Week 2: Foundational Skills

Alex Lishinski

August 24, 2021

Course organization

- Website
  - Syllabus
  - Presentations
  - Homework
  - Videos (recordings of class and about specific topics)
- Slack: https://introductiont-mej9811.slack.com
- Zoom: https://tennessee.zoom.us/j/94135227830
- Email: alishins@utk.edu

RECORD THE MEETING

Touching Base

- Introductions

- Alex Lishinski, Ph.D. (they/them)
- Contact:
  - alishins@utk.edu
- Postdoctoral researcher, CS Education, University of Tennessee, Knoxville
- Primary areas of interest:
  - Computer Science education
  - Quantitative research methods
  - Data science in education
- Former philosopher

- Recap

Foundational R skills

A general framework for you to use as a foundation and as a set of concepts to help you work through the class.

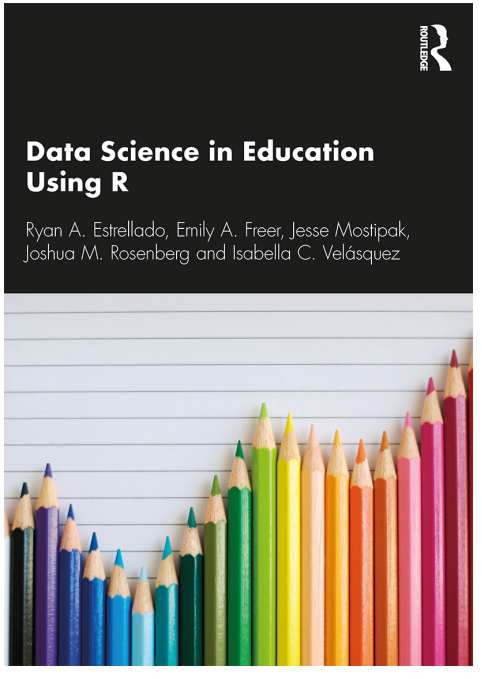
The four core concepts we will use to build our framework are:

1. Projects
2. Packages
3. Data
4. Functions

You will use each of these in most of your analyses with R.

Course texts

Data Science in Education Using R

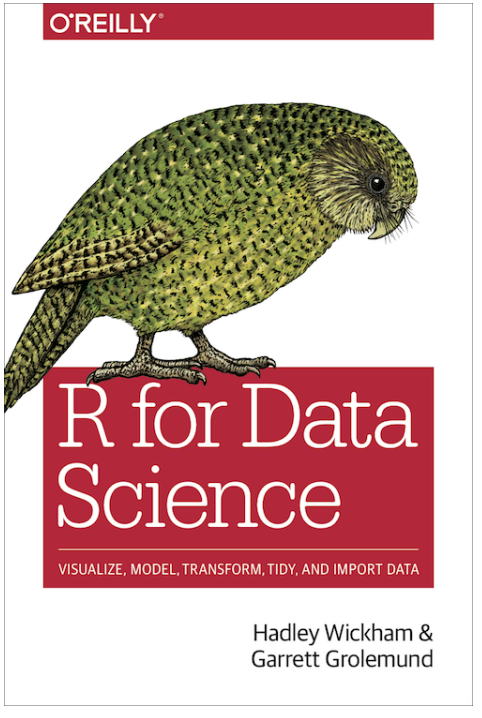


http://datascienceineducation.com/

https://github.com/data-edu/data-science-in-education

Course texts

R for Data Science



https://r4ds.had.co.nz/

1. Projects

Working Directories

```
getwd()
```

```
## [1] "/home/alishinski/Documents/work/FS21-Data-Sci-Methods/Course-Website/slides"
```

R Projects

Projects keep your work better contained and organized

Project data is saved using .RData - but we won't be using it!

The here package helps you to navigate around in a project:

```
library(here)

# data file
here("data", "fall-2018-data-file.csv")
```

2. Packages

1. What are packages?
    - Code bundles that add functionality to R
    - Examples: ggplot2, dplyr, rtweet, quanteda, lme4
2. Where do we get packages?
    - CRAN or GitHub
3. How do we install packages?
    - install.packages("pkg-name")
    - devtools::install_github("user/directory")
4. How do we know what packages to use?
    - Searching
    - People and news related to R (more later - there are *tons*)
    - CRAN task views
5. How do we use packages?
    - `library(pkgname)`

2. Installing another package

Tidyverse, a collection of R packages.

https://www.tidyverse.org/

Install via the following (do this now):

```
install.packages("tidyverse")
```

What issues have arisen?

3. Data

So far, we have used ***built-in data***. There is a lot of built-in data!

Loading different types of data

Comma-separated values (`.csv`)

```
library(readr)
readr::read_csv(here("data", "filename.csv"))
```

3. Data

.xlsx

```
library(readxl)
read_excel((here("data", "schedule.xlsx")))
```

3. Data

.sav

```
library(haven)
read_sav((here("data:, file-name.sav")))
```

3. Other data sources

Google Sheets

```
library(googlesheets4)
```

Web

```
read_csv("https://github.com/data-edu/dataedu/raw/master/data-raw/wt01_online-science-motivation/raw/
```

4. Functions

- A function is a reusable piece of code that allows us to consistently repeat a programming task
- Functions in R can be identified by a word followed by a set of parentheses, like so: `word()`.

More often than not, the word is a verb, such as `filter()`, suggesting that we're about to perform an action.

Indeed, functions act like verbs: they tell R what to do with our data.

The parentheses are where we can provide arguments.

4. Functions

- What is the name of the *package* used below?
- What is the name of the *data* used below?
- What is the name of the function used below?*

```
library(dplyr)
#mtcars
glimpse(mtcars)
```

4. select()

```
library(dplyr)

storms %>%
  select(name, year, month, day, hour, status)
```

```
## # A tibble: 10,010 × 6
##    name   year month   day  hour status
##    <chr> <dbl> <dbl> <int> <dbl> <chr>
##  1 Amy    1975     6    27     0 tropical depression
##  2 Amy    1975     6    27     6 tropical depression
##  3 Amy    1975     6    27    12 tropical depression
##  4 Amy    1975     6    27    18 tropical depression
##  5 Amy    1975     6    28     0 tropical depression
##  6 Amy    1975     6    28     6 tropical depression
##  7 Amy    1975     6    28    12 tropical depression
##  8 Amy    1975     6    28    18 tropical depression
##  9 Amy    1975     6    29     0 tropical storm
## 10 Amy    1975     6    29     6 tropical storm
## # … with 10,000 more rows
```

4. filter()

```
library(dplyr)

storms %>%
  filter(month == 8)
```

```
## # A tibble: 2,400 × 13
##    name      year month   day  hour   lat  long status   category  wind pressure
##    <chr>    <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>    <ord>    <int>    <int>
##  1 Caroline  1975     8    24    12  22.4 -69.8 tropica… -1          25     1011
##  2 Caroline  1975     8    24    18  21.9 -71.1 tropica… -1          25     1011
##  3 Caroline  1975     8    25     0  21.6 -72.5 tropica… -1          25     1010
##  4 Caroline  1975     8    25     6  21.2 -73.8 tropica… -1          25     1010
##  5 Caroline  1975     8    25    12  20.9 -75.1 tropica… -1          25     1011
##  6 Caroline  1975     8    25    18  20.6 -76.4 tropica… -1          25     1011
##  7 Caroline  1975     8    26     0  20.4 -77.7 tropica… -1          25     1011
##  8 Caroline  1975     8    26     6  20.3 -79   tropica… -1          25     1011
##  9 Caroline  1975     8    26    12  20.2 -80.3 tropica… -1          25     1012
## 10 Caroline  1975     8    26    18  20.2 -81.6 tropica… -1          25     1012
## # … with 2,390 more rows, and 2 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>
```

4. arrange()

```
library(dplyr)

storms %>%
  arrange(hour)
```

```
## # A tibble: 10,010 × 13
##    name      year month   day  hour   lat  long status   category  wind pressure
##    <chr>    <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>    <ord>    <int>    <int>
##  1 Amy       1975     6    27     0  27.5 -79   tropica… -1          25     1013
##  2 Amy       1975     6    28     0  31.5 -78.8 tropica… -1          25     1012
##  3 Amy       1975     6    29     0  34.4 -75.8 tropica… 0           35     1004
##  4 Amy       1975     6    30     0  34.3 -71.6 tropica… 0           50      998
##  5 Amy       1975     7     1     0  36.2 -69.8 tropica… 0           60      984
##  6 Amy       1975     7     2     0  37.4 -66.7 tropica… 0           60      984
##  7 Amy       1975     7     3     0  37.7 -62.8 tropica… 0           55      986
##  8 Amy       1975     7     4     0  42.5 -54.8 tropica… 0           50      986
##  9 Caroline  1975     8    25     0  21.6 -72.5 tropica… -1          25     1010
## 10 Caroline  1975     8    26     0  20.4 -77.7 tropica… -1          25     1011
## # … with 10,000 more rows, and 2 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>
```

4. Putting it together

```
library(dplyr)

storms %>%
  select(name, year, month, day, hour, status) %>%
  filter(month == 8) %>%
  arrange(hour)
```

```
## # A tibble: 2,400 × 6
##    name      year month   day  hour status
##    <chr>    <dbl> <dbl> <int> <dbl> <chr>
##  1 Caroline  1975     8    25     0 tropical depression
##  2 Caroline  1975     8    26     0 tropical depression
##  3 Caroline  1975     8    27     0 tropical depression
##  4 Caroline  1975     8    28     0 tropical depression
##  5 Caroline  1975     8    29     0 tropical depression
##  6 Caroline  1975     8    30     0 hurricane
##  7 Caroline  1975     8    31     0 hurricane
##  8 Doris     1975     8    30     0 tropical storm
##  9 Doris     1975     8    31     0 hurricane
## 10 Belle     1976     8     7     0 tropical storm
## # … with 2,390 more rows
```

4. Assignment operator

```
storms_in_august <- storms %>%
  select(name, year, month, day, hour, status) %>%
  filter(month == 8) %>%
  arrange(hour)
```

What is one thing that is different between `storms_in_august` and `storms` ?

4. Assignment operator

```
storms
```

```
## # A tibble: 10,010 × 13
##    name   year month   day  hour   lat  long status      category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>       <ord>    <int>    <int>
##  1 Amy    1975     6    27     0  27.5 -79   tropical d… -1          25     1013
##  2 Amy    1975     6    27     6  28.5 -79   tropical d… -1          25     1013
##  3 Amy    1975     6    27    12  29.5 -79   tropical d… -1          25     1013
##  4 Amy    1975     6    27    18  30.5 -79   tropical d… -1          25     1013
##  5 Amy    1975     6    28     0  31.5 -78.8 tropical d… -1          25     1012
##  6 Amy    1975     6    28     6  32.4 -78.7 tropical d… -1          25     1012
##  7 Amy    1975     6    28    12  33.3 -78   tropical d… -1          25     1011
##  8 Amy    1975     6    28    18  34   -77   tropical d… -1          30     1006
##  9 Amy    1975     6    29     0  34.4 -75.8 tropical s… 0           35     1004
## 10 Amy    1975     6    29     6  34   -74.8 tropical s… 0           40     1002
## # … with 10,000 more rows, and 2 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>
```

4. Assignment operator

```
storms_in_august
```

```
## # A tibble: 2,400 × 6
##    name      year month   day  hour status
##    <chr>    <dbl> <dbl> <int> <dbl> <chr>
##  1 Caroline  1975     8    25     0 tropical depression
##  2 Caroline  1975     8    26     0 tropical depression
##  3 Caroline  1975     8    27     0 tropical depression
##  4 Caroline  1975     8    28     0 tropical depression
##  5 Caroline  1975     8    29     0 tropical depression
##  6 Caroline  1975     8    30     0 hurricane
##  7 Caroline  1975     8    31     0 hurricane
##  8 Doris     1975     8    30     0 tropical storm
##  9 Doris     1975     8    31     0 hurricane
## 10 Belle     1976     8     7     0 tropical storm
## # … with 2,390 more rows
```

4. Assignment operator

```
ncol(storms)
```

```
## [1] 13
```

Tricky question: How many columns are present in `storms` after the following operation?

```
storms %>%
  select(name, year, status)
```

4. Assignment operators

How many columns are in storms after running the following two lines of code?

```
storms <- storms %>%
  select(name, year, status)
```

4. Pipe operator

We've been using the pipe operator `%>%` from the magrittr package

The pipe sends the results of a function (or object) from left side of pipe to next function after pipe.

So instead of this:

```
library(magrittr)
library(dplyr)

mtfilter <- dplyr::filter(mtcars, mpg < 20)
mtsubset <- dplyr::select(mtfilter, mpg, cyl, disp)
```

We can do this:

```
mtcars %>%
  dplyr::filter(mtcars, mpg < 20) %>%
  dplyr::select(mpg, cyl, disp)
```

4. Basic programming operators

Math

```
x <- 3 * 4
x
```

```
## [1] 12
```

```
y <- 2 + 3 -1
y
```

```
## [1] 4
```

```
z <- 4 / 2 ** 3
z
```

```
## [1] 0.5
```

4. Basic programming operators

Logic

```
x <- TRUE
y <- FALSE
# and
x & y
```

```
## [1] FALSE
```

```
# or
x | y
```

```
## [1] TRUE
```

```
# not
!x
```

```
## [1] FALSE
```

Demo

- Tour of Rstudio
- Projects
- Packages
- Files
- R scripts and Rmarkdown

Readings

One assigned readings for the next week:

- Data Science in Education Using R (DSIEUR) chapter #3: https://datascienceineducation.com/c03.html

Coming up

This week

- Readings (by next week)
- HW1 (Given Thursday)

Wrapping up

On slack:

- What is one thing you took away from today?
- What is something you want to learn more about?