Class 6: R Functions

Rimma Levina

1/24/2020

## R Markdown

This is regular text that does not need # like before like in an R script

## This is a level 2 heading

## This is a level 3 heading

**BOLD** will make whatever is between the asterisk bold once you press KNIT (word, pdf) *Will* make things italics

List w bullets (needs returns)

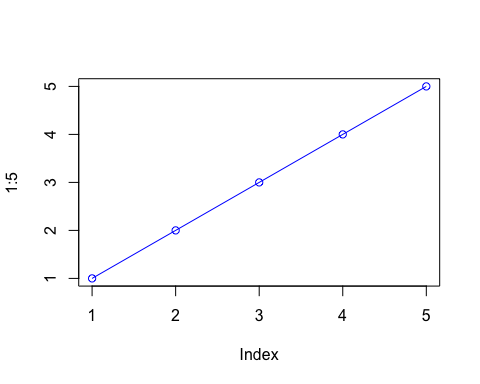
* 1
* 2
* 3

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

``` {} anything between this will be read as code, called a code chunk, press insert up there and chose the kind of code you want to work w: press play button to run code –>

plot(1:5, col = "blue", type = "o")

 the code will be in the document shortcut: option command i:

x <- 1:10

``` {python} tells the computer what type of code it needs to read

summary(cars)

## speed dist   
## Min. : 4.0 Min. : 2.00   
## 1st Qu.:12.0 1st Qu.: 26.00   
## Median :15.0 Median : 36.00   
## Mean :15.4 Mean : 42.98   
## 3rd Qu.:19.0 3rd Qu.: 56.00   
## Max. :25.0 Max. :120.00

here is my analysis of your new data. It looks ok, the mean of your data is 5.5 <– this is code , could also be python mean(x) when you knit it it runs and prints the answer to that code

There are templates that can help you format what you want your document to look like, class website is created in here

## Including Plots

You can also embed plots, for example:



Note that the echo = FALSE parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Lets insert a code chunk with the shortcut Option-CMD-i

read.table("test1.txt")

## V1  
## 1 Col1,Col2,Col3  
## 2 1,2,3  
## 3 4,5,6  
## 4 7,8,9  
## 5 a,b,c

x <- read.table("test1.txt")  
x

## V1  
## 1 Col1,Col2,Col3  
## 2 1,2,3  
## 3 4,5,6  
## 4 7,8,9  
## 5 a,b,c

separated by commas, so use read. csv = comma separated values our you could use read.table and say sep = “,”

x <- read.csv("test1.txt")  
x

## Col1 Col2 Col3  
## 1 1 2 3  
## 2 4 5 6  
## 3 7 8 9  
## 4 a b c

y <- read.table("test2.txt", header = TRUE, sep = "$")  
y

## Col1 Col2 Col3  
## 1 1 2 3  
## 2 4 5 6  
## 3 7 8 9  
## 4 a b c

z <- read.table("test3.txt")  
z

## V1 V2 V3  
## 1 1 6 a  
## 2 2 7 b  
## 3 3 8 c  
## 4 4 9 d  
## 5 5 10 e

## OUr first function

This is an example function named add with input x and y

add <- function(x,y=1) {  
 #sum the input x and y  
 x + y  
}

Lets use it

add(4)

## [1] 5

add(5,6)

## [1] 11

#here you define y

go crazy

add(x=c(3,6,1), 4)

## [1] 7 10 5

Rescale function

rescale <- function(x) {  
 rng <-range(x, na.rm = TRUE)   
 (x - rng[1]) / (rng[2] - rng[1])  
}

Test this function

rescale(c(1:10))

## [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667  
## [8] 0.7777778 0.8888889 1.0000000

rescale(c(1:10, NA))

## [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667  
## [8] 0.7777778 0.8888889 1.0000000 NA

even though there is a missing value, we took care of that by using na.rm=TRUE

rescale(c(1:10, NA))

## [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667  
## [8] 0.7777778 0.8888889 1.0000000 NA

## Rescale exercise 3

rescale3 <- function(x, na.rm=TRUE, plot=FALSE) {  
 if(na.rm) {  
 rng <-range(x, na.rm=na.rm)  
 } else {  
 rng <-range(x)  
 }  
 print("Hello")  
 answer <- (x - rng[1]) / (rng[2] - rng[1])  
 print("Please don't sing again!")  
 if(plot) {  
 plot(answer, typ="b", lwd=4)  
 }  
 print("I can see it in ...")  
}