Intro to R workshop, CORE 121, Professor Weissman-Unni

R cheat sheet

CORE 121, Professor Weissman-Unni

September 13th, 2021

Assigning variables.

Pick a variable name, and use the left pointing arrow (<-) to assign its value. Numbers don't need quotes, but strings do.

```
#Anything preceded by a "#" is a "comment". It does not get executed as code.
#Comments can be super helpful to provide info on your code.

# assigns 10 to x. This alone won't print it, just performs the value assignment.
x<-10

y<-"Hello"

# prints x
x
```

```
## [1] 10
```

```
#prints y
y
```

```
## [1] "Hello"
```

Functions.

Like most languages, R has functions that help you quickly execute common tasks. Functions typically take the form of:

functionName(argument1, argument2, etc.. . .)

For example, the function sqrt(number) takes the square root of a number:

```
sqrt(9)
```

```
## [1] 3
```

Getting help on functions

Use the 'help' function on a function, and info will appear in the lower-right window. Googling also is a good option.

```
help(sqrt)
```

How do I know what functions to use, or whether a function exists?

The best approach is to just Google it. For example, "how do I find the median in R".

Statistical Functions

Arrays/Vectors

Use the “c” (combine) function to create an array, or vector, of values:

```
# Create an array of numbers and assign it to a variable
myVector<-c(2,4,5,3,4,2,4,3,1)

#print it out
myVector
```

```
## [1] 2 4 5 3 4 2 4 3 1
```

Finding the mean.

Use the “mean” function, using an array variable as the argument:

```
myAvg<-mean(myVector)

myAvg
```

```
## [1] 3.111111
```

Finding the median.

Use the “median” function, using an array variable as the argument:

```
myMedian<-median(myVector)

myMedian
```

```
## [1] 3
```

Finding the standard deviation

Use the “sd” function, using an array variable as the argument:

```
mySd<-sd(myVector)

#print it out
mySd
```

```
## [1] 1.269296
```

Working with a Data Set

Loading the Tidyverse

In order to use the read_csv function, you must first load the Tidyverse:

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     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 a csv file from the Files area

```
# Assigns whole file to a variable as a data frame
myMinutes<-read_csv("JMExercise.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Timestamp = col_character(),
##   Date = col_character(),
##   `Walking(minutes)` = col_double(),
##   `Cycling(minutes)` = col_double(),
##   `Swimming(minutes)` = col_double()
## )
```

```
# Print it
myMinutes
```

```
## # A tibble: 8 x 5
##   Timestamp         Date     `Walking(minutes~ `Cycling(minute~ `Swimming(minut~
##   <chr>             <chr>                <dbl>            <dbl>            <dbl>
## 1 9/8/2021 14:52:18 9/1/2021                41               34               20
## 2 9/8/2021 14:51:34 9/2/2021                40               33               16
## 3 9/8/2021 14:50:51 9/3/2021                25               33               34
## 4 9/8/2021 14:50:03 9/4/2021                64                0                0
## 5 9/8/2021 14:49:31 9/5/2021                39              122                0
## 6 9/8/2021 14:48:53 9/6/2021                20                0                0
## 7 9/8/2021 14:48:15 9/7/2021                28               31               18
## 8 9/8/2021 14:47:07 9/8/2021                27               32                0
```

Create an array/vector from a data frame

The syntax to target a column is dataFrame$`Name of Column`.

Once you have your column data as an array, you can also perform statistical functions on them (mean, median, standard deviation, etc.)

```
# notice the column name is contained in quotes.
minWalked<-myMinutes$`Walking(minutes)`

#print values for "Walking(minutes)" column:
minWalked
```

```
## [1] 41 40 25 64 39 20 28 27
```

Loading an external data set

Now we're going to practice loading your data from your Google Spreadsheet. If you don't have yours readily available, you can use mine: https://bit.ly/jm-core121-data

- In another browser tab, find your data (or use mine above)
- Make sure the "Phase 2:Data" worksheet is selected
- In the Google Sheets menu, select File->Download->Comma separated values (csv). This should download the file to your computer.

- Now switch back to rstudio.cloud
- In the Files tab in the lower right-hand corner, click “Upload”
- Click “Choose File” to locate it on your computer.
- Once you’ve selected the file, click “Ok” to begin the upload.
- Once it’s loaded, you should see it listed among your files in the Files tab.
- Finally, you may want to rename the file to something simple, just to make it easier when you use the read_csv function. Click the checkbox next to the file, click ‘Rename’, and call it something like ‘myData.csv’

Now you can load it into your R environment by running the read_csv command as before

```
#Load the data to a variable:
myData<-read_csv("myData.csv")
```

Need help? Email Jeremy at jeremym@lclark.edu Jeremy is also hosting R office hours in Watzek 343: Tuesdays 2-3pm Thursdays 2-3pm or by appointment