Intro to R and the Tidyverse Quick Guide

Quick Guide

- **Install Packages:** You need to install new packages only the first time you want to use them. You do this with:

  `install.packages('PACKAGE_NAME')`

- **Load Packages:** You need to load a package after you have installed it, everytime you start a new R session. Typically you want to load all packages at the top of a script. You do this with the library function (Note you do not need to put an installed package name in quotes to use library)

  `library(PACKAGE_NAME)`

- **Variables in R:** We assign variables using the *assignment operator (<-)*. Variables save all kinds of objects (dataframes, numbers, lists, words) in R in memory. You can overwrite a variable when you assign it a new object.

- **Function:** Functions in R are actions you want performed, inside the function you put arguments that are the things you want the function action performed on or tell the function how specifically to perform an action.

  `function_name(arguments=thing_you_want_function_to_act_on)`

  You can find out more about what a function does and what arguments go in it by using `?function_name`

  If you want to search for a function that does something specific (but maybe you don't know the name exactly) you can use

  `??function_I_hope_exists`

  This searches all the documentation for that term.

- **read_csv()** This is the function in the *readr* package that loads csv data into R. Inside you want to put in quotes the file path that directs R to the file you want to read in.

- **filter()** from the *dplyr* package selects rows from a dataframe based on the value of a specific column. Inside the function the arguments are first the dataframe name then column you want to filter by, and what you want the column value to be filtered on. Example:

  `filter(dataframe, Column_Name == 'something_I_want')`

  or you can use greater than (>), less than (<), less than or equal to(<=), greater than or equal to(>=) to filter a column value.

  ‘`filter(dataframe, Column_Name > 100)`

- **select()** is a function from *dplyr* you can use it to grab just certain columns from a dataframe, inside the function you need to give the dataframe you want to grab columns from and then the name of each column you want (separate each by a comma)

  `select(dataframe, COLUMN_NAME1, COLUMN_NAME3)`

- **ggplot** from *ggplot2* package build graphs. Today we made scatterplots using the format

  ```
  ggplot(data=dataframe_name)+geom_point(mappings=aes(x=COLUMN_NAME, y=COLUMN_NAME,
  color=COLUMN_NAME, size=COLUMN_NAME))+xlab('X axis label')+ylab('Y axis Label')
  ```

- **group__by()** from *dplyr* tells R you want to perform any other actions on a whole group. Inside group__by() you list the variables (i.e. column names) you want to group by, you can include as many as you want.
- **summarise() and chaining** we used group__by with chaining (*magittr* package) and summarise (*dplyr* package) simulataneously. Summarise will create a new dataframe with summary statistics. You can tell it what summary statistics to calculate and for which columns.

  ```
  dataframe_name %>% group_by(COLUMN_NAME) %>% summarise(new_column_name = mean(COLUMN_NAME),
  another_new_column = sd(COLUMN_NAME))
  ```

  Chaining is done using the pipe operator %>% this basically tells R to take the thing before the pipe and do the action that is follows the pipe to that thing. So above we have a dataframe and we tell R to take that and group by a variable and then summarise it in a certain way. Check out the summarise() documentation (type ?summarise())to see other functions you can use to get summary values from your dataframe.