```
### Intro Script

### Coding is not about memorizing anything. It's about trial-and-error, copying and
pasting. Just know where to look for it!


# Section 1: Basics ----------------------

# Anything after has is not part of the code
## These are just comments

1+1 # To run the code click ctrl+enter or select
# the lines and click run
5*3

4-4*5 # is the operation correct?

x=20*5 # you can store values as variables. They will show up in the environment tab on
the right

x*20 # and you can use these variables in operations


# Section 2: Vectors and Matrices ---------------------

# We can create vectors as collection of value
c(2,5,7,1,40)

V=c(2,5,7,1,40) # and we can store them
V^2 ## You can also do operations on them!
sum(V*10)


#of course, they don't have to be just numbers, they can be strings too
#string can include letters, numbers etc

"CDMX" #is an example of a string, we write them with quotation marks

String_vector=c("Econometrics","Mexico","New York","Carlos")

# to see content of an object (vector, variable) you can execute it, or print it
String_vector
print(String_vector)


#We can also create matrices
my_matrix <- matrix(1:9, nrow = 3)




# Section 3: Sub-setting vectors and matrices ---------

#To show 3rd element of the vector
V[3]
String_vector[3]

## If vectors have the same length, you can add them together
length(V)
V2=c(10,23,12,50,100)
V_sum=V+V2

## for assessing equality we need == two equal signs
logical=String_vector=="Mexico"
```

```
String_vector[logical]

## or find elements larger than 3 (it creates a logical vector)
V[V>3]

#To show elements 2-4
elements_2_4=String_vector[2:4]
elements_2_4

##In a matrix subsetting, we have two indices. First one is for rows, second is for
columns
subset_matrix <- my_matrix[2:3, 1:2]

# for entire row, select the row and leave column empty (it gives all columns)
my_matrix[2, ]

# for entire column, leave the row space empty
my_matrix[,1]


#Practice: Create a vector with values from 2 to 6. Square it, then add 200, then divide
by -5. tell me the 3rd element of the new vector

# Section 4: Functions ----

### R has a bunch of useful built in functions
### It also has libraries for a lot of more functions that you may want to use!

#each function has some arguments that you need to fill
sequence= seq(from=1, to=10, by=2)

#We can also draw variables from distributions
help(rnorm)
?rnorm

X=rnorm(n=1000, mean=0, sd=1)

#Exercise: find sum of values larger than 2

sum(X[X>2])

##calculate share of observations larger than 1.96

# You can vizualize the values of these variables
min(X)
max(X)
mean(X)
sd(X)
var(X)
sum(X)

quantile(X,0.975) # compare it to the table, why is it different

#what happens when you increase sample size to 10000?

summary(X)

# Let's plot the histogram
hist(X)
# And a boxplot
boxplot(X)

# Create another vvector of 100 which is random normal with mean 15 and and standard
deviation 3.
# Sum them and call Z. What kind of distribution do you expect for sum Z?
```

```
# Plot histogram, Calculate the mean and standard deviation of new variable did you get
what we expected?

Y=rnorm(n=1000, mean=15, sd=3)

Z=X+Y
hist(Z)
sd(Z)
mean(Z)

### Now go back to Y, but create 100000. Calculate its 0.975 quantile. substract the mean
15 and divide it by 3. What do you get? why it makes sense?

Y=rnorm(n=100000, mean=15, sd=3) # it replaces old Y
(quantile(Y,0.975)-15)/3

###Using random normal, calculate 95th quantile of chi(1)


# Section 7: Dataframes ----

#Dataframes are collection of columns

df=data.frame(C_x=X,C_y=Y,C_z=Z)

## we access columns using $ sign
df$C_x
# and we can perform the same operations as on other vectors

df$C_x+df$C_y

#We can eliminate columns
df$C_z=NULL

#and add them
df$C_w=df$C_x+df$C_y

##we can look at correlations and covariances between vectors/colums
cov(df$C_x,df$C_y)
cor(df$C_x,df$C_y)

###we can plot a scatterplot
plot(df$C_y,df$C_w)


###What covariance do you expect between X and W?
cov(df$C_y,df$C_w)



# Section 6: Importing Data ---------

listings <-
read.csv("C:/Users/kzysi/Dropbox/Itam_teaching/Markdowns/Intro_to_r/listings.csv",
comment.char="#")

# how many columns we have?
nrow(listings)
# how many rows
ncol(listings)

#####Some data cleaning
##mean price
mean(listings$price) ## ops, it thinks it is a character variable
listings$price =gsub(",", "", listings$price)
```

```
listings$price = gsub("\\$", "", listings$price)
listings$price=as.numeric(listings$price)

mean(listings$price) ##It means we have missing values
##how many NAs we have?
sum(is.na(listings$price))

##Two ways to deal with this

##1. omit those with NA
mean(listings$price, na.rm=TRUE)

##2. Removing missing observations
listings=listings[!is.na(listings$price),]
mean(listings$price)

##find worst score with at least 10 reviews
listings$review_scores_rating=as.numeric(listings$review_scores_rating)
listings_10_reviews=listings$review_scores_rating[listings$number_of_reviews>10]
summary(listings_10_reviews)

##which one is it?
worst_listing=listings[listings$number_of_reviews>10 & listings$review_scores_rating==0,]


###there are usually multiple ways to do the same thing as there are multiple functions
install.packages("tidyverse")##this is how you install libraries - collection of functions
library(tidyverse)

worst_listing=listings%>%
  filter(number_of_reviews>10)%>%
  filter(review_scores_rating==0)


#Is mode host name Carlos?
#how to find a most frequent value of a variable? - use chatgpt?


#### Which neighborhoods have most of the airbnbs?
table(listings$neighbourhood_cleansed)

### Illustrate it with barplot
barplot(table(listings$neighbourhood_cleansed))


barplot(table(listings$neighbourhood_cleansed), las=2)

install.packages("ggplot2")
library(ggplot2)


### We will sort it so it looks nicer
freq_table=table(listings$neighbourhood_cleansed) # but we need data frame for ggplot!
freq_df <- data.frame(Frequency = freq_table)


# Create the plot ## it's all about adding subsequent layers
ggplot(freq_df, aes(x = Frequency.Var1, y = Frequency.Freq)) + #aes - aesthetics - so
basically what our axis will be about
  geom_bar(stat = "identity")

# Create the plot, rotate the text and add the labels
ggplot(freq_df, aes(x = Frequency.Var1, y = Frequency.Freq)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
```

```
  labs(title = "Bar Plot", x = "Neighborhood", y = "Frequency")

###order by the most frequent onesx
ggplot(freq_df, aes(x = reorder(Frequency.Var1, Frequency.Freq), y = Frequency.Freq))  +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  labs(title = "Bar Plot", x = "Neighborhood", y = "Frequency")


# Create the plot, make a nicer looking one
ggplot(freq_df, aes(x = reorder(Frequency.Var1, Frequency.Freq), y = Frequency.Freq)) +
  geom_bar(stat = "identity")+
  theme_minimal() + ##it needs to be before the other "theme" element
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  labs(title = "Neighborhoods by the number of listings", x = "Neighborhood", y =
"Frequency")


### Histogram of prices

ggplot(listings, aes(x=price)) +
  geom_histogram()+
  theme_minimal()


###adjust the number of bins
ggplot(listings, aes(x=price)) +
  geom_histogram(bins=100)+
  theme_minimal()

### we have some outliers!

###adjust the number of bins
ggplot(listings, aes(x=price)) +
  geom_boxplot()+
  theme_minimal()


##What are these guys?!!?!

expensive_listing=listings%>%
  filter(price>25000)

###let's eliminate these with price above 100000
listings_no_outliers=listings%>%
  filter(price<25000)

ggplot(listings_no_outliers, aes(x=price)) +
  geom_histogram()+
  theme_minimal()

ggplot(listings_no_outliers, aes(x=price)) +
  geom_boxplot()+
  theme_minimal()

### Task for you
#1. Choose a neighborhood
#2. Subset the data to this neigborhood only
#3. Calcuate a mean, median and standard deviation of price and rating
#4. Make a boxplot of prices and eliminate outliers if they exist
#5. Find the worst rated airbnb with more than 20 reviews
#6. Find the most expensive airbnb
#7. Use chatgpt to create a stacked barplot of room type

room_counts <- table(listings$room_type)
```

```
room_counts_df <- data.frame(RoomType = names(room_counts), Count =
as.vector(room_counts))

# Create a pie chart using ggplot2
pie_chart <- ggplot(room_counts_df, aes(x = "", y = Count, fill = RoomType)) +
  geom_bar(stat = "identity", width = 1, color = "white") +
  coord_polar("y", start = 0) +
  theme_void() +
  labs(title = "Distribution of Room Types")

#### What are some interesting questions you want to learn about Airbnb in CDMX?

#### Code interpreter


#### additional material, sampling distribution, tidyverse, ggplot -----

#### Illustrating confidence intervals in R

##What is the true mean price? we saw it before
mean(listings$price, na.rm=TRUE)
sd(listings$price, na.rm=TRUE)

##We will do it in the following steps

#Step 1. Create a sample of 300 listings. Calculate mean price, standard deviation, and
the confidence interval
sample <- listings %>%
  sample_n(500, replace = FALSE)
m=mean(sample$price, na.rm=TRUE)
sdev=sd(sample$price, na.rm=TRUE)
CI95_lower=m-sdev*1.96/sqrt(500)
CI95_upper=m+sdev*1.96/sqrt(500)

### I just remember that 1.96 is a 97.5% quantile. If I dont, I would do the following:
qnorm(0.975,mean=0,sd=1) #normal
qt(0.975,df=10) #student t

## Simpler method to get confidence interval
t.test(sample$price)
t.test(sample$price, conf.level = 0.99)

#Step 2. Repeat the sampling procedure 1000 times, save mean and confidence intervals for
each - this will create a sampling distribution of the sample mean

#construct a loop, it will create vectors of prices

for (i in 1:10){
  i*2
  print(i+10)
}

m=0
CI95_lower=0
CI95_upper=0

###Without outliers

##loop for us
for (i in 1:3000){
  sample <- listings_no_outliers %>%
    sample_n(500, replace = FALSE)
  m[i]=mean(sample$price, na.rm=TRUE)
  sdev[i]=sd(sample$price, na.rm=TRUE)
  CI95_lower[i]=m[i]-sdev[i]*1.96/sqrt(500)
```

```
  CI95_upper[i]=m[i]+sdev[i]*1.96/sqrt(500)
}


##put them together in a dataframe
sampling_distribution=data.frame(m,CI95_lower,CI95_upper)

#Step 3. Visualize the sampling distribution. Check the expectation and variance.
ggplot(sampling_distribution, aes(x=m)) +
  geom_histogram(bins=50)+
  theme_minimal()

mean(sampling_distribution$m)
sd(sampling_distribution$m)

#Step 4. Check what share of confidence intervals cover the true mean.
true_mean=mean(listings_no_outliers$price, na.rm=TRUE)
true_mean

mean_check=sampling_distribution$CI95_lower<true_mean &
sampling_distribution$CI95_upper>true_mean # it will give true only if two conditions are
true

sum(mean_check)/3000



###With outliers

##loop for us
for (i in 1:2000){
  sample <- listings %>%
    sample_n(500, replace = FALSE)
  m[i]=mean(sample$price, na.rm=TRUE)
  sdev[i]=sd(sample$price, na.rm=TRUE)
  CI95_lower[i]=m[i]-sdev[i]*1.96/sqrt(500)
  CI95_upper[i]=m[i]+sdev[i]*1.96/sqrt(500)
}


##put them together in a dataframe
sampling_distribution=data.frame(m,CI95_lower,CI95_upper)

#Step 3. Visualize the sampling distribution. Check the expectation and variance.
ggplot(sampling_distribution, aes(x=m)) +
  geom_histogram(bins=50)+
  theme_minimal()

mean(sampling_distribution$m)
sd(sampling_distribution$m)

#Step 4. Check what share of confidence intervals cover the true mean.
true_mean=mean(listings$price, na.rm=TRUE)

mean_check=sampling_distribution$CI95_lower<true_mean &
sampling_distribution$CI95_upper>true_mean # it will give true only if two conditions are
true

sum(mean_check)/2000
```