R Data Visualization Primmer

Load ggplot2

You will want to use the *ggplot2* package for all of your data visualization. Make sure to load it before you try to generate any graphs:

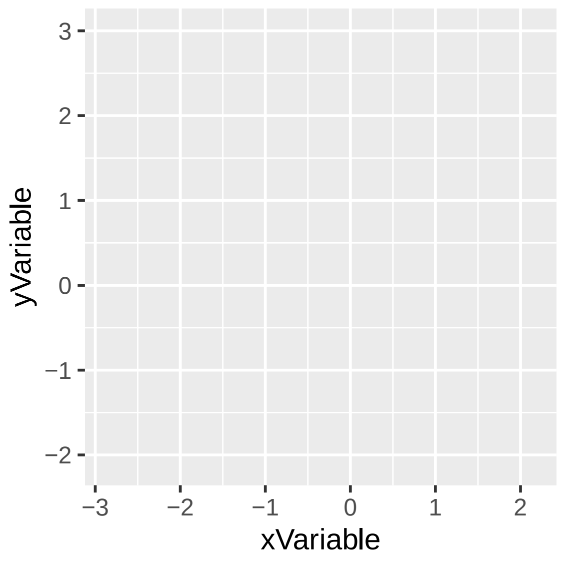
```
#load ggplot2
library(ggplot2)
```

Generating a plot

ggplot2 works by generating a plot base and adding elements to that plot one option at a time.

To generate this plot base, use the *ggplot()* function:

```
plotName <- ggplot(data = dataFrame,
                   aes(x = xVariable,
                       y = yVariable))
#plotName = a data structure for storing your plot
#dataFrame = the dataframe containing your data
#xVariable = your x-axis variable
#yVariable = your yaxis variable

#To view your plot, execute:
plotName
```



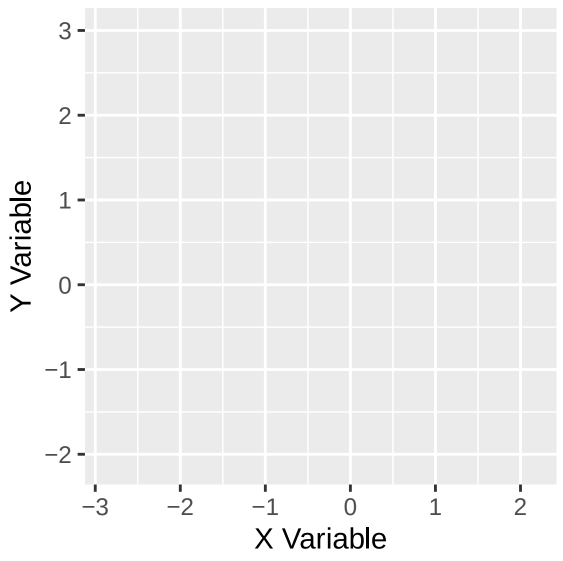
The *aes()* option inside the *ggplot2()* function sets the aesthetic options for your plot. You can use this to set, among other things:

- Which variables you plot and on which axis

- The colors and fills of your graph
- The size, transparency, and shape of graph elements

To add on to your base plot, you add additional functions using the "+". For example, to label your axes, use the *labs()* function:

```
plotName <- plotName +
  labs(x = "X Variable", y = "Y Variable")
#labs() = labels elements of your graph
#You can use this to label the axes, legend, graph title, and more
#Run ?labs to see all of your options

#To view your plot, execute:
plotName
```

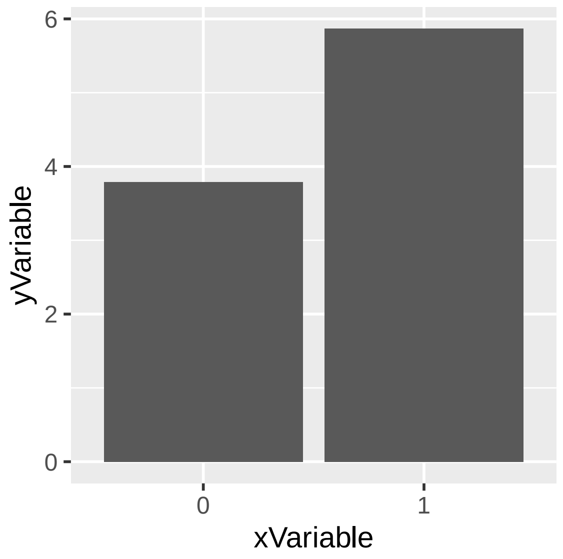


Bar Plots

To make a simple bar plot of group means across a categorical predictor, do:

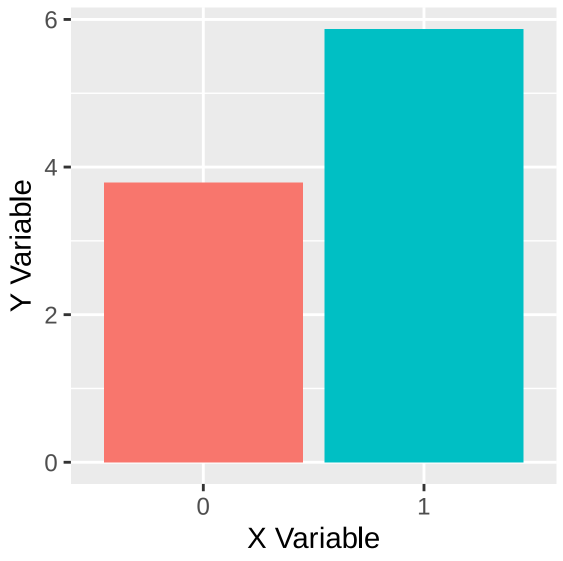
```
plotName <- ggplot(data = dataFrame,
                   aes(x = xVariable,
                       y = yVariable)) +
  stat_summary(fun = "mean",
               geom = "bar",
               position = "dodge")

#stat_summary = computes and plots a summary statistic based on your data
#fun = the statistic you want to compute
#geom = the shape you want to plot
#'position = "dodge"' = place the bars side by size, rather than stack
```

```
#To view your plot, execute:
plotName
```



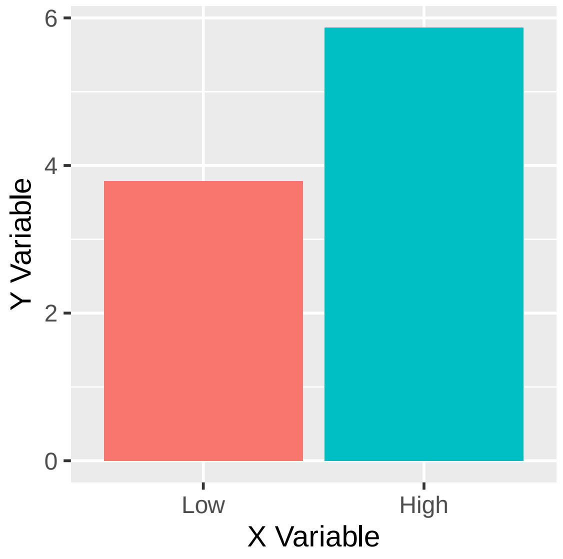
You can label using *labs()* and add color using the *fill* = option inside *aes()*:

```
plotName <- ggplot(data = dataFrame,
                   aes(x = xVariable,
                       y = yVariable,
                       fill = xVariable)) +
  stat_summary(fun = "mean",
               geom = "bar",
               position = "dodge") +
  labs(x = "X Variable",
       y = "Y Variable") +
  theme(legend.position = "none")

#fill = fills in geoms according to the variable specified
#theme(legend.position = "none") = hides the legend (as it is redundant in this case)

#To view your plot, execute:
plotName
```



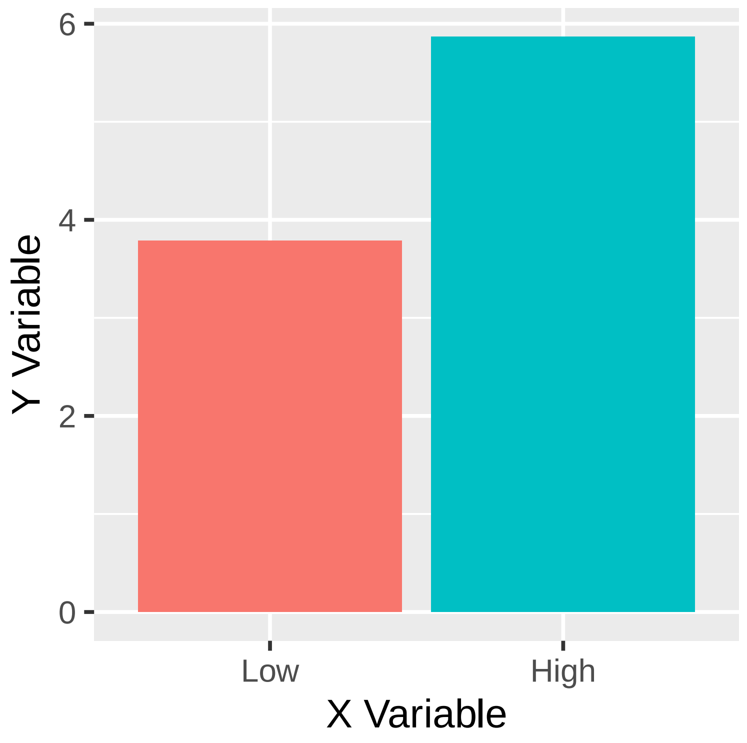
You can also relabel the x-axis ticks using *scale_x_discrete()*:

```
plotName <- plotName +
  scale_x_discrete(labels = c("Low", "High"))

#scale_x_discrete = lets you set options for the x-axis, including the axis ticks
#do ?scale_x_discrete to see all options
#Use scale_y_discrete for a categorical y-axis
#Use scale_x_continuous or scale_y_continuous for continuous variables

#To view your plot, execute:
plotName
```



You can set the text size for the whole graph using *base_size* = inside *theme_gray()*:

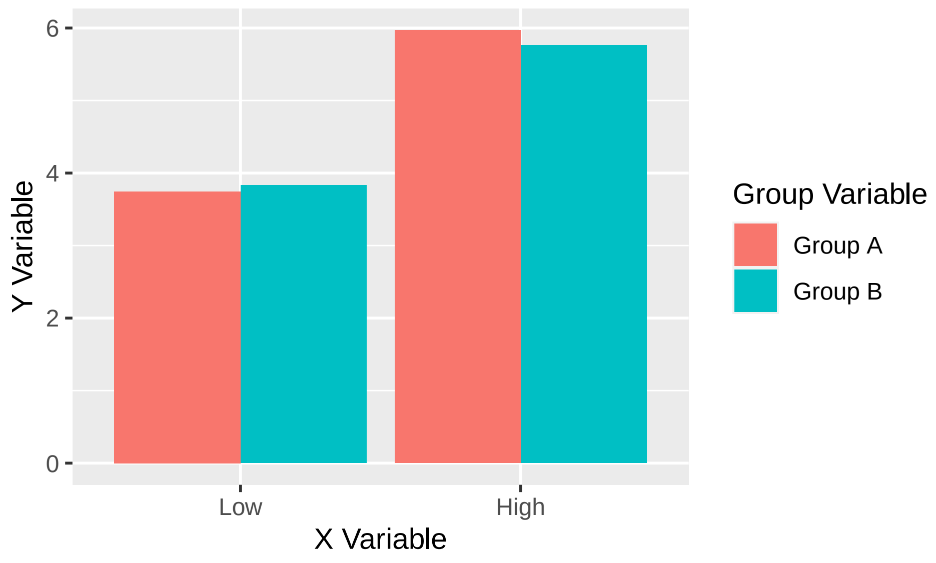
```r
plotName <- plotName +
  theme_gray(base_size = 15)+
  theme(legend.position="none")

#scale_x_discrete = lets you set options for the x-axis, including the axis ticks
#do ?scale_x_discrete to see all options
#Use scale_y_discrete for a categorical y-axis
#Use scale_x_continuous or scale_y_continuous for continuous variables

#theme() always has to follow theme_gray()

#To view your plot, execute:
plotName
```



To add a grouping variable, use it as the *fill()* variable:

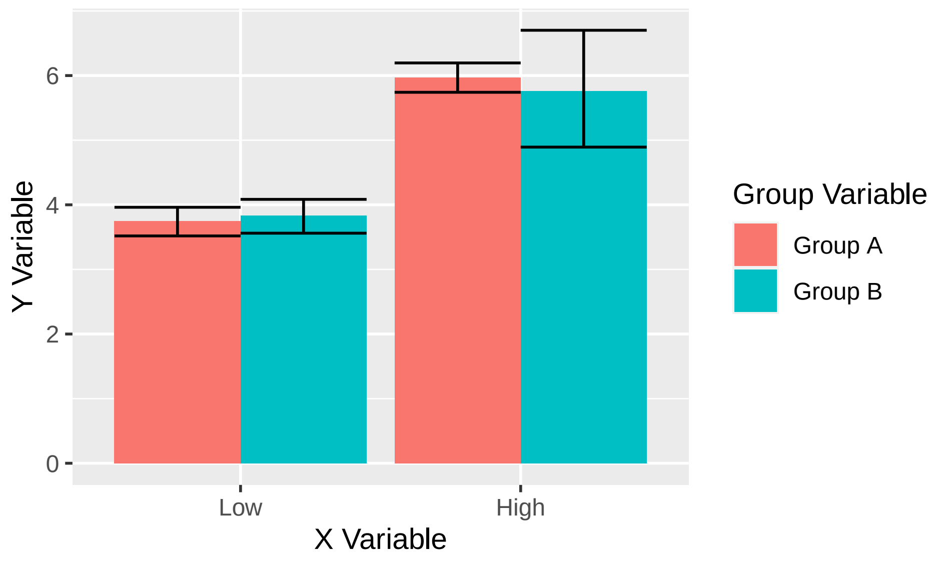
```r
plotName <- ggplot(data = dataFrame,
                   aes(x = xVariable,
                       y = yVariable,
                       fill = groupVariable)) +
  stat_summary(fun = "mean",
               geom = "bar",
               position = "dodge") +
  labs(x = "X Variable",
       y = "Y Variable",
       fill = "Group Variable") +
  scale_x_discrete(labels = c("Low", "High")) +
  scale_fill_discrete(labels = c("Group A", "Group B"))
```

```
#scale_fill_discrete() = sets options for fill and legend

#To view your plot, execute:
plotName
```



Finally, to add error bars representing 95% confidence intervals, you can use the *"mean_cl_boot"* option in *stat_summary*:

```
plotName <- plotName +
  stat_summary(fun.data = "mean_cl_boot",
               geom = "errorbar",
               position = "dodge")

#To view your plot, execute:
plotName
```

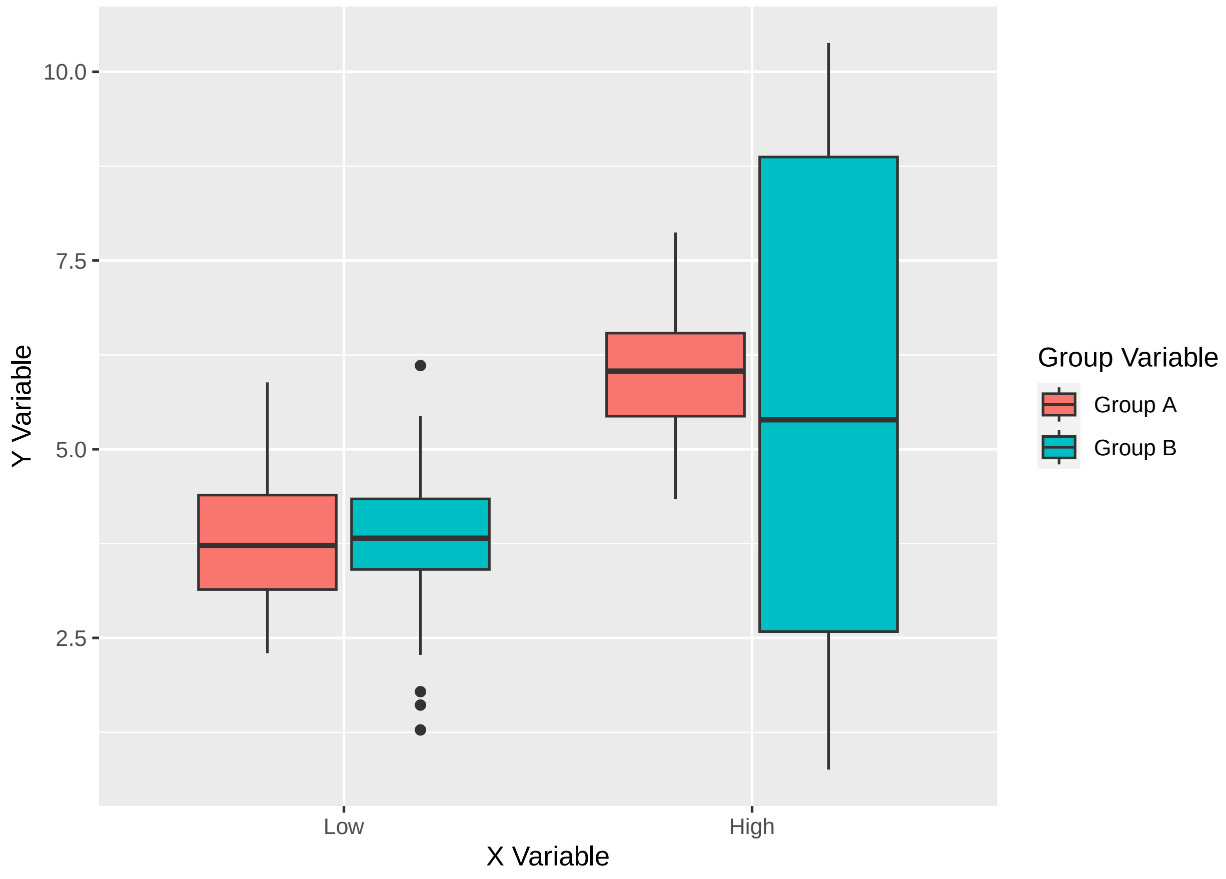


Box and Violin Plots

Bar plots are popular. But unless you are working with count data, bar plots are often not very good. Better are plots that actually show the distribution of the data.

One example is the boxplot:

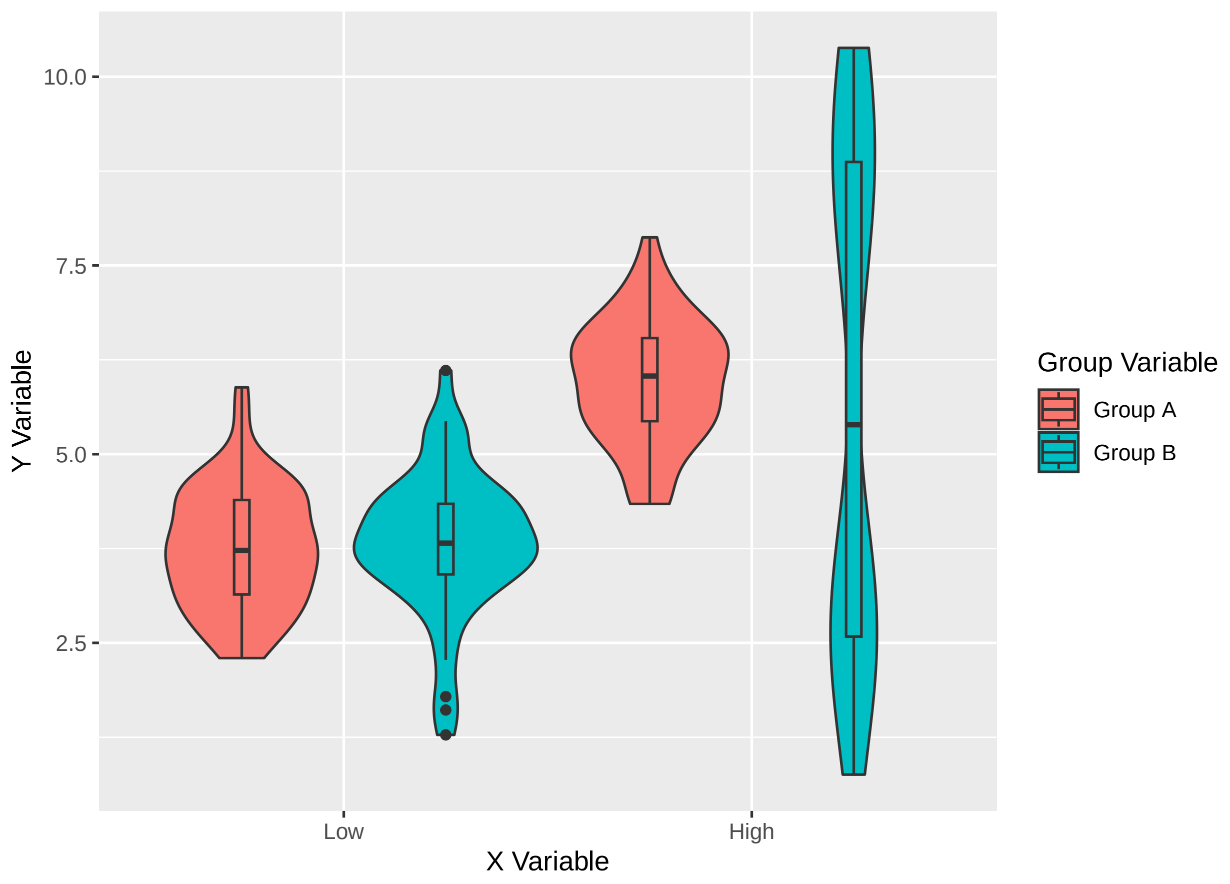
```r
plotName <- ggplot(data = dataFrame,
                   aes(x = xVariable,
                       y = yVariable,
                       fill = groupVariable)) +
  geom_boxplot() +
  labs(x = "X Variable",
       y = "Y Variable",
       fill = "Group Variable") +
  scale_x_discrete(labels = c("Low", "High")) +
  scale_fill_discrete(labels = c("Group A", "Group B"))

#geom_boxplot() = plots data as a boxplot

#To view your plot, execute:
plotName
```



```
#Thick line = median
#Box = IQR (bottom 25th percentile to top 75th percentile)
#Whiskers = 1.5 * IQR
#Dots = outliers
```

Another is the violin plot:

```
plotName <- ggplot(data = dataFrame,
                   aes(x = xVariable,
                       y = yVariable,
                       fill = groupVariable)) +
  geom_violin(position = position_dodge(1)) +
  geom_boxplot(position = position_dodge(1),
               width = .075) +
  labs(x = "X Variable",
       y = "Y Variable",
       fill = "Group Variable") +
  scale_x_discrete(labels = c("Low", "High")) +
  scale_fill_discrete(labels = c("Group A", "Group B"))

#geom_violin() = plots data as a violin plot
#position_dodge() = sets the horizontal positions of the geoms
#width = sets the width of the geom

#To view your plot, execute:
plotName
```



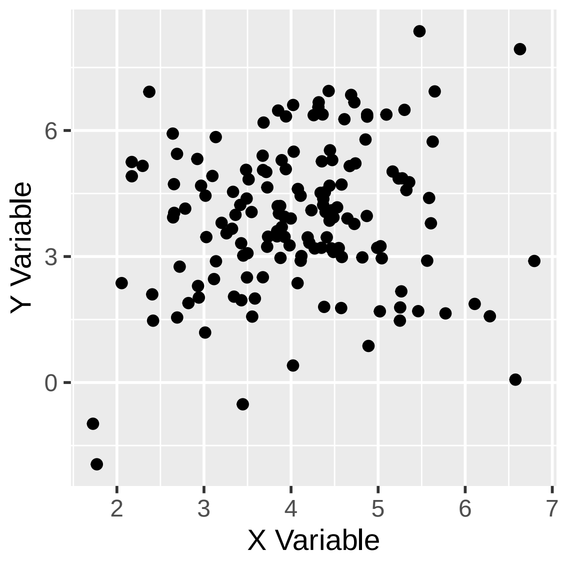
```
#Violin plot = sideways density distribution
```

Note: both of these graphs make clear in this case that the distribution of the 4th group is different from the others. This is obscured in the bar plot. It's particularly clear from the violin plot that the 4th group is bimodal.

Scatter Plots

To make a scatter plot showing the relationship between two continuous variables, do:

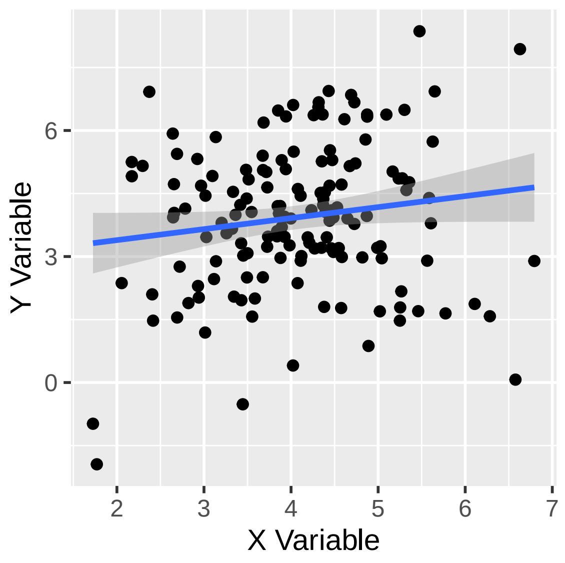
```
plotName <- ggplot(data = dataFrame,
                   aes(x = xVariable,
                       y = yVariable)) +
  geom_point()+
  labs(x = "X Variable",
       y = "Y Variable")

#geom_point() = plot data as points

#To view your plot, execute:
plotName
```



To add a trend line, use the *geom_smooth()* function:

```
plotName <- ggplot(data = dataFrame,
                   aes(x = xVariable,
                       y = yVariable)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(x = "X Variable",
       y = "Y Variable")

#geom_smooth() = add a "smoothed" trendline
#'method = "lm"' = add a regression line

#To view your plot, execute:
plotName
```

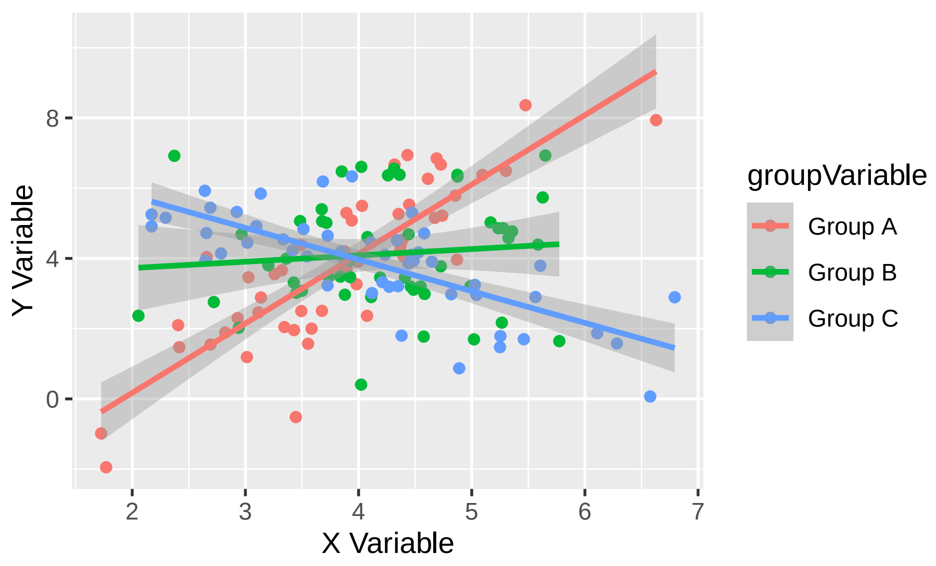
```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```



To add a grouping variable, use the *color =* option inside *aes()*:

```
plotName <- ggplot(data = dataFrame,
                   aes(x = xVariable,
                       y = yVariable,
                       color = groupVariable)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(x = "X Variable",
       y = "Y Variable",
       fill = "Group Variable") +
  scale_color_discrete(labels = c("Group A", "Group B", "Group C"))

#geom_smooth() = add a "smoothed" trendline
#'method = "lm"' = add a regression line

#To view your plot, execute:
plotName
```

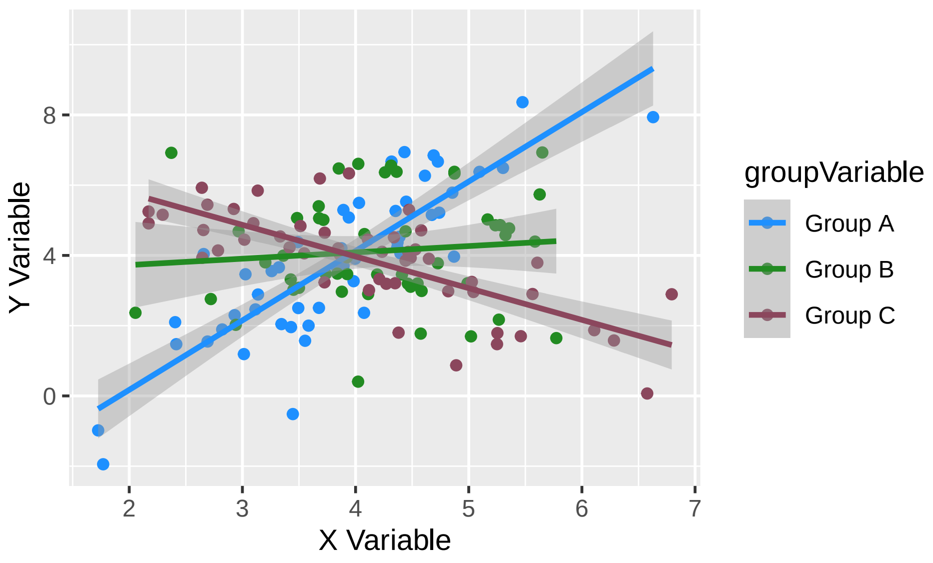
```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```



You can choose the color palette for the grouping:

```
plotName <- ggplot(data = dataFrame,
                   aes(x = xVariable,
                       y = yVariable,
                       color = groupVariable)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(x = "X Variable",
       y = "Y Variable",
       fill = "Group Variable") +
  scale_color_manual(
    values = c("dodgerblue", "forestgreen", "palevioletred4"),
    labels = c("Group A", "Group B", "Group C")
  )

#geom_smooth() = add a "smoothed" trendline
#'method = "lm"' = add a regression line

#To view your plot, execute:
plotName
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



8
4
0
Y Variable
2
3
4
5
6
7
X Variable
groupVariable
Group A
Group B
Group C

Data Visualization Cheat Sheet

Plotting option	Code

Base plot	plotName <- ggplot(data = dataFrame,
	aes(x = xVariable,
	y = yVariable,
	color = groupVariable))

Bar plot	+ stat_summary(fun="mean",
	geom="bar",
	position="dodge")

95% CI Error Bars	+ stat_summary(fun.data="mean_cl_boot",
	geom="errorbar",
	position="dodge")

Box plot	+ geom_boxplot(position = position_dodge(1)) |
Violin plot	+ geom_violin(position = position_dodge(1)) |
Scatter plot	+ geom_point()
Add trend line	+ geom_smooth(method="lm")

Add labels	+ labs(x="X Label",
	y = "Y Label",
	color = "Color Label",
	fill = "Fill Label")

Change axis labels	+ scale_x_discrete(labels = c("Low", "High"))
Change legend labels	+ scale_color_discrete(labels = c("Low", "High"))
Change legend colors	+ scale_color_manual(values = c("red", "black"))
Change font size	+ theme_gray(base_size=20)
Hide a legend	+ theme(legend.position="none")