own
load("C:/TopicsDig/Labs/datasets/ceosal2.RData")
```

You can see that two new objects appeared in your work environment:

- Data: This is a table with data on CEO salaries.
- Desc: This is a table with the description of each variable in Data.

When loading files to R, you need to use a different function according to the file's extension. The "load" function we just used opens files of the type ".RData". If you want to load a ".csv" file you should use the function "read.csv" instead of "load".

Exploring Data in R

During this class we will work with a data format called "data.table". In order to convert your data into this format you will need to do the following:

Install the data.table package:

install.packages("data.table")

Activate the data.table package:

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.1.3
```

Now we will convert our dataset "data" into the data.table format. For our own convenience, we will include the prefix "dt." in the name of all objects of the data.table type. Also, it is useful to give names to our objects or datasets that reflect their contents. In this case, a good name for our dataset would be, for instance, "dt.ceo.salaries":

```
dt.ceo.salaries <- data.table(data)
```

You can see that a new object "dt.ceo.salaries" appeared in your work environment. This is a duplicate of the dataset "data". As we don't need the original dataset we can delete it by typing the following:

```
rm(data)
```

The "rm" stands for "remove" and it deletes objects from your work environment. You cannot undo this action.

There are many options for exploring your data in R. You can get the names of the variables in your dataset by typing:

```
names(dt.ceo.salaries)
```

```
##  [1] "salary"   "age"      "college"  "grad"     "comten"   "ceoten"
##  [7] "sales"    "profits"  "mktval"   "lsalary"  "lsales"   "lmktval"
## [13] "comtensq" "ceotensq" "profmarg"
```

```
# or
colnames(dt.ceo.salaries)
```

```
##  [1] "salary"   "age"      "college"  "grad"     "comten"   "ceoten"
##  [7] "sales"    "profits"  "mktval"   "lsalary"  "lsales"   "lmktval"
## [13] "comtensq" "ceotensq" "profmarg"
```

You can count the number of columns in your dataset:

```
ncol(dt.ceo.salaries)
```

```
## [1] 15
```

And count the number of observations (rows):

```
nrow(dt.ceo.salaries)
```

```
## [1] 177
```

You can look at the first rows in your data:

```
head(dt.ceo.salaries)
```

```
##    salary age college grad comten ceoten sales profits mktval  lsalary   lsales
## 1:   1161  49       1    1      9      2  6200     966  23200 7.057037 8.732305
## 2:    600  43       1    1     10     10   283      48   1100 6.396930 5.645447
## 3:    379  51       1    1      9      3   169      40   1100 5.937536 5.129899
## 4:    651  55       1    0     22     22  1100     -54   1000 6.478509 7.003066
## 5:    497  44       1    1      8      6   351      28    387 6.208590 5.860786
## 6:   1067  64       1    1      7      7 19000     614   3900 6.972606 9.852194
##      lmktval comtensq ceotensq  profmarg
## 1: 10.051908       81        4 15.580646
## 2:  7.003066      100      100 16.961130
## 3:  7.003066       81        9 23.668638
## 4:  6.907755      484      484 -4.909091
## 5:  5.958425       64       36  7.977208
## 6:  8.268732       49       49  3.231579
```

Or the last rows:

```
tail(dt.ceo.salaries)
```

```
##    salary age college grad comten ceoten sales profits mktval  lsalary   lsales
## 1:    218  57       1    1     33      5   504      41    421 5.384495 6.222576
## 2:    264  63       1    0     42      3   334      43    480 5.575949 5.811141
## 3:    185  58       1    0     39      1   766      49    560 5.220356 6.641182
## 4:    387  71       1    1     32     13   432      28    477 5.958425 6.068426
## 5:   2220  63       1    1     18     18   277     -80    540 7.705263 5.624018
## 6:    445  69       1    0     23      0   249      31    828 6.098074 5.517453
##     lmktval comtensq ceotensq   profmarg
## 1: 6.042633     1089       25   8.134921
## 2: 6.173786     1764        9  12.874251
## 3: 6.327937     1521        1   6.396867
## 4: 6.167517     1024      169   6.481482
## 5: 6.291569      324      324 -28.880867
## 6: 6.719013      529        0  12.449800
```

You can also view your entire dataset by typing:

```
View(dt.ceo.salaries)
```

As R is capitalization and spelling sensitive, if you type "view" instead of "View" in the above command it will not work. Every time your code is not working you should first check if you got your spelling and capitalization right.

You can also use some of the data.table features to explore your data. In data.tables, when you write the table's name followed by square brackets with a comma in the middle, the space before the comma refers to the table's rows, and the space after the comma refers to the table's columns. When you leave either one of these spaces blank, it means that you are selecting either all rows (if space before the comma is blank) or all columns (if space after the comma is blank), or everything (if both spaces are blank).

```
dt.ceo.salaries[1, ]                    # shows first row and all columns
```

```
##    salary age college grad comten ceoten sales profits mktval  lsalary   lsales
## 1:   1161  49       1    1      9      2  6200     966  23200 7.057037 8.732305
##     lmktval comtensq ceotensq profmarg
## 1: 10.05191       81        4 15.58065
```

```
dt.ceo.salaries[ ,  salary]              # shows all rows of variable "salary"
```

```
##   [1] 1161  600  379  651  497 1067  945 1261  503 1094  601  355 1200  697 1041
##  [16]  245  817 1675  971  609  470  867  752  246  825  358 1162  270  829  300
##  [31] 1627 1237  540 1798  474 1336  541  129 1700 1750  624  791 1487 2021 1550
##  [46]  401 1295  449  456 1142  577  600  649  822 1080 1738  581  912  650 2199
##  [61]  609 1946  552  481  526  471  630  622  999  585 1107 1099  425 2792  350
##  [76]  363 2265  377  879  720  950 1143 1064 1253  462  174  474 1248 1101  348
##  [91]  650  875 1600 1500  323  459  925  375  447 1340 1749  491 5299  431  729
## [106] 1284 1373  989  515 1301  834  849  100  679  567  559  704  308 1392  389
## [121]  790  396  398  707  984  410 1095  694  834 1630  493  625  483  733 2102
## [136]  853  345  800  764  806  310 1119 1287 1170  880 1091 1100  650  607 1133
## [151]  393  605 1444 1033 1142  537  693  439  358 1276  873  537  713 1350 1268
## [166]  465  693  369  381  467  559  218  264  185  387 2220  445
```

```
dt.ceo.salaries[1, salary]               # shows first row of variable "salary"
```

```
## [1] 1161
```

```
dt.ceo.salaries[1:10, list(salary, age)] # shows first ten rows of the variables "salary" and "age"
```

```
##     salary age
##  1:   1161  49
##  2:    600  43
##  3:    379  51
##  4:    651  55
##  5:    497  44
##  6:   1067  64
##  7:    945  59
##  8:   1261  63
##  9:    503  47
## 10:   1094  64
```

Ordering the data:

```
dt.ceo.salaries[order(age)]  # order ascending (default)
```

```
##      salary age college grad comten ceoten sales profits mktval  lsalary
##   1:   1091  33       1    0      9      9   181      36   1300 6.994850
##   2:    607  38       1    1      7      3   231      38    599 6.408529
##   3:    323  39       1    1     15      3   637      63    517 5.777652
##   4:   1630  39       1    1      8      8   227      27    822 7.396335
##   5:    474  40       1    0     18      1  2700     117   2000 6.161207
##  ---
## 173:    971  72       1    1     33     24  1400      69    609 6.878326
## 174:   1946  73       1    0     25     21  7800     484   8000 7.573531
## 175:    300  77       0    0     45     26  6900     483   4700 5.703783
## 176:    396  80       1    0     58     28   513      53    963 5.981414
## 177:    425  86       1    1     13     13    36      11    644 6.052089
##        lsales  lmktval comtensq ceotensq  profmarg
##   1: 5.198497 7.170120       81       81 19.889503
##   2: 5.442418 6.395262       49        9 16.450216
##   3: 6.456769 6.248043      225        9  9.890110
##   4: 5.424950 6.711740       64       64 11.894273
##   5: 7.901007 7.600903      324        1  4.333333
##  ---
```

```
## 173: 7.244227 6.411819     1089      576  4.928571
## 174: 8.961879 8.987197      625      441  6.205128
## 175: 8.839276 8.455317     2025      676  7.000000
## 176: 6.240276 6.870053     3364      784 10.331384
## 177: 3.583519 6.467699      169      169 30.555555
```

```
dt.ceo.salaries[order(-age)] # order descending
```

```
##      salary age college grad comten ceoten sales profits mktval  lsalary
##   1:    425  86       1    1     13     13    36      11    644 6.052089
##   2:    396  80       1    0     58     28   513      53    963 5.981414
##   3:    300  77       0    0     45     26  6900     483   4700 5.703783
##   4:   1946  73       1    0     25     21  7800     484   8000 7.573531
##   5:   1200  72       1    0     37     37   796      35    678 7.090077
##  ---
## 173:    310  40       1    0     18      1  2400      60   1300 5.736572
## 174:    323  39       1    1     15      3   637      63    517 5.777652
## 175:   1630  39       1    1      8      8   227      27    822 7.396335
## 176:    607  38       1    1      7      3   231      38    599 6.408529
## 177:   1091  33       1    0      9      9   181      36   1300 6.994850
##        lsales  lmktval comtensq ceotensq  profmarg
##   1: 3.583519 6.467699      169      169 30.555555
##   2: 6.240276 6.870053     3364      784 10.331384
##   3: 8.839276 8.455317     2025      676  7.000000
##   4: 8.961879 8.987197      625      441  6.205128
##   5: 6.679599 6.519147     1369     1369  4.396985
##  ---
## 173: 7.783224 7.170120      324        1  2.500000
## 174: 6.456769 6.248043      225        9  9.890110
## 175: 5.424950 6.711740       64       64 11.894273
## 176: 5.442418 6.395262       49        9 16.450216
## 177: 5.198497 7.170120       81       81 19.889503
```

Subseting the data:

```
dt.ceo.salaries[age<=45,]   # select only CEOs with less than 45 years
```

```
##      salary age college grad comten ceoten sales profits mktval  lsalary
##  1:    600  43       1    1     10     10   283      48   1100 6.396930
##  2:    497  44       1    1      8      6   351      28    387 6.208590
##  3:    245  44       1    1      7      7   135      24    558 5.501258
##  4:    270  43       1    0     15      2   150      28    713 5.598422
##  5:    474  40       1    0     18      1  2700     117   2000 6.161207
##  6:    649  44       1    1      4      4   336      17    475 6.475433
##  7:    526  45       1    0      8      7  2400     106   2000 6.265301
##  8:   2792  40       1    0     11     11   534      35    888 7.934514
##  9:    377  45       1    0      7      5   238      57   1200 5.932245
## 10:    348  43       1    1     12     10   586      79   1400 5.852202
## 11:    323  39       1    1     15      3   637      63    517 5.777652
## 12:    491  43       1    1     21      2   561      54    521 6.196444
## 13:    989  40       1    0     18      5   439      30    582 6.896694
## 14:    308  45       1    1     14     14   210      39   1900 5.730100
## 15:   1630  39       1    1      8      8   227      27    822 7.396335
## 16:    310  40       1    0     18      1  2400      60   1300 5.736572
## 17:   1091  33       1    0      9      9   181      36   1300 6.994850
## 18:    607  38       1    1      7      3   231      38    599 6.408529
```

```
## 19:    693  42       1    0     17     12  1400     206   3000 6.541030
## 20:    873  41       1    1      2      2   149      21    567 6.771935
##       lsales  lmktval comtensq ceotensq  profmarg
##  1: 5.645447 7.003066      100      100 16.961130
##  2: 5.860786 5.958425       64       36  7.977208
##  3: 4.905275 6.324359       49       49 17.777779
##  4: 5.010635 6.569481      225        4 18.666666
##  5: 7.901007 7.600903      324        1  4.333333
##  6: 5.817111 6.163315       16       16  5.059524
##  7: 7.783224 7.600903       64       49  4.416667
##  8: 6.280396 6.788972      121      121  6.554307
##  9: 5.472270 7.090077       49       25 23.949579
## 10: 6.373320 7.244227      144      100 13.481229
## 11: 6.456769 6.248043      225        9  9.890110
## 12: 6.329721 6.255750      441        4  9.625669
## 13: 6.084499 6.366470      324       25  6.833713
## 14: 5.347107 7.549609      196      196 18.571428
## 15: 5.424950 6.711740       64       64 11.894273
## 16: 7.783224 7.170120      324        1  2.500000
## 17: 5.198497 7.170120       81       81 19.889503
## 18: 5.442418 6.395262       49        9 16.450216
## 19: 7.244227 8.006368      289      144 14.714286
## 20: 5.003946 6.340359        4        4 14.093960
```

```
dt.young.ceo.salaries <- dt.ceo.salaries[age<=45,] # creates a new data table
```

Subseting the data using multiple conditions (use the symbol "&"" for *and* and the symbol "|" for *or*):

```
dt.ceo.salaries[age<=45 & grad==1,]
```

```
##     salary age college grad comten ceoten sales profits mktval  lsalary
##  1:    600  43       1    1     10     10   283      48   1100 6.396930
##  2:    497  44       1    1      8      6   351      28    387 6.208590
##  3:    245  44       1    1      7      7   135      24    558 5.501258
##  4:    649  44       1    1      4      4   336      17    475 6.475433
##  5:    348  43       1    1     12     10   586      79   1400 5.852202
##  6:    323  39       1    1     15      3   637      63    517 5.777652
##  7:    491  43       1    1     21      2   561      54    521 6.196444
##  8:    308  45       1    1     14     14   210      39   1900 5.730100
##  9:   1630  39       1    1      8      8   227      27    822 7.396335
## 10:    607  38       1    1      7      3   231      38    599 6.408529
## 11:    873  41       1    1      2      2   149      21    567 6.771935
##       lsales  lmktval comtensq ceotensq  profmarg
##  1: 5.645447 7.003066      100      100 16.961130
##  2: 5.860786 5.958425       64       36  7.977208
##  3: 4.905275 6.324359       49       49 17.777779
##  4: 5.817111 6.163315       16       16  5.059524
##  5: 6.373320 7.244227      144      100 13.481229
##  6: 6.456769 6.248043      225        9  9.890110
##  7: 6.329721 6.255750      441        4  9.625669
##  8: 5.347107 7.549609      196      196 18.571428
##  9: 5.424950 6.711740       64       64 11.894273
## 10: 5.442418 6.395262       49        9 16.450216
## 11: 5.003946 6.340359        4        4 14.093960
```

Adding a new variable to the data.table:

```
dt.ceo.salaries[, log_salary:=log(salary)]
dt.ceo.salaries[, age_squared:=age^2]
```

Deleting a variable from the data table:

```
dt.ceo.salaries[, log_salary:=NULL]
```

References

http://dss.princeton.edu/training/RStudio101.pdf

Additional Resources

- Data table cheat sheet: https://s3.amazonaws.com/assets.datacamp.com/img/blog/data+table+cheat+sheet.pdf

Acknowledgements and Thanks:

This lab is based on material by M Godinho de Matos, R Belo and F Reis. Gratefully acknowledged!