Introduction to Using R for Data Analsysis

Gia Barboza, Fall 2021

Gia Elise Barboza

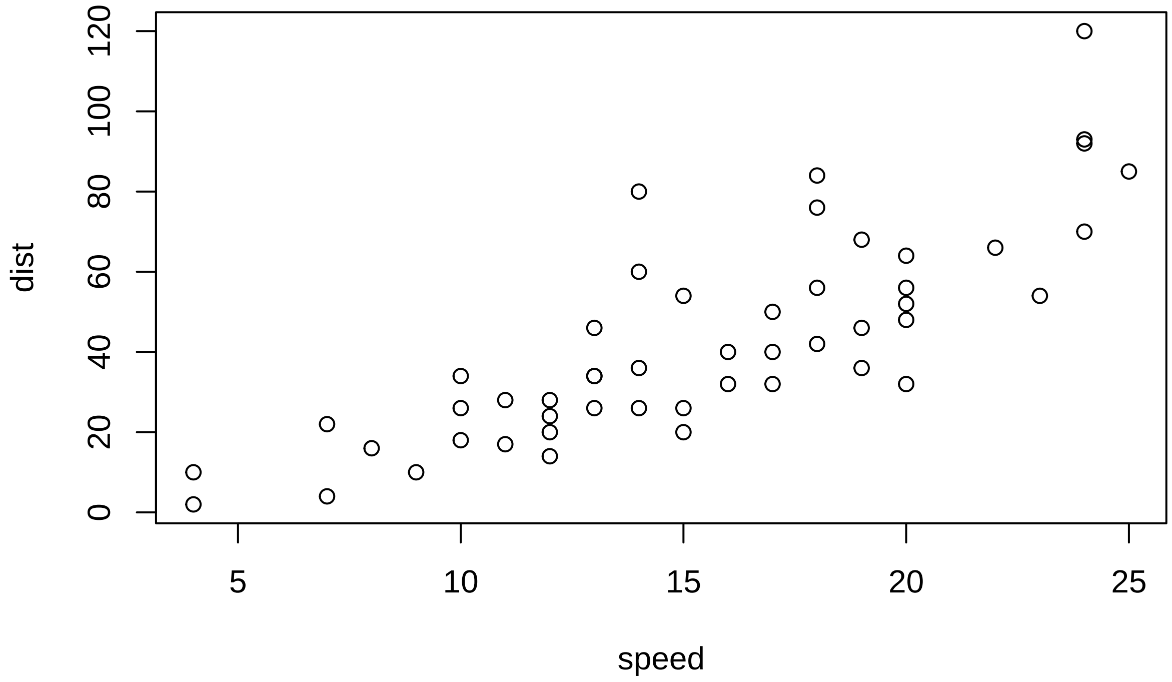
8/25/2023

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

If you see a figure of speed by distance then the notebook is working.

```
plot(cars)
```



Basic Operations Using R

```r
# add two numbers
1 + 3
```

R can be used as a calculator

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

```r
12 * (10 + 1)
```

```
## [1] 132
```

```r
four <- 1 + 3
four
```

R is an object oriented program. This means that the assignment operator is used to hold values. Here I am storing the results of 1 + 3 into the object called "four"

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

Working with vectors

```r
bunch_o_numbers <- c(1,4,2,8,11,100,8)
```

A vector is a really useful object in R. A vector essentially stores several numbers. Think about a column of numbers in a dataset as a vector that we can reference using a similar format.

```r
sum(bunch_o_numbers) # add up all of the numbers
```

We can also perform summary operations on numbers, vectors and other data formats. All of these are useful for different reasons. For example, we may want to know how many rows in a data set there are. For that, I used the length function below.

```
## [1] 134
```

```r
mean(bunch_o_numbers) # return the mean of the numbers
```

```
## [1] 19.14286
```

```
sd(bunch_o_numbers) # return the standard deviation of the numbers
```

```
## [1] 35.83494
```

```
length(bunch_o_numbers) # return the 'number' of numbers, i.e. the length
```

```
## [1] 7
```

```
var(bunch_o_numbers)
```

There is a big learning curve to R. There are many reasons for this. One reason is that there are many ways to do any one thing. Sometimes, one way is "better" or more efficient than another. Here is an example, there is a function to compute the variance. Of course, the variance is simply the standard deviation squared.

```
## [1] 1284.143
```

```
(sd(bunch_o_numbers))^2
```

```
## [1] 1284.143
```

```
sd(bunch_o_numbers) * sd(bunch_o_numbers)
```

```
## [1] 1284.143
```

```
standard_deviation <- sd(bunch_o_numbers)
variance <- standard_deviation^2
# Putting the parentheses around it tells R to display the output once the calculation is done
(variance <- standard_deviation^2)
```

```
## [1] 1284.143
```

```
# Make two vectors,
vec1 <- c(1,2,3,4,5)
vec2 <- c(11,12,13,14,15)

# Add a number, element-wise
vec1 + 10
```

Vectorized Operations

```
## [1] 11 12 13 14 15
```

```
# Element-wise quadratic:
vec1^2
```

```
## [1]  1  4  9 16 25
```

```
# Pair-wise multiplication:
vec1 * vec2
```

```
## [1] 11 24 39 56 75
```

```
# Pair-wise division
vec1 / vec2
```

```
## [1] 0.09090909 0.16666667 0.23076923 0.28571429 0.33333333
```

```
# Pair-wise difference:
vec2 - vec1
```

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

```
# Pair-wise sum:
vec1 + vec2
```

```
## [1] 12 14 16 18 20
```

```
# Compare the pair-wise sum to the sum of both vectors:
sum(vec1) + sum(vec2)
```

```
## [1] 80
```

```


# Mean of the vector 'vec1', *after* squaring the values
mean(vec1^2)
```

Applying multiple functions at once

```
## [1] 11
```

```
# Square the values of vector 'vec1' *before* squaring them
(mean(vec1))^2
```

```
## [1] 9
```

```
sum((vec1 - mean(vec1))^2) / (length(vec1) - 1) # you should recognize this as the variance
```

```
## [1] 2.5
```

```r
var(vec1)
```

```
## [1] 2.5
```

Working with Matrices

Another type of object is a matrix. A matrix is like a vector, except that it contains both rows and columns

```r
# Notice that, by default, R fills down columns, from left to right:
mat1 <- matrix(c(1,2,3,4,5,6,7,8,9), nrow=3, ncol=3)
mat1
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```r
t(mat1) # this will transpose the matrix (we will need to do this when we deal with network graphs)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

```r
# Get the sum of a vector
sum(mat1)
```

```
## [1] 45
```

```r
# Get mean, standard deviation, and number of observations (length):
mean(mat1)
```

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

```r
sd(mat1)
```

```
## [1] 2.738613
```

```r
length(mat1)
```

```
## [1] 9
```

```r
# We can also reference individual elements, rows or columns
mat1[1] # first element
```

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

```
mat1[3] # third element
```

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

```
mat1[, 1] #first column
```

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

```
mat1[1, ] #first row
```

```
## [1] 1 4 7
```

```
# Two other functions that come in handy are rowsums and colsums
rowSums(mat1)
```

```
## [1] 12 15 18
```

```
colSums(mat1)
```

```
## [1]  6 15 24
```

Working with data files

Each time you run R, it ’sees’ one of your folders (’directories’) and all the files in it. This folder is called the working directory. You can change the working directory in RStudio in a couple of ways. The first way is to set the working directory from the file menu. To do this go to Session from the file menu then Working Directory then ‘Choose Directory’. The other (easier) way is to do this via code. Why am I doing this? The reason is because unless you set the working directory you need to hard code the file paths (i.e. “C:/Users/gbarboza/Documents/PATH_TO_FILE”). Note: R uses forward slashes and not backslashes in files names, a backslash results in an error.

Hint: after typing setwd(“C:/) you can hit the tab key and the directory structure will appear to make your life easier.

```
setwd(
  "C:/Users/barboza-salerno.1/OneDrive - The Ohio State University/Documents/SWK8408/02_Lectures/Week 1/
  )
```

Working with R Packages

Almost every function in R depends on a package or library. These need to be installed independently. Warning: sometimes the package won’t install correctly. We can figure out why. The simplest reason is because the package runs on a version of R you don’t have (typically an older version). The website StackExchange will be your best friend if you are a serious R user. There is nothing I have not been able to figure out via this website. You are not the only one experiencing a problem, and people post the errors and issues on this website and offer solutions that helped them. Most of the packages we will use should run without issue, but for general problems this is also a good website.

As with everything, there are multiple ways to install a package. The easiest is from the file menu (Tools –> Install Packages). Another way is to use the built in function as follows

```
#install.packages("ggplot2") # this is the most popular graphics package in R
```

A couple of notes. Most of the packages are on the CRAN website. These are the ones available via the install.packages and the menu bar. Sometimes you may want to install a package that is not available from CRAN. If that happens, ask me.

Reading and Subsetting Data

There are many ways to read data into R, but we are going to keep things simple and show only a couple of options. Let's assume you have a fairly standard dataset, with variables organized in columns, and individual records in rows, and individual fields separated by a comma (a comma-separated values (CSV) file). Importing data into R is pretty straightforard. This can also be done via the import dataset button in the Environment tab (but I prefer to use code). Note that the button will also import SPSS files.

```
#mydata <- read.csv("PATH_TO_FILE/mydata.csv", fileEncoding="UTF-8-BOM") # the fileEncoding prevents R
# for example, if the path is "C:\Users\barboza-salerno.1\OneDrive - The Ohio State University\Document
mydata <- read.csv(
  "C:/Users/barboza-salerno.1/OneDrive - The Ohio State University/Documents/SWK8408/02_Lectures/Week 1
  fileEncoding="UTF-8-BOM"
  )
```

The most efficient and useful (not to mention cutting edge) way to wrangle data in R is via the dplyr package. Uncomment the following line and download it before trying the following code.

```
#UNCOMMENT THE LINE BY REMOVING THE "#" BEFORE RUNNING THE CODE
#install.packages("dplyr")
# Before you use a library you need to tell R as follows
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Let's do some data wrangling using dplyr (note there are many ways to do similar data tasks)
# Sometimes we need to rename the columns of the data
my_data_renamed <- mydata %>%
  rename(gender = var3)
```

```
# Sometimes we want to subset the data
my_data_selected <- mydata %>%
  select(var1, var3)
```

```r
# Sometimes we want to filter the data
my_data_filtered <- mydata %>%
  filter(var3 == "female")
```

```r
# The beauty of dplyr is that we can use %>% to perform multiple operations at once
my_data_subset <- mydata %>%
  select(var1, var3) %>%
  rename(gender = var3) %>%
  filter(gender == "female")
```

Notes: R is case sensitive which can be frustrating, so the following won't work

```r
# The beauty of dplyr is that we can use %>% to perform multiple operations at once
my_data_subset <- mydata %>%
  select(var1, var3) %>%
  rename(gender = var3) %>%
  filter(gender == "Female")
```

```r
# Let's rename both variables in one line
my_data_subset <- mydata %>%
  rename(age = var1, name = var2, gender = var3, inc = var4)
```

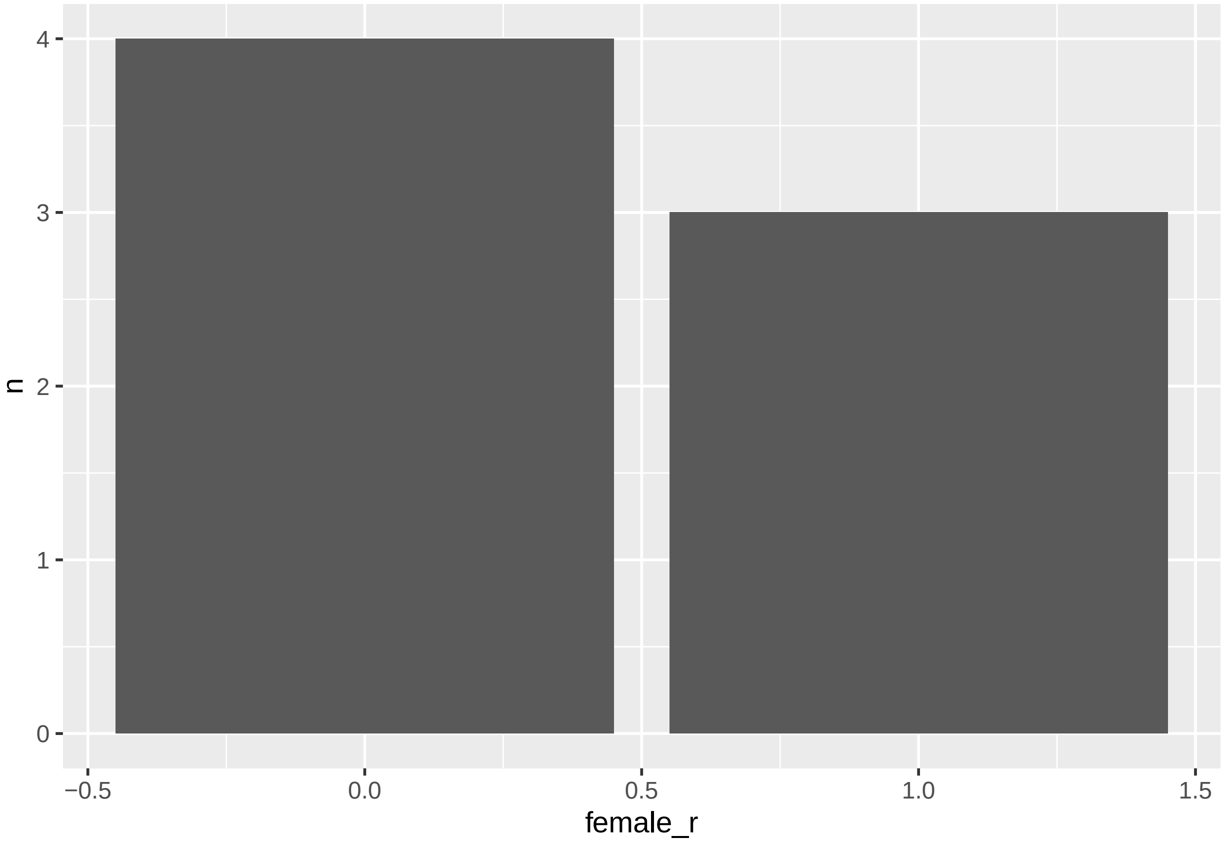
Notes: You need to be careful of the ordering. For example, you cannot rename the variable 'var3' to 'gender' and then select 'var3' (because it no longer exists). Also, you cannot filter gender == "female" before you actually rename the variable var3 to gender, etc. Being careful upfront saves a lot of frustration because some of the statements get very large and its hard to spot the error. The nice part is that you have a degree of flexibility that few statistical packages offer.

```r
my_data_subset <- my_data_subset %>%
  mutate(inc_r = case_when(inc < 20000 ~ "Low", inc >=20000 ~ "High"))

# Recoding dummy data using condition if_else since there are only two states (male or female)
my_data_subset <- my_data_subset %>%
  mutate(female_r = if_else(gender=="female", 1, 0))
```

We can also recode variables using dplyr as follows

```r
my_data_subset <- my_data_subset %>%
    group_by(female_r) %>%
    tally()

library(ggplot2)
ggplot(my_data_subset, aes(x = female_r, y = n)) + geom_bar(stat='identity')
```

We can even pipe the filtered, cleaned data into ggplot and create a nice graph, as follows



There are a few other data wrangling must-knows. One of them is removing empty rows so that only cases with full data are used. It is very difficult to teach all of the intricacies of R, it takes a while to build up your intuition. This is our first take at doing data in R.