R Workshop - Session 1

#### ***Chantal Koechli***

#### ***2 February 2019***

* [Learning/Skills Objectives](#gjdgxs)
* [What are R and RStudio?](#30j0zll)
* [Why R and RStudio?](#1fob9te)
* [Downloading/Installing R and Rstudio](#3znysh7)
* [Overview of RStudio Console](#2et92p0)
* [General R helpful hints](#tyjcwt)
* [Creating a dataframe](#3dy6vkm)
  + [Let’s practice](#1t3h5sf)
* [Reading in datasets](#4d34og8)
  + [Working directories and paths to data](#2s8eyo1)
  + [Reading in .txt files](#17dp8vu)
  + [Reading in .csv files](#3rdcrjn)
  + [Reading in Excel files (.xls, .xlsx)](#26in1rg)
  + [Reading in files stored on Google Drive](#lnxbz9)
  + [Reading in files via URL (can also be used with Google Drive)](#35nkun2)
  + [Reading in files formatted for other statistical software](#1ksv4uv)
  + [Let’s practice](#44sinio)
* [Troubleshooting](#2jxsxqh)
* [Practice : RSwirl](#z337ya)
* [Teaching R](#3j2qqm3)

Learning/Skills Objectives

1. Describe what R and RStudio can do

* Data import (Session 1)
* Data tidying (Session 1)
* Data visualization (Session 2)
* Statistical testing (Session 3)
* Reports/Presentations (optional, examples of student work)

2. Install R and RStudio

3. Explain RStudio console

4. Read in datasets multiple ways

5. Begin tidying/transforming/summarizing data sets (tutorials)

6. Identify resources for troubleshooting in R

7. Practice, practice, practice using swirl (homework!)

What are R and RStudio?

* R is a flexible and powerful statistics and visualization system.
* RStudio is an integrated development environment (IDE) for R.
  + Code editor
  + Debugging
  + Ability to see plots and datasets
  + Access code history
  + Reproducible analysis; workspace management

Why R and RStudio?

* Flexible and powerful statistics and visualization software
* Reproducible data analysis
* Does not cost anything (freeware)
  + Huge community of help, troubleshooting, learning
* Facilitates collaboration
* Adopted as key tool in many fields (graduate school, industry - data science)

Downloading/Installing R and Rstudio

To install and run Rstudio, base R needs to be installed first.

**Installing R**

1. Navigate to <https://cran.r-project.org/>
2. Click the “Download R for…” that matches your operating system.
3. **If** Mac OS X :  
     
   Click on the .pkg file relevant to your current OS (10.11 or higher vs. 10.9 or higher vs…)  
     
   **Else** :  
     
   Click the blue highlighted “install R for the first time”
4. After package download, standard installer package will run to walk through the installation.

**Installing RStudio**

1. Navigate to <https://www.rstudio.com/products/rstudio/download/#download>
2. Click “Download RStudio”
3. Click “Download” under “RStudio Desktop Open Source License”
4. Click on the Installer relevant to your operating system
5. Installer should install RStudio automatically

**Installing libraries**

R has base packages that come automatically installed. However, there are many additional packages that can be installed, as needed. One set of packages that we will be using today is called the tidyverse - a set of packages that allow for easy import, tidying, and transformation of data sets.

To install a package or set of packages, type the following into your R console

install.packages("tidyverse")

Note that R will tab complete commands!

This download will take a little time, as we are downloading a collection of packages.

To load a specific library for use, use the library() command. For instance :

library(ggplot2) #ggplot2 is a data visualization package we will use next session

Overview of RStudio Console

You can run R in the R console, but RStudio has some aspects that are more user-friendly and functional.

**Bottom left quadrant of screen :**

* *Console* - area where R code is input and output results are shown. Any code typed here is forgotten after the R session is closed.
* *Terminal* - access to your OS system shell (like Terminal in Mac OS X). Allows you to set your working directory, use text editors, and implement other command line operations. (I don’t use this very often)

**Top left quadrant of screen :**

* Files for code editing (R script, R Markdown)
  + Walk through example of both

**Top right quadrant of screen :**

* *Environment tab* - shows what is contained in your global environment (ex. what data files you’ve read in, variables that are named, etc.)
  + Can click on this data to see dataset
  + Can import data via “Import Dataset”
  + Broom clears objects from workspace (would need to re-load data)
* *History tab* - shows the history of commands executed during the R session
* *Git* - connection to Github, a version control service.
* *Connections* - connection to existing data sources/databases. Can be a collaborative tool.

**Bottom right quadrant of screen :**

* *Files* - shows files contained in current working directory
  + Can create new folder and delete or rename files
  + ‘More’ provides additional options, including the setting of the working directory
* *Plots* - visualizes plots from code
* *Packages* - list of packages that have been installed. Checkmarks indicate that the package is loaded for use in your R session. You can click on the box of the relevant package to load or unload it)
* *Help* - displays documentation relevant to specific commands or packages.
* *Viewer* - allows viewing of local web content (ex. web graphics generated by specific R packages)

General R helpful hints

* Avoid spaces in filenames and in naming variables
* In naming variables, <- operator is traditionally used in R. You can equivalently use = operator.
* Read error messages - they often have good information!
* A # operator in front of words signifies to R to not run that code - this can be used for annotating code or for troubleshooting
* Following that, annotate your code! You may think you will remember what the code was for…you will not. Annotation will also help reinforce your coding knowledge and models best practices for data science.
* To run R code in Markdown or in an Rscript, hit control + enter

Creating a dataframe

Dataframes are easily created in R. A dataframe stores a table (two-dimensional array-like structure) in which each column contains values of one variable and each row contains one set of values for each column.

Dataframes are built as follows :

Name <- c("Ned", "Sally", "Georgette")

Name is the first column. Ned, Sally, and Georgette are the values associated with the Name column. The type of variable used to represent name values is a “character”.

Height <- c(67, 60, 72)

Height is the second column. 67,60, 72 are the values associated with the Height column. As the first values of the first two columns correspond to the same row of data, 67 represents the height of Ned. The type of variable used to represent height values is a numeric. Numbers in R are generally treated as numeric (1.00), rather than integers (1), but you can explicitly specify a variable to be represented as integers : Height = as.integer(c(67, 60, 72))

Vitamin <- c(TRUE, FALSE, TRUE)

Vitamin (short for whether study participants take a multivitamin) is the third column. Again, the first value of this column corresponds to the first value of the other two columns - therefore, Ned does take a vitamin. The type of variable used to represent vitamin values is a logical. True or False are the only values a logical can take.

df = data.frame(Name, Height, Vitamin)  
df

## Name Height Vitamin  
## 1 Ned 67 TRUE  
## 2 Sally 60 FALSE  
## 3 Georgette 72 TRUE

The data.frame function merges the different columns into a dataframe.

Let’s practice

Ken’s class uses the following datasets during the semester. Practice creating a dataframe (no need to do the analysis) for both datasets :

1. Concentrations of nitrogen oxides and of hydrocarbons (µg/m3) were determined in a certain urban area on 11 successive days. Test the null hypothesis that both classes of air pollutants were present in the same concentration.

|  |  |  |
| --- | --- | --- |
| Day | N oxides | Hydrocarbons |
| 1 | 104 | 108 |
| 2 | 116 | 118 |
| 3 | 84 | 89 |
| 4 | 77 | 71 |
| 5 | 61 | 66 |
| 6 | 84 | 83 |
| 7 | 81 | 88 |
| 8 | 72 | 76 |
| 9 | 61 | 68 |
| 10 | 97 | 96 |
| 11 | 84 | 81 |

1. The following data are weights of food (kg) consumed per day by adult deer at different times of the year. Test the null hypothesis that food consumption is the same for all months tested.

Feb: 4.7, 4.9, 5.0, 4.8, 4.7

May: 4.6, 4.4, 4.3, 4.4, 4.1, 4.2

Aug: 4.8, 4.7, 4.6, 4.4, 4.7, 4.8

Nov: 4.9, 5.2, 5.4, 5.1, 5.6

Reading in datasets

If your dataset is already prepared in a file (ex. .xlsx, .csv, .sav, .dat, etc), it is quite easy to read the data into R, without needing to re-create the dataframe.

Working directories and paths to data

Prior to reading in data, it helps to know where R will be looking for files to read in. All the other software on your computer does this as well (ex. you need navigate to the correct folder for attaching a file to your email), but the process is abstracted via clicking on folders/files. In R, the file path needs to be written out in code instead of clicking!

For instance, if I wanted to read in or load a file from my desktop, I would use the absolute file path :

"/Users/chantalkoechli/Desktop/FILENAME.csv"

Note that R will auto-fill the directory path.

If you are on a Windows operating system, your path would be slightly different :

"C:/Users/chantalkoechli/Desktop/FILENAME.csv"

This is an absolute file path because it directs R to the file from the root directory (the base directory/tree trunk of the computer file system). The symbol for the root directory is /. Thus, the file path above goes from the root directory, to the Users directory (contained within the root directory), to the chantalkoechli directory (contained within the Users directory), to the Desktop directory (contained within the chantalkoehchli directory), to FILENAME.csv (contained within the Desktop directory).

You can easily find filepaths for specific files by right-clicking the file and selecting “Get Info” or “Properties”.

Relative filepaths can also be provided, if your working directory is already set to a directory other than the root directory. For instance, my current working directory is /Users/chantal.koechli/Desktop/Rcourse, which can be seen as the Rcourse folder contained on my desktop. This means that R will look for files in my Rcourse directory, unless I tell it to look somewhere else. So I can type example\_data.csv, and if example\_data.csv exists in the Rcourse folder, R will find it. You can set your working directory under the “Files” tab, or using the command setwd("/path/to/directory")

Using absolute filepaths is the most foolproof method for reading files in, but using relative directories is faster.

Now, we’ll see how various files can be read into R. There’s a handy walk-through in more detail, for more filetypes, here : <https://www.datacamp.com/community/tutorials/r-data-import-tutorial>

Reading in .txt files

data <- read.table("/Users/chantalkoechli/Desktop/example.txt", header = TRUE)

Reading in .csv files

data <- read.csv("Users/chantalkoechli/Desktop/example.csv")

Reading in Excel files (.xls, .xlsx)

library(xlsx)  
data <- read.xlsx("Users/chantalkoechli/Desktop/example.xlsx")

Reading in files stored on Google Drive

library(googledrive)  
drive\_download("data/mlb2016.csv", type = "csv")  
data <- read.csv("mlb2016.csv")

Reading in files via URL (can also be used with Google Drive)

my.url = "https://docs.google.com/spreadsheets/d/e/2PACX-1vSkaiah-LcTs1aBi--L6M0hQ5cfhn4zSMw50Oij1GEEsAy4\_A-hWL4pUTlXNWz\_y18pS948AJkJaR9P/pub?output=csv"   
# link from Google Drive after publishing data to web  
  
data <- read.csv(url(my.url))

Reading in files formatted for other statistical software

**SPSS files**

# Activate the `foreign` library  
library(foreign)  
  
# Read the SPSS data  
mySPSSData <- read.spss("example.sav", to.data.frame=TRUE)

**Stata files**

# Activate the `foreign` library  
library(foreign)  
  
# Read Stata data into R  
mydata <- read.dta("<Path to file>")

**SASS**

# Activate the `sas7bdat` library  
library(sas7bdat)  
  
# Read in the SAS data  
mySASData <- read.sas7bdat("example.sas7bdat")

**Minitab**

# Activate the `foreign` library  
library(foreign)  
  
# Read the Minitab data  
myMTPData <- read.mtp("example2.mtp")

Let’s practice

Using one of the methods illustrated above, or outlined in the tutorial cited above, try loading at least three data files of interest into R from your computer. If you want to load in a file type that is not specified in any resources, ask Google or Chantal.

Troubleshooting

One of R’s greatest strengths is that it is freeware and has an enormous community of users that help each other out. Using Google to troubleshoot an error message, an issue, or a “how do I…” is my main go-to.

Stackoverflow is a Q&A site for programming and is also an excellent resource : <https://stackoverflow.com/questions/tagged/r>.

Finally, R itself is well documented. In the R console, using ?<package name/function/etc> will bring up the relevant help file in the “Help” viewer.

Practice : RSwirl

To practice some of the basic functions of R, the package swirl offers a hands-on, guided practice. Just install.packages("swirl"), then load swirl through library(swirl) to begin.

Teaching R

* Pre-class survey
* Show your vulnerability
* Coding as exercise in grit-building
* Emphasize career-broadening ability of coding and R
* Let students explore! Data analysis and visualization is an exercise in creativity!