Getting Data into R

Brendan Apfeld

October 14, 2016

A Quick R Refresher

Packages

- R is super useful, but we supplement its functionality with *packages*
- You install packages (assuming they're in the normal place) using `install.packages("package_name")`
  - You only have to install a package once (though you may need to update it later)
- You have to *load* packages every time you start an R session
  - Load a package in R by using `library(package_name)`
    - Note there are no quotation marks there
  - Loading a package makes all of its functions available to you
  - If you don't need that, you can also do `package_name::function()`
    - E.g. (that we'll see in a second) `rio::import("file_name.csv")`

Objects and Operators

- Pretty much *everything* in R is an object
    - Which is probably why it's an "OO," or "object-oriented" language
- You "assign" objects to names in R using the <− operator

Data Sources

Online

- Sometimes you'll have a direct link to a csv or txt file
- Other online data may be accessed through an api (and an R package)
- Advantages: potentially easy, "cutting edge data"
- Disadvantages: might change/disappear, often slow

On Disk

- More often, you'll have some kind of data file on disk
- Advantages: fast, it shouldn't disappear
- Disadvantages: you can break it

Data Types

Common Data File Formats

- Spreadsheet-style
    - .xlsx or .xls (Excel)
    - .ods (OpenDocument)
    - Google Sheets
- Proprietary Formats
    - .dta (Stata)
    - .sav (SPSS)
    - .xpt (SAS)
    - .Rdata (R)
- Plain Text Data Formats
    - .csv (our good friend)
    - .dat (not very common)
    - .json
    - .xml

Getting Data into R

The (Bad) Old Days

- `read.csv()` or `readr::read_csv()`
- `gdata::read.xls()`
- `foreign::read.dta()` or `haven::read_dta()`
- Which one should I use?
  - Depended on the version of the program used to create the data
  - Each package required different arguments
  - You never really knew what to expect

One Import Package to Rule them All

Then the `rio` package came along

- Only two commands in the whole package: `rio::import()` and `rio::export()`
- The only argument you *need* for it to work is the file name
- Supports other arguments if you really *want* to change its defaults
- You (almost) never need to change the defaults

Saving and Opening Rdata Files

- R has its own data format that ends in .Rdata
  - Super compressed
  - Saves and loads quickly
- Save with `save(objects, file = "filename.Rdata")`
- Load with `load("filename.Rdata")`

Examples with Data on Disk

Examples in R

Examples with Data Online

- Can be as simple as

```
wunderground <-
read.csv("https://www.wunderground.com/history/airport/ZBAA
/2013/1/1/DailyHistory.html?format=1")
```

- The list of possibilities for APIs is far to great to include here
- For an example using the World Bank data API and the corresponding `wbstats` package, check out my shameless self-promotion

Visualizing Data

Visualizing Datasets

- Datasets are often large and unwieldy
- There are a number of ways to easily learn something about them

```
mtcars <- rio::import("mtcars.csv")
mtcars # a lot of times this is too much for the console
```

```
##      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## 1   21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## 2   21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## 3   22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## 4   21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## 5   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## 6   18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## 7   14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## 8   24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## 9   22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
```

Visualizing Datasets

```r
str(mtcars) # look at the structure of the data
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : int  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : int  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : int  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : int  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: int  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: int  4 4 1 1 2 1 4 2 2 4 ...
```

Visualizing Datasets

```r
names(mtcars) # get the names of the varibles in the dataframe
```

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "
## [11] "carb"
```

```r
head(mtcars) # print the first 6 lines
```

```
##    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## 1 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## 2 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## 3 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## 4 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## 5 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## 6 18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

R Also has a Viewer

```
wg <- read.csv("https://www.wunderground.com/history/airport/ZBAA/2013/1
# View(wg) # commented out because it won't run on a slide
# note the capital "V"
# Can also access by clicking on the object in the
# environment pane in RStudio
```

Graphing - Back to R!