R - Beginners Workshop Handout

Clinical Research Training Program

Ria Pinjani

Dr. Amar Ahmad

03/02/2020

The purpose of this handout is to give you a heads up for what you should expect in your R - Beginners Workshop. For the duration of the workshop we will be covering the following topics;

1. Introducing R & RStudio
2. R Basics
3. Importing, viewing and describing datasets in R
4. t-Test, Chi Squared test and Linear Regression in R
5. Basic Visualization

## 1. Introducing R & RStudio

### What is R?

R is a programming language for statistical computing and graphics. It’s open source, which means that it is free and open to the public.

### R & RStudio

RStudio is a integrated development environment for R. The inferface enables users to view graphs, data tables, R code and output all at the same time. Below is an image of how the R Studio interface looks like.

### Installing R & Rstudio

For the purposes of this workshop, R & Rstudio will already be installed on your PCs. To download it on your personal computers, You can use the links below.

* Download and install R from <http://cran.r-project.org>
* Download and install RStudio from <https://www.rstudio.com/products/rstudio/download/#download>

(You must install R before you can install RStudio)

### What are R packages

R packages are collections of functions and data sets developed by the community. They increase the power of R by improving existing base R functionalities, or by adding new ones. For the purpose of this workshop we will be using functions in base R for the most part. We will make use of the package “readr” to demonstrate importing and saving datasets in R. Moreover we will be using the package “utils”

### Getting Started

Once you open RStudio, you can create a new script by performing the following steps;

File –> New File –> R Script

## R Basics

### R as a calculator

Type the following on your R Script

3 + 1

This is what you will get on your Console when you Run the line above;

## [1] 4

You can take this a step further;

((3+1)^2)/(5\*0.5)

## [1] 6.4

### Creating a variable

In R, we can create a variables and then perform arithmetic calculations on them. Lets say we want to create a variable for the number 10. To create this variable, we can type the below line of code. Notice that the assignment operator “<-”, which consists of the tow characters “<” (“less than”) and “-” (minus) occuring strictly side by side and it “points” to the object recieving the value of the expression. Also keep in mind that we chose to name the variable “ten” here, we could have named it “x”, “y”, “TEN” etc. Once you run the below line of code, the variable will be saved to your Environment.

ten <- 10  
ten/5

We can also create this variable by using the assignment operator “=”

ten = 10  
ten/5

This is what you will get on your Console when you Run either of the lines above;

## [1] 2

Suppose you want to create a list for the cups of coffee you had everyday for the past week. You can do this by using the “c” function. The c(…) function combines one or more values into a vector. It is always a good idea to name your vector in a way that represents the information in this vector. Again, the variable coffee\_pastweek will be saved to your Environment once you run the line below.

coffee\_pastweek <- c(2,3,2,1,4,1,1)

You can make your code more informative and clear by adding a comment! To add a comment simply type # and insert your text after. Let’s add a comment to the above code.

# cups of coffee consumed in the past week (17/02/2020 -23/02/2020) from monday to sunday  
coffee\_pastweek <- c(2,3,2,1,4,1,1)

Adding comments to your code helps you, and anyone reading your code understand it better.

You can view the contents of your vector by simply typing out the name of your vector in Souce. The contents will be displayed on your Console.

coffee\_pastweek

## [1] 2 3 2 1 4 1 1

You can also choose to view a single item in your vector. Let’s suppose you want to look at the first element in your vector. You can do this by indexing;

coffee\_pastweek[1]

## [1] 2

### Arithmetic Functions

R has biult in functions that you can use. These functions can take one or more arguments. Multiple arguments are seperated by commas.

The sqrt function allows you to take the square root of number.Let us take a look at the value we get in the console when we take a square roof of our variable ten.

sqrt(ten)

## [1] 3.162278

Here are some more examples of the many arithmetic functions one can use.

This provides a sum of all the elements in your vector coffee\_pastweek, therefore 14 is the total cups of coffee for the entire week.

sum(coffee\_pastweek)

## [1] 14

This provides a sum of all the elements in your vector coffee\_pastweek and 1.

sum(coffee\_pastweek, 1)

## [1] 15

This provides an average of all the elements in your vector, therefore 2 is the average cups of coffee per day.

mean(coffee\_pastweek)

## [1] 2

This gives us the standard deviation;

sd(coffee\_pastweek)

## [1] 1.154701

This gives us the range (minimum and maximum value in our vector);

range(coffee\_pastweek)

## [1] 1 4

Besides these, there are numerous other arithmetic and statistical functions in R that one can use.

### Logical Operators

The table below presents logical vectors than can be used in R. These can be used along with & (and) / | (or).

|  |  |
| --- | --- |
| Logical Operator | Definition |
| == | equal to |
| != | not equal to |
| > | more than |
| >= | more than or equal to |
| < | less than |
| <= | less than or equal to |
| %in% | contains |

Below are some examples of how they can be put into use;

When you drink more than 1 cup of coffee in a day in the past week.

coffee\_pastweek > 1

The ouput in the console is shown below. This indicates that you drank more than one cup of coffee on four out of seven days in the past week. (The result displays TRUE four out of seven times)

## [1] TRUE TRUE TRUE FALSE TRUE FALSE FALSE

When did you drink 2 cups of coffee in a sigle day?

coffee\_pastweek == 2

## [1] TRUE FALSE TRUE FALSE FALSE FALSE FALSE

The use of these logical operators will you put further into use when carrying out simple data manupilations using the dplyr package in R.

## Constants

|  |  |
| --- | --- |
| Constant | Definition |
| pi | The number π = 3.141593 |
| Inf, -Inf | ∞, -∞ |
| NA | Not Available: Missing values |
| NaN | Not a Number: e.g. 0/0 |
| NULL | Empty set |

## Importing, viewing and describing datasets in R

### Reading your data into R

To read data into R, You can go to File –> Import Dataset –> Choose file format.

Below are the many options that you will see on your screen.

Another way to read data into R is by running a line of code. To import a csv file, we can simply use the function “read\_csv” from the package readr and enter the file path in brackets. You must always remember to put the file path in quotation marks.

install.packages("readr")  
library(readr)  
  
dat <- read\_csv("MyH/MyFolder/Dat.csv")

For the purpose of this workshop we will be using data on Smoking, Alcohol and Oesophageal Cancer from the utils package in R. The dataset is called esoph. It is from a case-control study of oesophageal cancer in llle-et-Vilaine, France. We can save it in our global environment with the name “data” for simplicity sake.

install.packages("utils")  
library(utils)

dat <- esoph

## Display your data

You can use the function “View” to look at your data on a seperate tab in RStudio. Remember that R is case sensitive. Therefore when you run

view(dat)

Your console will show you the following error.

You have to instead type;

View(dat)

OR

You can simply type esoph into your Source and view the data in your Console.

## Describe your data

Let’s look at the number of rows and columns in our dataset. We can do so by using the functions “nrow” and “ncol”

nrow(dat)

## [1] 88

ncol(dat)

## [1] 5

Another way of doing this is using the “dim” function in R. This shows us number of rows, followed by the number of columns.

dim(dat)

## [1] 88 5

We now know that our data set has 88 rows and 5 columns/variables.

To take a look at what variables we have; we can use the function “colnames”

colnames(data)

## NULL

As often is the case, the variable names do not clearly indicate what information they contain. For the purposes of this dataset, we can simply run ?esoph in our Source. Data description will be displayed in the Help tab on the bottom right corner of RStudio.

Here is a description of the variables;

1. agegp - Age group
2. alcgp - Alcohol consumption
3. tobgp - Tobacco consumption
4. ncases - Number of cases
5. ncontrols - Number of controls

You can use the summary function in base R to summarize the variables in your dataset. You can do so for one or more variables. Here we use it to describe our entire dataset.

summary(dat)

This is what will be displayed in your console;

## agegp alcgp tobgp ncases ncontrols   
## 25-34:15 0-39g/day:23 0-9g/day:24 Min. : 0.000 Min. : 1.00   
## 35-44:15 40-79 :23 10-19 :24 1st Qu.: 0.000 1st Qu.: 3.00   
## 45-54:16 80-119 :21 20-29 :20 Median : 1.000 Median : 6.00   
## 55-64:16 120+ :21 30+ :20 Mean : 2.273 Mean :11.08   
## 65-74:15 3rd Qu.: 4.000 3rd Qu.:14.00   
## 75+ :11 Max. :17.000 Max. :60.00

The output shows that agegp, aclgp and tobgp are ordered factor/categorical variables. It also shows us the frequency for each group/level in all of these variables. for example; 15 individuals in our dataset belong to the age group 25-34 and so on.

ncases and ncontrols are numeric variables. The variable ncases has a mean equal to 2.273 whereas the the variable ncontrol has a mean equal to 11.08.

You can also choose to summarize only one variable in your dataset. How do you select one variable from your data set? You can do this by typing out data$variable\_name.

Let’s try below.

summary(dat$agegp)

## 25-34 35-44 45-54 55-64 65-74 75+   
## 15 15 16 16 15 11

You can also use the “class” function to see what type of variables you are dealing with.

class(dat$agegp)

## [1] "ordered" "factor"

class(dat$ncases)

## [1] "numeric"

## t-Test, Chi Squared test and Linear Regression in R

### t-Test

t.test(dat$ncases , mu = 0)

##   
## One Sample t-test  
##   
## data: dat$ncases  
## t = 7.7438, df = 87, p-value = 1.636e-11  
## alternative hypothesis: true mean is not equal to 0  
## 95 percent confidence interval:  
## 1.689387 2.856068  
## sample estimates:  
## mean of x   
## 2.272727

### Chi Squared tests in R

chisq.test(dat$ncases, dat$agegp)

## Warning in chisq.test(dat$ncases, dat$agegp): Chi-squared approximation may be  
## incorrect

##   
## Pearson's Chi-squared test  
##   
## data: dat$ncases and dat$agegp  
## X-squared = 96.827, df = 45, p-value = 1.169e-05

### Simple Linear Regression in R

dat$percent\_cases <- (dat$ncases\*100)/(dat$ncontrols)

summary(lm(percent\_cases ~ as.numeric(tobgp) + as.numeric(alcgp), data = dat))

##   
## Call:  
## lm(formula = percent\_cases ~ as.numeric(tobgp) + as.numeric(alcgp),   
## data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -67.727 -14.656 0.091 18.129 67.654   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -17.382 10.568 -1.645 0.104   
## as.numeric(tobgp) 3.587 2.864 1.252 0.214   
## as.numeric(alcgp) 17.690 2.857 6.192 2.03e-08 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 29.9 on 85 degrees of freedom  
## Multiple R-squared: 0.3159, Adjusted R-squared: 0.2998   
## F-statistic: 19.62 on 2 and 85 DF, p-value: 9.841e-08

## Basic Visualization

## Visualizing your data in base R

plot(dat$agegp, dat$percent\_cases)

## Saving your dataset

require(readr)

## Loading required package: readr

write\_csv(dat, 'my\_esoph.csv')

## Let’s Wrap up

We hope that the above information gets you more aquainted with the RStudio interface and helps you a get started with statistical analysis in R. We are going to wrap things up for the purpose of this handout. For the workshop; we will be running this code live and looking at the output, while using the same dataset.You are expected to run the same code on your own desktop as well - this is simply because you do learn R best by trying yourself! We will be helping you throughout the process.