Wherever you see **red characters**, these need to be replaced by your information. This includes the < > symbols!

Plotting a One-Variable Bar Plot with Counts

```
ggplot(data = <NAME OF DATASET>,
       mapping = aes(x = <NAME OF VARIABLE>)) +
  geom_bar() +
  labs(x = "<TITLE FOR THE X-AXIS>")
```

Note: This bar plot has the variable names on the x-axis. If the names are squished, then you should use **`y =`** instead of **`x =`** .

Plotting a One-Variable Bar Plot with Proportions

```
ggplot(data = <NAME OF DATASET>,
       mapping = aes(x = <NAME OF VARIABLE>, y = ..prop.., group = 1)) +
  geom_bar() +
  labs(x = "<TITLE FOR THE X-AXIS>")
```

Note: This bar plot has the variable names on the x-axis. If the names are squished, then you should use **`y =`** instead of **`x =`** .

Plotting a Two-Variable Bar Plot

```
ggplot(data = <NAME OF DATASET>,
       mapping = aes(x = <NAME OF VARIABLE 1>,
                     fill = <NAME OF VARIABLE 2>)) +
  geom_bar(position = "stack") +
  labs(x = "<TITLE FOR THE X-AXIS>")
```

Note: You should fill by whichever variable has **fewer** values.
Note: If you want a side-by-side bar plot you need to change `position` to "dodge". If you want a filled bar plot, you need change `position` to "fill".

Creating a Summary Table of Observations of One Variable

```
<NAME OF DATASET> |>
   count(<NAME OF VARIABLE>)
```

Wherever you see **red characters**, these need to be replaced by your information. This includes the < > symbols!

Creating a Summary Table of Observations from Two Variables

```
<NAME OF DATASET> |>
   count(<NAME OF VARIABLE 1>, <NAME OF VARIABLE 2>)
```

Creating a Contingency Table of Observations from Two Variables

```
<NAME OF DATASET> |>
   count(<NAME OF VARIABLE 1>, <NAME OF VARIABLE 2>)|>
   pivot_wider(names_from = <NAME OF VARIABLE FOR COLUMNS>,
               values_from = n) |>
   adorn_totals(where = c("row", "col"))
```

Note: Your explanatory variable should be in the rows and your response variable should be in the columns. So, the variable you insert into `names_from` should be the response variable you are interested in.

Performing a Chi-Squared Goodness-of-Fit Test (One Variable)

```
chisq_test(x = <NAME OF DATASET>,
           response = <NAME OF VARIABLE>)
```

Performing a Chi-Squared Independence / Homogeneity Test (Two Variables)

```
chisq_test(x = <NAME OF DATASET>,
           response = <NAME OF RESPONSE VARIABLE>,
           explanatory = <NAME OF EXPLANATORY VARIABLE>)
```

Obtaining the Sample X-Squared Statistic

```
obs_xsq <- <NAME OF DATASET> |>
  specify(response = <NAME OF RESPONSE VARIABLE>,
          explanatory = <NAME OF EXPLANATORY VARIABLE>) |>
  calculate(stat = "Chisq")
```

Note: *This step **must** be done **before** you find your p-value!*

Obtaining 1000 Permuted X-Squared Statistics – Assuming the Null Hypothesis is True

```
null_dist <- <NAME OF DATASET> |>
  specify(response = <NAME OF RESPONSE VARIABLE>,
          explanatory = <NAME OF EXPLANATORY VARIABLE>) |>
  generate(reps = 1000, type = "permute") |>
  calculate(stat = "Chisq")
```

Wherever you see **red characters**, these need to be replaced by your information. This includes the < > symbols!

Plotting the Simulated Null Distribution

```
visualize(data = null_dist,
          method = "simulation")
```

Note: *This step **must** come after you have obtained the permuted differences in means!*

Obtaining a p-value from a Null Distribution

```
get_pvalue(x = null_dist,
           obs_stat = obs_xsq,
           direction = "greater")
```

Note: *This step **must** come after you have obtained the bootstrapped differences in means **and** the observed difference in means!*
Note: In a Chi-Squared test we **always** use a greater than alternative, since we only look in the right tail!