Intro to Data Science

Midterm 1

PT1 Getting Started

R & R Studio

- Data science is statistics meets computer science, so much of the work we do is in R Studio w/ coding language R.
- Some basics about R...
  - we will work w/ variables, numbers, vectors, tibbles/dataframes and more
  - we can use normal calculator functions in R too (/ * + -)
  - typically functions are called in this gen manner...
    `function(argument1 = _, arg2 = _, ...)`
  - In R Studio we have 4 places we work in...
    - Editor → where we write code
    - Console → see our code run
    - Enviroment → shows all obj you created

	env
Editor	
console	output

- In R Studio your computer looks for files in the working directory so it's a good idea to keep all files you need for a prj in 1 folder.

R Markdown

- R markdown is a file type in R studio that allows you to mix code and normal writing
- Use Knit to export to a .doc, .html, or .pdf.
- Code chunks are started w/ ```` ```{r} ```` and end w/ ```` ``` ````
- We can include some helpful options in headers...

Stops...	Run code	Show code	output	plots	mess.	warn.
eval=F	x		x	x	x	x
include=F		x	x	x	x	x
echo=F		x	x	x		
results="hide"			x			
fig.show="hide"				x		
message=F					x	
warning=F						x

- Can also name chunks... `{r Name, ...}`
- Chunks named "setup" will automatically run before any other code.

Code Used
- `seq(start, stop, length.out=)` makes reg sequence of #'s
- `getwd()` finds working directory
- `Knitr::kable(data, caption)` prints df as tables in knitted file

Librarys Used
- Knitr

PT2 Organizing Data

how do we get data into R and how do we make it workable?

Data Import

- we can use the `readr` library for a convient way to import data files
- works best w/ csv's but can also specify the delimeter
- but all csv's only contain character type vectors so to change the type of a column we use a `parse_*()` function
- R also expresses characters in strings w/ underlying code and is that encoding can't express certain characters we get weird strings.
- Use `guess_encoding()` to try and fix
- can also use `readr` to save to csv

Code Used
- `filter(data, var == ___)`
- `is.na()` returns TF is value NA
- `arrange(data, var)`
- `select("data", columns)`
- `rename()` select but renames vars
- `mutate(data, newvar = var + var)`
- `transmute()` mutate but only keeps new variables
- `summarize(data, newvar = func(var))`
- `group_by()` groups data based on a var. (good w/ sum.)
- `ungroup()`
- `read_csv("file path/.csv", col_names = c("a", "b"))`
- `read_delim()`
- `guess_encoding()`

- `parse_*()` ex: `parse_double()`, `parse_logical()`, ...
- `write_csv(data, "filename.csv")`
- `as_tibble()` turns df to tibble
- `tibble()` creates new tib.
- `print()` prints df or tib
- `as.data.frame()` tib → df
- `gather(cols to combine, Key = "old became", value = "new")`
- `spread(data, var to split, values = var of values)` Key = var to split value = var of values
- `separate(data, col to split, into = c("new names"), sep = "seperator")`
- `unite(data, new col, col to combine, sep = "")`

Librarys Used
- `dplyr`
- `tibbles`
- `readr`
- `tidyr`

Tibbles

- Tibbles → better data frames
- main differences...
  - tibble doesn't print whole dataframes
  - can extract from a tibble by name or position

- `complete()` turns implicit missing values into explicit missing.
- `fill()` can fill in NAs or w/ other values.

Data Transformation

- we will use 5 main functions...
  - `filter()` → pick observations
  - `arrange()` → reorder rows
  - `select()` → pick variables by name
  - `mutate()` → new variables made w/ func. of existing variables.
  - `summarize()` → collapse many values to a single summary.
- Data transformation: often use logical arguments and operators
- when transforming data we often use the pipe `%>%` which allows us to put multiple steps into one. ex... $f(g(x)) \Rightarrow x \%>\% g() \%>\% f()$
- when using summarize we typically also have to use the `group_by()` function.
- most of these funcs have helper functions that allow you to do more complicated things w/ them.

Tidy Data

- you can often represent the same data in many diff ways, but some ways are better than others.
- Tidy data (for our purposes) means ① each variable has it's own column, ② each observation has it's own row, ③ each value has it's own cell.
- Some untidy examples...
  - problem: column names are actually values of a variable
    solution: `gather()`
  - problem: one observation is in multiple rows. Sol: `spread()`
  - problem: two pieces of data in each cell of a column.
    solution: `separate()`
  - Problem: info in 2 rows in two sep. columns needs to be tog. in one column. Solution: `unite()`
- Missing values are either explicitly missing (NA's) or implicitly missing (we should have an entry for this month but we don't). We have to decide how to handle both types. (get rid of? comment on?)

Librarys Used
- `dplyr`
- `tibbles`
- `readr`
- `tidyr`

PT3 Exploring Data

Data Visualization

- We start by making visualizations for data exploration (later we'll make some for communication)
- We use `ggplot2` for graphing.
- the ggplot graphics take arguments in the "aes" category for splitting up variables.
- you can get many variables in one graph by mapping variables to the color of your graphs using `facet_wrap()`
- geom objects are the functions we use to tell ggplot2 what kind of graph to make.
- we can add text, labels, legends, and themes to our graphs too!
- general graphing layout...

```
ggplot(data) +
  geom_*(aes(variables))+
  func for themes and text
```

- Some common graphs → line graph, scatter plot, boxplot, tile plot, bar plot, histogram, ...

Code Used
- `ifelse(logical test, do if T, otherwise)` tests if something is T
- `ggplot()` start a plot
- `geom_*()` adds type of graph to your plot...
  - `_point()`
  - `_smooth()`
  - `_bar()`
  - `_boxplot()`
  - `_tile()`
  - `_histogram()`
- `aes()` the argument in ggplot where you specify variables
- `?funcname` get info on a function.
- `facet_wrap(~ var)` create multiple graphs grouped by a specific variable
- `facet_grid()`
- `position = " "` control dodge's position in a grid.
- `coord_*()` controls coord graph.
- `labs()` add labels to graph and legend.
- `geom_text()` label individual points
- `theme_*()` set a theme for your graph (color, text, positions)

Librarys Used
- `ggplot2`

Exploratory Data Analysis

- EDA → Exploratory Data Analysis.
- generate questions → search for answers → refine questions → visualize, transform, and model
- Questioning
  - mainly ways we EDA
  - Two types of questions → what kind of variation exists within my vars? what type of covariation occurs between my vars?
- Variation
  - how variables change between obsv.
  - visualize w/ histograms, bar charts
  - look for unexpected values (outliers)
  - look at subgroups.
- Covariation
  - tendency for 2 values to vary together in a related way
  - cat. & cont. → boxplot
  - 2 cat. → tile plot
  - 2 cont. → scatterplots!
- Once we see patterns we get to ask more questions about what causes these patterns!

(gen as a unusual values w/ NAs)

Good Luck ♡