# R Workshop 2020: Session 1

*Anisha Khosla*

*20th July, 2020*

## Contents



## R Studio

#### 4 panels

Top left: Script viewer - write your code to save it for future use Bottom left: Console - this is where R is actually running Top right: Variable/history displayer Bottom right: Files/plots/packages/help displayer

#### Script vs Console

Console: type commands directly to see their result. This is not saved Script editor: Save your code for later.

**Shortcuts**

TAB – autocomplete control/cmd + enter – run current line from the script in console

### How to run a line of code?

When you are on the line, press cmd (mac) / ctrl (windows), then enter (more later)

## Comments

Comments are lines in the script that are skipped and not run. They can be used to annotate your code and make it more friendly when you are trying out something new or for others if you are sharing your code. This is also VERY helpful to remind yourself of what you did when you look back at the code years after you wrote it.

If you add a hashtag, everything written after is a comment.

```
# This is a comment. We can use comments using a hash sign
This is not a comment
```

```
## Error: <text>:3:6: unexpected symbol
## 2: # This is a comment. We can use comments using a hash sign
## 3: This is
##         ^
```

## Variables & Data Types

Variables are used to store information and you can reference to the information later using the variable name.

Variables can be used to store different types of data like numerics (1,4), characters('a','g'), booleans (TRUE/FALSE)

If you are wondering: Why '<-' and not '=' for variable assignment? – indicates the difference between function arguments and assignation. See https://www.r-bloggers.com/why-do-we-use-arrow-as-an-assignment-operator/

How to run a line of code?: When you are on the line, press cmd (mac) / ctrl (windows), then enter (more later)

```
#%# Numeric
# assigning a value to a variable
n <- 1

# str() displays the type/ structure of object
str(n)
```

```
##  num 1
```

```
#%# Character
c <- '4'
str(c)
```

```
##  chr "4"
```

```
# Is 3 the same as '3'?
identical(3, '3')
```

```
## [1] FALSE
```

```
#%# Boolean/Logical: TRUE, FALSE, NA
# NA: not available

l <- TRUE
str(l)
```

```
##  logi TRUE
```

```
# Converting from one data type to another
# Ex. Converting numeric to character
as.numeric(c)
```

```
## [1] 4
```

```
as.character(n)
```

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

## Getting Help in RStudio

Below are some ways to get help within RStudio. You can, of course, google functions and specific things you are looking for. R Documentation is usually a good first source because it details everything a function can do, but if you are looking for something more specific, often someone else has asked the same question on forums like StackOverflow

```
#%# Getting help in RStudio


help.start()   # general help
```

```
## starting httpd help server ... done
```

```
help(mean)    # help for function mean()
?mean         # same as above

example(mean) # shows an example of function mean()
```

## Vectors, Factors, Converting between data types

vectors, factors, matrix, array, dataframe, list for more see https://www.statmethods.net/input/datatypes.html

```r
#%# Vectors
my_vector <- c(8,0,9.9,6,-1,4)

#%# Indexing vectors
my_vector[1] # access first element
```

```
## [1] 8
```

```r
# note: in R, the indexing begins with 1 (0 in Python)
my_vector[3:5] # access 3rd to 5th elements
```

```
## [1]  9.9  6.0 -1.0
```

```r
my_vector[c(1, 3)] # access first and last element
```

```
## [1] 8.0 9.9
```

```r
mean(my_vector)
```

```
## [1] 4.483333
```

```r
max(my_vector)
```

```
## [1] 9.9
```

```r
#%# Factors
# how catgorical variables are stored in R

groups <- factor(c("red","green", "red", "green"))
groups
```

```
## [1] red   green red   green
## Levels: green red
```

```r
levels(groups) # all possible values of a factor
```

```
## [1] "green" "red"
```

```r
nlevels(groups) # number of possible values of a factor
```

```
## [1] 2
```

```r
# why is it important to specify data type? -- especially important with numerical factors
groups<- c(4,5,6)
str(groups)
```

```
##  num [1:3] 4 5 6
```

```r
# convert numeric to character
groups<- as.character(groups)
str(groups)
```

```
##  chr [1:3] "4" "5" "6"
```

```r
levels(groups) # this will not work because 'groups' is not a factor yet
```

```
## NULL
```

```r
# convery character to factor
groups<- as.factor(groups)
str(groups)
```

```
##  Factor w/ 3 levels "4","5","6": 1 2 3
```

```r
levels(groups)
```

```
## [1] "4" "5" "6"
```

## Setting Working Directory

directory = folder

The folder R will load files from and save files to. You can check and change the current directory to a different one

```r
# Where am I? i.e. where is R currently looking for files or saving files to?
getwd()
```

```
## [1] "/Users/anishakhosla/Dropbox/Baycrest_Rworkshop/2020/part1"
```

```r
# Set the location i.e.working directory.  This is where R will look for files you want to load.
# setwd()

# sample working directory for MAC users
setwd('/Users/anishakhosla/Dropbox/Baycrest_Rworkshop/2020/part1')

# if I had the file on my Desktop
# setwd("/Users/anishakhosla/Desktop")

# sample working directory for windows users
#setwd("C:\\Users\\akhosla\\Desktop")
```

## Load Data

You can read in .csv files into sheet-like data format in R. This sheet-like data is called a 'dataframe' in R

Coffee Data: There was a pilot participant and 4 other participants who either drank coffee or water (between-subject variable) and then did some task. We have the reaction time and accuracy from the task.

```
# read in data from .csv to a dataframe
df <- read.csv('coffee_data.csv')

# EXPLAIN DATA
View(df)
str(df)
```

```
## 'data.frame':    50 obs. of  5 variables:
##  $ participantID: Factor w/ 5 levels "1","2","3","4",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ trial_number : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ drink        : Factor w/ 2 levels "coffee","water": 1 1 1 1 1 1 1 1 1 1 ...
##  $ rt           : int  300 344 311 501 665 612 700 598 399 297 ...
##  $ accuracy     : int  1 0 0 0 1 1 1 0 0 1 ...
```

## Dataframes – Explore Data

```
summary(df) # summary for each column of the dataframe
```

```
##  participantID  trial_number     drink          rt            accuracy
##  1    :10      Min.   : 1.0   coffee:30   Min.   :  50.0   Min.   :0.0
##  2    :10      1st Qu.: 3.0   water :20   1st Qu.: 409.5   1st Qu.:0.0
##  3    :10      Median : 5.5               Median : 612.5   Median :0.0
##  4    :10      Mean   : 5.5               Mean   : 675.5   Mean   :0.4
##  pilot:10      3rd Qu.: 8.0               3rd Qu.: 757.2   3rd Qu.:1.0
##                Max.   :10.0               Max.   :4078.0   Max.   :1.0
```

```
dim(df) # number of rows and columns
```

```
## [1] 50  5
```

```
head(df) # prints the first 5 rows of a dataframe
```

```
##   participantID trial_number  drink  rt accuracy
## 1         pilot            1 coffee 300        1
## 2         pilot            2 coffee 344        0
## 3         pilot            3 coffee 311        0
## 4         pilot            4 coffee 501        0
## 5         pilot            5 coffee 665        1
## 6         pilot            6 coffee 612        1
```

```r
names(df) # name of columns
```

```
## [1] "participantID" "trial_number"  "drink"         "rt"
## [5] "accuracy"
```

```r
# to select/print a particular column
df$participantID # OR df[['participantID']] (both are the same)
```

```
##  [1] pilot pilot pilot pilot pilot pilot pilot pilot pilot pilot 1
## [12] 1     1     1     1     1     1     1     1     1     2     2
## [23] 2     2     2     2     2     2     2     2     3     3     3
## [34] 3     3     3     3     3     3     3     4     4     4     4
## [45] 4     4     4     4     4     4
## Levels: 1 2 3 4 pilot
```

```r
df['participantID'] # while the output of $ or [['']] is a factor, output of this type of indexing is a
```

```
##    participantID
## 1          pilot
## 2          pilot
## 3          pilot
## 4          pilot
## 5          pilot
## 6          pilot
## 7          pilot
## 8          pilot
## 9          pilot
## 10         pilot
## 11             1
## 12             1
## 13             1
## 14             1
## 15             1
## 16             1
## 17             1
## 18             1
## 19             1
## 20             1
## 21             2
## 22             2
## 23             2
## 24             2
## 25             2
## 26             2
## 27             2
## 28             2
## 29             2
## 30             2
## 31             3
## 32             3
## 33             3
```

```
## 34             3
## 35             3
## 36             3
## 37             3
## 38             3
## 39             3
## 40             3
## 41             4
## 42             4
## 43             4
## 44             4
## 45             4
## 46             4
## 47             4
## 48             4
## 49             4
## 50             4
```

```
# Use what we learned about indexing into vectors to index into columns of a dataframe
df$participantID[1]  # participant number from the 1st row
```

```
## [1] pilot
## Levels: 1 2 3 4 pilot
```

```
df$participantID[40] # participant number from the 40th row
```

```
## [1] 3
## Levels: 1 2 3 4 pilot
```

```
df$participantID[1:3] # grabs the first 3 entries in a particular column
```

```
## [1] pilot pilot pilot
## Levels: 1 2 3 4 pilot
```

## Packages

Packages in R (and really any programming language) allow you to add functionality to base R. So you can are downloading a package of functions that did not exist before in base R.

For example, if you want to do mixed models in R, you would download a package called 'lme4'. You can't do that with base R. We also download packages which make our lives easier and code easier to read and understand. For example, we can create plots to visualize data in base R but they are pretty tedious if you want to make them pretty and publication-ready, you would probably use 'ggplot2'.

There are alot of packages that add functionality or make our code easier to read. Hadley Wickham (http://hadley.nz/) made a universe of useful packages so you don't have to install each one of them individually. 'tidyverse' is a universe of such useful packages. When you load 'tidyverse', you will get a huge print out in your console detailing the packages that come with tidyverse

```
## Installing Packages
# uncomment the line below and run it if you don't have this package installed
#install.packages('tidyverse')

## to view a list of installed packages
# installed.packages()

## Loading Packages
# If you haven't installed the 'tidyverse' package, you should get an error when you run the line below
# If you have it installed, you will get a huge print out in your console detailing the packages that c

library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------- tidyverse 1.2.1 -

## v ggplot2 3.3.0     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.2     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()
```

# Piping

Pipe operator: %>%

- shortcut is cmd/ctrl + shift + m
- it allows us to do step-by-step actions in an intuitive way
- insert results from prev step as output for the next step OR inserts the results from the previous function into the next function

```
x <- 1:4
x
```

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

```
sum(x) # without pipe
```

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

```
x %>% sum() # with pipe
```

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

# Dataframes – tidyverse

```r
# back to our dataset...
df$participantID
```

```
##  [1] pilot pilot pilot pilot pilot pilot pilot pilot pilot pilot 1
## [12] 1     1     1     1     1     1     1     1     1     2     2
## [23] 2     2     2     2     2     2     2     2     3     3     3
## [34] 3     3     3     3     3     3     3     4     4     4     4
## [45] 4     4     4     4     4     4
## Levels: 1 2 3 4 pilot
```

```r
# distinct() shows all distinct values in a column
df %>%
  distinct(participantID)
```

```
##   participantID
## 1         pilot
## 2             1
## 3             2
## 4             3
## 5             4
```

```r
# count() shows the number of distinct values in a column
# example, how many rows per subject?
df %>%
  count(participantID)
```

```
## # A tibble: 5 x 2
##   participantID     n
##   <fct>         <int>
## 1 1                10
## 2 2                10
## 3 3                10
## 4 4                10
## 5 pilot            10
```

```r
# filter--we can define a set of conditions that we want to filter the rows of our data frame by
df %>%
  filter(participantID == 'pilot')
```

```
##    participantID trial_number  drink  rt accuracy
## 1          pilot            1 coffee 300        1
## 2          pilot            2 coffee 344        0
## 3          pilot            3 coffee 311        0
## 4          pilot            4 coffee 501        0
## 5          pilot            5 coffee 665        1
## 6          pilot            6 coffee 612        1
## 7          pilot            7 coffee 700        1
## 8          pilot            8 coffee 598        0
## 9          pilot            9 coffee 399        0
## 10         pilot           10 coffee 297        1
```

```r
# select--same as filter but allows for subsetting columns instead
df %>%
  select(participantID, drink, rt)
```

```
##    participantID  drink   rt
## 1          pilot coffee  300
## 2          pilot coffee  344
## 3          pilot coffee  311
## 4          pilot coffee  501
## 5          pilot coffee  665
## 6          pilot coffee  612
## 7          pilot coffee  700
## 8          pilot coffee  598
## 9          pilot coffee  399
## 10         pilot coffee  297
## 11             1 coffee  331
## 12             1 coffee  402
## 13             1 coffee  731
## 14             1 coffee  788
## 15             1 coffee  639
## 16             1 coffee  256
## 17             1 coffee  755
## 18             1 coffee  245
## 19             1 coffee  354
## 20             1 coffee  623
## 21             2  water  890
## 22             2  water  917
## 23             2  water 1087
## 24             2  water  455
## 25             2  water 1198
## 26             2  water  501
## 27             2  water  667
## 28             2  water  568
## 29             2  water  899
## 30             2  water  601
## 31             3 coffee  270
## 32             3 coffee   50
## 33             3 coffee  351
## 34             3 coffee  432
## 35             3 coffee  758
## 36             3 coffee  642
## 37             3 coffee  712
## 38             3 coffee  788
## 39             3 coffee  678
## 40             3 coffee  444
## 41             4  water  592
## 42             4  water  587
## 43             4  water  972
## 44             4  water 4078
## 45             4  water  652
## 46             4  water  613
## 47             4  water  987
## 48             4  water 1056
```

```
## 49             4  water  472
## 50             4  water 1009
```

```
# add new column
df$age <- 21
df
```

```
##    participantID trial_number  drink   rt accuracy age
## 1          pilot            1 coffee  300        1  21
## 2          pilot            2 coffee  344        0  21
## 3          pilot            3 coffee  311        0  21
## 4          pilot            4 coffee  501        0  21
## 5          pilot            5 coffee  665        1  21
## 6          pilot            6 coffee  612        1  21
## 7          pilot            7 coffee  700        1  21
## 8          pilot            8 coffee  598        0  21
## 9          pilot            9 coffee  399        0  21
## 10         pilot           10 coffee  297        1  21
## 11             1            1 coffee  331        1  21
## 12             1            2 coffee  402        1  21
## 13             1            3 coffee  731        0  21
## 14             1            4 coffee  788        0  21
## 15             1            5 coffee  639        0  21
## 16             1            6 coffee  256        0  21
## 17             1            7 coffee  755        0  21
## 18             1            8 coffee  245        1  21
## 19             1            9 coffee  354        1  21
## 20             1           10 coffee  623        0  21
## 21             2            1  water  890        0  21
## 22             2            2  water  917        0  21
## 23             2            3  water 1087        1  21
## 24             2            4  water  455        0  21
## 25             2            5  water 1198        1  21
## 26             2            6  water  501        0  21
## 27             2            7  water  667        0  21
## 28             2            8  water  568        1  21
## 29             2            9  water  899        0  21
## 30             2           10  water  601        1  21
## 31             3            1 coffee  270        0  21
## 32             3            2 coffee   50        1  21
## 33             3            3 coffee  351        0  21
## 34             3            4 coffee  432        0  21
## 35             3            5 coffee  758        0  21
## 36             3            6 coffee  642        0  21
## 37             3            7 coffee  712        0  21
## 38             3            8 coffee  788        0  21
## 39             3            9 coffee  678        1  21
## 40             3           10 coffee  444        1  21
## 41             4            1  water  592        0  21
## 42             4            2  water  587        0  21
## 43             4            3  water  972        1  21
## 44             4            4  water 4078        0  21
## 45             4            5  water  652        0  21
## 46             4            6  water  613        1  21
```

```
## 47             4            7  water  987        0  21
## 48             4            8  water 1056        0  21
## 49             4            9  water  472        1  21
## 50             4           10  water 1009        1  21
```

```
# the - operator can be used to exclude a column
df %>%
  select(-age)
```

```
##    participantID trial_number  drink   rt accuracy
## 1          pilot            1 coffee  300        1
## 2          pilot            2 coffee  344        0
## 3          pilot            3 coffee  311        0
## 4          pilot            4 coffee  501        0
## 5          pilot            5 coffee  665        1
## 6          pilot            6 coffee  612        1
## 7          pilot            7 coffee  700        1
## 8          pilot            8 coffee  598        0
## 9          pilot            9 coffee  399        0
## 10         pilot           10 coffee  297        1
## 11             1            1 coffee  331        1
## 12             1            2 coffee  402        1
## 13             1            3 coffee  731        0
## 14             1            4 coffee  788        0
## 15             1            5 coffee  639        0
## 16             1            6 coffee  256        0
## 17             1            7 coffee  755        0
## 18             1            8 coffee  245        1
## 19             1            9 coffee  354        1
## 20             1           10 coffee  623        0
## 21             2            1  water  890        0
## 22             2            2  water  917        0
## 23             2            3  water 1087        1
## 24             2            4  water  455        0
## 25             2            5  water 1198        1
## 26             2            6  water  501        0
## 27             2            7  water  667        0
## 28             2            8  water  568        1
## 29             2            9  water  899        0
## 30             2           10  water  601        1
## 31             3            1 coffee  270        0
## 32             3            2 coffee   50        1
## 33             3            3 coffee  351        0
## 34             3            4 coffee  432        0
## 35             3            5 coffee  758        0
## 36             3            6 coffee  642        0
## 37             3            7 coffee  712        0
## 38             3            8 coffee  788        0
## 39             3            9 coffee  678        1
## 40             3           10 coffee  444        1
## 41             4            1  water  592        0
## 42             4            2  water  587        0
## 43             4            3  water  972        1
## 44             4            4  water 4078        0
```

```
## 45             4            5  water  652        0
## 46             4            6  water  613        1
## 47             4            7  water  987        0
## 48             4            8  water 1056        0
## 49             4            9  water  472        1
## 50             4           10  water 1009        1
```

```
#%# why pipe?
# note: arrange() sorts data

# select columns participantID, drink, rt
# filter for the pilot participant
# arrange reraction time in descending order

# piping
df %>%
  select(contains('part'), drink, rt) %>%
  filter(participantID=='pilot') %>%
  arrange(rt)
```

```
##    participantID  drink  rt
## 1          pilot coffee 297
## 2          pilot coffee 300
## 3          pilot coffee 311
## 4          pilot coffee 344
## 5          pilot coffee 399
## 6          pilot coffee 501
## 7          pilot coffee 598
## 8          pilot coffee 612
## 9          pilot coffee 665
## 10         pilot coffee 700
```

```
# base R
arrange(filter((select(df, contains('part'), drink, rt)), participantID=='pilot'), rt)
```

```
##    participantID  drink  rt
## 1          pilot coffee 297
## 2          pilot coffee 300
## 3          pilot coffee 311
## 4          pilot coffee 344
## 5          pilot coffee 399
## 6          pilot coffee 501
## 7          pilot coffee 598
## 8          pilot coffee 612
## 9          pilot coffee 665
## 10         pilot coffee 700
```

```
# subset data--print rows where the rt is above 1000ms
subset(df, df$rt > 1000)
```

```
##    participantID trial_number drink   rt accuracy age
## 23             2            3 water 1087        1  21
```

```
## 25            2            5 water 1198        1  21
## 44            4            4 water 4078        0  21
## 48            4            8 water 1056        0  21
## 50            4           10 water 1009        1  21
```

# R Resources

Free online edX course by Rafael Irizarry, Harvard University: https://www.edx.org/course/data-science-r-basics

**Cheatsheets** *note that some of these are links to PDFs which will download when you click on them

https://rstudio.com/resources/cheatsheets/ (various cheatsheets) Tidyverse cheatsheet: https://www.datacamp.com/community/blog/tidyverse-cheat-sheet-beginners RStudio cheatsheet: https://rstudio.com/wp-content/uploads/2016/01/rstudio-IDE-cheatsheet.pdf Base R cheatsheet: https://rstudio.com/wp-content/uploads/2016/10/r-cheat-sheet-3.pdf

Course by Tobi Gerstenberg (stats and data visualization with R) PSY252: https://psych252.github.io/ Book (PSY252): https://psych252.github.io/psych252book/

U of T coders: https://uoftcoders.github.io/studyGroup/lessons/

# Packages to download for following sessions:

- afex, emmeans, ggthemes, cowplot, gridExtra
- run the line of code below in the console to install the packages listed above

install.packages(c("afex","emmeans","ggthemes", "cowplot","gridExtra"))