Managing and Manipulating Data Using R

Lecture 8, Acquiring data in R

Patricia Martin

1. Introduction

2. Common data formats

3. `readr` package

4. `haven` package

5. `readxl` package

6. Downloading data from web

7. Data sources (maybe)

Introduction

Libraries we will use today [install if you don’t have them]

```r
library(dplyr)
library(readr)
library(haven)
library(readxl)
library(labelled)
```

Data we will use today

- Integrated Postsecondary Education Data System (IPEDS)
- High School Longitudinal Surveys (HSLS)
- Federal Student Aid Data
- Equality of Opportunity Project

Integrated Postsecondary Education Data System (IPEDS)

- Postsecondary education data from NCES
- There are 12 survey components and 3 collection periods

We will be working with Institutional Characteristics data of 2017

High school longitudinal surveys from National Center for Education Statistics (NCES)

- Follow U.S. students from high school through college, labor market

We will be working with High School Longitudinal Study of 2009 (HSLS:09)

- Follows 9th graders from 2009
- Data collection waves
  - Base Year (2009)
  - First Follow-up (2012)
  - 2013 Update (2013)
  - High School Transcripts (2013-2014)
  - Second Follow-up (2016)

Federal Student Aid

- Federal Student Aid Data Center provides information for federal assistance programs and is divided into four categories:
  - Student Aid Data
  - School Data
  - Federal Family Education Loan (FFEL) Program
  - Business Information Resources

We will be working with School Data

Equality of Opportunity Project

- Equality of Opportunity Project uses two data sources– federal tax recoards and Department of Education records (1999-2013)– to investigate intergenerational income mobility at colleges in the US.

We will use Mobility Report Cards: The Role of Colleges in Intergenerational Mobility data

Not sure if to simply list datasets. Lecture will go over acquiring data and not so much about manipulating data. Not sure if students need to know each dataset in detail?

Common data formats

Common data formats

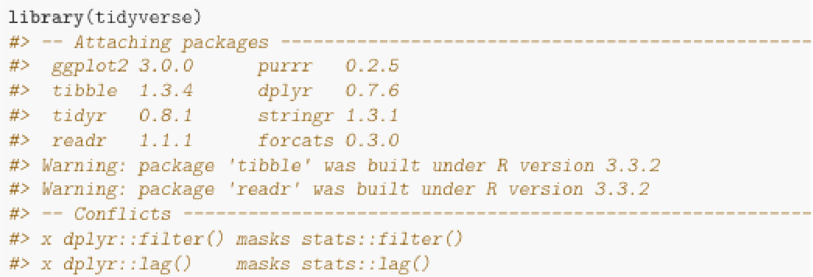
- Comma-separated values (.csv)
- Excel (.xls or .xlsx)
- Text-formated data (.txt)
- Tab-separated values (.tsv)
- R (.Rdata or .rds)
- Stata (.dta)
- SPSS (.sav)
- SAS (.sas)

`readr` package

readr

The `readr` package is part of tidyverse, which is designed to read in flat data files in R and transform them into data frames.

- We could load **library(tidyverse)** if we wanted to load all packages in tidyverse (e.g. ggplot2, dplyr, tidyr, stringr, readr, etc…)



```
library(tidyverse)
#> -- Attaching packages ---------------------------------------------
#>  ggplot2 3.0.0      purrr   0.2.5
#>  tibble  1.3.4      dplyr   0.7.6
#>  tidyr   0.8.1      stringr 1.3.1
#>  readr   1.1.1      forcats 0.3.0
#> Warning: package 'tibble' was built under R version 3.3.2
#> Warning: package 'readr' was built under R version 3.3.2
#> -- Conflicts ------------------------------------------------------
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
```

- For the purpose of this lecture, we will just need to load **library(readr)**

readr

No matter the flat file format you are working with, there are two important steps for reading in data with `readr`:

1. **a function to parse the file (read_csv)**
2. **column specification**

`readr functions`

readr's (tidyverse) functions

Format	**Function**
Comma-separated values (csv)	`read_csv`
Semicolon separated files	`read_csv2`
Tab-separated values (tsv)	`read_tsv`
Any delimiter	`read_delim`
Fixed width files	`read_fwf`
Text-formated data (txt)	`read_table`
Web log files	`read_log`

readr column specification

readr is pretty good at guessing each column's data type (e.g. character, double, etc.), however it is good practice to manually specify the data type for each column.

```
mtcars <- read_csv(readr_example("mtcars.csv"))
#> Parsed with column specification:
#> cols(
#>   mpg = col_double(),
#>   cyl = col_integer(),
#>   disp = col_double(),
#>   hp = col_integer(),
#>   drat = col_double(),
#>   wt = col_double(),
#>   qsec = col_double(),
#>   vs = col_integer(),
#>   am = col_integer(),
#>   gear = col_integer(),
#>   carb = col_integer()
#> )
```

readr column specification

The output of the previous example shows us the column specification readr gave us. However, we could manually change column specification if we do not like readr's guess.

```r
mtcars <- read_csv(readr_example("mtcars.csv"), col_types =
  cols(
    mpg = col_double(),
    cyl = col_integer(),
    disp = col_double(),
    hp = col_integer(),
    drat = col_double(),
    vs = col_integer(),
    wt = col_double(),
    qsec = col_double(),
    am = col_integer(),
    gear = col_integer(),
    carb = col_integer()
  )
)
```

`readr` features

- **skip**: `read_csv`(csv file, skip = n)
- **comment**: `read_csv`(csv file, comment = “#”)
- **col_names**: `read_csv`(csv file, col_names = c(“x”, “y”, “z”))

readr demonstration csv

readr automatically treats the first line of data as column names.

```
read_csv("column 1, column 2, column 3
         1,2,3
         4,5,6"
         )
#> # A tibble: 2 x 3
#>   `column 1` `column 2` `column 3`
#>        <int>      <int>      <int>
#> 1          1          2          3
#> 2          4          5          6
```

There are instances where you may want to tell R from what line to begin reading in data.

readr demonstration csv

Notice the example below. The first two lines are comments about the data. We would need to use **skip = n** to skip n lines.

```
read_csv("This file contains data on student charges for the acdemic year.
         File name: IC2016_AY
         a, b, c
         1,2,3
         4,5,6", skip = 2
         )
#> # A tibble: 2 x 3
#>         a     b     c
#>     <int> <int> <int>
#> 1     1     2     3
#> 2     4     5     6
```

readr demonstration csv

We could also tell R to drop lines we specify as comments. With **comment = n**

```
read_csv("# This file contains data on student charges for the acdemic year.
         a, b, c
         1,2,3
         4,5,6", comment = "#"
         )
#> # A tibble: 2 x 3
#>       a     b     c
#>   <int> <int> <int>
#> 1     1     2     3
#> 2     4     5     6
```

```
read_csv("* This file contains data on student charges for the acdemic year.
         a, b, c
         1,2,3
         4,5,6", comment = "*"
         )
#> # A tibble: 2 x 3
#>       a     b     c
#>   <int> <int> <int>
#> 1     1     2     3
#> 2     4     5     6
```

readr column names

We could tell R there are no column names with **col_names = FALSE** or we could manually give R column names with **col_names = c(“”, “”, “”)**

```
read_csv("1,2,3
         4,5,6", col_names = FALSE
         )
#> # A tibble: 2 x 3
#>      X1    X2    X3
#>   <int> <int> <int>
#> 1     1     2     3
#> 2     4     5     6
```

```
read_csv("1,2,3
         4,5,6", col_names = c("column 1", "column 2", "column 3")
         )
#> # A tibble: 2 x 3
#>   `column 1` `column 2` `column 3`
#>        <int>      <int>      <int>
#> 1          1          2          3
#> 2          4          5          6
```

readr Student exercise

- Get in your homework groups
- Create a 3x3 tibble like the examples above (e.g. read_csv("a,b,c. . . .")), treating the first line as column names
- Now on the first line add a sentence
- This time add a special character (*, #, !) at the beginning of the sentence and indicate it is a comment
- Delete the sentence and column names (should have a 2x2 tibble) and manually tell R column names

`readr` demonstration csv

NOT SURE IF TO MAKE THIS A DEMONSTRATION WHERE STUDENTS FOLLOW ALONG OR ANOTHER STUDENT EXERCISE
Tying it all together

Use `read_csv()` function from `readr` to import csv dataset into R without column specification. Follow along on your computers.

```
ipeds <- read_csv(file="~/Desktop/GitHub/rclass/data/ipeds/ic/ipeds_hd_2017_smal
#> Parsed with column specification:
#> cols(
#>   unitid = col_integer(),
#>   instnm = col_character(),
#>   stabbr = col_character(),
#>   sector = col_integer(),
#>   iclevel = col_integer(),
#>   control = col_integer()
#> )
# glimpse(ipeds)
```

`readr` demonstration csv

Use `read_csv()` function from `readr` to import csv dataset into R with column specification **[Would it be better to change to integer or double?]**

```
ipeds <- read_csv(file="~/Desktop/GitHub/rclass/data/ipeds/ic/ipeds_hd_2017_smal
                  col_types =
                    cols(
                      unitid = col_number(),
                      instnm = col_character(),
                      stabbr = col_character(),
                      sector = col_integer(),
                      iclevel = col_integer(),
                      control = col_integer()
  )
)
```

We changed unitid to number, but could be left as is or changed to character type for example.

`readr` variable and value labels

Let’s view variable and value labels

```
ipeds %>% select(sector) %>% var_label()
#> $sector
#> NULL
ipeds %>% select(sector) %>% val_labels()
#> $sector
#> NULL
```

There are no variable and value labels for this data. IPEDS has a separate do file with variable and value labels.

- Let’s practice manually adding variable and value labels using the `labelled` package.

readr labelled data

- Open the data dictionary file for hd2017 data and select “Frequencies” sheet
- We are only working with these 6 variables (unitid, instnm, stabbr, sector, iclevel, control)
- We need to add variable labels for all 6 variables
- We need to add value labels for sector, iclevel, and control

```
# Lets view values for sector
ipeds %>%
  count(sector)
#> # A tibble: 11 x 2
#>    sector     n
#>     <int> <int>
#>  1      0    75
#>  2      1   775
#>  3      2  1701
#>  4      3   661
#>  5      4   981
#>  6      5   169
#>  7      6   864
#>  8      7   248
#>  9      8    85
#> 10      9  1562
#> 11     99    32
```

readr manually add variable and value labels

```r
# Need to manually assign variable and value labels using labelled package
ipeds_labelled <- ipeds %>%
  set_variable_labels(unitid = "Unit identification number",
                      instnm = "Institution name",
                      stabbr = "State abbreviation",
                      sector = "Sector of institution",
                      iclevel = "Level of institution",
                      control = "Control of institution") %>%
  set_value_labels(sector = c("Administrative Unit" = 0,
                              "Public, 4-year or above" = 1,
                              "Private not-for-profit, 4-year or above" = 2,
                              "Private for-profit, 4-year or above" = 3,
                              "Public, 2-year" = 4,
                              "Private not-for-profit, 2-year" = 5,
                              "Private for-profit, 2-year" = 6,
                              "Public, less-than 2-year" = 7,
                              "Private not-for-profit, less-than 2-year" = 8,
                              "Private for-profit, less-than 2-year" = 9,
                              "Sector unknown (not active)" = 99),
                   iclevel = c("Four or more years" = 1,
                               "At least 2 but less than 4 years" = 2,
                               "Less than 2 years (below associate)" = 3,
                               "{Not available}" = -3),
                   control = c("Public" = 1, "Private not-for-profit" = 2,
                               "Private for-profit" = 3,
                               "{Not available}" = -3))
```

readr Let's view new labelled data

```
typeof(ipeds_labelled$iclevel)
#> [1] "integer"
class(ipeds_labelled$iclevel)
#> [1] "labelled"
attributes(ipeds_labelled$iclevel)
#> $label
#> [1] "Level of institution"
#>
#> $labels
#>                  Four or more years    At least 2 but less than 4 years
#>                                   1                                   2
#> Less than 2 years (below associate)                     {Not available}
#>                                   3                                  -3
#>
#> $class
#> [1] "labelled"
```

Let's change class to factor

`readr` Class to factor

Approach #1

```
ipeds_factor <- as_factor(ipeds_labelled, only_labelled = TRUE)
typeof(ipeds_factor$sector)
#> [1] "integer"
class(ipeds_factor$sector)
#> [1] "factor"
attributes(ipeds_factor$sector)
#> $levels
#>  [1] "Administrative Unit"
#>  [2] "Public, 4-year or above"
#>  [3] "Private not-for-profit, 4-year or above"
#>  [4] "Private for-profit, 4-year or above"
#>  [5] "Public, 2-year"
#>  [6] "Private not-for-profit, 2-year"
#>  [7] "Private for-profit, 2-year"
#>  [8] "Public, less-than 2-year"
#>  [9] "Private not-for-profit, less-than 2-year"
#> [10] "Private for-profit, less-than 2-year"
#> [11] "Sector unknown (not active)"
#>
#> $class
#> [1] "factor"
#>
#> $label
#> [1] "Sector of institution"
```

readr Class to factor

Approach #2

```
ipeds_factor2 <- to_factor(ipeds_labelled, ordered = TRUE)
typeof(ipeds_factor2$sector)
#> [1] "integer"
class(ipeds_factor2$sector)
#> [1] "ordered" "factor"
attributes(ipeds_factor2$sector)
#> $levels
#>  [1] "Administrative Unit"
#>  [2] "Public, 4-year or above"
#>  [3] "Private not-for-profit, 4-year or above"
#>  [4] "Private for-profit, 4-year or above"
#>  [5] "Public, 2-year"
#>  [6] "Private not-for-profit, 2-year"
#>  [7] "Private for-profit, 2-year"
#>  [8] "Public, less-than 2-year"
#>  [9] "Private not-for-profit, less-than 2-year"
#> [10] "Private for-profit, less-than 2-year"
#> [11] "Sector unknown (not active)"
#>
#> $class
#> [1] "ordered" "factor"
#>
#> $label
#> [1] "Sector of institution"
```

1. readr Running into errors

1. Make sure you have downloaded and saved flat file
2. Make sure to know the file path of where data is downloaded or saved (~/Desktop/educ263/data)
3. Make sure you set your working `setwd()` directory in R. To check your current working directory type `getwd()` in console.

haven package

haven

Recap from lecture 5

haven is part of **tidyverse**, which enables users to import and export data from the following statistical packages:

- SAS
- SPSS
- Stata

Similar to readr, we could load the entire **library(tidyverse)** package to get haven. For the purpose of this lecture, we will just need to load **library(haven)**.

haven functions

haven's (tidyverse) functions

Format	**Function**
SPSS	`read_sav`
SAS	`read_sas`
Stata	`read_dta`

haven read and write Stata arguments

```
read_dta(file, encoding = NULL)
write_data(data, path, version = 14)
```

Arguments
- **file**: file path to data
- **encoding**: files prior to Stata 14 did not declare text encoding, files after Stata 14 do not need to declare encoding value
- **data**: data frame to save (write)
- **path**: file path to where data will be saved
- **version**: file version

Link

haven Student exercise

- Use read_dta() function from haven to import Stata dataset into R
- Use write_dta() funtction from haven to save Stata dataset
- If you have time, explore data (View, glimpse, head, etc.)
  - View variable and value labels
  - Change class == labelled to class == factor

`haven` Student exercise Solution

Use `read_dta` function from `haven` to import State data

```
hsls <- read_dta("~/Desktop/GitHub/rclass/data/hsls/hsls_sch_small.dta", encodin

# View data
head(hsls)
glimpse(hsls)
```

Use `write_dta` function from `haven` to write State data

```
write_dta(hsls, path = "~/Desktop/GitHub/rclass/data/hsls/hsls_sch_small.dta")
```

haven Student exercise Solution cont...

Variable and Value labels

```
# View variable labels
hsls %>% var_label()
#> $sch_id
#> [1] "School ID"
#>
#> $x1control
#> [1] "X1 School control"
#>
#> $x1locale
#> [1] "X1 School locale (urbanicity)"
#>
#> $x1region
#> [1] "X1 School geographic region"
#>
#> $a1schcontrol
#> [1] "A1 A02 School control"
```

haven Student exercise Solution cont...

```
#View value label for x1locale
hsls %>% select(x1locale) %>% val_labels()
#> $x1locale
#>                                    Missing
#>                                          1
#> Unit non-response/component not applicable
#>                                          2
#>                    Item legitimate skip/NA
#>                                          3
#>                                       City
#>                                          4
#>                                     Suburb
#>                                          5
#>                                       Town
#>                                          6
#>                                      Rural
#>                                          7
```

haven Student exercise Solution cont...

```r
# Change class == labelled to class == factor
hsls <- as_factor(hsls, only_labelled = TRUE)

typeof(hsls$x1region)
#> [1] "integer"
class(hsls$x1region)
#> [1] "factor"
attributes(hsls$x1region)
#> $levels
#> [1] "Missing"
#> [2] "Unit non-response/component not applicable"
#> [3] "Item legitimate skip/NA"
#> [4] "Northeast"
#> [5] "Midwest"
#> [6] "South"
#> [7] "West"
#>
#> $class
#> [1] "factor"
#>
#> $label
#> [1] "X1 School geographic region"
```

`readxl` package

readxl

The `readxl` package is part of tidyverse, which is designed to easily read data from Excel and into R.
- We could load **library(tidyverse)** if we wanted to load all packages in tidyverse. For the purpose of this lecture, we just need to load **library(readxl)**.

readxl

readxl supports both .xls and .xlsx formats and is designed to work with tabular data. It does not require dependencies– making installing and operating fairly simple.

readxl has several example files where we could use as practice. The files include:

```
readxl_example()
#>  [1] "clippy.xls"    "clippy.xlsx"   "datasets.xls"  "datasets.xlsx"
#>  [5] "deaths.xls"    "deaths.xlsx"   "geometry.xls"  "geometry.xlsx"
#>  [9] "type-me.xls"   "type-me.xlsx"
```

For now, lets use “datasets.xlsx”

```
excel_example <- readxl_example("datasets.xlsx")
```

`readxl` features

- **sheet**: `read_excel`(excel file, sheet = “sheet name”)
- **n_max**: `read_excel`(excel file, n_max = n)
- **range**: `read_excel`(excel file, range = “A:D”)
- **cell_rows**: `read_excel`(excel file, range = cell_rows(1:n))
- **cell_cols**: `read_excel`(excel file, range = cell_cols(“A:D”))
- **na**: `read_excel`(excel file, na = “n”)

readxl sheet

```
#To view sheets in excel file
excel_sheets(excel_example)
#> [1] "iris"     "mtcars"   "chickwts" "quakes"
```

```
xl_example <- read_excel(excel_example, sheet = "quakes")
head(xl_example)
#> # A tibble: 6 x 5
#>      lat   long depth   mag stations
#>    <dbl>  <dbl> <dbl> <dbl>    <dbl>
#> 1 -20.42 181.62   562   4.8       41
#> 2 -20.62 181.03   650   4.2       15
#> 3 -26.00 184.10    42   5.4       43
#> 4 -17.97 181.66   626   4.1       19
#> 5 -20.42 181.96   649   4.0       11
#> 6 -19.68 184.31   195   4.0       12
```

readxl n_max

```
read_excel(excel_example, sheet = "quakes", n_max = 3)
#> # A tibble: 3 x 5
#>       lat   long depth   mag stations
#>     <dbl>  <dbl> <dbl> <dbl>    <dbl>
#> 1 -20.42 181.62   562   4.8       41
#> 2 -20.62 181.03   650   4.2       15
#> 3 -26.00 184.10    42   5.4       43
```

readxl range

```
read_excel(excel_example, sheet = "quakes", range = "C1:E4")
#> # A tibble: 3 x 3
#>   depth   mag stations
#>   <dbl> <dbl>    <dbl>
#> 1   562   4.8       41
#> 2   650   4.2       15
#> 3    42   5.4       43

read_excel(excel_example, sheet = "quakes", range = cell_rows(1:3))
#> # A tibble: 2 x 5
#>      lat   long depth   mag stations
#>    <dbl>  <dbl> <dbl> <dbl>    <dbl>
#> 1 -20.42 181.62   562   4.8       41
#> 2 -20.62 181.03   650   4.2       15

head(read_excel(excel_example, sheet = "quakes", range = cell_cols("A:C")))
#> # A tibble: 6 x 3
#>      lat   long depth
#>    <dbl>  <dbl> <dbl>
#> 1 -20.42 181.62   562
#> 2 -20.62 181.03   650
#> 3 -26.00 184.10    42
#> 4 -17.97 181.66   626
#> 5 -20.42 181.96   649
#> 6 -19.68 184.31   195
# using head() to only view first 6 rows
```

readxl na

```
read_excel(excel_example, sheet = "quakes", na = "-20.42")
#> # A tibble: 1,000 x 5
#>       lat   long depth   mag stations
#>     <dbl>  <dbl> <dbl> <dbl>    <dbl>
#>  1     NA 181.62   562   4.8       41
#>  2 -20.62 181.03   650   4.2       15
#>  3 -26.00 184.10    42   5.4       43
#>  4 -17.97 181.66   626   4.1       19
#>  5     NA 181.96   649   4.0       11
#>  6 -19.68 184.31   195   4.0       12
#>  7 -11.70 166.10    82   4.8       43
#>  8 -28.11 181.93   194   4.4       15
#>  9 -28.74 181.74   211   4.7       35
#> 10 -17.47 179.59   622   4.3       19
#> # ... with 990 more rows
```

`readxl` Student exercise

Save data - Download and save Federal Student Financial Aid Data
- Read in data using `readxl` function
- Read in first four rows (n_max) - Read in column Names to column State **hint** `cell_cols` - Set value "A" to missing (na) **note** : you need to investigate in detail before setting anything to missing

`readxl` Student exercise solution

```
#Read in data using readxl function
setwd("~/Desktop/GitHub/rclass/data/fsa")
fsa <- read_excel("peps300.xlsx")
```

```
#Read in first four rows (n_max)
setwd("~/Desktop/GitHub/rclass/data/fsa")
read_excel("peps300.xlsx", n_max = 4)
#> # A tibble: 4 x 29
#>    OPEID                                         Name
#>    <chr>                                        <chr>
#> 1 001002 ALABAMA AGRICULTURAL & MECHANICAL UNIVERSITY
#> 2 001003                          FAULKNER UNIVERSITY
#> 3 001004                     UNIVERSITY OF MONTEVALLO
#> 4 001005                     ALABAMA STATE UNIVERSITY
#> # ... with 27 more variables: Address <chr>, City <chr>, State <chr>,
#> #   `State Desc` <chr>, `Zip Code` <chr>, `Zip Ext` <chr>,
#> #   `Prog\r\nLength` <dbl>, `School\r\nType` <dbl>, `Year 1` <dbl>,
#> #   `Dual\r\nNum 1` <dbl>, `Dual\r\nDenom 1` <dbl>, `DRate 1` <dbl>,
#> #   `PRate 1` <chr>, `Ethnic Code` <chr>, Program <chr>, `Cong Dis` <chr>,
#> #   Region <chr>, `Year 2` <dbl>, `Dual\r\nNum 2` <dbl>, `Dual\r\nDenom
#> #   2` <dbl>, `DRate 2` <dbl>, `PRate 2` <chr>, `Year 3` <dbl>,
#> #   `Dual\r\nNum 3` <dbl>, `Dual\r\nDenom 3` <dbl>, `DRate 3` <dbl>,
#> #   `PRate 3` <chr>
```

readxl Student exercise solution cont…

```
#Read in column Names to column State
setwd("~/Desktop/GitHub/rclass/data/fsa")
head(read_excel("peps300.xlsx", range = cell_cols("B:E")))
#> # A tibble: 6 x 4
#>                                           Name                  Address
#>                                          <chr>                    <chr>
#> 1 ALABAMA AGRICULTURAL & MECHANICAL UNIVERSITY     4900 MERIDIAN STREET
#> 2                          FAULKNER UNIVERSITY     5345 ATLANTA HIGHWAY
#> 3                     UNIVERSITY OF MONTEVALLO            PALMER CIRCLE
#> 4                     ALABAMA STATE UNIVERSITY 915 SOUTH JACKSON STREET
#> 5            CENTRAL ALABAMA COMMUNITY COLLEGE       1675 CHEROKEE ROAD
#> 6                      ATHENS STATE UNIVERSITY   300 NORTH BEATY STREET
#> # ... with 2 more variables: City <chr>, State <chr>
```

```
setwd("~/Desktop/GitHub/rclass/data/fsa")
read_excel("peps300.xlsx", n_max = 4,  na = "A")
#> # A tibble: 4 x 29
#>    OPEID                                         Name
#>    <chr>                                        <chr>
#> 1 001002 ALABAMA AGRICULTURAL & MECHANICAL UNIVERSITY
#> 2 001003                          FAULKNER UNIVERSITY
#> 3 001004                     UNIVERSITY OF MONTEVALLO
#> 4 001005                     ALABAMA STATE UNIVERSITY
#> # ... with 27 more variables: Address <chr>, City <chr>, State <chr>,
#> #   `State Desc` <chr>, `Zip Code` <chr>, `Zip Ext` <chr>,
#> #   `Prog\r\nLength` <dbl>, `School\r\nType` <dbl>, `Year 1` <dbl>,
#> #   `Dual\r\nNum 1` <dbl>, `Dual\r\nDenom 1` <dbl>, `DRate 1` <dbl>,
```

`readxl` Running into problems

1. Make sure you have downloaded and saved excel file
2. Make sure to know the file path of where data is downloaded or saved (~/Desktop/educ263/data)
3. Make sure you set your working `setwd()` directory in R. To check your current working directory type `getwd()` in console.
4. Make sure to choose the correct sheet (if applicable)
5. Pay attention to column names when setting range

Downloading data from web

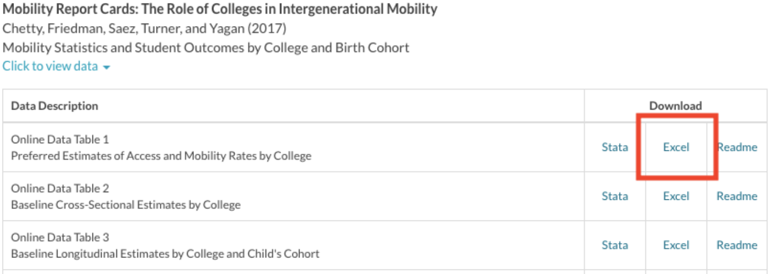
Downloading data from web

- Save time
  - Reduce the steps of downloading, saving, and reading in data
  - Read in data directly from internet
  - **note** not all packages will working with downloading data from the web (read_excel)

For example, rather than downloading ipeds data and saving it in a folder, we could download the data directly from the web.

Downloading data from web example using Raj Chetty data

1. Follow this link and under the “Mobility Report Cards…” tab select “click to view data”.
2. Choose “Online Data Table 1”
3. Right click and copy link address for “Excel” (Note: it is actually a csv file)



Mobility Report Cards: The Role of Colleges in Intergenerational Mobility
Chetty, Friedman, Saez, Turner, and Yagan (2017)
Mobility Statistics and Student Outcomes by College and Birth Cohort
Click to view data ▾

Data Description	Download		
Online Data Table 1 Preferred Estimates of Access and Mobility Rates by College	Stata	Excel	Readme
Online Data Table 2 Baseline Cross-Sectional Estimates by College	Stata	Excel	Readme
Online Data Table 3 Baseline Longitudinal Estimates by College and Child's Cohort	Stata	Excel	Readme

Downloading data from web

[FIX OUTPUT (CUTTING OFF)]

```
#Paste url to excel "csv" file
data_url <- "http://www.equality-of-opportunity.org/data/college/mrc_table1.csv"

#Download data and read in using read_csv (readr)
mrc <- read_csv(data_url)

#View first 4 rows and 4 columns
mrc[1:4, 1:4]
#> # A tibble: 4 x 4
#>   super_opeid                                         name   czname state
#>         <int>                                        <chr>    <chr> <chr>
#> 1        2665 Vaughn College Of Aeronautics And Technology New York    NY
#> 2        7273               CUNY Bernard M. Baruch College New York    NY
#> 3        2688              City College Of New York - CUNY New York    NY
#> 4        7022                          CUNY Lehman College New York    NY
```

Downloading data from web

Alternative approach

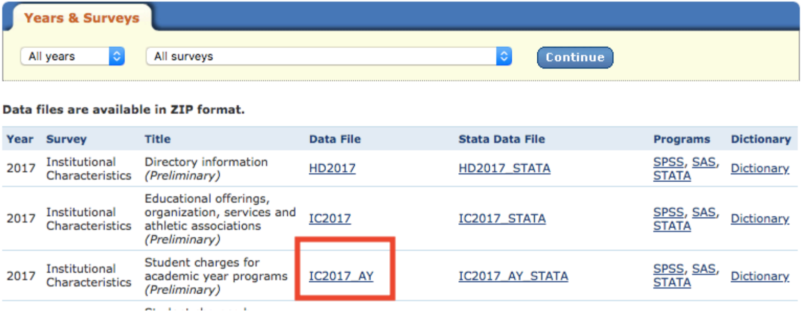
```
#Download data and read in link directly using read_csv (readr)
mrc <- read_csv("http://www.equality-of-opportunity.org/data/college/mrc_table1.
#> Parsed with column specification:
#> cols(
#>   super_opeid = col_integer(),
#>   name = col_character(),
#>   czname = col_character(),
#>   state = col_character(),
#>   par_median = col_integer(),
#>   k_median = col_integer(),
#>   par_q1 = col_double(),
#>   par_top1pc = col_double(),
#>   kq5_cond_parq1 = col_double(),
#>   ktop1pc_cond_parq1 = col_double(),
#>   mr_kq5_pq1 = col_double(),
#>   mr_ktop1_pq1 = col_double(),
#>   trend_parq1 = col_double(),
#>   trend_bottom40 = col_double(),
#>   count = col_double()
#> )
```

Downloading data from web

```
#View first 4 rows and 4 columns
mrc[1:4, 1:4]
#> # A tibble: 4 x 4
#>   super_opeid                                         name   czname state
#>         <int>                                        <chr>    <chr> <chr>
#> 1        2665 Vaughn College Of Aeronautics And Technology New York    NY
#> 2        7273               CUNY Bernard M. Baruch College New York    NY
#> 3        2688              City College Of New York - CUNY New York    NY
#> 4        7022                          CUNY Lehman College New York    NY
```

Problems downloading data (zip files) using IPEDS

1. Follow this link and under the “Survey Data” tab select “Complete data files”.
2. Choose “All years” and “All surveys” and click continue
3. Right click and copy link address for “IC2017_AY”



Data files are available in ZIP format.

Year	Survey	Title	Data File	Stata Data File	Programs	Dictionary
2017	Institutional Characteristics	Directory information *(Preliminary)*	HD2017	HD2017_STATA	SPSS, SAS, STATA	Dictionary
2017	Institutional Characteristics	Educational offerings, organization, services and athletic associations *(Preliminary)*	IC2017	IC2017_STATA	SPSS, SAS, STATA	Dictionary
2017	Institutional Characteristics	Student charges for academic year programs *(Preliminary)*	IC2017_AY	IC2017_AY_STATA	SPSS, SAS, STATA	Dictionary

Downloading data (zip files) using IPEDS

Paste url and read in using `read_csv`
What happens when you try reading in this zip file?
Need to download **and** unzip

Downloading data (zip files) using IPEDS

```
#Set path to where data will be saved
setwd("~/Desktop/lecture8")
#download file and pa
download.file("https://nces.ed.gov/ipeds/datacenter/data/IC2017_AY.zip",
              destfile = "ic2017_ay", mode = 'wb')
#unzip zip file and keep original name
unzip(zipfile = "ic2017_ay" , unzip = "unzip")
#> arguments 'minimized' and 'invisible' are for Windows only

ic2017_ay <- read_csv("ic2017_ay.csv")
#> Parsed with column specification:
#> cols(
#>   .default = col_character(),
#>   UNITID = col_integer()
#> )
#> See spec(...) for full column specifications.
```

Student exercise

Tying it all together

- Using everything we learned today read in a csv data file from the web
- Go back to the ipeds data center here
- Right click and copy the link address to a different data file (“HD2017”, “EFFY2017”)
- Make sure to download the link first (download.file) before reading in
- Change column names to lowercase
- Report dimensions of data
- Create a subset of your data (filter, select, etc.)

Data sources (maybe)