# Tutorial 1

Revati Shivnekar

2023-08-12

References other sources for learning: 1. learnr package 2. https://www.sas.upenn.edu/~baron/from_cattell/rpsych/rpsych.html#htoc2 3. https://intro2r.com/basics_r.html 4. Discovering Statistics with R by Andy Field

NOTE: This worksheet is for you to get a hands-on experience of R. If you are unfamiliar with R or coding in general, this should help, but you must explore more from the references (1, 2, and 3) above to get a better hang of all things R.

There are also some OPTIONAL bits in this worksheet which you can skip.

Contents: * SETTING WORKING DIRECTORY * IMPORTING AND LOADING LIBRARIES * OBJECTS AND FUCNTIONS: vectors, lists, matrices, and data frames ==========

**# 1. SETTING WORKING DIRECTORY**

Saves your progress, plots, files in one folder. When you import files, you can simply put them in the same folder as your wd for easy access.

GUI path for setting wd(): Session -> set working directory.

## Some important commands

```
getwd() # check where you are currently
```

```
## [1] "/home/swarag/Downloads"
```

```
setwd("/home/swarag/Documents")
```

if you know the path to the folder. Or check "Properties/Get Info" ``r getwd() # check if this is where you want to be`` ### OPTIONAL: You can also use the terminal in the console window to manipulate directories by shell scripting

**2. IMPORTING AND LOADING LIBRARIES**

Installing makes the library available to your PC. Loading makes it available to the R environment. You need to install a package once but load it every time you want to run the script.

A package is a bundle of functions that you can use in your code. When you talk to these functions in the syntax they understand, these functions will save you tons of time and lines of complicated code. Best thing about them is you (most often) do not need to know how they are doing any of this. Just knowing the syntax is enough.

GUI for packages: bottom right pane has a tab for packages. You can install and then load (by checking off) packages from there.

```
install.packages(c("ggplot2", "dplyr", "tidyverse", "tidyr", "shiny", "plotly"))
```

```
## Installing packages into '/home/swarag/R/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
## also installing the dependencies 'gargle', 'curl', 'systemfonts', 'textshaping', 'googledrive', 'googlesheets
4', 'httr', 'ragg', 'rvest', 'xml2'
```

```
## Warning in install.packages(c("ggplot2", "dplyr", "tidyverse", "tidyr", :
## installation of package 'curl' had non-zero exit status
```

```
## Warning in install.packages(c("ggplot2", "dplyr", "tidyverse", "tidyr", :
## installation of package 'systemfonts' had non-zero exit status
```

```
## Warning in install.packages(c("ggplot2", "dplyr", "tidyverse", "tidyr", :
## installation of package 'xml2' had non-zero exit status
```

```
## Warning in install.packages(c("ggplot2", "dplyr", "tidyverse", "tidyr", :
## installation of package 'textshaping' had non-zero exit status
```

```
## Warning in install.packages(c("ggplot2", "dplyr", "tidyverse", "tidyr", :
## installation of package 'httr' had non-zero exit status
```

```
## Warning in install.packages(c("ggplot2", "dplyr", "tidyverse", "tidyr", :
## installation of package 'gargle' had non-zero exit status
```

```
## Warning in install.packages(c("ggplot2", "dplyr", "tidyverse", "tidyr", :
## installation of package 'ragg' had non-zero exit status
```

```
## Warning in install.packages(c("ggplot2", "dplyr", "tidyverse", "tidyr", :
## installation of package 'rvest' had non-zero exit status
```

```
## Warning in install.packages(c("ggplot2", "dplyr", "tidyverse", "tidyr", :
## installation of package 'plotly' had non-zero exit status
```

```
## Warning in install.packages(c("ggplot2", "dplyr", "tidyverse", "tidyr", :
## installation of package 'googledrive' had non-zero exit status
```

```
## Warning in install.packages(c("ggplot2", "dplyr", "tidyverse", "tidyr", :
## installation of package 'googlesheets4' had non-zero exit status
```

```
## Warning in install.packages(c("ggplot2", "dplyr", "tidyverse", "tidyr", :
## installation of package 'tidyverse' had non-zero exit status
```

```
library(ggplot2) # for plotting
```

what do other packages do? ?{package/function} is a help command to get more info ?dplyr
find out what other packages from the list above do by using ?

OPTIONAL: look up pacman for installing and loading multiple packages

# 3. OBJECTS AND FUNCTIONS

Most things that can be named is an object, from strings and numbers to complex data frames and matrices. You can create an object by giving it a name ie by using <-

There are mainly four kinds of objects: vectors, matrices, lists, and dataframes.

Lastly, once you create an object, you can use functions on them.

Some examples of objects and functions are below. You can make your own objects for practice.

---

++++++++++++ # 3.1 VECTORS ++++++++++++

```
my_age <- 27 # can be a number
my_name <- 'revati' # can be a string or a bunch of characters

my_fam_names <- c('vandana', 'vijay', 'saurabh') # multiple elements are bound together
                                                 # by c(), called concatenate
my_fam_ages <- c(59, 63, 32) # another multi-element vector
```

# vector functions

write new vectors and try them out

```
length(my_fam_names)
```

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

```
mean(my_fam_ages) # what if i use my_fam instead? try it
```

```
## [1] 51.33333
```

```
var(my_fam_ages) #variance
```

```
## [1] 284.3333
```

```
sd(my_fam_ages) #standard deviation
```

```
## [1] 16.86219
```

```
my_fam_names[2] # get the 3rd element from vec
```

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

```
my_fam_names[4] <- 'chelsea' # adds my cat to to the my_fam vector
my_fam_names #this is what it looks like now
```

```
## [1] "vandana" "vijay"   "saurabh" "chelsea"
```

Chelseas is 8 years old. can you add my cat's age to my_fam_ages vector? Try:

++++++++++++ # 3.2 MATRIX ++++++++++++

make a matrix with cbind()

```
my_fam_ages[4] <- 8 # all arrays need to be of the same length or weird things
#                   happen
my_fam <- cbind(names = my_fam_names, ages = my_fam_ages)
my_fam
```

```
##      names     ages
## [1,] "vandana" "59"
## [2,] "vijay"   "63"
## [3,] "saurabh" "32"
## [4,] "chelsea" "8"
```

another way of making a matrix

```
new_mat <- matrix(1:16, nrow = 4, byrow = FALSE) # Matrix of numbers 1 to 16 with
#                                                  4 rows, NOT assigning row-wise
new_mat
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
## [3,]    3    7   11   15
## [4,]    4    8   12   16
```

matrix functions

```
rownames(new_mat) <- c('A', 'B', 'C', 'D')
colnames(new_mat) <- c('col1', 'col2', 'col3', 'col4')
transposed_mat <- t(new_mat) # what does this do?
diag(new_mat)
```

```
## [1]  1  6 11 16
```

```
dim(my_fam)
```

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

++++++++++++++++ # 3.3 LISTS ++++++++++++++++

while vectors and matrices are of one type (num or char), lists can be mixed

```
lst <- list(c('i', 'want', 'ice', 'cream'),
            my_fam,
            c(TRUE, FALSE, TRUE, TRUE, TRUE, TRUE))

lst
```

```
## [[1]]
## [1] "i"     "want"  "ice"   "cream"
##
## [[2]]
##      names     ages
## [1,] "vandana" "59"
## [2,] "vijay"   "63"
## [3,] "saurabh" "32"
## [4,] "chelsea" "8"
##
## [[3]]
## [1]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
```

this list is made of an array (1st row), a matrix (my_fam), and an array of boolean variables. Notice the lengths/dimensions are not the same.

OPTIONAL: what functions can you use on lists?