Week 2 Demonstration

Get: Importing, Scraping and Exporting Data with R

Dr. Anil Dolgun

7/03/2018

Course Announcements

Course Announcements

- A few students are still trickling into the course. Please read the course information pack and let me know if you have any questions about the course.

2 / 28

Course Announcements

- A few students are still trickling into the course. Please read the course information pack and let me know if you have any questions about the course.
- Lecture recordings are available on Canvas Echo360.

Course Announcements

- A few students are still trickling into the course. Please read the course information pack and let me know if you have any questions about the course.
- Lecture recordings are available on Canvas Echo360.
- Our class time will be composed of four parts
  - Announcements and Questions (~ 5-10 mins)
  - Demonstration (~1 hr)
  - Class Activities (~ 1 hr, exercises on Class Worksheets)
  - Supervised self-directed learning (~ 1 hr, work on module skill builders and/or assignments)

2 / 28

Course Announcements

- A few students are still trickling into the course. Please read the course information pack and let me know if you have any questions about the course.
- Lecture recordings are available on Canvas Echo360.
- Our class time will be composed of four parts
  - Announcements and Questions (~ 5-10 mins)
  - Demonstration (~1 hr)
  - Class Activities (~ 1 hr, exercises on Class Worksheets)
  - Supervised self-directed learning (~ 1 hr, work on module skill builders and/or assignments)
- Answers to Week 1 Worksheet are available here

Course Announcements

- A few students are still trickling into the course. Please read the course information pack and let me know if you have any questions about the course.
- Lecture recordings are available on Canvas Echo360.
- Our class time will be composed of four parts
    - Announcements and Questions (~ 5-10 mins)
    - Demonstration (~1 hr)
    - Class Activities (~ 1 hr, exercises on Class Worksheets)
    - Supervised self-directed learning (~ 1 hr, work on module skill builders and/or assignments)
- Answers to Week 1 Worksheet are available here
- Answers to Module 1 Skill Builders are available here

Course Announcements

- A few students are still trickling into the course. Please read the course information pack and let me know if you have any questions about the course.
- Lecture recordings are available on Canvas Echo360.
- Our class time will be composed of four parts
    - Announcements and Questions (~ 5-10 mins)
    - Demonstration (~1 hr)
    - Class Activities (~ 1 hr, exercises on Class Worksheets)
    - Supervised self-directed learning (~ 1 hr, work on module skill builders and/or assignments)
- Answers to Week 1 Worksheet are available here
- Answers to Module 1 Skill Builders are available here
- If you haven't received the DataCamp invitation let me know.

Course Announcements

- A few students are still trickling into the course. Please read the course information pack and let me know if you have any questions about the course.
- Lecture recordings are available on Canvas Echo360.
- Our class time will be composed of four parts
    - Announcements and Questions (~ 5-10 mins)
    - Demonstration (~1 hr)
    - Class Activities (~ 1 hr, exercises on Class Worksheets)
    - Supervised self-directed learning (~ 1 hr, work on module skill builders and/or assignments)
- Answers to Week 1 Worksheet are available here
- Answers to Module 1 Skill Builders are available here
- If you haven't received the DataCamp invitation let me know.
- Assignment 1 details are available here. Due: End of Week 4, 11:59pm Sunday 25/03/2018.

Get: Importing, Scraping and Exporting Data with R

Module 2: Outline

- Reading Data from Text Files
  - Base R functions
  - `readr` package
- Reading Data from Excel files
  - `xlsx` package **
  - `readxl` package
- Importing Data from statistical software
  - `foreign` package
- Reading from Databases
  - export and then import
  - connect directly to a database from R
- Scraping Data from Web
  - Importing tabular and Excel files stored online (`read.csv` and `gdata`)
  - Scraping HTML Table Data (using `rvest`)
- Exporting Data to text files
  - Base R functions
  - `readr` package
- Exporting data to Excel files
  - `xlsx` package **
  - export to a csv file then save as xlsx
- Saving Data as an R object File

Preliminaries: Setting The Working Directory

- R is always pointed at a directory on your computer. You can find out which directory by running the `getwd` (get working directory) function.

```
getwd()
```

```
## [1] "/Users/anildolgun/Google Drive/Data Preprocessing/project-1/src/main/
```

- Remember that your working directory may be different from my path. The location of working directory will be the place that R stores everything.

Preliminaries: Changing The Working Directory

- There are a number of ways to change the current working directory:

Preliminaries: Changing The Working Directory

- There are a number of ways to change the current working directory:

```
setwd
```

Preliminaries: Changing The Working Directory

- There are a number of ways to change the current working directory:

  `setwd`

Let's create a folder on desktop and name it "   " and set this as our working directory.

```
setwd("~/Desktop/Week2")
```

Preliminaries: Changing The Working Directory

- There are a number of ways to change the current working directory:

  `setwd`

Let's create a folder on desktop and name it " " and set this as our working directory.

```
setwd("~/Desktop/Week2")
```

To check:

```
getwd()
```

Preliminaries: Changing The Working Directory

- There are a number of ways to change the current working directory:

  `setwd`

Let's create a folder on desktop and name it "   " and set this as our working directory.

```
setwd("~/Desktop/Week2")
```

To check:

```
getwd()
```

Remember, you must use the forward slash / or double backslash || in R while specifying the file path. The Windows format of single backslash will not work.

Preliminaries: Changing The Working Directory Cont.

7 / 28

Preliminaries: Changing The Working Directory Cont.

This will also change directory location of the Files pane.

Preliminaries: Changing The Working Directory Cont.

This will also change directory location of the Files pane.

Reading Data from Text Files

- Text files are a popular way to hold and exchange tabular data as almost any data application supports exporting data to the CSV (or other text file) formats.

Reading Data from Text Files

- Text files are a popular way to hold and exchange tabular data as almost any data application supports exporting data to the CSV (or other text file) formats.
- Text file formats use delimiters to separate the different elements in a line, and each line of data is in its own line in the text file.

Reading Data from Text Files

- Text files are a popular way to hold and exchange tabular data as almost any data application supports exporting data to the CSV (or other text file) formats.
- Text file formats use delimiters to separate the different elements in a line, and each line of data is in its own line in the text file.
- are the built-in functions that are already available when you download R and RStudio.

Reading Data from Text Files

- Text files are a popular way to hold and exchange tabular data as almost any data application supports exporting data to the CSV (or other text file) formats.
- Text file formats use delimiters to separate the different elements in a line, and each line of data is in its own line in the text file.
- are the built-in functions that are already available when you download R and RStudio.
- **`readr`** : The `readr` functions are around 10× faster. You need to install and load the `readr` package using the following commands:

Reading Data from Text Files

- Text files are a popular way to hold and exchange tabular data as almost any data application supports exporting data to the CSV (or other text file) formats.
- Text file formats use delimiters to separate the different elements in a line, and each line of data is in its own line in the text file.
- are the built-in functions that are already available when you download R and RStudio.
- **`readr`** : The `readr` functions are around 10× faster. You need to install and load the `readr` package using the following commands:

```
install.packages("readr")
library(readr)
```

Reading Data from Text Files: Base R functions

- For the demonstration, we will use the "iris.csv" data available in our Data Repository right click on here and save iris.csv data in your working directory.
- The following command will read iris.csv data and store it in the `iris` object in R as a data frame:

```
# iris.csv file is located in the working directory

iris <- read.csv( "iris.csv" )
```

Reading Data from Text Files: Base R functions

- For the demonstration, we will use the "iris.csv" data available in our Data Repository right click on here and save iris.csv data in your working directory.
- The following command will read iris.csv data and store it in the `iris` object in R as a data frame:

```
# iris.csv file is located in the working directory

iris <- read.csv( "iris.csv" )
```

- If the data file is under a different folder you need to specify the path to the data file explicitly:

```
#iris.csv file is located in the "~/Desktop/data/iris.csv" path

iris <- read.csv( file="~/Desktop/data/iris.csv" )
```

9 / 28

Reading Data from Text Files: `readr` package functions

- `readr` functions are around 10× faster. This will make a remarkable difference in reading time if you have a very large data set.
- `read_csv()` function is equivalent to base R 's `read.csv()` function (note the distinction between these two function names!)

```
install.packages("readr")
library(readr)
```

```
iris <- read_csv("iris.csv")
```

- `read_csv()` maintains the full variable name whereas, `read.csv` eliminates any spaces in variable names and fills it with '.'
- `read_csv()` automatically sets `stringsAsFactors = FALSE`, which can be a controversial topic.

Reading Data from Text Files: `readr` package functions

- RStudio has the built in " " dialog box on the upper-right " " pane.
- You can also use this dialog box to import a wide range of file types including csv, Excel, SPSS, SAS and Stata data files. The following slides (taken from Dr. James Baglin's R Bootcamp notes) will briefly explain the process of importing a csv data set into RStudio.

RStudio - Importing Data

Slide 1

Reading Data from Excel files: `xlsx` Package

- Excel is the most commonly used spreadsheet software. If the dataset is stored in the .xls or .xlsx format, we have to use certain R packages to import those files; one of the packages is `xlsx`.

Reading Data from Excel files: `xlsx` Package

- Excel is the most commonly used spreadsheet software. If the dataset is stored in the .xls or .xlsx format, we have to use certain R packages to import those files; one of the packages is `xlsx`.
- We will use the iris.xlsx data available here but first install and load the `xlsx` package:

```
install.packages(xlsx)
library(xlsx)
```

13 / 28

Reading Data from Excel files: `xlsx` Package

- Excel is the most commonly used spreadsheet software. If the dataset is stored in the .xls or .xlsx format, we have to use certain R packages to import those files; one of the packages is `xlsx`.
- We will use the iris.xlsx data available here but first install and load the `xlsx` package:

```
install.packages(xlsx)
library(xlsx)
```

- The `xlsx` package has external dependencies (i.e., rJava).
- Often installation and loading this package would be problematic.
- I recommend using `readxl` package instead.

Reading Data from Excel files: `readxl` Package

- `readxl` was developed by Hadley Wickham and the RStudio team who also developed the `readr` package.
- This package works with both .xls and .xlsx formats.
- Unlike `xlsx` package, the `readxl` package has no external dependencies (like Java or Perl).

```
install.packages(readxl)
library(readxl)
```

```
# read in xlsx worksheet using a sheet index or name

iris<- read_excel("../data/iris.xlsx", sheet = "iris")
```

- Help on arguments: Use package documentation.

14 / 28

Importing Data from statistical software

- The `foreign` package provides functions that help you read data files from other statistical software such as SPSS, SAS, Stata, and others into R.
- Here is an example of importing an SPSS data file called iris.sav:

```
install.packages("foreign")
library(foreign)
```

```
# read in spss data file and store it as data frame

iris_spss <- read.spss("../data/iris.sav", to.data.frame = TRUE)
```

Reading from Databases

- Large-scale data sets are generally stored in database software.

Reading from Databases

- Large-scale data sets are generally stored in database software.
- Commonly, large organisations and companies keep their data in relational databases. Therefore, we may need to import and process large-scale data sets in R.

Reading from Databases

- Large-scale data sets are generally stored in database software.
- Commonly, large organisations and companies keep their data in relational databases. Therefore, we may need to import and process large-scale data sets in R.
- One of the best approaches for working with data from a database is to export the data to a text file and then import the text file into R.

Reading from Databases

- Large-scale data sets are generally stored in database software.
- Commonly, large organisations and companies keep their data in relational databases. Therefore, we may need to import and process large-scale data sets in R.
- One of the best approaches for working with data from a database is to export the data to a text file and then import the text file into R.
- According to Adler (2010), importing data into R at a much faster rate from text files than you can from database connections, especially when dealing with very large data sets (1 GB or more).

Reading from Databases

- Large-scale data sets are generally stored in database software.
- Commonly, large organisations and companies keep their data in relational databases. Therefore, we may need to import and process large-scale data sets in R.
- One of the best approaches for working with data from a database is to export the data to a text file and then import the text file into R.
- According to Adler (2010), importing data into R at a much faster rate from text files than you can from database connections, especially when dealing with very large data sets (1 GB or more).
- This approach is considered to be the best approach if you plan to import a large amount of data once and then analyse.
- However, if you need to produce regular reports or to repeat an analysis many times, then it might be better to import data into R directly through a database connection.
- Database connection is an advanced topic, for more information refer to the Module 2 notes

16 / 28

Importing Tabular files Stored Online

- Vast amount of information is now being stored online, both in structured and unstructured forms. The most basic form of getting data from online is to import tabular (i.e. . txt , .csv) files that are being hosted online.
- Importing tabular data is common for the government data available online like Domestic Airlines On Time Performance
- Reading online .csv or .txt file is just like reading tabular data. The only difference is, we need to provide the URL of the data instead of the file name as follows:

```
# the url for the online csv file

url <- "https://data.gov.au/dataset/29128ebd-dbaa-4ff5-8b86-d9f30de5
```

- Next, as the online data is a .csv file, we can read this data file using `read.csv` function.

```
# use read.csv to import

ontime_data <- read.csv(url, stringsAsFactors = FALSE)

# display first six rows and four variables in the data

ontime_data[1:6,1:4]
```

```
##                   Route Departing_Port Arriving_Port      Airline
## 1   Adelaide-Brisbane       Adelaide      Brisbane All Airlines
## 2   Adelaide-Canberra       Adelaide      Canberra All Airlines
## 3 Adelaide-Gold Coast       Adelaide    Gold Coast All Airlines
## 4  Adelaide-Melbourne       Adelaide     Melbourne All Airlines
## 5      Adelaide-Perth       Adelaide         Perth All Airlines
## 6     Adelaide-Sydney       Adelaide        Sydney All Airlines
```

Scraping HTML Table Data

- Web pages contain several HTML tables and we may want to read the data from that HTML table.
- The simplest approach to scraping HTML table data directly into R is by using the `rvest` package.
- HTML tables are contained within `<table>` tags; therefore, to extract the tables, we need to use the `html_nodes()` function to select the `<table>` nodes.
- We will use the example from the help page for `rvest`, which loads all tables from the U.S. Social Security webpage: https://www.ssa.gov/oact/babynames/numberUSbirths.html
- First, we will install and load the `rvest` package:

```
# first install and load the rvest package

install.packages("rvest")
library(rvest)
```

- We will use `read_html` to locate the URL of the HTML table. When we use `read_html`, all table nodes that exist on the webpage will be captured.

```
births <- read_html("https://www.ssa.gov/oact/babynames/numberUSbirth
```

- In this example, using the `length` function we can see that the `html_nodes` captures 2 HTML tables.

```
length(html_nodes(births, "table"))
```

```
## [1] 2
```

```
html_nodes(births, "table")
```

```
## {xml_nodeset (2)}
## [1] <table class="table-layout">\n<colgroup>\n<col span="1" width="20%"> .
## [2] <table width="100%" class="border">\n<caption><b>Number of Social Se .
```

- This includes data from a few additional tables used to format other parts of the page (i.e. table of contents, table of figures, advertisements, etc.).

- The second table on the webpage is the place where our data is located, thus, we will select the second element of the `html_nodes`.

```
# select the second element of the html_nodes
births_data<- html_table(html_nodes(births, "table")[[2]])

# view the header of the births_data

head(births_data)
```

```
##   Year ofbirth    Male  Female   Total
## 1         1880 118,400  97,604 216,004
## 2         1881 108,282  98,855 207,137
## 3         1882 122,031 115,695 237,726
## 4         1883 112,477 120,059 232,536
## 5         1884 122,739 137,586 260,325
## 6         1885 115,945 141,949 257,894
```

Exporting Data to text files : Base R functions

- Exporting data out of R is equally important as importing data into R.
- I will introduce the base R and `readr` package functions to export data to text files.

```
# create a data frame and assign it to an object named df

df <- data.frame (cost = c(10, 25, 40),
                  color = c ("blue", "red", "green"),
                  suv = c (TRUE, TRUE, FALSE),
                  row.names = c ("car1", "car2", "car3"))

df
```

```
##      cost color   suv
## car1   10  blue  TRUE
## car2   25   red  TRUE
## car3   40 green FALSE
```

- To export `df` to a CSV file we will use `write.csv()`.

```
# write to a csv file in our working directory

write.csv(df, file = "cars_csv")
```

- To save the data frame in a different directory we will use:

```
# write to a csv and save in a different directory (i.e., ~/Desktop)

write.csv(df, file = "~/Desktop/cars_csv")
```

Exporting Data to text files : `readr` functions

- The `readr` package functions, `write_csv` and `write_delim` are twice as fast as base R functions and they are very similar in usage.

```
# load the library

library(readr)

# write to a csv file in the working directory

write_csv(df, path = "cars_csv2")
```

- Note that the base R write functions use the `file =` argument whereas, `readr` write functions use `path =` to specify the name of the file.

Saving Data as an R object File

- Sometimes we may need to save data or other R objects outside of the workspace or may want to store, share, or transfer between computers.
- We can use the .rda or .RData file types when we want to save several, or all, objects and functions that exist in the global environment.
- On the other hand, if we only want to save a single R object such as a data frame, function, or statistical model results, it is best to use the .rds file type.

To illustrate let's create two objects named x and y and save them to a .RData file using `save()` function.

```
# generate random numbers from uniform and normal distribution and as

x <- runif(10)
y <- rnorm(10, 0, 1)

# Save both objects in .RData format in the working directory

save(x, y, file = "xy.RData")
```

Saving Data as an R object File Cont.

- Also, the `save.image()` function will save your all current workspace as .RData.

```
# save all objects in the global environment

save.image()
```

- The following example will illustrate how a single object will be saved using `saveRDS()`

```
# save a single object to file

saveRDS(x, "x.rds")

# restore it under a different name

x2 <- readRDS("x.rds")
```

What do you need to know by Week 2

- Week 1 tasks +
- Understand how to get data from tabular and spreadsheet files
- Understand how to get data from statistical software and databases
- Know how to scrape data files stored online
- Know how to export to tabular and spreadsheet files
- Know how to save R objects
- Know how to get further help: Use R data import/export manual https://cran.r-project.org/doc/manuals/R-data.html
- RStudio's "Data Import Cheatsheet" is a compact resource for all importing functions available in the `readr` package.
- Practice!

Class Worksheet

- Working in small groups, complete the following class worksheet

Week 2 Class Worksheet

- Once completed, feel free to work on your Assignment and/or Skill Builders

Return to Data Preprocessing Website