Intro to R

Part 1: A Microcosm of Data Science

Prof. Bisbee

Vanderbilt University

Lecture Date: 2023/01/11

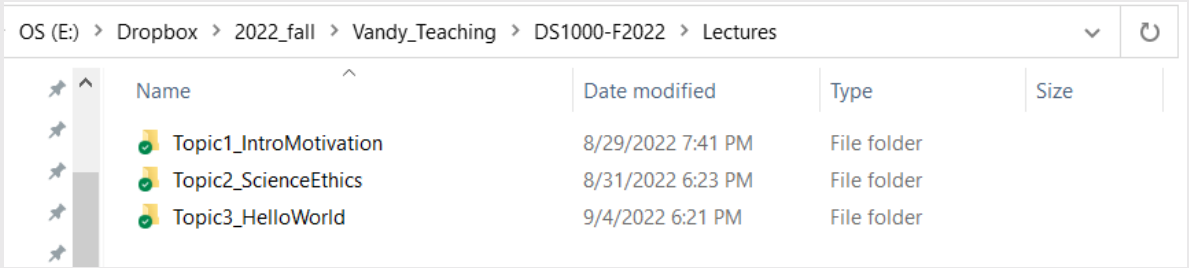
Slides Updated: 2023-01-03

Agenda

1. Getting set up
    - Folder structure + `setwd()`
2. Installing software
    - `R: https://cran.r-project.org/`
    - `RStudio: https://rstudio.com/products/rstudio/download/`
3. Requiring packages
    - `install.packages("tidyverse")`
    - `require(tidyverse)`
4. Loading and manipulating data
    - `readRDS()`
    - `%>%`

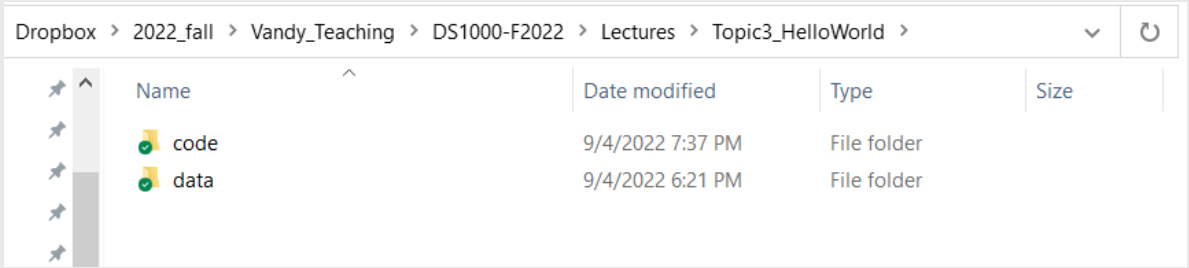
Getting set up

- Folder structure + `setwd()`
  - Concept: keep everything together...
  - ...and **related**



Getting set up

- Folder structure + `setwd()`
  - Concept: keep everything together...
  - ...and **related**



Installing software

- `R: https://cran.r-project.org/`
  - Accept all defaults
- `RStudio: https://rstudio.com/products/rstudio/download/`
  - Download the version for your OS
- Open `RStudio` and create a new rmarkdown (`.Rmd`) file
  - Accept defaults, give it a sensible name, delete the default text, then save it to your folder (again with a sensible name)
  - You should follow along with the lecture in this file! Take notes here! Try code here!

How to type in `.Rmd`

```
# This is a header

## This is a subheader

### This is a subsubheader

This is plain text.
```

This is a header

This is a subheader

This is a subsubheader

This is plain text.

How to type in `.Rmd`

```
- This is
- a bulleted
  - List
1. This is
2. a numbered list
```

- This is
- a bulleted
    - List

1. This is
2. a numbered list

How to type in `.Rmd`

```
*Bold font**, *italic font*, `code font`
```

Bold font, *italic font*, `code font`

- Most Important! `R` code!

````
```{r}
2+2
```
````

```
2+2
```

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

How R Works

- **O**bject **O**riented **L**anguage (**OOL**)
  - Objects are created with the `<-` command
  - You *can* run code directly...

```
2+2
```

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

How R Works

- **O**bject **O**riented **L**anguage (**OOL**)
  - Objects are created with the `<-` command
  - ...but most of what we'll do involves objects

```
object1 <- 2+2
object1
```

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

How R Works

- **O**bject **O**riented **L**anguage (**OOL**)
  - Objects are created with the `<-` command
  - They can be named anything (so be intuitive!)

```
three_plus_three <- 2+2
three_plus_three
```

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

How R Works

- **O**bject **O**riented **L**anguage (**OOL**)
  - Objects are created with the `<-` command
  - Objects can store many different things

```
an_element <- 2+2
a_vector <- c(1,2,3)
a_list <- list('element1' = 2+2,
               'element2' = "hello world!",
               'element3' = runif(n = 10,min = 0,max = 10))
a_function <- function(x) {
  avg_of_x <- sum(x) / length(x)
  return(avg_of_x)
}
```

How R Works

- Objects **persist!**

```
an_element # This object stores 2+2
```

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

```
a_vector   # This object stores the integers 1, 2, and 3
```

```
## [1] 1 2 3
```

```
an_element*a_vector
```

```
## [1]  4  8 12
```

```
an_element-a_vector
```

```
## [1] 3 2 1
```

How R Works

- Objects **persist!**

```
# This object stores:
  # 1) 2+2 (named "element1")
  # 2) the text "hello world!" (named "element2")
  # 3) 10 numbers randomly drawn between 0 and 10
a_list
```

```
## $element1
## [1] 4
##
## $element2
## [1] "hello world!"
##
## $element3
##  [1] 2.875775 7.883051 4.089769 8.830174 9.404673 0.455565
##  [7] 5.281055 8.924190 5.514350 4.566147
```

How R Works

- Objects **persist!**

```
# Let's apply our function ("a_function") to "element3" in "a_list"
a_function(x = a_list[['element3']])
```

```
## [1] 5.782475
```

- We could also call `element3` from `a_list` with a \$ sign

```
a_function(x = a_list$element3)
```

```
## [1] 5.782475
```

How `RStudio` Works

- `RStudio` is a powerful way to interact with `R`
- In base `R`, you interact with the program via the "command line"
    - For example...
- But to save your work, you can write "scripts"
    - For example...
- *This is all cumbersome!*
- Enter, `RStudio`

How `RStudio` Works

- `RStudio` allows us to:

    1. Write scripts
    2. Run scripts
    3. See results

- It is **deeply interactive**

    - We can highlight a line and press `ctrl+enter` / `cmd+enter` and see the result

    - **We can even do this with single objects!**

Functions & Packages

- What are `packages`?
  - Basically, **functions** that someone else wrote
- `R` has many functions already installed
  - These are known as "base `R`" and contain many useful functions
  - For example, `sum()` will add up a vector of numbers

```
sum(object1)
```

- Quiz: What does the `mean()` function do? The `median()`? The `range()`?
- Other base `R` functions interact with other files
  - For example, `read.csv()` will load a `.csv` file
- And there are MANY **MANY** more

Installing Packages

- In addition to the functions included in base `R`, we want more
- For this class, we want one called `tidyverse`
  - `tidyverse` contains many (hundreds?) of functions that make `R` easier
  - But it is NOT included in the base `R` set of functions
  - Therefore, we need to add it
- Use the base `R` function `install.packages("[PACKAGE NAME]")`
  - Specifically, `install.packages("tidyverse")`

Requiring Packages

- Once installed, a package will live somewhere on your computer
- However, any new *instance* of `R` will not automatically load the packages
- We need to `require()` them to tell `R` to load them
    - Alternatively, we can use `library()` (but it's the same result)
- So load the `tidyverse` package with `require(tidyverse)`
- NB: you need quotes for the `install.packages()` function...
    - i.e., `install.packages("tidyverse")`
- but NOT for the `require()` function
    - i.e., `require(tidyverse)`

Loading Data

- So you should be using `R` via `RStudio` with the `tidyverse` package loaded
- Now let's load some data
- First, download the `sc_debt.Rds` file from the course github page
  - Save it to the `./data` folder that you created
- Second, load the data with `readRDS("[PATH TO DATA]/sc_debt.Rds")`
  - NB: `R` is an "object-oriented language" (OOL)
  - We **create** an "object" to store the data using a left-arrow: `<-`

```
df<-readRDS("../data/sc_debt.Rds")
```

- `../` means "go up one folder"

Loading Data

- We now have the contents of `sc_debt.Rds` stored in the object `df`
- We can look at this object directly

```
df
```

```
## # A tibble: 2,546 × 16
##    unitid instnm       stabbr grad_…¹ control region preddeg
##     <int> <chr>        <chr>    <int> <chr>   <chr>  <chr>
##  1 100654 Alabama A &… AL       33375 Public  South… Bachel…
##  2 100663 University … AL       22500 Public  South… Bachel…
##  3 100690 Amridge Uni… AL       27334 Private South… Associ…
##  4 100706 University … AL       21607 Public  South… Bachel…
##  5 100724 Alabama Sta… AL       32000 Public  South… Bachel…
##  6 100751 The Univers… AL       23250 Public  South… Bachel…
##  7 100760 Central Ala… AL       12500 Public  South… Associ…
##  8 100812 Athens Stat… AL       19500 Public  South… Bachel…
##  9 100830 Auburn Univ… AL       24826 Public  South… Bachel…
## 10 100858 Auburn Univ… AL       21281 Public  South… Bachel…
## # … with 2,536 more rows, 9 more variables: openadmp <int>,
## #   adm_rate <dbl>, ccbasic <int>, sat_avg <int>,
## #   md_earn_wne_p6 <int>, ugds <int>, costt4_a <int>,
```

Loading Data

- Or we can look at its columns

```
names(df)
```

```
##  [1] "unitid"          "instnm"          "stabbr"
##  [4] "grad_debt_mdn"   "control"         "region"
##  [7] "preddeg"         "openadmp"        "adm_rate"
## [10] "ccbasic"         "sat_avg"         "md_earn_wne_p6"
## [13] "ugds"            "costt4_a"        "selective"
## [16] "research_u"
```

Loading Data

Name	Definition
unitid	Unit ID
instnm	Institution Name
stabbr	State Abbreviation
grad_debt_mdn	Median Debt of Graduates
control	Control Public or Private
region	Census Region
preddeg	Predominant Degree Offered: Assocates or Bachelors
openadmp	Open Admissions Policy: 1=Yes, 2=No, 3=No 1st time students
adm_rate	Admissions Rate: proportion of applications accepted
ccbasic	Type of institution*
sat_avg	Average SAT scores
md_earn_wne_p6	Average Earnings of Recent Graduates
ugds	Number of undergraduates
costt4_a	Average cost of attendance (tuition-grants)
selective	Institution admits fewer than 10% of applications, 1=Yes, 0=No
research_u	Institution is a research university, 1=Yes, 0=No

Manipulating the Data

- These data are cool!
- But TMI at first
- I want to know...
  - Where is `Vanderbilt University`?
  - Who is the most selective?
  - Which schools produce the richest grads?

Manipulating: `filter()`

- `filter` will select **rows** of the data based on some criteria

```
df %>%
  filter(instnm == "Vanderbilt University")
```

```
## # A tibble: 1 × 16
##   unitid instnm         stabbr grad_…¹ control region preddeg
##    <int> <chr>          <chr>    <int> <chr>   <chr>  <chr>
## 1 221999 Vanderbilt U… TN       14962 Private South… Bachel…
## # … with 9 more variables: openadmp <int>, adm_rate <dbl>,
## #   ccbasic <int>, sat_avg <int>, md_earn_wne_p6 <int>,
## #   ugds <int>, costt4_a <int>, selective <dbl>,
## #   research_u <dbl>, and abbreviated variable name
## #   ¹grad_debt_mdn
```

Manipulating: `select()`

- Still TMI!
- `select` will select **columns**

```
df %>%
  filter(instnm == "Vanderbilt University") %>%
  select(instnm,adm_rate,selective,sat_avg,md_earn_wne_p6)
```

```
## # A tibble: 1 × 5
##   instnm                adm_rate selective sat_avg md_earn…¹
##   <chr>                    <dbl>     <dbl>   <int>     <int>
## 1 Vanderbilt University   0.0912         1    1515     53400
## # … with abbreviated variable name ¹md_earn_wne_p6
```

Manipulating: `arrange()`

- How does Vandy compare?
- `arrange` will sort the data based on a column (ascending!)

```
df %>%
  filter(adm_rate < .1) %>%
  arrange(sat_avg,adm_rate) %>%
  select(instnm,adm_rate,sat_avg)
```

```
## # A tibble: 25 × 3
##    instnm                                    adm_r…¹ sat_avg
##    <chr>                                       <dbl>   <int>
##  1 Colby College                              0.0967    1456
##  2 Swarthmore College                         0.0893    1469
##  3 Pomona College                             0.074     1480
##  4 Dartmouth College                          0.0793    1500
##  5 Stanford University                        0.0434    1503
##  6 Northwestern University                    0.0905    1506
##  7 Columbia University in the City of New Y…  0.0545    1511
##  8 Brown University                           0.0707    1511
##  9 University of Pennsylvania                 0.0766    1511
## 10 Vanderbilt University                      0.0912    1515
```

How does Vandy compare?

- `arrange` in descending order

```
df %>%
  filter(adm_rate < .1) %>%
  arrange(-sat_avg,adm_rate) %>%
  select(instnm,adm_rate,sat_avg)
```

```
## # A tibble: 25 × 3
##    instnm                                  adm_r…¹ sat_avg
##    <chr>                                     <dbl>   <int>
##  1 California Institute of Technology       0.0642    1557
##  2 Massachusetts Institute of Technology    0.067     1547
##  3 University of Chicago                    0.0617    1528
##  4 Duke University                          0.076     1522
##  5 Rice University                          0.0872    1520
##  6 Harvard University                       0.0464    1517
##  7 Princeton University                     0.0578    1517
##  8 Yale University                          0.0608    1517
##  9 Vanderbilt University                    0.0912    1515
## 10 Columbia University in the City of New Y… 0.0545    1511
## # … with 15 more rows, and abbreviated variable name
## #   ¹adm_rate
```

More complicated? More %>%!

- Less selective schools by SAT with debt and state

```
df %>%
  filter(adm_rate > .2 & adm_rate < .3) %>%
  arrange(stabbr,-sat_avg) %>%
  select(instnm,sat_avg,grad_debt_mdn,stabbr)
```

```
## # A tibble: 37 × 4
##    instnm                                sat_avg grad_…¹ stabbr
##    <chr>                                   <int>   <int> <chr>
##  1 Heritage Christian University              NA      NA AL
##  2 University of California-Santa Ba…       1370   15000 CA
##  3 California Polytechnic State Univ…       1342   19501 CA
##  4 University of California-Irvine          1306   15488 CA
##  5 California Institute of the Arts           NA   27000 CA
##  6 University of Miami                      1371   17125 FL
##  7 Georgia Institute of Technology-M…       1418   23000 GA
##  8 Point University                          986   26000 GA
##  9 Grinnell College                         1457   17500 IA
## 10 St Luke's College                          NA   17750 IA
## # … with 27 more rows, and abbreviated variable name
## #   ¹grad_debt_mdn
```

A quick aside on missingness

- Some rows have `NA` in some columns
  - `NA` is the standard code for **missing data** in `R`
  - Data can be missing for many different reasons (i.e., some schools don't require SAT scores or record them)
- We will return to this in the lectures on **data wrangling**

```
df %>%
  filter(is.na(sat_avg)) %>%
  select(instnm,stabbr)
```

```
## # A tibble: 1,317 × 2
##    instnm                                    stabbr
##    <chr>                                     <chr>
##  1 Amridge University                        AL
##  2 Central Alabama Community College         AL
##  3 Athens State University                   AL
##  4 Chattahoochee Valley Community College    AL
##  5 Coastal Alabama Community College         AL
##  6 Gadsden State Community College           AL
```

Stepping back

- Thus far, lots of data
- Not a lot of science
- RQ: How might admissions and SAT scores be **related**?
  - Theory: selective schools have stricter criteria
  - Hypothesis: admissions and SAT scores should be **negatively** related
- How can we test this hypothesis?

Summarizing Data: `summarise()` + `mean()`

- We can combine base `R` functions with `tidyverse` functions!
  - Base `R`: `mean()`
  - `tidyverse`: `summarise()` (aka `summarize()`)
- Overall average SAT scores

```
df %>%
  summarise(mean_sat = mean(sat_avg,na.rm=T))
```

```
## # A tibble: 1 × 1
##   mean_sat
##      <dbl>
## 1    1141.
```

Summarizing Data

- Let's unpack this

```
df %>%
  summarise(mean_sat = mean(sat_avg,na.rm=T))
```

- Create new variable `mean_sat` that contains the `mean()` of every school's average SAT score
- `na.rm=T` means we want to ignore missing data. If not?

```
df %>%
  summarise(mean_sat = mean(sat_avg))
```

```
## # A tibble: 1 × 1
##   mean_sat
##      <dbl>
## 1       NA
```

Summarizing Data

- Recall we want see if more selective schools have higher SAT scores

```
df %>%
  filter(adm_rate < .1) %>%
  summarise(mean_sat_LT10 = mean(sat_avg,na.rm=T))
```

```
## # A tibble: 1 × 1
##   mean_sat_LT10
##           <dbl>
## 1         1510.
```

```
df %>%
  filter(adm_rate > .1 & adm_rate < .2) %>%
  summarise(mean_sat_1020 = mean(sat_avg,na.rm=T))
```

```
## # A tibble: 1 × 1
##   mean_sat_1020
##           <dbl>
## 1         1424.
```

Summarizing Data: `group_by()`

- One final `tidyverse` function: `group_by()`

```
df %>%
  group_by(selective) %>%
  summarise(mean_sat = mean(sat_avg,na.rm=T))
```

```
## # A tibble: 3 × 2
##   selective mean_sat
##       <dbl>    <dbl>
## 1         0    1135.
## 2         1    1510.
## 3        NA     NaN
```

Conclusion

- What we've done today is a microcosm of data science
    1. Opened data (`readRDS`)
    2. Looked at data (`tidyverse` + `select()`, `filter()`, `arrange()`)
    3. Generated hypotheses (Admissions versus SAT scores)
    4. Tested hypotheses (`summarise()` + `mean()`)

Quiz & Homework

- Go to Brightspace and take the **2nd** quiz
    - The password to take the quiz is 4695
- **Homework:**

    1. Work through Intro_to_R_Part1_hw.Rmd