



Statistics in R: A Basic Guide

Abby K. Johnson
Thao P. Tran
Department of Psychology
Colorado State University



Section 1: Downloading R and RStudio

R is a free, open-source language, graphics, and computing environment available for Windows and Mac computers.

To install R: go to <http://www.r-project.org/> , click on the download R link under “Getting Started” and choose <https://cloud.r-project.org/>. Next, choose your system, and click “install R for the first time.” Be sure to download the latest version.

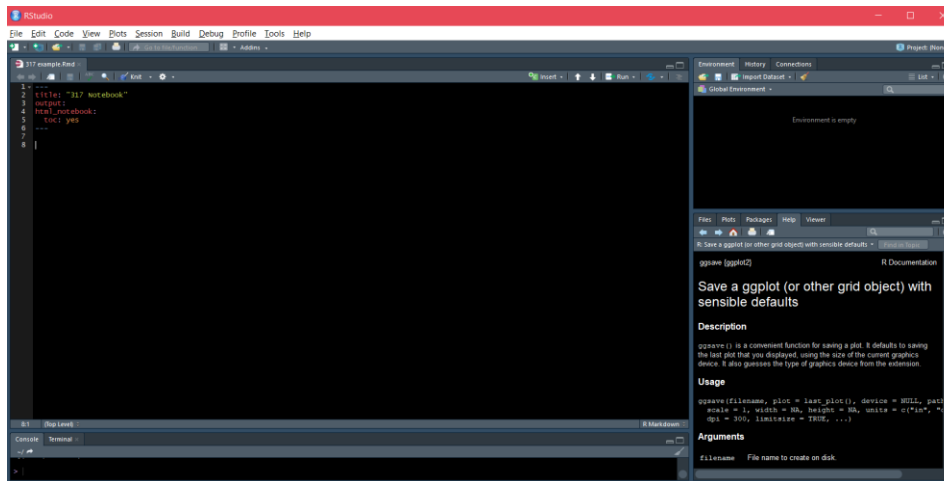
R studio is an integrated development environment (IDE) for R. Download it at <http://www.rstudio.com/> . Click “Download Now”

Section 2: Getting Started

Basics

The basic layout of RStudio

1. Open RStudio



Starting a New Project

The first thing to do when creating a new project is to create a project folder. Before doing this, please create a folder on your computer just for this class (or your work in R). For example the project folder on my example is called RM and its path is Downloads/RM

1. To create a new project, File > New Project > New Directory > Empty Project

2. Name your project “317Activities”

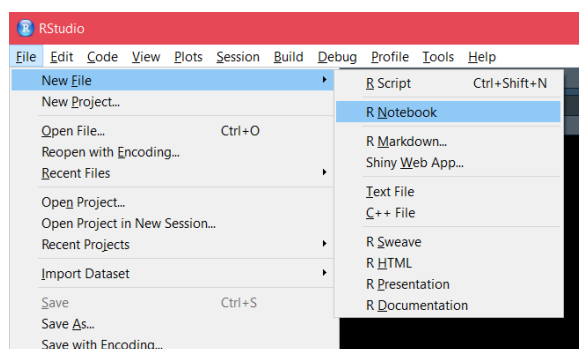
Remember where you save your project- save it in the `_folder` on your computer where you keep files for this project (or all of your R projects). So on my computer, it looks like

Downloads/RM/317Activities

When you begin working on this project and open RStudio again, the first thing you’ll do is open the project via File > Open Project

Creating an R Notebook

1. Create an R Notebook by clicking File > New File > R Notebook



2. Rename your notebook, “317 Notebook” after where it says *title*

****optional - `toc: yes` and `toc_float: true` are helpful if you want to create a floating table of content on the left of the output document**

```
1 ---
2 title: "317 Notebook"
3 output:
4   html_notebook:
5     toc: yes
6     toc_float: true
7 ---
```

3. Save your R Notebook by clicking File > Save As > 317Notebook

Make sure that you save your R Notebook in your R Folder

Install Packages

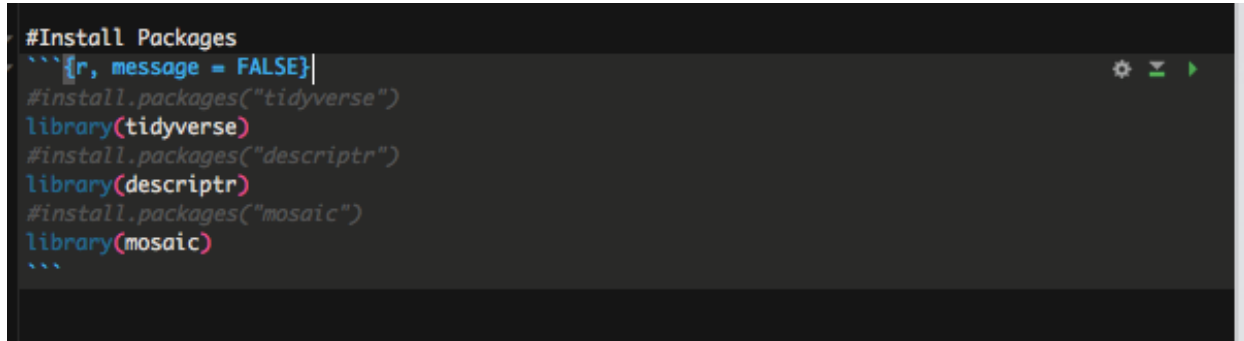
Packages allow us to use different functions in R. One of the most popular packages is called “tidyverse” by Hadley Wickham, chief scientist at RStudio. You only need to install a package once via the **`install.packages()`** function. After you install the package, to use it in your R Notebook, you will use the **`library()`** function.

1. Type the following code into your R chunk to install the tidyverse, descriptr, and mosaic packages

```
install.packages("tidyverse")
```

```
library(tidyverse)
install.packages("descriptr")
library(descriptr)
install.packages("mosaic")
library(mosaic)
```

We can delete the automated text that follows this code by typing `message = FALSE` after your ````{r, }` to clean up our notebook a bit. It will look like this:

A screenshot of an R chunk in a notebook. The chunk is titled "#Install Packages" and contains the following code:

```
```{r, message = FALSE}
#install.packages("tidyverse")
library(tidyverse)
#install.packages("descriptr")
library(descriptr)
#install.packages("mosaic")
library(mosaic)
```
```

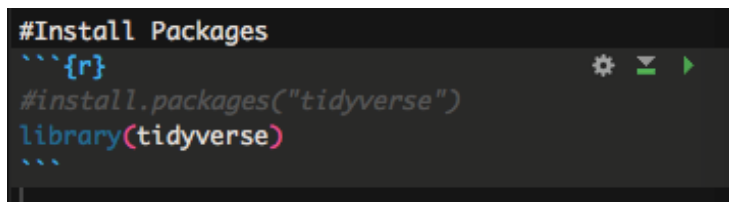
 The code is syntax-highlighted. On the top right of the chunk, there are three icons: a gear (settings), a downward arrow (collapse), and a green right-pointing triangle (run).

Click the green arrow on the top right of your R chunk to run the code.

You can create a title for this R chunk by writing your intended title behind a hashtag above your chunk. # Indicates a first level header, ## a second level header, etc.

Hashtags, when used in Rchunks (rather than outside of the chunk) null code in your Rchunks so you can write yourself notes in the code. This is helpful especially when you're first learning R, You'll see examples of these in my notebook.

E.g., #Install Packages

A screenshot of an R chunk in a notebook. The chunk is titled "#Install Packages" and contains the following code:

```
```{r}
#install.packages("tidyverse")
library(tidyverse)
```
```

 The code is syntax-highlighted. On the top right of the chunk, there are three icons: a gear (settings), a downward arrow (collapse), and a green right-pointing triangle (run).

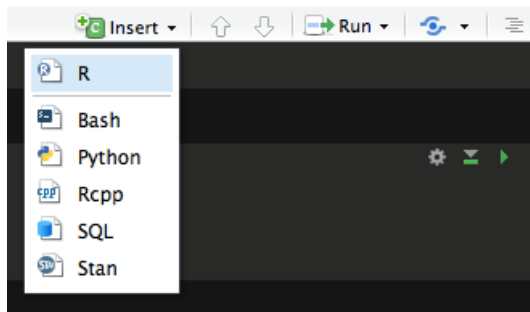
Import a Dataset

Your dataset should be saved as a .csv file from Qualtrics. Before installing any .csv file into R you need to make sure that it is in your project folder that you created on your computer. So for example, you might export a .csv file from qualtrics (recommended). For a refresher- <https://www.qualtrics.com/support/survey-platform/data-and-analysis-module/data/download-data/export-formats/>

When you save your .csv file from qualtrics be sure you save it in your project folder (otherwise R will not be able to find it!) E.g., click Save As > Downloads > RM > 317Activities

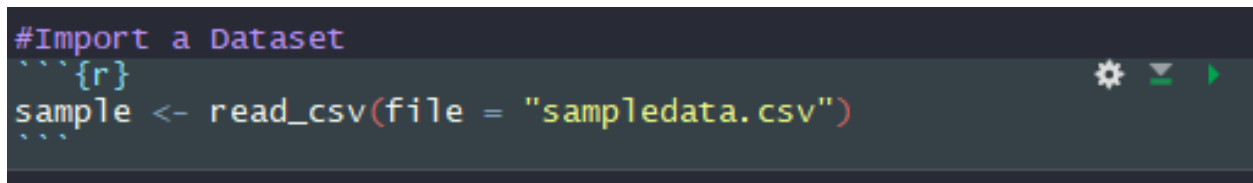
You will use your own data. However, for learning purposes, we will use the **sampladata.csv**. The data was created based on an application of Terror Management Theory¹. To start, download and save the **sampladata.csv** in your project folder.

1. To import the dataset into RStudio, **you'll need to create an R chunk**. To do this, find the “Insert” button on the top of your RNotebook and click on it. In the dropdown menu, click R (OR you can use Ctrl/Command + Alt + I). The chunk will appear wherever you left your cursor.



2. Next, type the following code into a separate R chunk:

[title the chunk “#Import a Dataset”]
sample <- read_csv(file= "sampladata.csv")



- “sample” is the name that we are giving to our file.
- “read_csv()” is our function
- “sampladata.csv” is the name of the file exactly as it appears in my RM folder

Now click on the Environment tab. Notice that when you click on the Data file called “sample,” a new tab will come up in your Notebook with all of your data

¹ Orit Taubman, B. (2011). Is the Meaning of Life Also the Meaning of Death? A Terror Management Perspective Reply. *Journal of Happiness Study*, 12(3), 385-399. doi: 10.1007/s10902-010-9201-2

| | cond | est_1 | est_2 | est_3 | est_4 | est_5 | est_6 | est_7 |
|----|------|-------|-------|-------|-------|-------|-------|-------|
| 1 | 1 | 2 | 2 | 2 | 3 | 3 | 4 | 3 |
| 2 | 1 | 1 | 4 | 2 | 4 | 3 | 3 | 2 |
| 3 | 1 | 2 | 4 | 4 | 1 | 3 | 2 | 1 |
| 4 | 1 | 3 | 3 | 2 | 3 | 1 | 1 | 4 |
| 5 | 1 | 4 | 4 | 1 | 2 | 2 | 3 | 1 |
| 6 | 1 | 4 | 4 | 3 | 4 | 3 | 4 | 4 |
| 7 | 1 | 3 | 4 | 2 | 2 | 3 | 2 | 2 |
| 8 | 1 | 2 | 4 | 4 | 2 | 1 | 3 | 2 |
| 9 | 1 | 4 | 2 | 2 | 4 | 2 | 4 | 4 |
| 10 | 1 | 2 | 3 | 4 | 1 | 2 | 4 | 3 |
| 11 | 1 | 3 | 2 | 1 | 4 | 1 | 1 | 1 |
| 12 | 1 | 3 | 1 | 3 | 3 | 1 | 4 | 1 |
| 13 | 1 | 2 | 3 | 1 | 1 | 2 | 2 | 2 |

Section 3: Cleaning the Data

Omitting Cases with Missing Values

It's reasonable to assume that all of our participants did not answer all of the questions. I.e., we have some missing data to work with.

Consider the missing variables that are non-negotiable for your study, e.g., participants' scores on one of your IV/DV scales, participants' gender, etc. You'll want to omit these cases because the missing data will affect your hypothesis testing.

To see the cases in which we have missing data, execute the following code:

```
sample[!complete.cases(sample),]
```

To omit cases in which data is non-existent for these variables use the following code:

```
sample <- na.omit(sample)
```

In the above code, we are simply overwriting the sample dataset and leaving out missing data. Na.omit is the function that omits cases with missing data from our dataset.

```
#cleaning the Data
{r}
sample[!complete.cases(sample),]

sample <- na.omit(sample)
```

Reverse Coding and creating scoring scales

Like in the case of missing data, you will have instances in which you'll need to reverse code items on your scales. Finally, you'll want to score a scale for your participants. For example, in our sample dataset, each participant had a total score on the MIL scale, and there will be a mean score for the MIL within your entire sample.

Below is the key for the items in the sample dataset:

| Name of scale | Item | Statement | Response option | Scoring |
|--|--------|--|---|--|
| Meaning in life questionnaires
Steger, M. F., Frazier, P., Oishi, S., & Kaler, M. (2006) | mil_1 | I understand my life's meaning | Likert scale 1-7
1 - Absolutely untrue
2 - Mostly untrue
3 - Somewhat untrue
4 - Can't say true or false
5 - Somewhat true
6 - Mostly true
7 - Absolutely true | MIL is compiled of two subscales:
<u>Presence of meaning</u> = 1, 4, 5, 6, & 9
(reverse-coded)
<u>Search for meaning</u> = 2, 3, 7, 8, & 10 |
| | mil_2 | I am looking for something that makes my life feel meaningful | | |
| | mil_3 | I am always looking to find my life's purpose | | |
| | mil_4 | My life has a clear sense of purpose | | |
| | mil_5 | I have a good sense of what makes my life meaningful | | |
| | mil_6 | I have discovered a satisfying life purpose | | |
| | mil_7 | I am always searching for something that makes my life feel significant | | |
| | mil_8 | I am seeking a purpose or mission for my life | | |
| | mil_9 | My life has no clear purpose | | |
| | mil_10 | I am searching for meaning in my life | | |
| Self-esteem scale
Rosenberg (1979) | est_1 | I feel that I am a person of worth, at least on an equal plane with others | Likert scale 1-4
1 - Strongly disagree
2 - Disagree
3 - Agree
4 - Strongly agree | The scale ranges from 0-30. Scores between 15 and 25 are within normal range; scores below 15 suggest low self-esteem
Item 3, 5, 9, 10 should be reverse-coded |
| | est_2 | I feel that I have a number of good qualities | | |
| | est_3 | All in all, I am inclined to feel that I am a failure | | |
| | est_4 | I am able to do things as well as most other people | | |
| | est_5 | I feel I do not have much to be proud of | | |
| | est_6 | I take a positive attitude toward myself | | |
| | est_7 | On the whole, I am satisfied with myself | | |
| | est_8 | I wish I could have more respect for myself | | |
| | est_9 | I certainly feel useless at times | | |
| | est_10 | At times I think I am no good at all | | |
| Condition | cond | | 1 - Control
2 - Mortality salience - self
3 - Mortality salience - loved ones | |

Using the multicon package: In a new code chunk, type in these codes!

```
#Creating the scoring scales using multicon package
```{r, message = FALSE}
#install.packages("multicon")
library(multicon)

MLQset <- sample[, grep("mil", names(sample))]
MLQlist <- list(MLQP = c(1, 4, 5, 6, -9),
 MLQS = c(2, 3, 7, 8, 10))

MLQscores <- scoreTest(MLQset, MLQlist, nomiss = 0.01, rel = F)

sample <- data.frame(sample, MLQscores)
```
```

Let's break this down from the top!

- We first create a new object called **MLQset**, and we use the "<-" to assign value to it
- sample[row, column] allows us to draw a subset from the dataset.
 - In the square brackets, we leave the row argument blank because we don't want to pick any specific row of data
 - We specify the column argument with "grep()" – Grab all the columns whose names include "mil"
- Then, we create **MLQlist** - a list with the scoring keys: These are the column numbers in MLQset associated with the items for each dimension and whether they're reverse coded or not. Item 9 (as noted) should be reverse-coded
- "scoreTest()" will calculate the mean score for each subscale that we specify. We save the result in an object called **MLQscores**
- Finally, we merge the **MLQscores** into our original dataset - **sample**

Follow this, and create the mean score for the Self-esteem scale!

Examining reliability of the scales

Before testing our hypothesis, we need to make sure that the scales being used are reliable. We can use the "alpha()" function in R to examine the internal consistency of our scales.

alpha(MLQset, keys = "mil_9")

- MLQset is the subset of data that we draw from the original dataset (earlier)
 - keys = "mil_9" indicates that item 9 should be reverse-scored before alpha is calculated
- If we have more than 2 items that should be reverse-coded, use: **keys = c("x", "y", "z")**

```
#Examining reliability of the scales
```{r}
alpha(MLQset, keys = "mil_9")
```
```


Section 4: Preliminary Info and Descriptive Stats

Preliminary Info

The following functions will give us some preliminary info on our dataset. Remember that you'll insert your dataframe where I write "sample"

names(sample) will give us the names of all of the variables in the dataset

str(sample) will give us the structure of the dataset

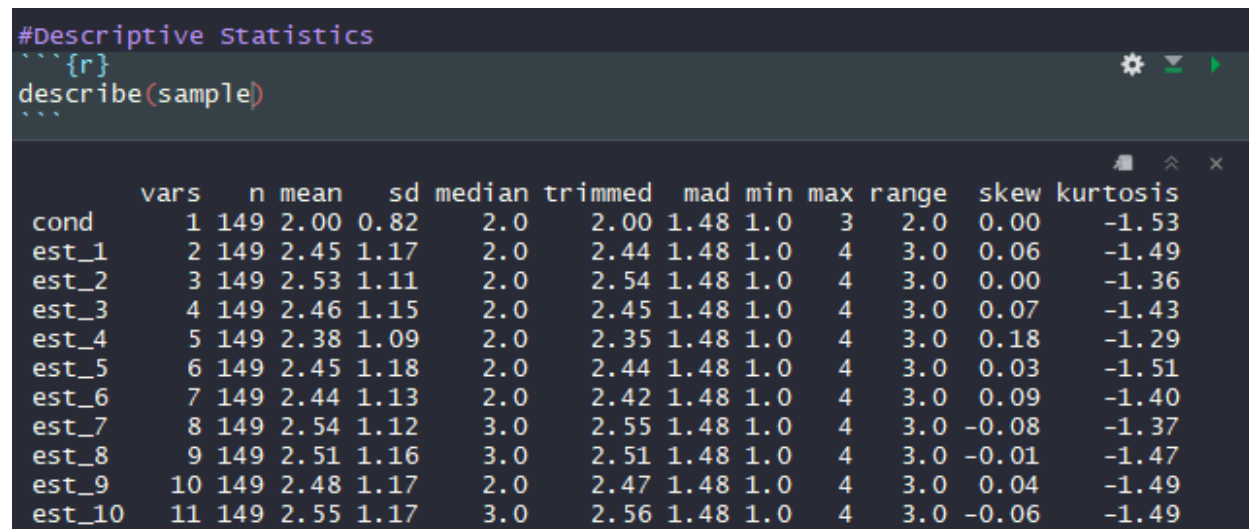
head(sample) will give us the first six rows of the dataset

Descriptive Stats

Use the following code to obtain descriptive stats for each variable in your dataset:

describe(sample)

remember, sample = our dataset



```
#Descriptive Statistics
##{r}
describe(sample)
```

| | vars | n | mean | sd | median | trimmed | mad | min | max | range | skew | kurtosis |
|--------|------|-----|------|------|--------|---------|------|-----|-----|-------|-------|----------|
| cond | 1 | 149 | 2.00 | 0.82 | 2.0 | 2.00 | 1.48 | 1.0 | 3 | 2.0 | 0.00 | -1.53 |
| est_1 | 2 | 149 | 2.45 | 1.17 | 2.0 | 2.44 | 1.48 | 1.0 | 4 | 3.0 | 0.06 | -1.49 |
| est_2 | 3 | 149 | 2.53 | 1.11 | 2.0 | 2.54 | 1.48 | 1.0 | 4 | 3.0 | 0.00 | -1.36 |
| est_3 | 4 | 149 | 2.46 | 1.15 | 2.0 | 2.45 | 1.48 | 1.0 | 4 | 3.0 | 0.07 | -1.43 |
| est_4 | 5 | 149 | 2.38 | 1.09 | 2.0 | 2.35 | 1.48 | 1.0 | 4 | 3.0 | 0.18 | -1.29 |
| est_5 | 6 | 149 | 2.45 | 1.18 | 2.0 | 2.44 | 1.48 | 1.0 | 4 | 3.0 | 0.03 | -1.51 |
| est_6 | 7 | 149 | 2.44 | 1.13 | 2.0 | 2.42 | 1.48 | 1.0 | 4 | 3.0 | 0.09 | -1.40 |
| est_7 | 8 | 149 | 2.54 | 1.12 | 3.0 | 2.55 | 1.48 | 1.0 | 4 | 3.0 | -0.08 | -1.37 |
| est_8 | 9 | 149 | 2.51 | 1.16 | 3.0 | 2.51 | 1.48 | 1.0 | 4 | 3.0 | -0.01 | -1.47 |
| est_9 | 10 | 149 | 2.48 | 1.17 | 2.0 | 2.47 | 1.48 | 1.0 | 4 | 3.0 | 0.04 | -1.49 |
| est_10 | 11 | 149 | 2.55 | 1.17 | 3.0 | 2.56 | 1.48 | 1.0 | 4 | 3.0 | -0.06 | -1.49 |

Section 5: Data Visualization

Density Plots

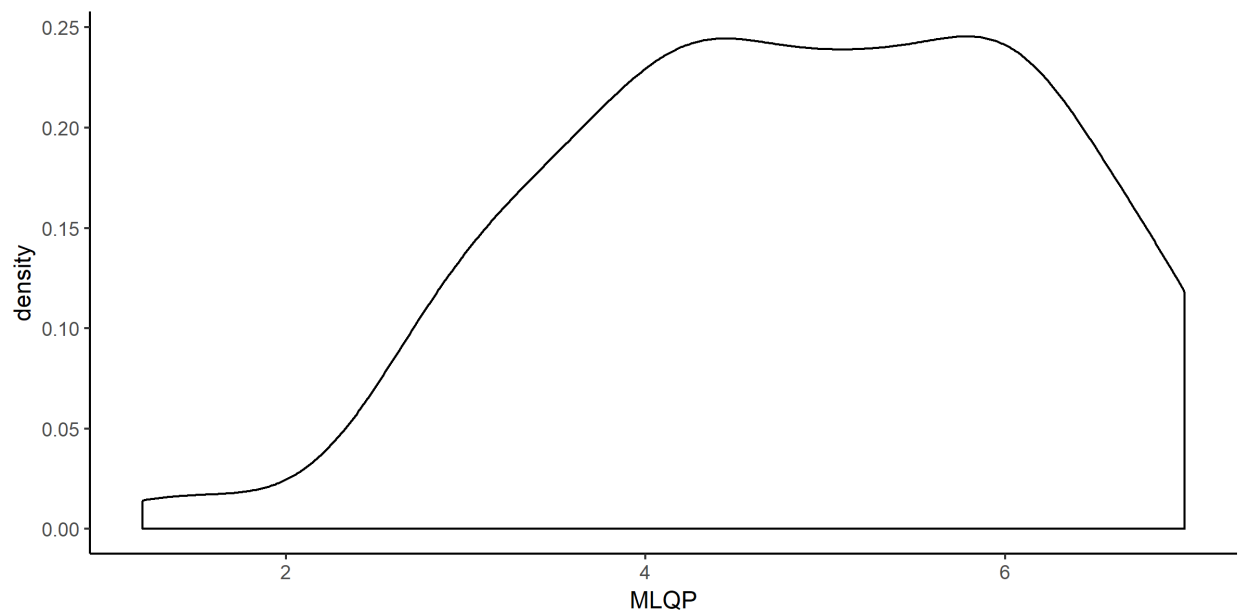
One of the best things about R is that it allows us to create really visually appealing graphs. To work with data viz, you'll need to install the package called "ggplot2."

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

Let's start with a simple density plot.

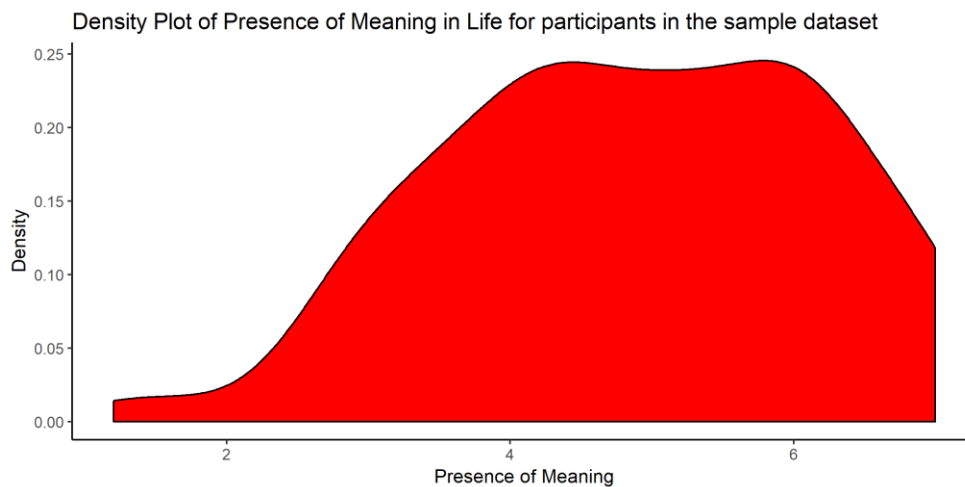
```
#Data visualization  
##Density plot  
`{r}  
ggplot(sample, aes(x = MLQP)) +  
  geom_density() +  
  theme_classic()  
  
ggsave("density_plot.png", width = 8, height = 4)
```

- "ggplot()" is our function
- "sample" designates the dataset
- "aes(x =)" allows us to pick the variable for the x axis
- "geom_density()" designates what kind of plot/graph we want. There are lots of ways to enhance this
- "theme_classic()" removes all the unnecessary grid lines in the background (which are default in R)
- "ggsave()" allows us to save the plot in picture format. This is particularly helpful when we want to include the plot in our Appendix!



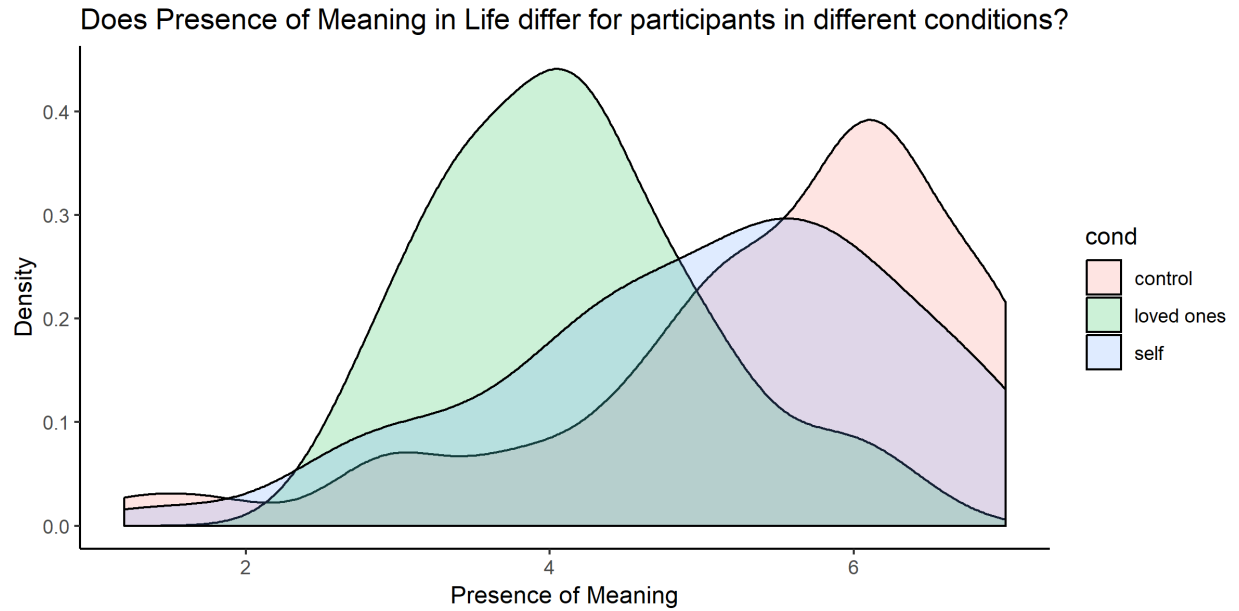
Enhance your code to add color and labels:

```
ggplot(sample, aes(x = MLQP)) +  
  geom_density(fill = "red") +  
  theme_classic() +  
  labs(title = "Density Plot of Presence of Meaning in Life for participants in the sample  
dataset",  
        x = "Presence of Meaning", y = "Density")  
  
ggsave("density_plot.png", width = 8, height = 4)
```



Group by condition:

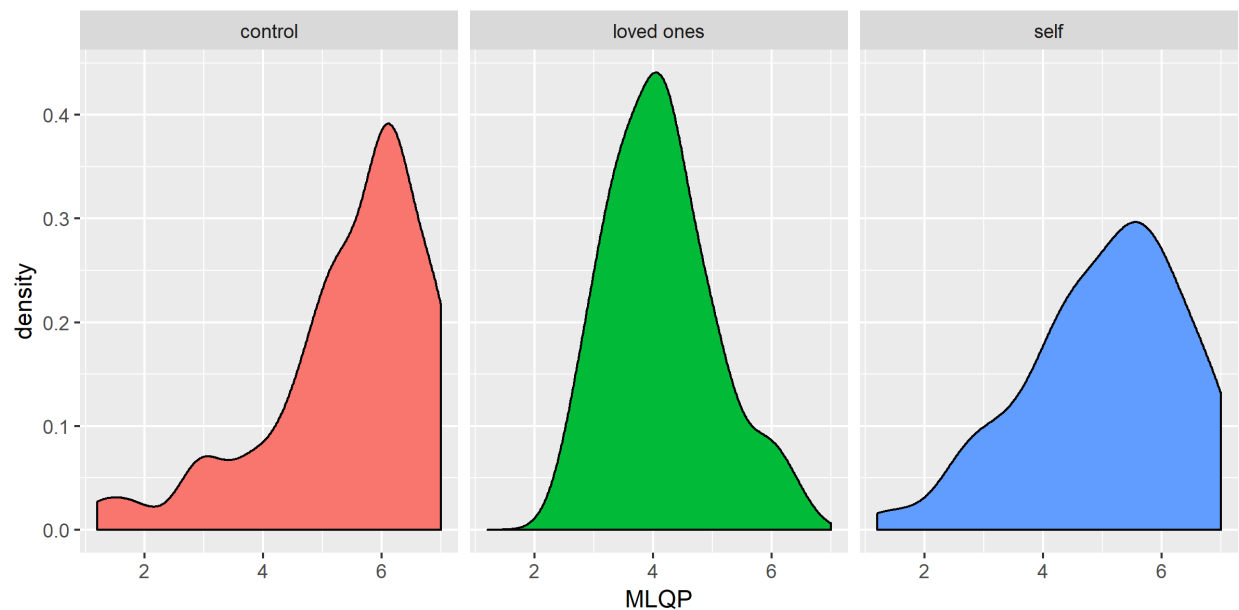
```
ggplot(sample, aes(x = MLQP, group = cond, fill = cond)) +  
  geom_density(alpha = .2) +  
  theme_classic() +  
  labs(title = "Does Presence of Meaning in Life differ for participants in different  
conditions?",  
        x = "Presence of Meaning", y = "Density")  
  
ggsave("density_plot.png", width = 8, height = 4)
```



Alternative method:

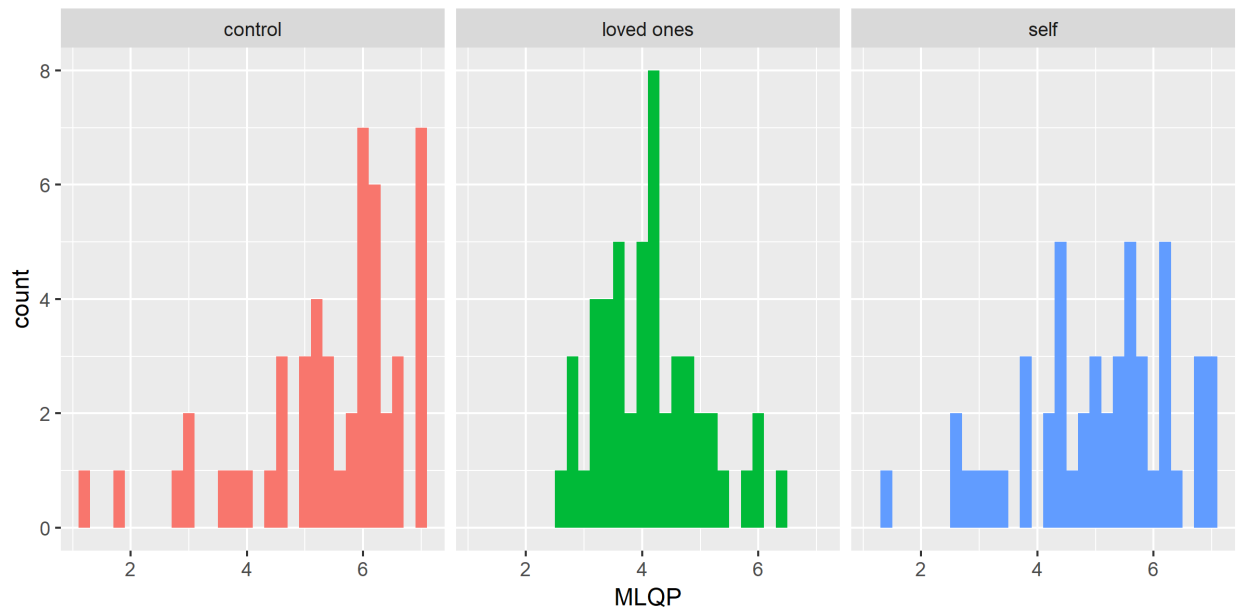
```
ggplot(sample, aes(x = MLQP, group = cond, fill = cond)) +
  geom_density(show.legend = FALSE) +
  facet_wrap(~cond) +
  labs(title = "Does Presence of Meaning in Life differ for participants in different
conditions?",
       x = "Presence of Meaning", y = "Density")

ggsave("density_plot.png", width = 8, height = 4)
```



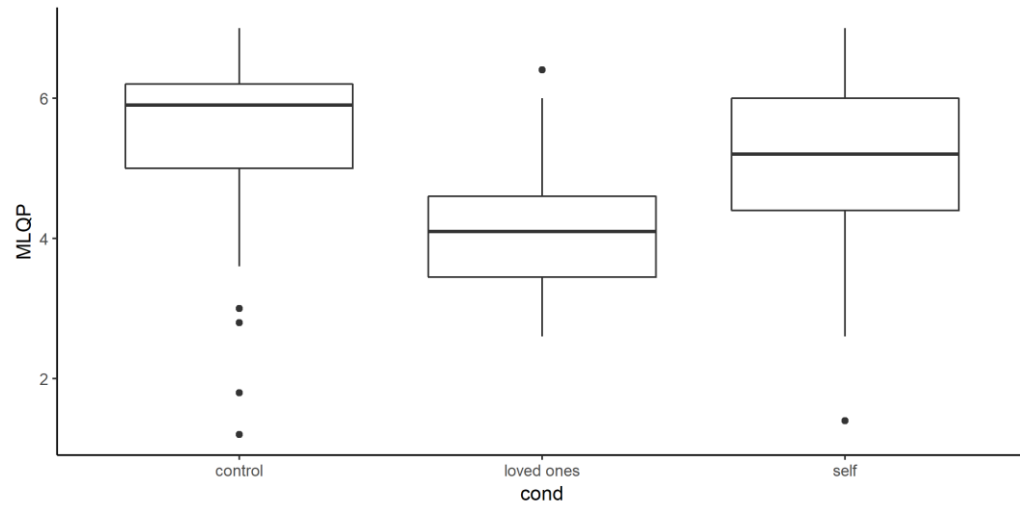
Try a histogram:

```
ggplot(sample, aes(x = MLQP, group = cond, fill = cond)) +  
  geom_histogram(show.legend = FALSE) +  
  facet_wrap(~cond) +  
  labs(title = "Does Presence of Meaning in Life differ for participants in different  
conditions?",  
        x = "Presence of Meaning", y = "Count")  
  
ggsave("histogram_plot.png", width = 8, height = 4)
```



A box plot:

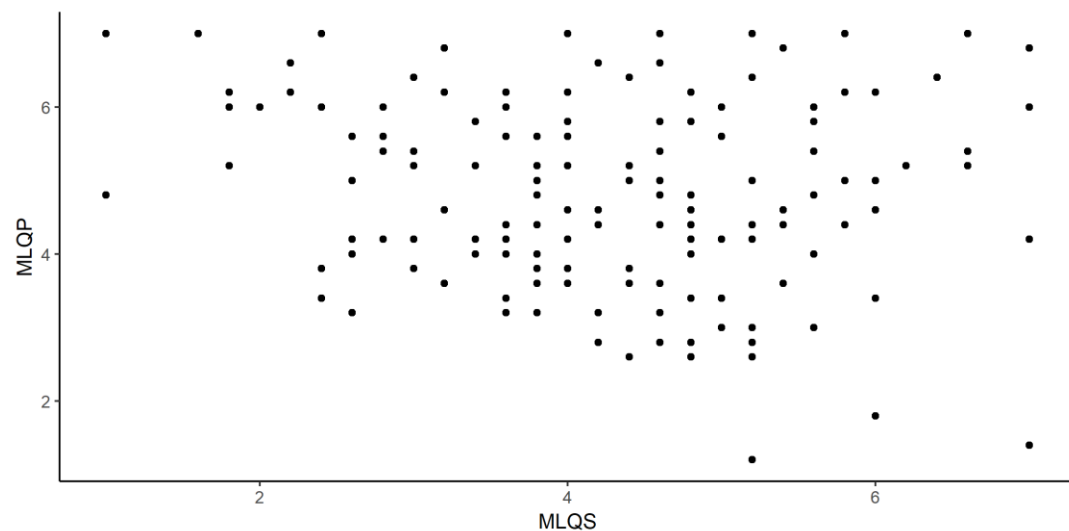
```
ggplot(sample, aes(x = cond, y = MLQP, group = cond)) +  
  geom_boxplot(show.legend = FALSE) +  
  theme_classic() +  
  labs(title = "Does Presence of Meaning in Life differ for participants in different  
conditions?",  
        x = "Experimental condition", y = "Presence of Meaning in Life")  
  
ggsave("boxplot.png", width = 8, height = 4)
```



A scatter plot:

```
ggplot(sample, aes(x = MLQS, y = MLQP)) +
  geom_point() +
  theme_classic() +
  labs(title = "What is the relationship between Presence and Search for Meaning in Life",
        x = "Search for Meaning in Life", y = "Presence of Meaning in Life")
```

```
ggsave("scatterplot.png", width = 8, height = 4)
```



Section 6: Inferential Statistics (stats to test your hypotheses!)

Correlations

It's always good to see how your variables are correlated. A correlation matrix can paint a clear picture for you.

First, let's install and load the apaTables package

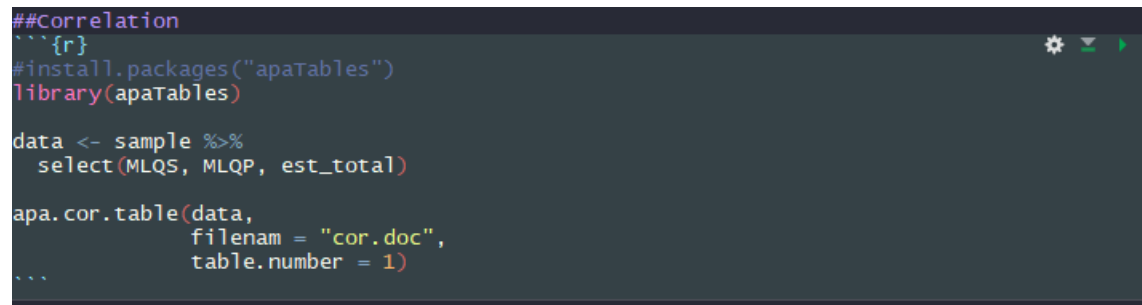
```
install.packages("apaTables")  
library(apaTables)
```

Select variables that you are interested in: Presence of Meaning, Search for Meaning, and Self-esteem

```
data <- sample %>%  
  select(MLQS, MLQP, est_total)
```

Create a correlation matrix

```
apa.cor.table(data,  
  filename = "cor.doc",  
  table.number = 1)
```

A screenshot of the RStudio code editor with a dark theme. The code is as follows:

```
##Correlation  
...{r}  
#install.packages("apaTables")  
library(apaTables)  
  
data <- sample %>%  
  select(MLQS, MLQP, est_total)  
  
apa.cor.table(data,  
  filename = "cor.doc",  
  table.number = 1)  
...
```

- We create an object called “data” including 3 variables from the sample dataset
- `apa.cor.table()` is the function used to generate a correlation matrix and export it as a separate document
The first argument is data, the object we just create, “filename” allows us to name the document and choose the extension for it. Here, we export a file named cor.doc that can be found in the Project folder that you create! The neat thing about this is that the table is already APA formatted.

Output: this can be found in the project folder!

Table 1

Means, standard deviations, and correlations with confidence intervals

| Variable | M | SD | 1 | 2 |
|--------------|------|------|-----------------------|---------------------|
| 1. MLQS | 4.22 | 1.36 | | |
| 2. MLQP | 4.87 | 1.32 | -.18*
[-.33, -.02] | |
| 3. est_total | 2.49 | 0.35 | .07
[-.09, .23] | -.13
[-.29, .03] |

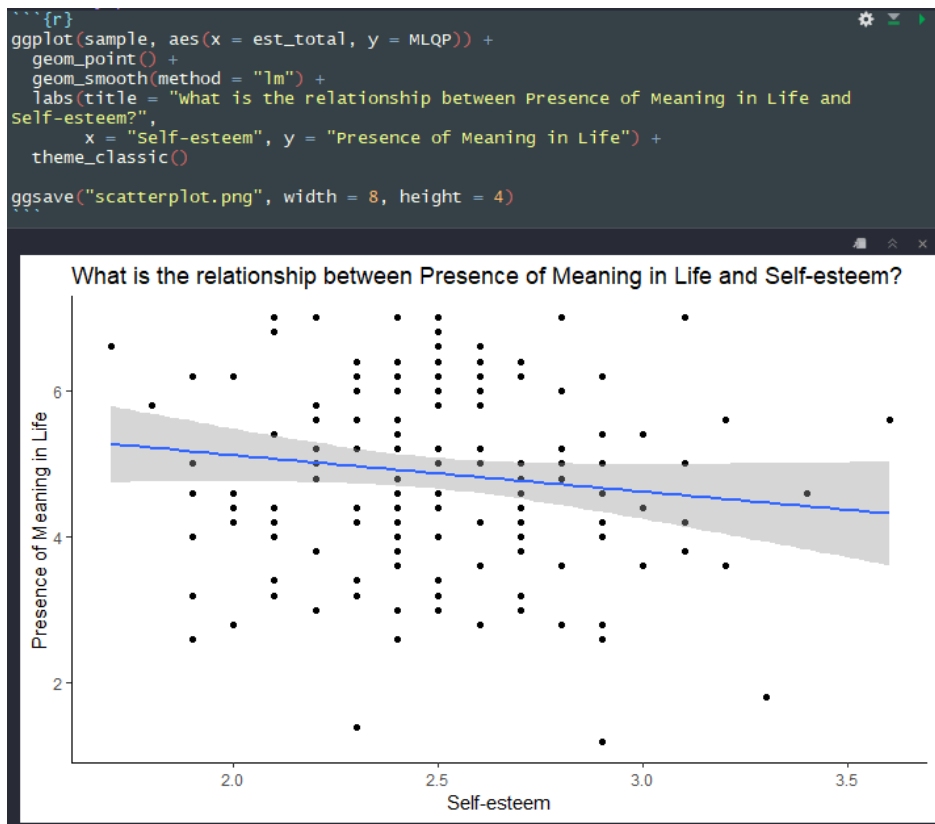
Note. M and SD are used to represent mean and standard deviation, respectively. Values in square brackets indicate the 95% confidence interval. The confidence interval is a plausible range of population correlations that could have caused the sample correlation (Cumming, 2014).
* indicates $p < .05$. ** indicates $p < .01$.

A Simple Correlation Plot:

Though a correlation matrix is helpful, sometimes we might just want to visualize the correlations between two variables. We can do this by adding one more line of code to our normal scatterplot!

```
ggplot(sample, aes(x = est_total, y = MLQP)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "What is the relationship between Presence of Meaning in Life and Self-
esteem?",
        x = "Self-esteem", y = "Presence of Meaning in Life") +
  theme_classic()

ggsave("scatterplot.png", width = 8, height = 4)
```

`geom_smooth()` allows us to adjust the scatterplot
`method = lm` means we want to fit a linear model to the data points!

Independent Samples t-test (AKA a two sample t-test)

Now let's say we want to compare means for two groups of people- people who are in the control condition, and people who were assigned in the self-mortality salience condition. We want to see if people report significantly higher meaning in life as they are reminded of their own mortality. We have 50 people in the control and 50 people in the self-mortality condition. To compare group means difference, we use this code:

```

sample1 <- sample %>%
  filter(cond != "loved ones")

```

```

t.test(data = sample1, MLQP ~ cond, pair = FALSE)

```

- since t-test is used to compare 2 groups, `filter()` allows us to remove 1 condition
`!=` means “not equal to”.
- the `t.test()` function executes the test. We specify `pair = FALSE` because it is an independent samples t-test
- `data` = name of your dataset!
- On the left of the “`~`” is our variable of interest – Presence of Meaning in Life
On the right of the “`~`” is the variable that contains the group assignment - condition

```

{r}
sample1 <- sample %>%
  filter(cond != "loved ones")

t.test(data = sample1, MLQP ~ cond, pair = FALSE)

```

Welch Two Sample t-test

data: MLQP by cond
 t = 1.3589, df = 97.868, p-value = 0.1773
 alternative hypothesis: true difference in means is not equal to 0
 95 percent confidence interval:
 -0.167585 0.895585
 sample estimates:
 mean in group control mean in group self
 5.428 5.064

Dependent Samples t-test (AKA a paired t-test)

Let's think of an experiment in which we measure the Presence of Meaning in Life (MLQP) for people before exposing them to questions that prime them to think of their own mortality, then we measure MLQP again. In total, we only have 50 people, but we have 100 observations collected in 2 time points. We'll conduct a dependent t-test using this code:

t.test(data = sample1, MLQP ~ cond, paired = TRUE)

the t.test function execute the test. We specify paired = TRUE because it is a dependent samples t-test

```

##Dependent sample t-test
{r}
sample1 <- sample %>%
  filter(cond != "loved ones")

t.test(data = sample1, MLQP ~ cond, pair = TRUE)

```

Paired t-test

data: MLQP by cond
 t = 1.2315, df = 49, p-value = 0.224
 alternative hypothesis: true difference in means is not equal to 0
 95 percent confidence interval:
 -0.2299968 0.9579968
 sample estimates:
 mean of the differences
 0.364

Simple Linear Regression

A simple linear regression (SLR) model can tell us a lot about the relationship between two variables. Consider that we want to model the relationship between Self-esteem and Presence of Meaning in Life. We would *expect* that the two variables be positively related (people with higher self-esteem are likely to report higher meaning in life). A SLR can tell us *how* they are related (i.e., there is a positive linear relationship, a negative linear relationship, etc.)

First, let's install the package "olsrr"

```
install.packages("olsrr")  
library(olsrr)
```

Run a linear regression

```
mod1 <- lm(MLQP ~ est_total, data = data)
```

```
ols_regress(mod1)
```

- data = name of your dataset. In this case, we use the dataset named "data" that we create above!
- lm() allows us to conduct a linear regression
 - $lm(y \sim x_1 + x_2 + x_3 + \dots, data = \text{name of your dataset})$
 - On the left of "~" is the y-variable – the outcome/dependent variable
 - On the right of "~" is the x-variable – the predictor/independent variable. You can include as many x-variables as you want!

```
##Regression  
```{r}  
#install.packages("olsrr")
library(olsrr)

mod1 <- lm(MLQP ~ est_total, data = data)

ols_regress(mod1)
```

Model Summary					
R	0.133	RMSE	1.312		
R-Squared	0.018	Coef. Var	26.910		
Adj. R-Squared	0.011	MSE	1.721		
Pred R-Squared	-0.009	MAE	1.093		

RMSE: Root Mean Square Error  
MSE: Mean Square Error  
MAE: Mean Absolute Error

ANOVA					
	Sum of Squares	DF	Mean Square	F	Sig.
Regression	4.564	1	4.564	2.652	0.1055
Residual	254.679	148	1.721		
Total	259.244	149			

Parameter Estimates							
model	Beta	Std. Error	Std. Beta	t	Sig.	lower	upper
(Intercept)	6.120	0.772		7.928	0.000	4.594	7.645
est_total	-0.500	0.307	-0.133	-1.629	0.106	-1.107	0.107

ANOVAS

Since R recognizes the cond column as a character string, there are some data wrangling we need to do before doing the analysis

```
sample <- sample %>%
```

```
 mutate(self = ifelse(cond == "self", 1, 0),
```

```
 control = ifelse(cond == "control", 1, 0),
```

```
 cond.f = factor(cond, levels = c("control", "self", "loved ones")))
```

- “mutate()” allows us to create new variables: self, control, and cond.f
- self and control are two dummy coded variables demonstrating the group assignment
  - people in the control group will have self = 0 and control = 1
  - people in the self-mortality-salience group will have self = 1 and control = 0
  - people in the loved-ones-mortality-salience group will have self = 0 & control = 0
- ifelse() is a logistic function that takes 1 argument
  - E.g.: R will run a logistic test on the “cond” column: If cond equals “self is TRUE, the variable “self” (our new variable) will be assigned with a 1, otherwise “self” will be 0.
- cond.f is the factored version of the condition variable

Visualizing the data helps us to identify group differences:

Box plots

Plot MLQP by group and color by group

```
#Boxplot
```

```
ggplot(sample, aes(x = cond.f, y = MLQP, group = cond.f, color = cond.f)) +
```

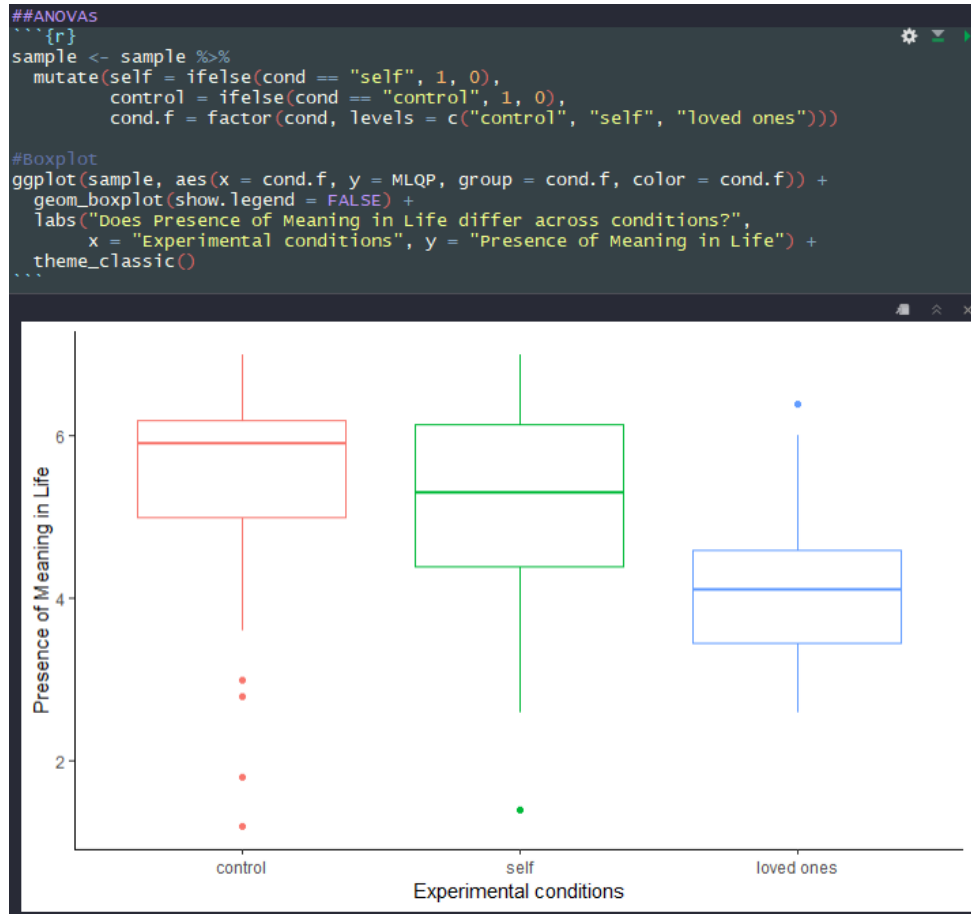
```
 geom_boxplot(show.legend = FALSE) +
```

```
 labs("Does Presence of Meaning in Life differ across conditions?",
```

```
 x = "Experimental conditions", y = "Presence of Meaning in Life") +
```

```
 theme_classic()
```

- ggplot() is the function that makes our plot
- sample is the name of our dataset
- x is our group assignment – cond.f (remember to use the factor)
- y is the variable of interest – Presence of Meaning in Life
- labs() allows us to rename the axes
- theme\_classic() gets rid of the unnecessary grid lines



Compute the analysis of variance:

```
mod2 <- aov(MLQP ~ cond.f, data = sample)
```

Summary of the analysis:

```
summary(mod2)
```

```
TukeyHSD(mod2)
```

```

```{r}
#Analysis of variance
mod2 <- aov(MLQP ~ cond.f, data = sample)

summary(mod2)

TukeyHSD(mod2)
```

```

|           | Df  | Sum Sq | Mean Sq | F value | Pr(>F)       |
|-----------|-----|--------|---------|---------|--------------|
| cond.f    | 2   | 44.68  | 22.34   | 15.3    | 9.16e-07 *** |
| Residuals | 147 | 214.56 | 1.46    |         |              |

---  
 Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
 Tukey multiple comparisons of means  
 95% family-wise confidence level

Fit: aov(formula = MLQP ~ cond.f, data = sample)

```

$`cond.f`
 diff lwr upr p adj
self-control -0.364 -0.9361054 0.2081054 0.2908587
loved ones-control -1.296 -1.8681054 -0.7238946 0.0000009
loved ones-self -0.932 -1.5041054 -0.3598946 0.0004998

```