Coding Lab: Reading in data

Ari Anisfeld

Summer 2020

```
## Warning: package 'haven' was built under R version 3.6.3
```

So you found some data

Say you find a spreadsheet on the internet and want to start exploring it with R.

Sometimes loading data is as easy as

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

But often you'll need to consider:

- File location
- File type
- Funky formatting

detour: directory structure

Computer hard drives are organized using a file system. In this way, each file has a unique “address” or **file path**.

- `~/Documents/coding_lab_examples/texas_housing_data.csv`

The files are stored in folders or directories which are analagous to “zip codes”.

- `~/Documents/coding_lab_examples/`

In Windows, file paths start with `C://...`

detour: working directory

The ‘working directory’ in an R session is the folder your script knows about. If the data you want is in that folder you can refer to it directly.

```
fed_data <-
  read_xlsx("SCE-Public-LM-Quarterly-Microdata.xlsx")
```

`getwd()` shows your current working directory .

detour: directory structure

if the data were not in your current working directory you could:

- give the full address:
  `read_csv("~/Documents/coding_lab_examples/file.csv")`
- give a relative address:
  `read_csv("coding_lab_examples/file.csv")`
- change the current working directory:
  `setwd("~/Documents/coding_lab_examples")`
- move the file to the current working directory: `drag and drop`

loading data of various formats

We can load data into R with different functions depending on the data format.

file type	package	function
.csv	readr	`read_csv()`
.dta (stata)	haven	`read_dta()`
.xlsx	readxl	`read_xlsx()`

Note: `readr` is loaded with `tidyverse`

loading data?

Sometimes, as we experienced with our csv import, reading data is a straightforward process

```
drug_war_data <- read_dta(
  "../data/Dataset_HighProfileCriminalViolence.dta")
head(drug_war_data)
```

```
## # A tibble: 6 x 36
##   cve_inegi state municipality  year aggr_sum aggr_dummy
##       <dbl> <chr> <chr>        <dbl>    <dbl>      <dbl
## 1      1001 Agua~ Aguascalien~  2007        0          0
## 2      1001 Agua~ Aguascalien~  2008        0          0
## 3      1001 Agua~ Aguascalien~  2009        0          0
## 4      1001 Agua~ Aguascalien~  2010        0          0
## 5      1001 Agua~ Aguascalien~  2011        0          0
## 6      1002 Agua~ Asientos      2007        0          0
## # ... with 29 more variables: menfeje1000 <dbl>, juxt1 <
## #   juxt3 <dbl>, juxt4 <dbl>, juxt5 <dbl>, juxt6 <dbl>,
## #   juxt8 <dbl>, juxt9 <dbl>, juxtaposition <dbl>, juxt
```

loading data?

While often you can just load the data directly, we often require finesse

```r
fed_data <-
  read_xlsx(
    "../data/SCE-Public-LM-Quarterly-Microdata.xlsx")
head(fed_data)
```

```
## # A tibble: 6 x 1
##   `License for Survey of Consumer Expectations Data and
##   <chr>
## 1 <NA>
## 2 The Survey of Consumer Expectations (the "SCE") was de
## 3 FRBNY launched the SCE in 2013.  The subject matter ar
## 4 questions have been informed by or adapted from other
## 5 FRBNY has published the SCE questions and most data it
## 6 and resuse. FRBNY permits use of the SCE questions and
```

```r
fed_data <-
  read_xlsx(
    "../data/SCE-Public-LM-Quarterly-Microdata.xlsx",
    sheet = "Data 2013",
    skip = 1)
head(fed_data)
```

```
## # A tibble: 6 x 488
##   userid weight L1_rc L2_rc    L4    L5   L5b    L6    L6
##    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl
## 1 7.00e7  0.662     2     1     0    NA    NA     2    N
## 2 7.00e7  0.738     1     1     0    NA    NA     3
## 3 7.00e7  0.473     1     3     0    NA    NA     2    N
## 4 7.00e7  4.62      9    NA    NA     0     2    NA    N
## 5 7.00e7  0.491     2     1     1    NA    NA    NA    N
## 6 7.00e7  0.348     1     3     0    NA    NA     1    N
## # ... with 476 more variables: L9 <dbl>, L10 <dbl>, L11_
## #   L11_weekly <dbl>, L11_annual <dbl>, L11_flag <dbl>,
## #   JH9_weekly <dbl>, JH9_annual <dbl>, L14self_1_rc <db
## #   L14self_2_rc <dbl>, L14self_3_rc <dbl>, L14self_4_rc
```

looking at data: head(), glimpse() or View()

- head() and glimpse() provide ways to see part of your data.
- View() provides a more spreadsheet-like experience.

```r
head(texas_housing_data)
```

```
## # A tibble: 6 x 9
##   city     year month sales   volume median listings inv
##   <chr>   <int> <int> <dbl>    <dbl>  <dbl>    <dbl>
## 1 Abilene  2000     1    72  5380000  71400      701
## 2 Abilene  2000     2    98  6505000  58700      746
## 3 Abilene  2000     3   130  9285000  58100      784
## 4 Abilene  2000     4    98  9730000  68600      785
## 5 Abilene  2000     5   141 10590000  67300      794
## 6 Abilene  2000     6   156 13910000  66900      780
```

getting meta data

Get number of rows

```
nrow(texas_housing_data)
```

```
## [1] 8602
```

See column names

```
names(texas_housing_data)
```

```
## [1] "city"       "year"       "month"      "sales"      "vol
## [7] "listings"   "inventory" "date"
```

Recap

- For most file types there's a function of form `read_xxx()` that will get the data into R.
- Use `getwd()` and `setwd()` to ensure you're in the right directory.
- When you have funky formatting use `?` to see if R can help you fix the problem on read.
- R has useful functions like `View()`, `glimpse()`, `head()`, `names()` and `nrow()` to get to know your data.