Welcome to Data Science!

Today, we'll be working on getting you set up with the tools you will need for this class. Once you are set up, we'll do what we're here to do: analyze data!

Here's what we need to get done today:

1. Getting started with R
2. Getting started with RStudio
3. Starting assignment 1, "Hello, world!"

Introductions

We need three basic sets of tools for this class. We will need `R` to analyze data. We will need `RStudio` to help us interface with R and to produce documentation of our results.

Installing R

R is going to be the only programming language we will use. R is an extensible statistical programming environment that can handle all of the main tasks that we'll need to cover this semester: getting data, analyzing data and communicating data analysis.

If you haven't already, you need to download R here: https://cran.r-project.org/.

Installing RStudio

When we work with R, we communicate via the command line. To help automate this process, we can write scripts, which contain all of the commands to be executed. These scripts generate various kinds of output, like numbers on the screen, graphics or reports in common formats (pdf, word). Most programming languages have several **I** ntegrated **D** evelopment **E** nvironments (IDEs) that encompass all of these elements (scripts, command line interface, output). The primary IDE for R is RStudio.

If you haven't already, you need to download RStudio here: https://rstudio.com/products/rstudio/download/. You need the free RStudio desktop version.

Yes We Code! Running R Code

The following code chunks will be our first use of R in this class. We're going to grab some data that's part of the college scorecard and do a bit of analysis on it.

.Rmd files

Open the `Lecture2_HelloWorld.Rmd` file. In RStudio, go to File–>Open, then find the `Lecture2_HelloWorld.Rmd` file in the directory.

.Rmd files will be the only file format we work in this class. .Rmd files contain three basic elements:

1. Script that can be interpreted by R.
2. Output generated by R, including tables and figures.
3. Text that can be read by humans.

From a .Rmd file you can generate html documents, pdf documents, word documents, slides . . . lots of stuff. All class notes will be in .Rmd. Most assignments will be turned in as .Rmd files, and the guided exercise we'll have you do? You guessed it, .Rmd.

In the `.Rmd` file you'll notice that there are three open single quotes in a row, like so: ``` This indicates the start of a "code chunk" in our file. The first code chunk that we load will include a set of programs that we will need all semester long.

Using R Libraries

When we say that R is extensible, we mean that people in the community can write programs that everyone else can use. These are called "packages." In these first few lines of code, I load a set of packages using the library command in R. The set of packages, called `tidyverse` were written by Hadley Wickham and others and play a key role in his book. To install this set of packages, simply type in `install.packages("tidyverse")` at the R command prompt.

To run the code below in R, you can:

- Press the "play" button next to the code chunk
- In OS X, place the cursor in the code chunk and hit `CMD+RETURN`
- In Windows, place the cursor in the code chunk and hit `CTRIL+RETURN`

```
## Get necessary libraries-- won't work the first time, because you need to install them!
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Loading Datasets

Now we're ready to load in data. The data frame will be our basic way of interacting with everything in this class. The data frame contains information from the college scorecard on different colleges and universities.

However, we first need to make sure that R is looking in the right place. When you opened up your project, RStudio automagically took you to the directory for that project. But because we keep lessons in a separate directory, we need to point R to the right place. This is called setting the working directory, and can be done either by using the command `setwd` or in RStudio by going to "Session–>Set Working Directory–>Choose Directory." Choose the directory where the file currently resides on your computer. Make sure to always set the working directory at the beginning of each session—not doing so causes a lot of headaches for new users.

```
df<-readRDS("sc_debt.Rds")
names(df)
```

```
##  [1] "unitid"         "instnm"         "stabbr"         "grad_debt_mdn"
##  [5] "control"        "region"         "preddeg"        "openadmp"
##  [9] "adm_rate"       "ccbasic"        "sat_avg"        "md_earn_wne_p6"
## [13] "ugds"           "selective"      "research_u"
```

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– see here
selective	Institution admits fewer than 10 % of applicants, 1=Yes, 0=No
research_u	Instituition is a research university 1=Yes, 0=No
sat_avg	Average Sat Scores
md_earn_wne_p6	Average Earnings of Recent Graduates
ugds	Number of undergraduates

Looking at datasets

We can look at the first few rows and columns of `df` by typing in the data name.

```
df
```

```
## # A tibble: 2,555 x 15
##    unitid instnm   stabbr grad_debt_mdn control region preddeg openadmp adm_rate
##     <int> <chr>    <chr>          <dbl> <chr>   <chr>  <chr>      <int>    <dbl>
##  1 132657 Lynn Un~ FL             17556 Private Soutw~ Bachel~        2    0.704
##  2 130217 Quineba~ CT                NA Public  New E~ Associ~        1   NA
##  3 132851 College~ FL             13140 Public  Soutw~ Associ~        1   NA
##  4 135364 Luther ~ GA             29875 Private Soutw~ Bachel~        2    0.591
##  5 135391 State C~ FL             10413 Public  Soutw~ Associ~        1   NA
##  6 134097 Florida~ FL             19002 Public  Soutw~ Bachel~        2    0.368
##  7 135717 Miami D~ FL              9500 Public  Soutw~ Associ~        1   NA
##  8 138947 Clark A~ GA             27000 Private Soutw~ Bachel~        2    0.518
##  9 138187 Valenci~ FL              9208 Public  Soutw~ Associ~        1   NA
## 10 138354 The Uni~ FL             17250 Public  Soutw~ Bachel~        2    0.424
## # ... with 2,545 more rows, and 6 more variables: ccbasic <int>, sat_avg <dbl>,
## #   md_earn_wne_p6 <int>, ugds <int>, selective <dbl>, research_u <dbl>
```

We can look at the whole dataset using View. Just delete the `#` sign below to make the code work. That `#` sign is a comment in R code, which indicates to the computer that everything on that line should be ignored. To get it to run, we need to drop the `#`.

```
#View(df)
```

You'll notice that this data is arranged in a rectangular format, with each row showing a different college, and each column representing a different characteristic of that college. Datasets are always structured this way— cases (or units) will form the rows, and the characteristics of those cases– or variables— will form the columns. Unlike working with spreadsheets, this structure is always assumed for datasets.

Filter, Select, Arrange

In exploring data, many times we want to look at smaller parts of the dataset. There are three commands we'll use today that help with this.

-`filter` selects only those cases or rows that meet some logical criteria.

-`select` selects only those variables or columns that meet some criteria

-`arrange` arranges the rows of a dataset in the way we want.

For more on these, please see this vignette.

Let's grab just the data for Vanderbilt, then look only at the average test scores and admit rate. We can use filter to look at all of the variables for Vanderbilt:

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

```
## # A tibble: 1 x 15
##   unitid instnm    stabbr grad_debt_mdn control region preddeg openadmp adm_rate
##    <int> <chr>     <chr>          <dbl> <chr>   <chr>  <chr>      <int>    <dbl>
## 1 221999 Vanderbi~ TN             14962 Private Soutw~ Bachel~        2   0.0961
## # ... with 6 more variables: ccbasic <int>, sat_avg <dbl>,
## #   md_earn_wne_p6 <int>, ugds <int>, selective <dbl>, research_u <dbl>
```

What's that weird looking `%>%` thing? That's called a pipe. This is how we chain commands together in R. Think of it as saying "and then" to R. In the above case, we said, take the data *and then* filter it to be just the data where the institution name is Vanderbilt University.

Many times, though we don't want to see everything, we just want to choose a few variables. `select` allows us to select only the variables we want. In this case, the institution name, its admit rate, and the average SAT scores of entering students.

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

```
## # A tibble: 1 x 3
##   instnm                adm_rate sat_avg
##   <chr>                    <dbl>   <dbl>
## 1 Vanderbilt University   0.0961    1514
```

`filter` takes logical tests as its argument. The code `insntnm=="Vanderbilt University"` is a logical statement that will be true of just one case in the dataset– when institution name is Vanderbilt University. The `==` is a logical test, asking if this is equal to that. Other common logical and relational operators for R include

- `>`, `<`: greater than, less than
- `>=`, `<=`: greater than or equal to, less than or equal to
- `!` :not, as in `!=` not equal to
- `&` AND
- `|` OR

Next, we can use `filter` to look at colleges with low admissions rates, say less than 10% (or .1 in the proportion scale used in the dataset).

```
df%>%
  filter(adm_rate<.1)%>%
  select(instnm,adm_rate,sat_avg)%>%
  arrange(sat_avg,adm_rate)%>%print(n=20)
```

```
## # A tibble: 23 x 3
##    instnm                                    adm_rate sat_avg
##    <chr>                                        <dbl>   <dbl>
##  1 Claremont McKenna College                   0.0931    1446
##  2 Swarthmore College                          0.0949    1465
##  3 Pomona College                              0.0761    1468
##  4 Dartmouth College                           0.0874    1488
##  5 Brown University                            0.0767    1492
##  6 University of Pennsylvania                  0.0841    1492
##  7 Stanford University                         0.0436    1497
##  8 Princeton University                        0.0548    1503
##  9 Northwestern University                     0.0847    1508
## 10 Columbia University in the City of New York   0.0591    1512
## 11 Vanderbilt University                       0.0961    1514
## 12 Duke University                             0.0891    1516
## 13 Yale University                             0.0635    1517
## 14 Harvard University                          0.0473    1520
## 15 University of Chicago                       0.0726    1520
## 16 Massachusetts Institute of Technology       0.0674    1545
## 17 California Institute of Technology          0.0662    1566
## 18 Saint Elizabeth College of Nursing          0           NA
## 19 Saint Anthony College of Nursing            0           NA
## 20 Belanger School of Nursing                  0           NA
## # ... with 3 more rows
```

Now let's look at colleges with low admit rates, and order them by SAT scores (`-sat_avg` gives descending order).

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

```
## # A tibble: 23 x 3
##    instnm                                    adm_rate sat_avg
##    <chr>                                        <dbl>   <dbl>
##  1 California Institute of Technology          0.0662    1566
##  2 Massachusetts Institute of Technology       0.0674    1545
##  3 Harvard University                          0.0473    1520
##  4 University of Chicago                       0.0726    1520
##  5 Yale University                             0.0635    1517
##  6 Duke University                             0.0891    1516
##  7 Vanderbilt University                       0.0961    1514
##  8 Columbia University in the City of New York   0.0591    1512
##  9 Northwestern University                     0.0847    1508
## 10 Princeton University                        0.0548    1503
## # ... with 13 more rows
```

And one last operation: all colleges that admit between 20 and 30 percent of students, looking at their SAT scores, earnings of attendees six years letter, and what state they are in, then arranging by state, and then SAT score.

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

```
  print(n=20)
```

```
## # A tibble: 40 x 4
##    instnm                                        sat_avg grad_debt_mdn stabbr
##    <chr>                                           <dbl>         <dbl> <chr>
##  1 Scripps College                                  1409         14000 CA
##  2 University of California-Irvine                  1316         15488 CA
##  3 Hope International University                    1022         24422 CA
##  4 California Institute of the Arts                   NA         27000 CA
##  5 California State University-Bakersfield            NA         16003 CA
##  6 Georgia Institute of Technology-Main Campus      1465         23000 GA
##  7 Savannah State University                        1251         31000 GA
##  8 Grinnell College                                 1450         17500 IA
##  9 Alice Lloyd College                              1054         15838 KY
## 10 Boston College                                   1429         17500 MA
## 11 Boston University                                1420         25000 MA
## 12 Babson College                                   1367         22985 MA
## 13 University of Michigan-Ann Arbor                 1434         17500 MI
## 14 Grace Christian University                         NA         23080 MI
## 15 University of North Carolina at Chapel Hill      1398         15400 NC
## 16 Carolinas College of Health Sciences             1012         12525 NC
## 17 Wake Forest University                             NA         19500 NC
## 18 Yeshiva Gedolah Shaarei Shmuel                     NA            NA NJ
## 19 New Mexico Institute of Mining and Technology    1264         19211 NM
## 20 Hamilton College                                 1446         16500 NY
## # ... with 20 more rows
```

Quick Exercise Choose a different college and two different things about that college. Have R print the output.

Summarizing Data

To summarize data, we use the summarize command. Inside that command, we tell R two things: what to call the new object (a data frame, really) that we're creating, and what numerical summary we would like. The code below summarizes median debt for the colleges in the dataset by calculating the average of median debt for all institutions.

```
df%>%
  summarize(mean_debt=mean(grad_debt_mdn,na.rm=TRUE))
```

```
## # A tibble: 1 x 1
##   mean_debt
##       <dbl>
## 1    19662.
```

```
df%>%
  summarize(median_debt=median(grad_debt_mdn,na.rm=TRUE))
```

```
## # A tibble: 1 x 1
##   median_debt
##         <dbl>
## 1       21500
```

Quick Exercise Summarize the average entering SAT scores in this dataset.

Combining Commands

We can also combine commands, so that summaries are done on only a part of the dataset. Below, we summarize median debt for selective schools, and not very selective schools.

```
df%>%
  filter(adm_rate<.1)%>%
  summarize(mean_debt=mean(grad_debt_mdn,na.rm=TRUE))
```

```
## # A tibble: 1 x 1
##   mean_debt
##       <dbl>
## 1    15680.
```

What about for not very selective schools?

```
df%>%
  filter(adm_rate>.3)%>%
  summarize(mean_debt=mean(grad_debt_mdn,na.rm=TRUE))
```

```
## # A tibble: 1 x 1
##   mean_debt
##       <dbl>
## 1    23261.
```

Quick Exercise Calculate average earnings for schools where SAT>1200

Plotting Data

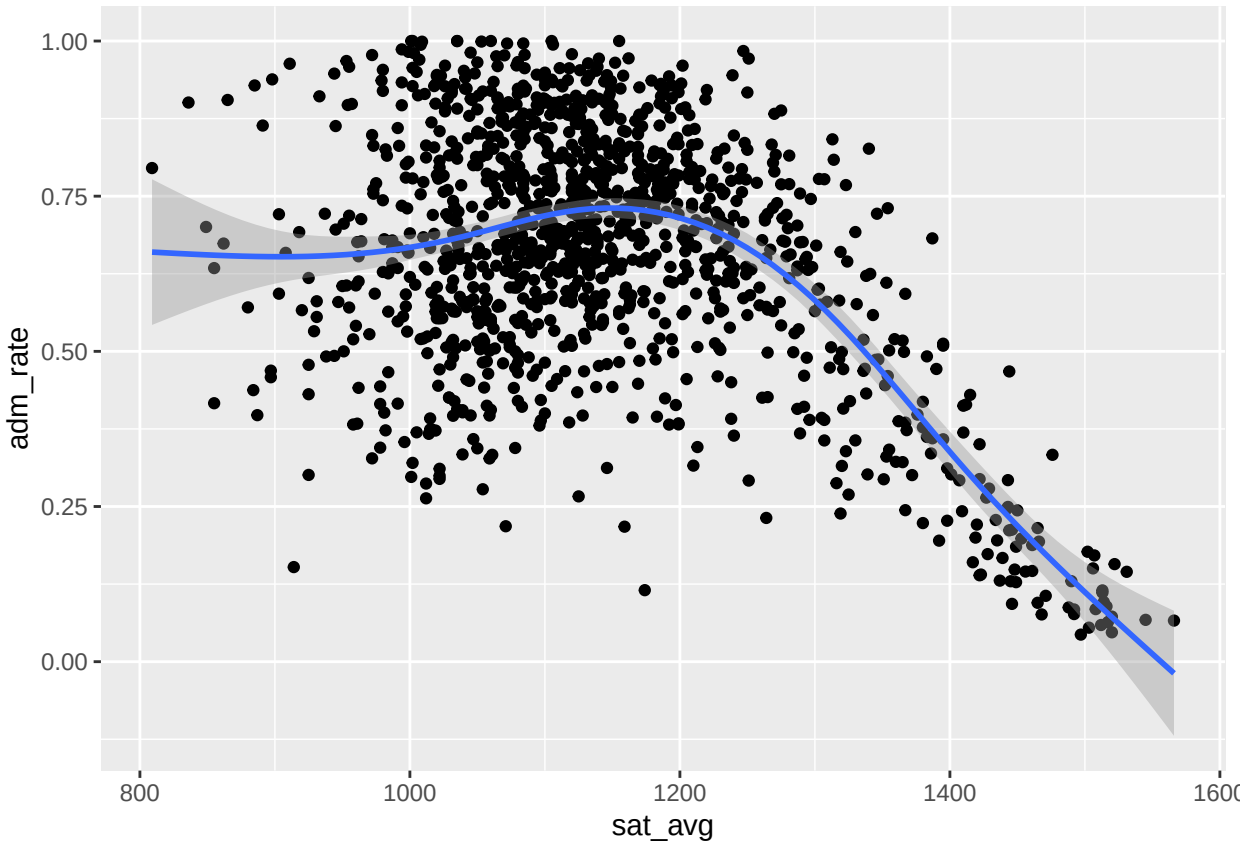
The last basic tool for looking at a dataset is plotting the data. The code below creates a scatterplot of admission rates by average SAT scores

```
## Plotting: bivariate
gg<-ggplot(data=df,aes(x=sat_avg,y=adm_rate))
gg<-gg+geom_point()
gg<-gg+geom_smooth()
gg
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 1325 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1325 rows containing missing values (geom_point).
```



Quick exercise Replicate the above plots, but put debt level on the y axis.

Your first commit: Hello, World!

For today, I want you to create a file called `01-assignment_<lastname>.Rmd` in your GitHub repo for assignments. It should contain the following elements:

1. A sentence that says "Hello, World"
2. R output that summarizes one of the variables in the `sc_debt.Rds` dataset
3. R output that shows a scatterplot for two of the variables in the `sc_debt.Rds` dataset.

Lucky for you this is is also your assignment! Submit it under assignments, using the format `01-assignment_<lastname>.Rmd`. All assignments should be turned in using this format. Since my last name is Doyle, I would use `01-assignment_doyle.Rmd` as my file name. Unless your name is also Doyle, you should use a different name.

Stretch Items

If you have extra time, you can do the following:

1. Calculate the average earnings for individuals at the most selective colleges, then compare that with individuals at the least selective colleges in the dataset.
2. Find a way to determine whether colleges with very high SAT scores tend to have higher or lower debt than colleges with low SAT scores.
3. Plot the relationship between SAT and debt. What do you see? Does this surprise you?
4. Now, provide separate plots for SAT and debt by control of the institution.