SOLUTIONS
Skill Task 1: Rmd, `here()`

PS 811: Statistical Computing

February 25, 2020

Setup

In this class, we take advantage of RStudio's "Project" capabilities. We open an R project by opening the `.Rproj` file for a project, or opening the desired project from within RStudio.

When you work within an R project, R's working directory should automatically be set to the project "root," a.k.a. the top of the project folder. You can check this using the `here` package, which gives you feedback about its behavior when you load it.

```r
library("here")
## here() starts at /Users/michaeldecrescenzo/Box Sync/teaching/811-computing-s20
```

If this does print your project directory, two things could have gone wrong: (1), you aren't working in your RStudio project, or (2) you have not given your project an `.Rproj` file.

Basic R commands

I create a code chunk that saves two variables, `first` and `last`.[1]

```r
first <- "Michael"
last <- "DeCrescenzo"
```

I now use the `c()` function to combine my first and last name into one vector. The `c()` function is ubiquitous in R. Any time you need to supply a *series of values* to a function argument, for example, you can create that series (a.k.a. vector) using `c()`.

[1]If you open the `.Rmd` file for this document, you will notice that I have *named* these code chunks. Names are helpful to keep yourself organized, and they can come in handy when debugging an R Markdown document that fails to compile successfully.

```
c(first, last)
## [1] "Michael"     "DeCrescenzo"
```

Import data

Now I import the CAFE data into R. I need to load the tidyverse package to get the read_csv() function.[2]

```
library("tidyverse")
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

When you load the tidyverse package, it will loudly tell you a few things. First, because tidyverse is technically a bundle of packages, it tells you that several packages have actually been loaded. They are:

- readr: data import/export functions. Technically this is where read_csv() lives!
- tibble: improves the behaviors of data frames in R. Learn more.
- dplyr: data manipulation functions. We've seen this already!
- ggplot2: graphics. We've also seen this!
- tidyr: a package for data reshaping. This package is an update to other packages that you may have seen before, reshape and reshape2. We will explore this soon!
- stringr and forcats: two packages that provide helpful functions for working with character strings and factor data. We will also explore this soon!
- purrr: tools for working with lists, list-columns, and nested data. If this sounds like "wtf" stuff, that's because it's advanced. But buckle up because we're going to dive in at the end of the semester.

There is also a message that tells you that functions have been "masked." What this means is that there are functions within tidyverse packages that have the same names as other R functions, and so the new functions are prioritized over the old functions. Never fear; you can always access a masked function by typing package_name::function_name().

Now I read the data using read_csv(), which requires a path to the file that we want

[2] You should be aware that read_csv(), a function contained within tidyverse, is different from read.csv(), a function that R has by default. We will generally prefer reading functions whose names have *underscores*, so be mindful.

to import. We use `here()` to build the path, so we call `here()` within `read_csv()`. The functions are evaluated inside out, so `here()` builds the path, and then `read_csv()` uses the path to import the data. Note that I also pipe into `print()`, which lets me look at the results even as I store the data in the object named `cafe`.

```
cafe <-
  read_csv(here("data", "CAFE.csv")) %>%
  print()
## Parsed with column specification:
## cols(
##   Senator = col_character(),
##   State = col_character(),
##   Contribution = col_double(),
##   Party_Code = col_double(),
##   Vote = col_character()
## )
## # A tibble: 100 x 5
##    Senator           State Contribution Party_Code Vote
##    <chr>             <chr>        <dbl>      <dbl> <chr>
##  1 Murkowski, Frank  AK           19700        200 Yea
##  2 Stevens, Ted      AK           13000        200 Yea
##  3 Sessions, Jeff    AL            9500        200 Yea
##  4 Shelby, Richard   AL           25000        200 Yea
##  5 Hutchinson, Tim   AR            4900        200 Yea
##  6 Lincoln, Blanche  AR            5500        100 Yea
##  7 McCain, John      AZ           29350        200 Nay
##  8 Kyl, Jon          AZ           14500        200 Yea
##  9 Boxer, Barbara    CA            1500        100 Nay
## 10 Feinstein, Dianne CA            9750        100 Nay
## # ... with 90 more rows
```

Reading data with `read_csv()` (and other `tidyverse`-style `read_*()` functions) will also print a noisy message. It isn't an error. It's telling you how the data were interpreted by `read_csv()`. It's saying, "I made a variable called 'Senator' that is a character (string) variable..." and so on. These are inferences that the function is making about the data, but it's possible to modify this inferential behavior in advanced applications.

If your code looks like mine, and the document builds ("knits") to PDF, then you did great!