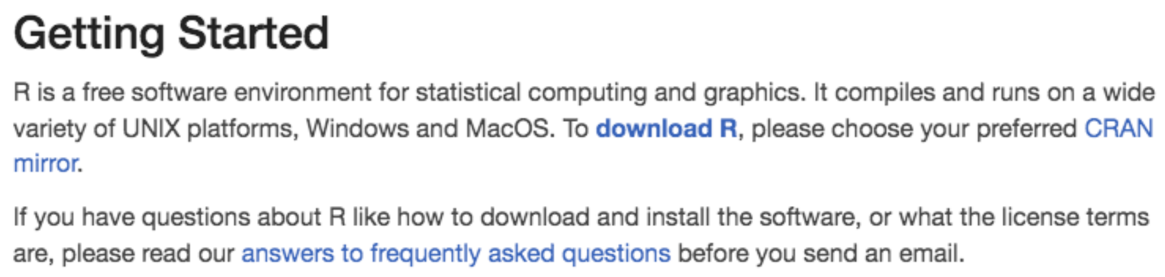
R Basic Workshop

Yuqing | yhe050@uottawa.ca

Instructions for downloading R

1. Go to https://www.r-project.org/ and click “To download R” (as shown below);



Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

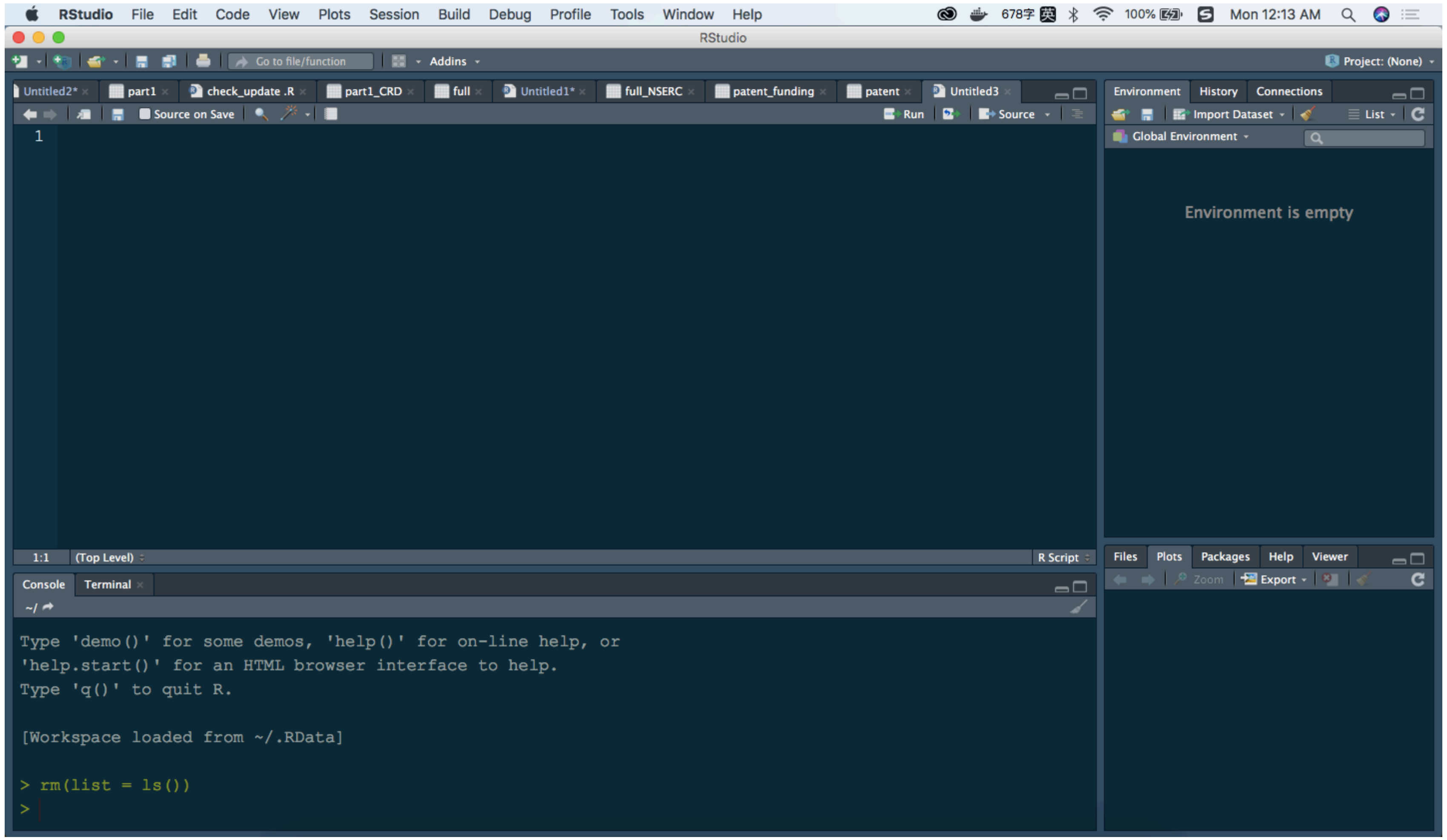
If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

2. Choose one of the shown locations you like and click;
3. Download the version that works for your computer. It has versions for MAC, Windows and Linux
4. Install it and you finish this step!

Instructions for downloading R studio

1. Go to *https://rstudio.com/products/rstudio/download/#download*
2. In ‘Installers for Supported Platforms’ section, choose and click the R Studio installer based on your operating system. The download should begin as soon as you click
3. Install it. (just click next and you can do it! ☺)

R basic: the interface of R



```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> rm(list = ls())
>
```

R basic: data types

1. Character, e.g. “hello world”
2. Numeric, e.g. “123.45” / integer, e.g. “999”
3. Logical, e.g. “TRUE” or “FALSE”

```
> as.numeric(x)     # change to another type
> as.integer(x)
> as.character(x)
```

```
class(x)           # have a look at what type the object is
is.integer(x)  # whether the object is integer
```

R basic: data types

we could store data by

1. creating a vector, e.g. x <- c (1,2,3)
2. creating a list, e.g. x <- list(22, “ab”, TRUE, 1+2i)
3. creating a data frame, e.g. x <- data.frame(x = data1, y=data2, header = TRUE, sep=“,”)

R basic: computations

- x <- 2 + 3
- y <- 6 / 3
- z <- log(12)
- sqrt(), log()...
- c <- (a + sqrt(a))/(exp(2)+1)

- Basic Numerical Descriptions: mean, min, SD, etc... (see scripts)
- round() / ceiling() / floor()

R b

1. If… el

if (<cond

#

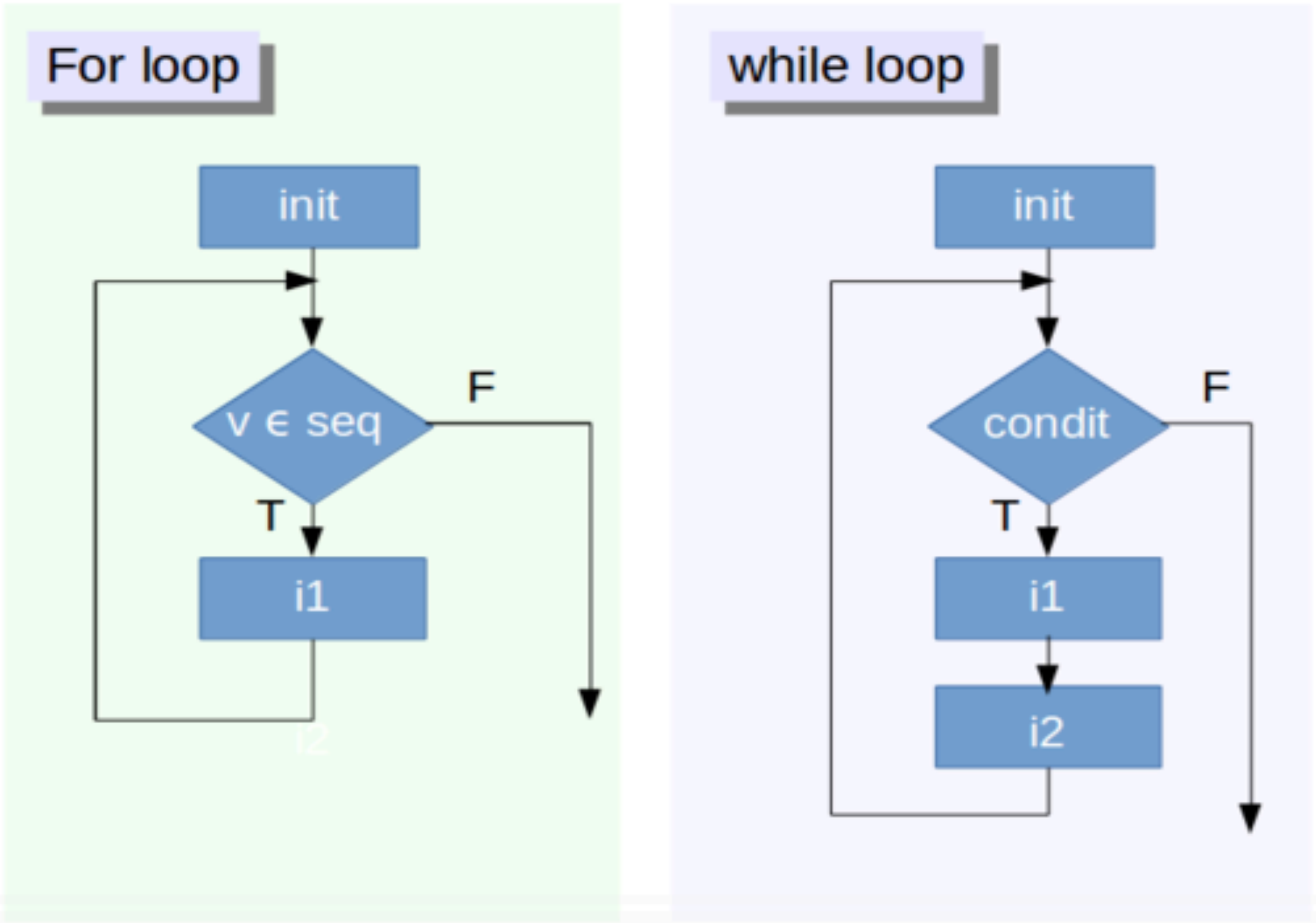
} else {

#

}

condition>){

mething



uctures:

es an infinite loop

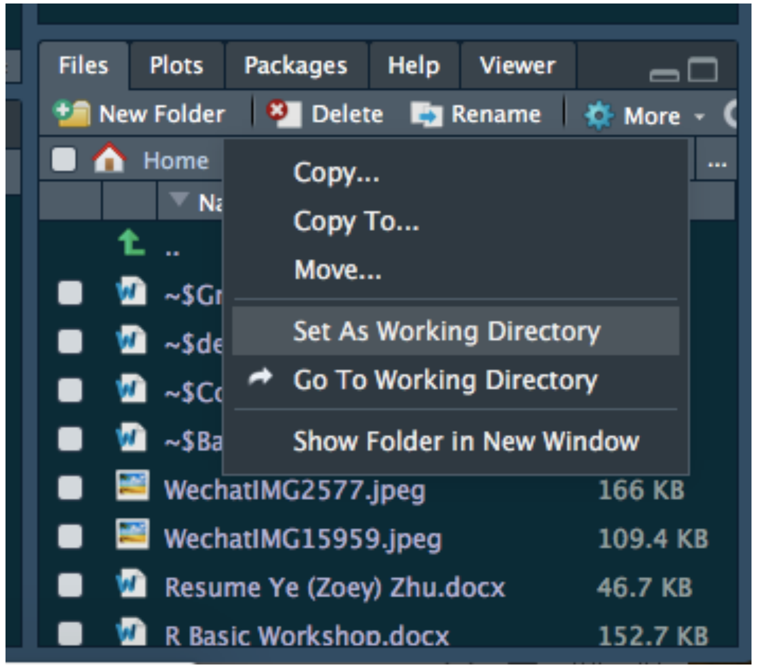
he execution of a loop

next – skips an iteration in a loop

return – exit a function

Data Preparation:
Importing Data and Installing R packages

- Importing Data: two methods for csv file
- 1. set a working directory
- e.g. patent <- read.csv('part2_info_update.csv')



```
..
~$Gr
~$de
~$Co
~$Ba
WechatIMG2577.jpeg        166 KB
WechatIMG15959.jpeg       109.4 KB
Resume Ye (Zoey) Zhu.docx 46.7 KB
R Basic Workshop.docx     152.7 KB
```

2. set an absolute path

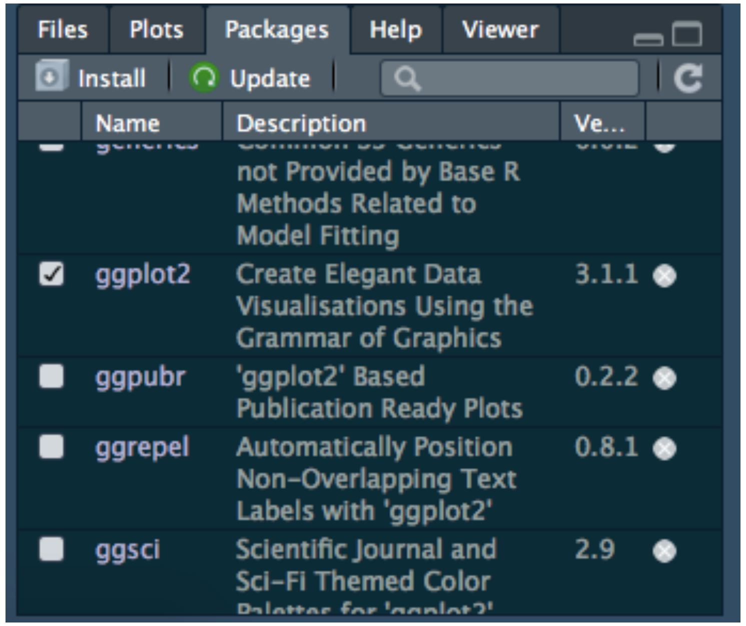
e.g. patent <- read.csv(file = /Users/mayhe/Dropbox/R basic workshop/data processing.csv", header = T, as.is = T, encoding = "UTF=8")

For importing other formats, check packages: data.tables, readr, RMySQL, jsonlite, etc.

Data Preparation:

Importing Data and Installing R packages

- Installing R Packages: two ways
- 1.



	Name	Description	Ve...
		not Provided by Base R Methods Related to Model Fitting	
✓	ggplot2	Create Elegant Data Visualisations Using the Grammar of Graphics	3.1.1
☐	ggpubr	'ggplot2' Based Publication Ready Plots	0.2.2
☐	ggrepel	Automatically Position Non-Overlapping Text Labels with 'ggplot2'	0.8.1
☐	ggsci	Scientific Journal and Sci-Fi Themed Color	2.9

2. install.packages('stringr')
   library(stringr)

Data Preparation:
Some basic functions that are frequently used

- summary()
- sort()/order()/rank()
- unique()
- length()
- attribute()
- dim()
- class()
- nrow()/ncol(), colnames()/row.names()
- is.na()
- table()
- na.omit()
- complete.cases()

Data Preparation:
Basic Steps for Data Pre-Processing

- import data
- check missing data
- transfer categorical data
- splitting the dataset into the training set and test set
- feature scaling

Data Preparation:
Packages for Data Processing

1. ggplot2, for data visualization.
2. dplyr, for data manipulation.
3. tidyr, for data tidying.
4. readr, for data import.
5. stringr, for strings.

(see cheatsheet provided!)

Data Preparation:
Some suggestions for practicing R

1. Find the packages you want from the website:

   https://www.rdocumentation.org/

2. Bonus! you can find some useful cheat sheets here:

   https://github.com/rstudio/cheatsheets

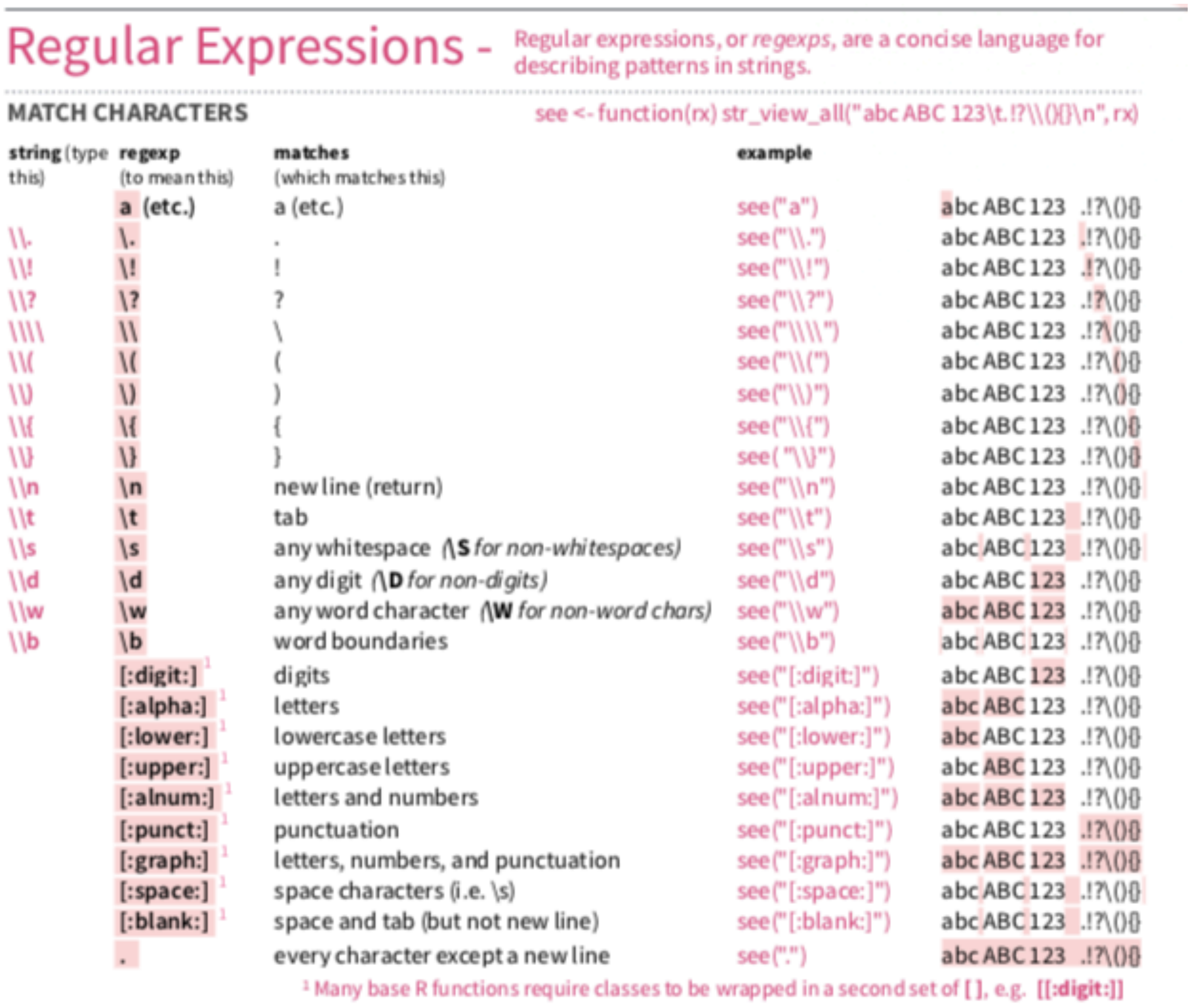
3. Google is a good teacher!
4. It is a good forum to ask questions and search for answers:

   https://stackoverflow.com/

5. Online courses are good options to learn R, no matter you are a beginner or advanced user!

Appendix

from cheat sheet produced by R studio…



Regular Expressions - Regular expressions, or *regexps*, are a concise language for describing patterns in strings.

MATCH CHARACTERS see <- function(rx) str_view_all("abc ABC 123\t.!?\\(){}\n", rx)

string (type this)	**regexp** (to mean this)	**matches** (which matches this)	**example**	
	a (etc.)	a (etc.)	see("a")	abc ABC 123 .!?\(){}
\\.	\.	.	see("\\.")	abc ABC 123 .!?\(){}
\\!	\!	!	see("\\!")	abc ABC 123 .!?\(){}
\\?	\?	?	see("\\?")	abc ABC 123 .!?\(){}
\\\\	\\	\	see("\\\\")	abc ABC 123 .!?\(){}
\\(	\(	(	see("\\(")	abc ABC 123 .!?\(){}
\\)	\)	)	see("\\)")	abc ABC 123 .!?\(){}
\\{	\{	{	see("\\{")	abc ABC 123 .!?\(){}
\\}	\}	}	see("\\}")	abc ABC 123 .!?\(){}
\\n	\n	new line (return)	see("\\n")	abc ABC 123 .!?\(){}
\\t	\t	tab	see("\\t")	abc ABC 123 .!?\(){}
\\s	\s	any whitespace (**\S** *for non-whitespaces*)	see("\\s")	abc ABC 123 .!?\(){}
\\d	\d	any digit (**\D** *for non-digits*)	see("\\d")	abc ABC 123 .!?\(){}
\\w	\w	any word character (**\W** *for non-word chars*)	see("\\w")	abc ABC 123 .!?\(){}
\\b	\b	word boundaries	see("\\b")	abc ABC 123 .!?\(){}
	[:digit:][1]	digits	see("[:digit:]")	abc ABC 123 .!?\(){}
	[:alpha:][1]	letters	see("[:alpha:]")	abc ABC 123 .!?\(){}
	[:lower:][1]	lowercase letters	see("[:lower:]")	abc ABC 123 .!?\(){}
	[:upper:][1]	uppercase letters	see("[:upper:]")	abc ABC 123 .!?\(){}
	[:alnum:][1]	letters and numbers	see("[:alnum:]")	abc ABC 123 .!?\(){}
	[:punct:][1]	punctuation	see("[:punct:]")	abc ABC 123 .!?\(){}
	[:graph:][1]	letters, numbers, and punctuation	see("[:graph:]")	abc ABC 123 .!?\(){}
	[:space:][1]	space characters (i.e. \s)	see("[:space:]")	abc ABC 123 .!?\(){}
	[:blank:][1]	space and tab (but not new line)	see("[:blank:]")	abc ABC 123 .!?\(){}
	.	every character except a new line	see(".")	abc ABC 123 .!?\(){}

[1] Many base R functions require classes to be wrapped in a second set of [ ], e.g. **[[:digit:]]**