

# Intro to R

## Part 1: A Microcosm of Data Science

Prof. Bisbee

Vanderbilt University

Lecture Date: Slides Updated: 2024-08-10

# 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**

OS (E:) > Dropbox > 2022_fall > Vandy_Teaching > DS1000-F2022 > Lectures					▼	🔄
➤	^	Name	Date modified	Type	Size	
➤		📁 Topic1_IntroMotivation	8/29/2022 7:41 PM	File folder		
➤		📁 Topic2_ScienceEthics	8/31/2022 6:23 PM	File folder		
➤		📁 Topic3_HelloWorld	9/4/2022 6:21 PM	File folder		
➤						

# Getting set up

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

Dropbox > 2022_fall > Vandy_Teaching > DS1000-F2022 > Lectures > Topic3_HelloWorld >					▼	🔄
↗	^	Name	Date modified	Type	Size	
↗		code	9/4/2022 7:37 PM	File folder		
↗		data	9/4/2022 6:21 PM	File folder		
↗						
↗						

# 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 Importantly! R code!

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

```
2+2
```

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



# How R Works

- **Object Oriented Language (OOL)**
  - Objects are created with the `<-` command
  - You *can* run code directly...

```
2+2
```

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

# How R Works

- **Object Oriented Language (OOL)**
  - Objects are created with the `<-` command
  - ...but most of what we'll do involves objects

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

- Object assignment operator **saves** the output
- It **does not print** the output
- To see, just call the object

```
object1
```

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

# How R Works

- **Object Oriented Language (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

- **Object Oriented Language (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
```

# A comment on comments

- If you use a # sign inside a code chunk, you can write a comment

```
# This is a comment. If I compile the code, nothing will happen.
```

```
# This is another comment. These are helpful for annotating my code.
```

# 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 dollar sign

```
# This does the same thing as the previous slide...it just accesses  
element3 differently.  
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!**

# Give it a try

- Comment **everything**
- Create two objects
  - `a` contains the product of 3 and 5 (`3*5`)
  - `b` contains five numbers `c(10,21,43,87,175)`
- Now create object `c` which is `a - b`

```
# INSERT CODE HERE
```

# 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
- You can save it locally from the course [github page](#) and then load it from your computer
- Or you can load it directly from the internet with the **read\_rds()** function from **tidyverse**
  - NB: **R** is an "object-oriented language" (OOL)
  - We **create** an "object" to store the data using a left-arrow: **<-**

```
df <-  
read_rds('https://github.com/jbisbee1/DS1000_F2024/raw/main/data/sc_deb')
```

# Loading Data

- We now have the contents of `sc_debt.Rds` stored in the object `df`
  - This is a "tabular data frame", aka a `tibble`
  - **Rows** are observations
  - **Columns** are values
- We can look at this object directly

```
df
```

```
## # A tibble: 2,546 × 16
##   unitid instnm stabbr grad_debt_mdn control region preddeg
##   <int> <chr>   <chr>         <int> <chr>   <chr>   <chr>
## 1 100654 Alaba... AL             33375 Public  South... Bachel...
## 2 100663 Unive... AL             22500 Public  South... Bachel...
## 3 100690 Amrid... AL             27334 Private South... Associ...
## 4 100706 Unive... AL             21607 Public  South... Bachel...
## 5 100724 Alaba... AL             32000 Public  South... Bachel...
## 6 100751 The U... AL             23250 Public  South... Bachel...
## 7 100760 Centr  AL             12500 Public  South   Associ
```



# 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: Associates 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?
- There are `tidyverse` functions to answer all of these questions

# Manipulating with tidyverse

- The code process of tidyverse relies on a "pipe" symbol: %>%
  - I don't like this name
  - I think it should be called a "chain" because it **links code together**
  - Or maybe a "do" symbol because it tells R what to do
- The basic grammar of R is: object, %>%, verb

```
object %>% # This is the object  
function() # This is the verb
```

# Manipulating with `tidyverse`

- `tidyverse` has many useful "verbs" (i.e., functions)
  - `filter()`: subsets **rows**
  - `select()`: subsets **columns**
  - `arrange()`: sorts **rows** based on **columns**
  - `summarise()`: collapses **rows**
  - `group_by()`: groups **rows** by **columns**

# Manipulating: `filter()`

- So let's look at Vandy
- `filter` will select **rows** of the data based on some criteria

```
df %>%  
  filter(instnm == "Vanderbilt University") # Only select rows with  
  Vandy
```

```
## # A tibble: 1 × 16  
##   unitid instnm  stabbr grad_debt_mdn control region preddeg  
##   <int> <chr>   <chr>         <int> <chr>   <chr> <chr>  
## 1 221999 Vander... TN             14962 Private South... Bachel...  
## # i 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>
```

# Manipulating: `select()`

- Still TMI!
- I only care about the admissions rate (`adm_rate`), the SAT scores (`sat_avg`), and the future earnings (`md_earn_wne_p6`)
- `select` will select **columns**

```
df %>%  
  filter(instnm == "Vanderbilt University") %>%  
  select(instnm, adm_rate, sat_avg, md_earn_wne_p6) # Only select four  
columns
```

```
## # A tibble: 1 × 4  
##   instnm          adm_rate sat_avg md_earn_wne_p6  
##   <chr>          <dbl>   <int>         <int>  
## 1 Vanderbilt University  0.0912    1515         53400
```

# Manipulating: `arrange()`

- How does Vandy compare...?
  - to other schools in terms of SAT scores?
  - to other schools in terms of future earnings?
  - to other schools in terms of admissions rates?
- `arrange` will sort the data based on a column (ascending!)

```
df %>%  
  arrange(sat_avg) %>% # Sort data by SAT scores  
  select(instnm, sat_avg) # Only look at name and SAT scores
```

```
## # A tibble: 2,546 × 2  
##   instnm                sat_avg  
##   <chr>                <int>  
## 1 Morgan State University      737  
## 2 Saint Augustine's University  847  
## 3 Albany State University      849  
## 4 Holy Names University        851  
## 5 Livingstone College          854
```



# Manipulating: `arrange()`

- Vandy is not in the bottom 10 schools

```
df %>%  
  arrange(sat_avg) %>% # Sort data by SAT scores  
  select(instnm,sat_avg) # Only look at name and SAT scores
```

```
## # A tibble: 2,546 × 2  
##   instnm                sat_avg  
##   <chr>                <int>  
## 1 Morgan State University      737  
## 2 Saint Augustine's University  847  
## 3 Albany State University      849  
## 4 Holy Names University        851  
## 5 Livingstone College          854  
## 6 Virginia Union University    855  
## 7 Manor College                861  
## 8 Saint Louis Christian College  865  
## 9 Bacone College               875  
## 10 Paine College               876  
## # i 2,536 more rows
```

# Manipulating: `arrange()`

- Use `desc()` to order in descending values...Vandy not in top 10 either

```
df %>%  
  arrange(desc(sat_avg)) %>% # Sort data by SAT scores (descending)  
  select(instnm,sat_avg) # Only look at name and SAT scores
```

```
## # A tibble: 2,546 × 2  
##   instnm                                sat_avg  
##   <chr>                                <int>  
## 1 California Institute of Technology    1557  
## 2 Massachusetts Institute of Technology 1547  
## 3 University of Chicago                 1528  
## 4 Harvey Mudd College                   1526  
## 5 Duke University                       1522  
## 6 Franklin W Olin College of Engineering 1522  
## 7 Washington University in St Louis     1520  
## 8 Rice University                       1520  
## 9 Yale University                       1517  
## 10 Harvard University                   1517  
## # i 2,536 more rows
```

# Manipulating: `arrange()`

- What if we look only at "selective" schools (i.e., those who accept less than 10% of applicants)?

```
df %>%  
  filter(adm_rate < .1) %>% # Only look at schools who accept less  
  than 10%  
  arrange(sat_avg,adm_rate) %>% # Sort data by SAT scores AND THEN  
  admissions rates (breaks ties)  
  select(instnm,adm_rate,sat_avg) # Only look at name, admissions  
  rate, and SAT scores
```

```
## # A tibble: 25 × 3  
##   instnm                adm_rate 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 ... 0.0545   1511  
## 8 Brown University     0.0707   1511
```

# How does Vandy compare?

- `arrange` in descending order

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

```
## # A tibble: 25 × 3  
##   instnm                                adm_rate 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 ... 0.0545   1511  
## # i 15 more rows
```

# More complicated? More %>%!

- Less selective schools by SAT with debt and state

```
df %>%  
  filter(adm_rate > .2 & adm_rate < .3) %>% # Less selective schools  
  (accept between 20% and 30%)  
  arrange(stabbr, desc(sat_avg)) %>% # Sort by state name, then by SAT  
  scores  
  select(instnm, sat_avg, grad_debt_mdn, stabbr) # Only look at some  
  columns
```

```
## # A tibble: 37 × 4  
##   instnm          sat_avg grad_debt_mdn stabbr  
##   <chr>          <int>         <int> <chr>  
## 1 Heritage Christian Universi...    NA            NA AL  
## 2 University of California-Sa...  1370          15000 CA  
## 3 California Polytechnic Stat...  1342          19501 CA  
## 4 University of California-Ir...  1306          15488 CA  
## 5 California Institute of the...    NA          27000 CA  
## 6 University of Miami           1371          17125 FL  
## 7 Georgia Institute of Techno...  1418          23000 GA  
## 8 Point University             986          26000 GA  
## 9 Grinnell College             1457          17500 IA
```

# 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 can use a base **R** function called `is.na()` which will be **TRUE** if the value is **NA** or **FALSE** otherwise
  - And we can combine `is.na()` with the `filter()` function from **tidyverse**
- We will return to this in the lectures on **data wrangling**
- For now, how many schools don't report SAT scores?

```
df %>%  
  filter(is.na(sat_avg)) %>% # Only look at schools that DON'T report  
SATs  
  select(instnm, stabbr) # Only look at the name and the state
```

# Stepping back

- Thus far, lots of **data**
- Not a lot of **science**
- But remember the **Research** camp!
  1. **Observation** → **Question**
  2. **Theory** → **Hypothesis**
  3. **Data Collection / Wrangling** → **Analysis**
  4. **Results** → **Conclusion**
- We have been doing lots of **Observation**!
- Do we have any good **Research questions**?

# Stepping back

- **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)) # Average SAT scores  
for entire data
```

```
## # 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) %>% # Only look at schools who accept less  
  than 10% of applicants  
  summarise(mean_sat_LT10 = mean(sat_avg, na.rm=T)) # Calculate the  
  average SAT score
```

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

```
df %>%  
  filter(adm_rate > .1) %>% # Only look at schools who accept more  
  than 10% of applicants  
  summarise(mean_sat_GT20 = mean(sat_avg, na.rm=T)) # Calculate the  
  average SAT score
```

```
## # A tibble: 1 × 1  
##   mean_sat_GT20
```

# Summarizing Data: `group_by()`

- One final `tidyverse` function: `group_by()`
- There is a column called `selective` which is either 1 or 0
  - 1: the admissions rate is less than 10%
  - 0: otherwise

```
df %>%  
  select(instnm,selective,adm_rate)
```

```
## # A tibble: 2,546 × 3  
##   instnm                selective adm_rate  
##   <chr>                <dbl>   <dbl>  
## 1 Alabama A & M University      0     0.918  
## 2 University of Alabama at Birmingham 0     0.737  
## 3 Amridge University           NA     NA  
## 4 University of Alabama in Huntsville 0     0.826  
## 5 Alabama State University      0     0.969  
## 6 The University of Alabama      0     0.827  
## 7 Central Alabama Community College NA     NA  
## 8 Athens State University       NA     NA
```

# Summarizing Data: `group_by()`

- Instead of running two separate `filter()` commands, use `group_by()`

```
df %>%  
  group_by(selective) %>% # Group the data by selective (either 1 or  
  0)  
  summarise(mean_sat = mean(sat_avg, na.rm=T)) # Calculate average SAT  
  for each group
```

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

# Results

- Do more selective schools have higher SAT scores?
- Yes
- This **Result** **confirms** our **Hypothesis** and **answers** our **Research Question**

# 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 **1st** quiz
  - The password to take the quiz is ####
- **Homework:**
  1. Work through ds1000\_hw\_2.Rmd
  2. Problem set 1 due Friday by midnight