

Introduction to R

Main objective <- Review the main features of R (remember there is a lot of documentation packaged in R itself!)

Berenica Vejvoda, Data Librarian

bvejvodauwindsor.ca

February 12, 2020

Collaboratory, Leddy Library, University of Windsor

Academic Data Centre:

Drop-In Hours: 12:30-4:30PM

By appointment: libdata@uwindsor.ca

Contents

Introduction to what is R?	3
Why R?	3
What is RStudio?	4
What is the R Console?	5
Basics of R	6
R as a Calculator	6
Entering and Manipulating Data in R	6
What is an R script?	6
Creating an R script	6
Comments in R	7
Setting working directory	7
Data types	8
Scaler	8
Vector	9
Character	10
Data Frames	11
Install R Packages	11
Load Installed R Packages	11
Loading Data in R	12
Viewing the data	13
More on Character Variables	14
Searching for Patterns of Text	14

Introduction to what is R?

R is an open source programming language and software environment for statistical computing and graphics that is supported by the R Foundation for Statistical Computing.

25 years old now!

Similar to the S language developed by Bell Labs in 1976 (43 years ago!). R is considered a different “implementation” of S. Unlike S (MathSoft) is open source!!!

Much of code written in R will run in S.

R packages give R its functionality

- 3500+ packages available for installation. (packages are similar to extensions or apps for other products like Google or Apple).
- Each package serves a purpose and has specific commands you can use.
- Packages are user-created programs which can be used to run a specific task or set of tasks.

R supports a very wide variety of statistical techniques:

- Linear modelling techniques such as multiple linear regression (MLR) (predictive modelling)
- Non-linear modelling techniques such as non-linear regression
- Classical statistical tests such as:
 - Nominal: Chi-Square (e.g. # of correct responses)
 - Ordinal: Factor Analysis
 - Interval or Ratio: t test, ANOVA (linear relationship between two continuous variables), Pearson’s r (linear relationship between two variables)
- Time series analysis
- Clustering (partitioning data into the same class e.g. text data mining)
- Graphical techniques – especially great if you need to create a large number of plots (very efficient compared to Excel)

Why R?

- Free
- Works on all operating systems
- Excellent and efficient graphical capacities
- Large and active community
- Many methods not available in other commercial software
- Good integration between programming language and statistical functions
- Good integration with a number of database systems and other languages

What is RStudio?

Also open source

Need to install R first

Can use R without R Studio but you can't use R Studio without R

RStudio is an IDE (integrated development environment) for R

Most generally, it can help you work more efficiently by providing further functionality

RStudio makes it easier to:

- view datasets as spreadsheets
- see the results of your script (e.g. plot)
- change your plot in the plot display without re-running your code (e.g. size)
- install packages (collections of R functions, data and compiled code)
- set your working directory and access your files

The screenshot displays the RStudio integrated development environment. The main window shows a data table with the following columns: Service.Request.Description, Department, Method.Received, and Crea. The table contains six rows of data, all with 'Pothole' in the first column. The Department column lists 'Contracts, Field Services & Maintenance' for all entries. The Method.Received column lists 'E-Mail' for the first three rows and 'Phone' for the last three. The Crea column shows the number '2' for all entries. Below the table, it indicates 'Showing 1 to 7 of 1,714 entries, 8 total columns'. The console pane at the bottom left shows the following R code and output:

```
> Pothole_YTD <- read.csv("C:/collaboratory_workshops/data/Pothole_YTD.csv")
> View(Pothole_YTD)
> |
```

The environment pane on the right shows the 'Global Environment' with a data object 'Pothole_YTD' containing '1714 obs. of 8 variables'. The bottom pane shows the 'Viewer' tab with the documentation for the 'rgdal' package, titled 'Bindings for the 'Geospatial' Data Abstraction Library'.

What is the R Console?

There are two ways of interacting with R:

1. Console
2. Script files (plain text files that contain code)

The console window (in RStudio, the bottom left panel) is the place where R is waiting for you to tell it what to do. Once you run the command, the console will show the results of a command. You can type commands directly into the console, but they will be forgotten when you close the session.

It is better to enter the commands in the script editor and save the script. This way, you have a complete record of what you did. That is you can easily show others how you did it and you can do it again later on if needed. You can copy-paste into the R console, but the RStudio script editor allows you to 'send' the current line or the currently selected text to the R console using the Ctrl+Return shortcut.

Table 1: Summary of RStudio Windows/Tabs

R Studio Windows/Tabs	Location	Description
Console Windows	lower-left	location where commands are entered and the output is printed
Source Tabs	upper-left	built-in text editor
Environment Tab	upper-right	interactive list of loaded R objects
History Tab	upper-right	list of key strokes entered into the Console
Files Tab	lower-right	file explorer to navigate C drive folders
Plots Tab	lower-right	output location for plots
Packages Tab	lower-right	list of installed packages
Help Tab	lower-right	output location for help commands and help search window
Viewer Tab	lower-right	advanced tab for local web content

Table 2: Commonly Used Keyboard Shortcuts in Console

Keyboard Shortcut	Action
Ctrl + L	Clearing Console window
Up/Down	Navigate command history
Esc	Interrupt currently executing command
Ctrl + 1	Move focus to the Console
Ctrl + 2	Move focus to the Source Editor

Basics of R

R as a Calculator

```
Console Terminal x Jobs x
C:/collaboratory_workshops/data/
> 2+3
[1] 5
> pi
[1] 3.141593
> 2+3*pi
[1] 11.42478
> log(2+3*pi)
[1] 2.435785
> exp(2.435785)
[1] 11.42478
> 6/7
[1] 0.8571429
> |
```

Entering and Manipulating Data in R

In R you have variables, functions and arguments

A key operator which is used to assign the value on the right to the variable on the left

can use `<-` or `=` (also: ALT +- for shortcut keys; Option +- for Mac)

What is an R script?

An R script is a plain text file that you save R code in. Scripts are a good way to keep track of what you are doing. They allow you to a reproducible record of your work which you can run at a later date. In RStudio, you can run one command at a time or a series of commands at a time in the script file.

Creating an R script

You can create a R script in RStudio by going to File > New File > R script in the menu bar. RStudio will open a fresh script above the console pane. To save the R script go to File > Save As in the menu bar.

It is important to start your R script with a header which is an annotated description of what the code does. The header will include the creator name, creator email, script name, script description and date of creations. Let's create a header for our R script file.

```
## -----
## ##
```

Script name: FirstRScript.R

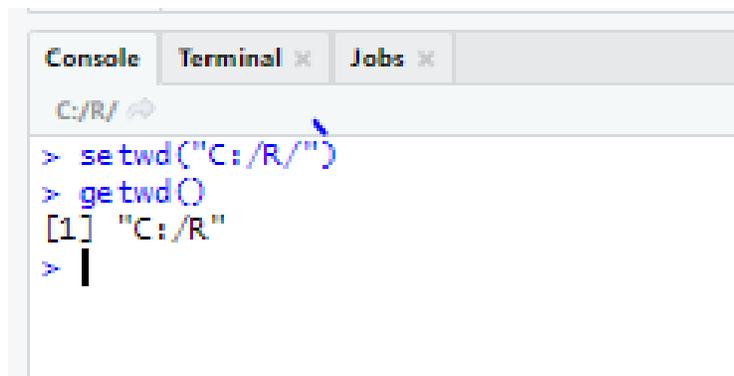
```
##  
## Purpose of script: This is the code I ran during Intro R  
##  
## Author:  
##  
## Date Created: 2019-12-04  
## ## Email:  
##  
## -----  
##  
## Notes:  
## ## ## -----
```

Comments in R

When R sees the # as the start of a line, it indicates the line is a comment. The comment continues from the initial # until the end of the line on which that occurs and no further.

You can comment and uncomment entire selections of code clicking Code -> Comment/Uncomment Lines in the menu bar. As well, the key shortcut for Windows is Ctrl+Shift+C or for Macs Command+Shift+C keyboard shortcut.

Set Working Directory



```
Console Terminal x Jobs x  
C:/R/ ↗  
> setwd("C:/R/")  
> getwd()  
[1] "C:/R/"  
> |
```

Data types

Scalar (simple single numeric value such as 1, 2/3, 3.14)

```
Console Terminal x Jobs x
C:/collaboratory_workshops/data/
> a <- 10
> a
[1] 10
> b <- 5
> b
[1] 5
> b-a
[1] -5
> b/a
[1] 0.5
> |
```

Vector (is simply a series of variables)

To combine values into a vector use the “c ()” function.

```
Console Terminal x Jobs x
C:/collaboratory_workshops/data/ ↗
> x <- c(2,5,8,9,7,4,6,8)
> x
[1] 2 5 8 9 7 4 6 8
> y <- c(9,6,4,7,4,6,8,9,4,3,2)
> y
[1] 9 6 4 7 4 6 8 9 4 3 2
> x[7]
[1] 6
> y[7]
[1] 8
> x[1:4]
[1] 2 5 8 9
> y[y>4]
[1] 9 6 7 6 8 9
> |
```

You can also apply functions to vectors.

```
C:/collaboratory_workshops/data/ ↗  
> x <- c(2,5,8,9,7,4,6,8)  
> x  
[1] 2 5 8 9 7 4 6 8  
> y <- c(9,6,4,7,4,6,8,9,4,3,2)  
> y  
[1] 9 6 4 7 4 6 8 9 4 3 2  
> x[7]  
[1] 6  
> y[7]  
[1] 8  
> x[1:4]  
[1] 2 5 8 9  
> y[y>4]  
[1] 9 6 7 6 8 9  
> length(y)  
[1] 11  
> length(x)  
[1] 8  
> mean(x)  
[1] 6.125  
> mean(y)  
[1] 5.636364  
> var(x)  
[1] 5.553571  
> sqrt(var(x))  
[1] 2.356602
```

And summary statistics

summary () mean, median, 25th and 75th quartiles, min, max

```
> summary(x)  
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   
 2.000  4.750   6.500   6.125  8.000   9.000   
> summary(y)  
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   
 2.000  4.000   6.000   5.636  7.500   9.000   
> |
```

Character variables

```

Source

Console Terminal x Jobs x
C:/collaboratory_workshops/data/ ↗
> walks_everyday <- c("yes", "no", "yes", "no", "no", "yes")
> walks_everyday
[1] "yes" "no" "yes" "no" "no" "yes"
> |

```

Data Frames

Most data are data frames (columns and rows)

	Service.Request.Description	Department	Method.Received	Created.Date
1	Pothole	Contracts, Field Services & Maintenance	E-Mail	2019-02-08 14:00
2	Pothole	Contracts, Field Services & Maintenance	E-Mail	2019-02-12 9:20
3	Pothole	Contracts, Field Services & Maintenance	E-Mail	2019-02-11 15:00
4	Pothole	Contracts, Field Services & Maintenance	Phone	2019-02-14 10:00
5	Pothole	Contracts, Field Services & Maintenance	Phone	2019-02-06 15:00

Install R Packages

We need either the readr or rcpp package installed to load a csv (if done by command) so let's install "readr". This will take a few minutes.

```

Console Terminal x Jobs x
C:/collaboratory_workshops/data/ ↗
> install.packages("readr")

```

Load Installed R Packages

Now we need to load the installed package into R from the /library location.

```
Console Terminal x Jobs x
C:/collaboratory_workshops/data/ ↗
> library(readr)|
```

Loading Data in R

Read a file in table format and create a data frame from it.

Note that there are different functions for different data formats.

Note also the argument “header = TRUE” which holds if the file contains the names of the variables as its first line.

Will use a .csv file found here: <https://opendata.citywindsor.ca/opendata/details/151>.

```
Console Terminal x Jobs x
C:/collaboratory_workshops/data/ ↗
> potholes_windsor <- read.csv(file="Pothole_YTD.csv", header = TRUE)|
```

Viewing the data

To display the data as a spreadsheet in the data viewer window:

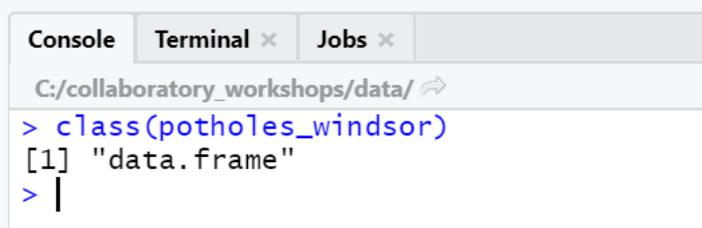
```
> view(potholes_windsor)
> |
```

To list just the variables:

```
> ls(potholes_windsor)
[1] "Block.Address"          "Created.Date"
[3] "Department"            "Method.Received"
[5] "Service.Request.Description" "Street"
[7] "Sub.Type"              "Ward"
> |
```

OR: potholes\$ (variables will display in “environment window”)

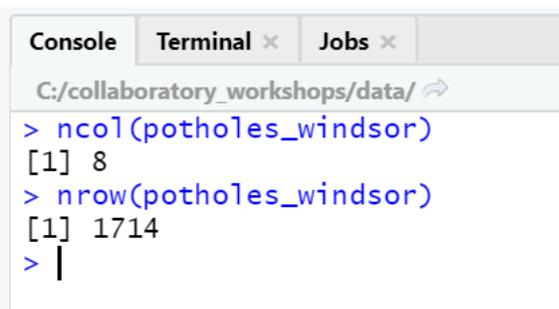
To display the class of an object (i.e. data type):



The screenshot shows a R console window with tabs for 'Console', 'Terminal x', and 'Jobs x'. The current directory is 'C:/collaboratory_workshops/data/'. The command executed is `> class(potholes_windsor)`, which returns `[1] "data.frame"`.

```
> class(potholes_windsor)
[1] "data.frame"
> |
```

To display number of columns and rows:



The screenshot shows a R console window with tabs for 'Console', 'Terminal x', and 'Jobs x'. The current directory is 'C:/collaboratory_workshops/data/'. The commands executed are `> ncol(potholes_windsor)` returning `[1] 8` and `> nrow(potholes_windsor)` returning `[1] 1714`.

```
> ncol(potholes_windsor)
[1] 8
> nrow(potholes_windsor)
[1] 1714
> |
```

More on Character Variables

Data frames automatically convert character(string) variables to factors – i.e. each unique string variable is assigned a unique number. This becomes problematic when you want to treat string as string. Sometimes a string is just a string!

```
Console Terminal x Jobs x
C:/collaboratory_workshops/data/ ↗
> class(potholes_windsor$Street)
[1] "factor"
> is.character(potholes_windsor)
[1] FALSE
> |
```

One solution with this dataset is to re-import it using a `stringsAsFactors` argument. `stringsAsFactors` is an argument specific to the `data.frame` function.

```
Console Terminal x Jobs x
C:/collaboratory_workshops/data/ ↗
> potholes_windsor_string <- read.csv(file = "Pothole_YTD.csv", header = TRUE,
  stringsAsFactors = FALSE)
> View(potholes_windsor_string)
> is.character(potholes_windsor_string$Street)
[1] TRUE
> |
```

Searching for Patterns of Text

The **grep** and **grepl** functions use regular expressions (or literal values) as patterns to conduct pattern matching on a character vector.

The **grep** returns indices of matched items or matched items themselves while **grepl** returns a logical vector with TRUE to represent a match **and** FALSE otherwise.

For more sophisticated text data mining you might be interested in checking out the “tm” package.

```
Console Terminal x Jobs x  
C:/collaboratory_workshops/data/ ↗  
> length(grep("UNIVERSITY AVE W", potholes_windsor_string$Street))  
[1] 25  
> |
```