Basic Graphing w/ GGPlot2

—

An Introduction to Visualizations

Choosing a Visualization

Consider Your Variables:

Are you plotting one, two, or even three variables?

Are your variables discrete or continuous?

Continuous: Can your variable take any value within a range of possible values? Ex. Age, Height, Distance

Discrete: Does your variable take one of a few specified values? Is your variable organized into groups? Ex. Color, Species, Type

*Note: Discrete data can also be expressed in numeric terms (ie. Species 1 through 4)

Example Plots

The following plots were created using the data from the mpg data set found in the GGPlot library (using the GGPlot library will automatically let you use the mpg data set). The mpg data set contains several variables relating to cars. Use *head(mpg)* to learn more about these variables.

To follow along, first create a GGPlot object for the variables you are interested in:

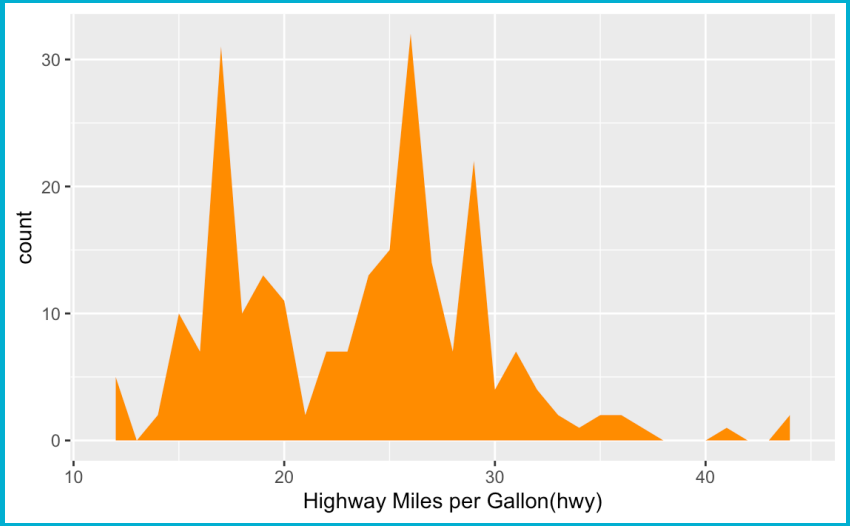
```
library(ggplot2)

a <- ggplot(mpg, aes(hwy))  #hwy is the name of a column in the mpg data set

a2 <- ggplot(mpg)
```

These functions create 2 GGPlot objects we will use to create visualizations

Plotting w/ One Continuous Variable



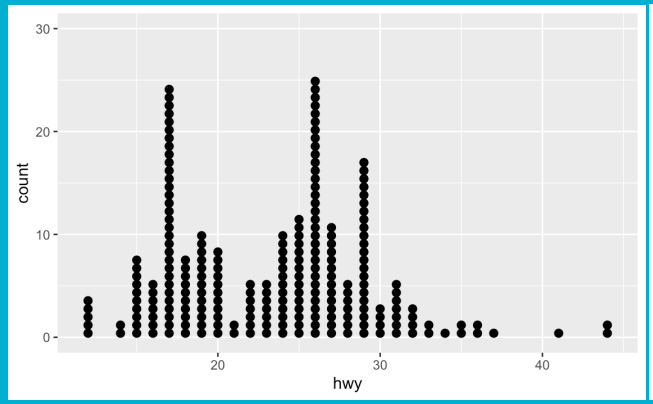
Area Plots: Similar to dot plots; histograms; and frequency plots, area plots show the amount of times a value is present in a dataset. Any time you want to measure one continuous variable, these plots are a good option.

Graph Code:

a + geom_area(stat="bin", binwidth =1, fill = "Dark Orange") + xlab("Highway Miles per Gallon(hwy)")

- *binwidth = 1:* is a parameter used to set the ranges over which your variable is measured
- *fill = "Dark Orange":* is an aesthetic parameter which lets you select a color for your graph
- *Xlab():* is a function that sets the label of the X-axis

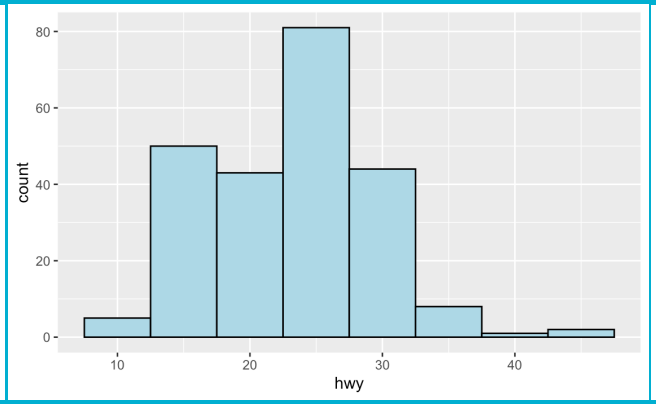
Plotting w/ One Continuous Variable



a + geom_dotplot(binwidth = 1/2) + ylim(c(0,30))

- *Ylim(c(0,30)):* manually sets the y-axis to go from 0 to 30. Without *ylim()* your count will be a scaled value between 0 & 1

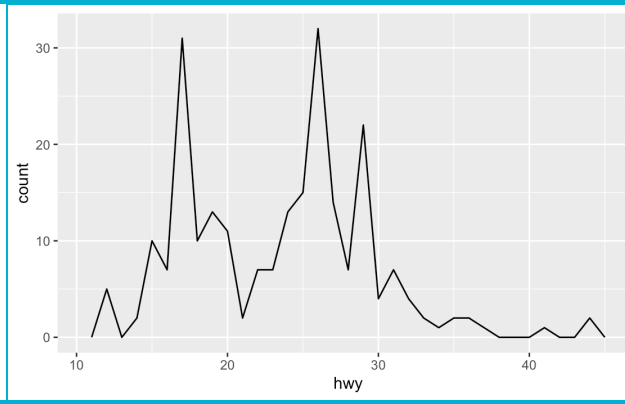
- *binwidth =1/2:* here is used to set the size of the dots



a + geom_histogram(binwidth = 5, color = "Black", fill = "Light Blue")

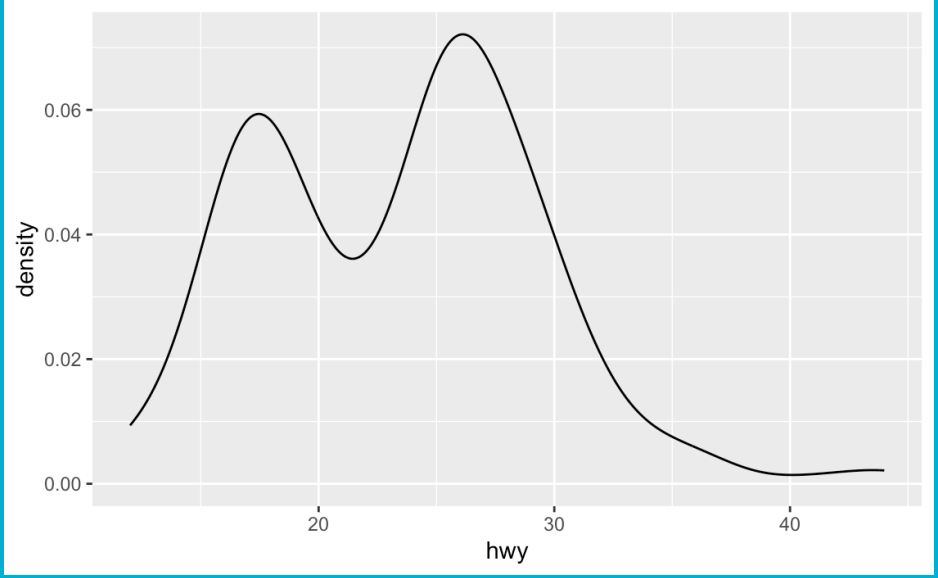
- *color = "Black":* is a parameter that sets an outline color

- *binwidth = 5:* here is used to specify the range for the bars in your histogram



a + geom_freqpoly(binwidth = 1)

Plotting w/ One Continuous Variable

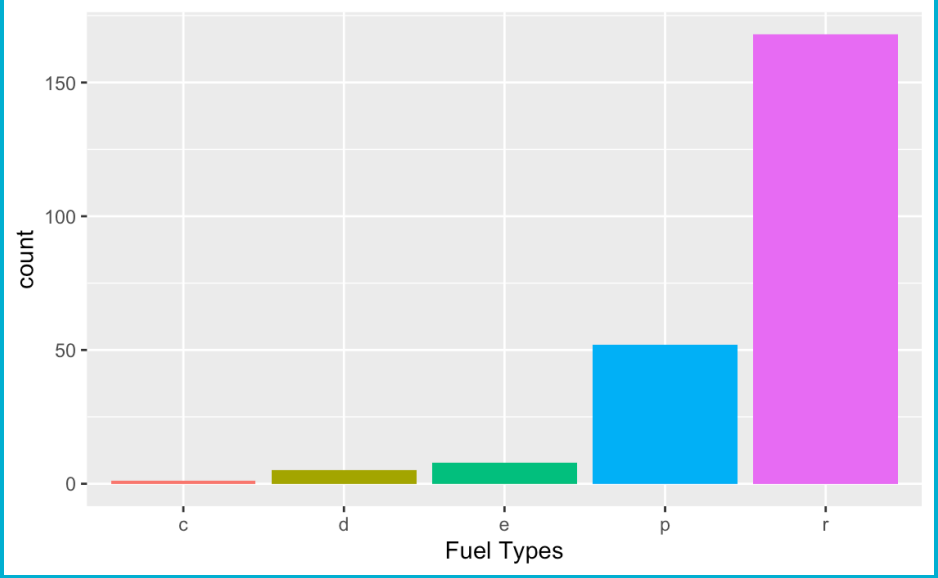


Density Plots: Density plots are similar to frequency plots; however, instead of measuring counts they measure frequency as a percentage of the total data. (Ie. About 4% of the cars measured would get 30 miles per gallon on the highway).

Graph Code:

a + geom_density()

Plotting w/ One Discrete Variable



Box Plots: Measure the occurrences of each factor within a discrete variable. A useful graph for anytime you want to visualize a discrete variable.

Graph Code:

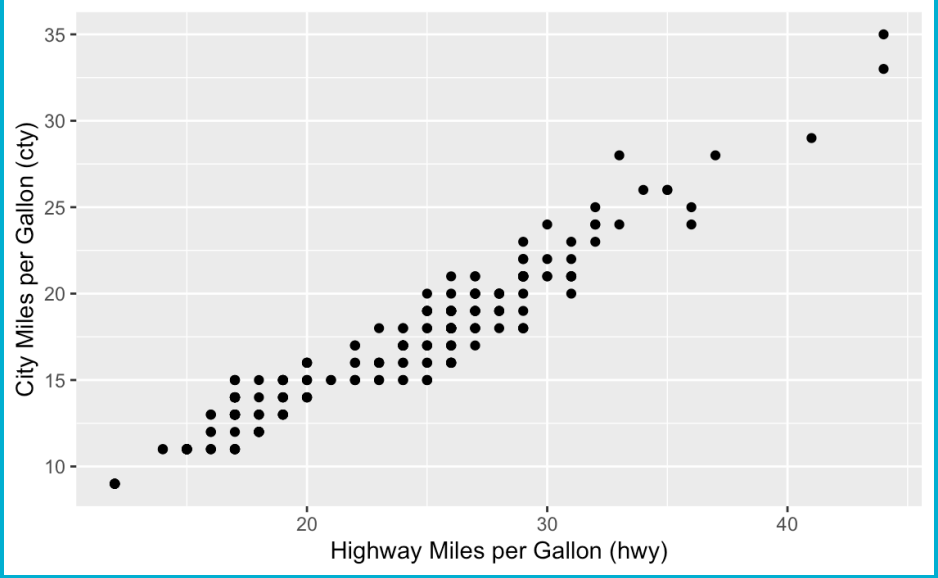
b <- ggplot(mpg, aes(fl)) #Creates a GGPlot object

b + geom_bar(aes(fill=fl), show.legend = FALSE) + xlab("Fuel Types")

- *aes(fill = fl):* Maps different colors to different factors of Fuel Types (in the data set, the fuel types column is called fl)

- *Show.legend = FALSE:* Since *aes(fill)* will automatically create a legend that, for our purposes, is redundant so we manually turn it off

Plotting w/ Two Continuous Variables



Continuous Point Plots are a simple way to display the relationship between two continuous variables.

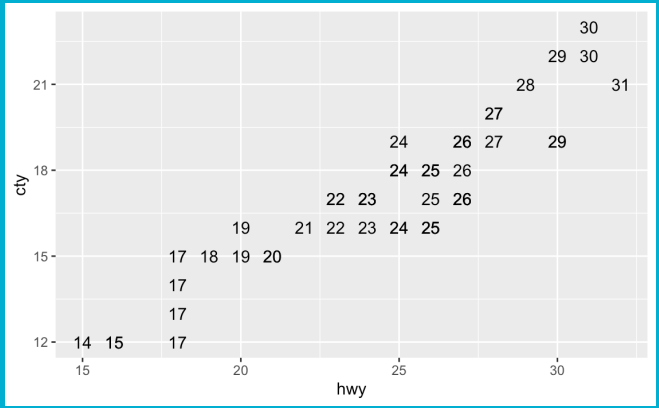
Graph Code:

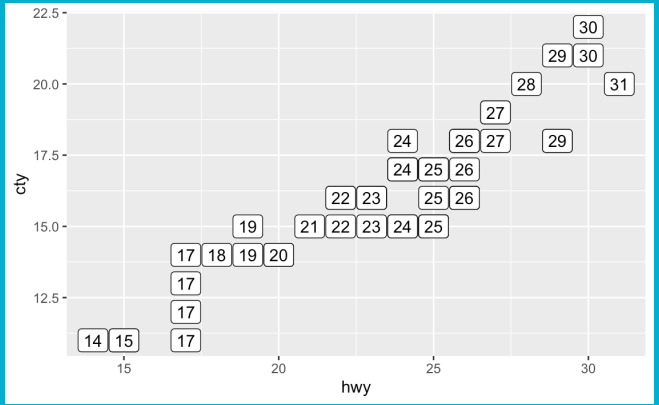
c <- ggplot(mpg, aes(hwy, cty))

c + geom_point() + xlab("Highway Miles per Gallon (hwy)") + ylab("City Miles per Gallon (cty)")

- In this example, *aes()* is used to map the two variables to the x and y axis, respectively

Plotting w/ Two Continuous Variables





Geom_text and Geom_label are two types of labelled continuous plots. They function very similarly to a continuous point plot, only labeled.

Graph Code:

#Creating a subset of the mpg data for cleaner graphs:
c_short <- ggplot(mpg_short, aes(hwy, cty))

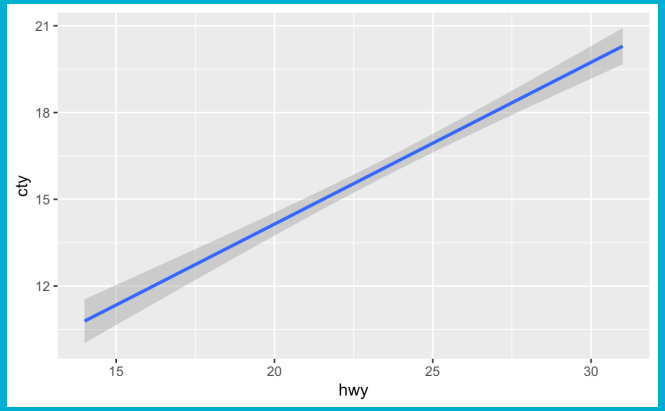
cs + geom_label(aes(label = hwy))

cs + geom_text(aes(label = hwy), nudge_x = 1, nudge_y = 1)

- *aes(label = hwy):* manually sets the labels to display the highway mpg values, but either variable can be specified

- *nudge_x & _y:* can be used to offset the labels from each other so there is no overlap of the labels

Plotting w/ Two Continuous Variables



Graph Code:

#Creating a subset of the mpg data for cleaner graphs:
c_short <- ggplot(mpg_short, aes(hwy, cty))

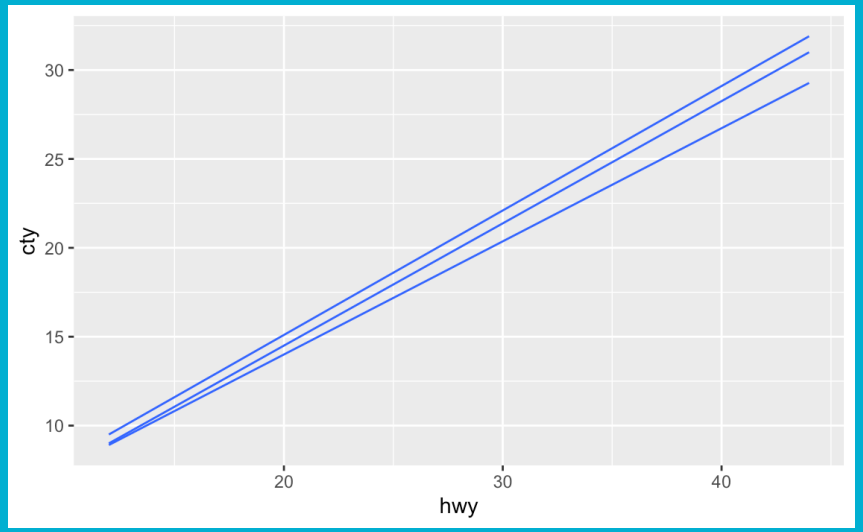
c_short + geom_label(aes(label = hwy))

c_short + geom_text(aes(label = hwy), nudge_x = 1, nudge_y = 1)

- *aes(label = hwy):* manually sets the labels to display the highway mpg values, but either variable can be specified

- *nudge_x & nudge_y:* can be used to offset the labels from each other so there is no overlap of the labels

Plotting w/ Two Continuous Variables



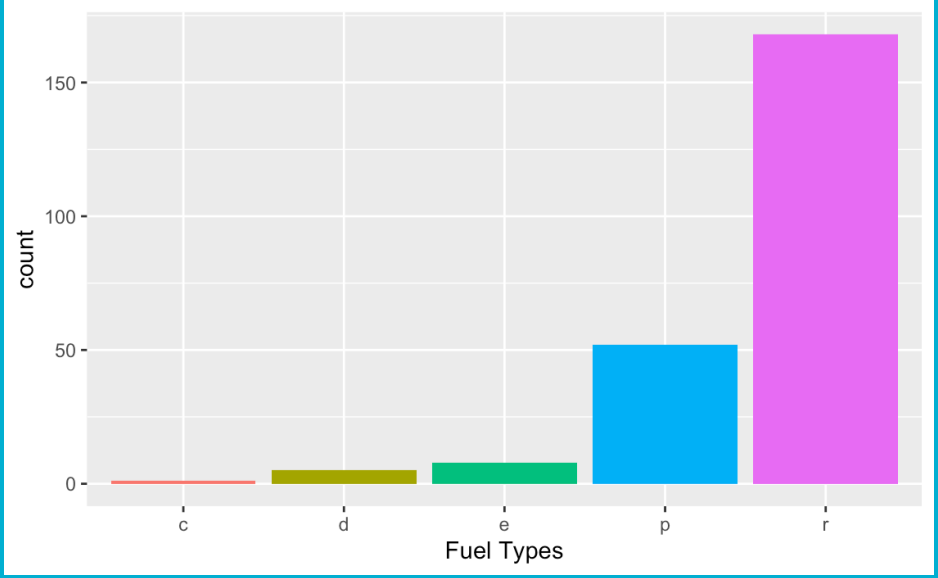
Quantile Plots overlay three trend lines correlating to different percentiles. These plots should be used if there is a reasonable assumption that the trend will change based on the severity of one variable.

Graph Code:

c + geom_quantile()

- *quantiles(c(Q1, Q2, Q3))*: can be added within the geom_quantile() function to manually set desired percentiles. By default the values are set to .25, .50, and .75, respectively

Plotting w/ Discrete Variable



Box Plots: Measure the occurrences of each factor within a discrete variable. A useful graph for anytime you want to visualize a discrete variable.

Graph Code:

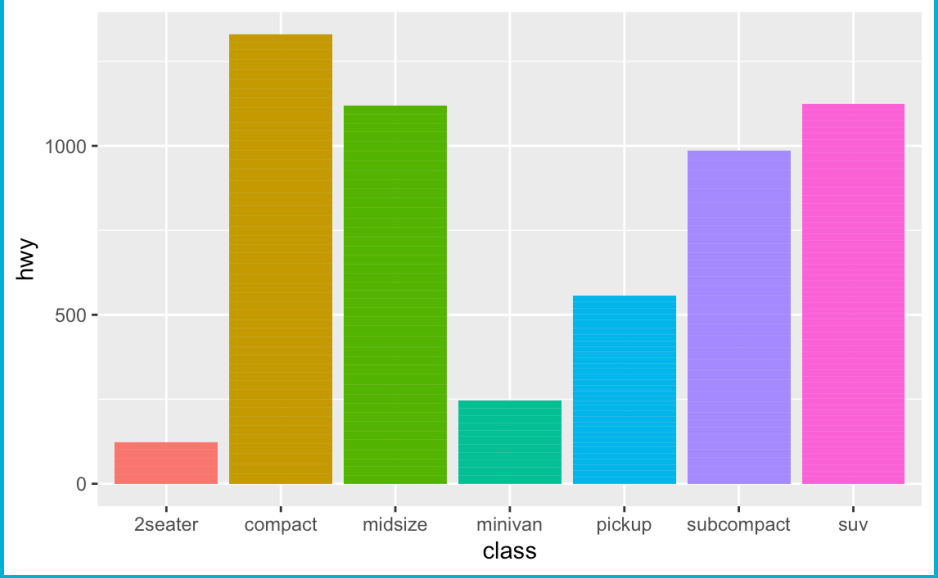
b <- ggplot(mpg, aes(fl)) #Creates a GGPlot object

b + geom_bar(aes(fill=fl), show.legend = FALSE) + xlab("Fuel Types")

- *aes(fill = fl):* Maps different colors to different factors of Fuel Types (in the data set, the fuel types column is called fl)

- s*how.legend = FALSE*: Since aes(fill) will automatically create a legend that, for our purposes, is redundant so we manually turn it off

Plotting w/ a Continuous and Discrete Variable



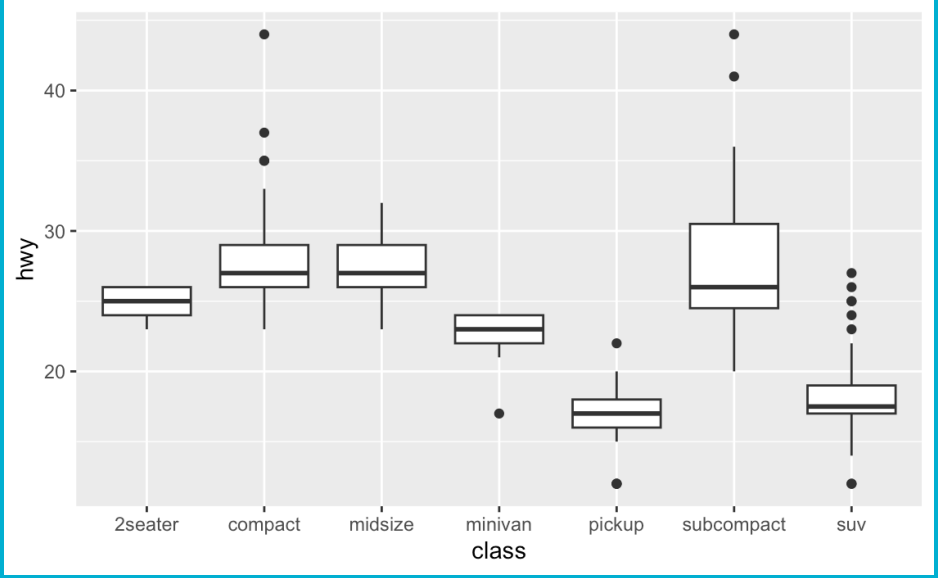
The Column Plot displays the respective counts of a continuous variable by each factor of a discrete variable. These graphs are useful for visualizing c.

Graph Code:

```
f <- ggplot(mpg, aes(class, hwy))

f + geom_col(aes(fill = class), show.legend = FALSE)
```

Plotting w/ a Continuous and Discrete Variable

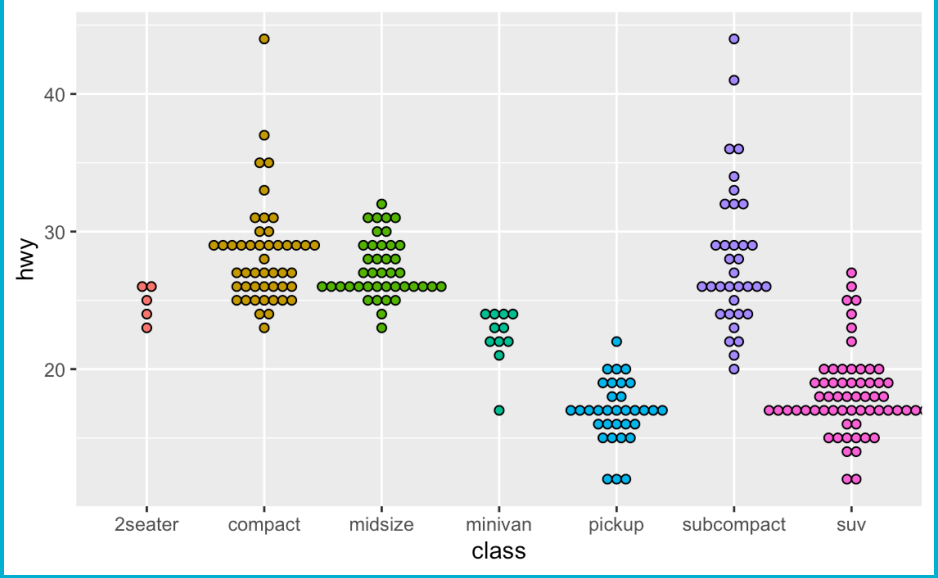


The grouped box plot creates a visualization to easily compare the quantiles of a continuous variable based on the different factors you're interested in.

Graph Code:

f + geom_boxplot()

Plotting w/ a Continuous and Discrete Variable



Dot plots can also be used with 2 variables to create a graph that shows the counts and how values of a continuous variable are distributed for each factor level of a discrete distribution.

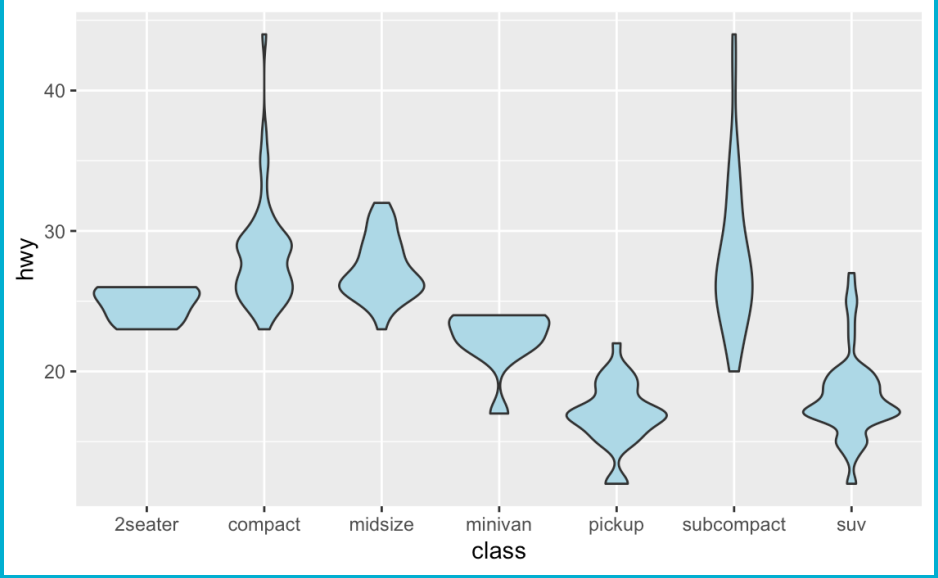
Graph Code:

f + geom_dotplot(binaxis = "y", stackdir = "center", binwidth = 2/3, aes(fill = class), show.legend = FALSE)

- *stackdir = "center":* is a parameter that sets which direction the points extend from. Other options include "up" and "down"

- *binaxis = "y":* sets the axis on which data points are compiled

Plotting w/ a Continuous and Discrete Variable



Similar to the grouped dot plot, the violin graph displays an approximate distribution of your continuous variable for different factor levels.

Graph Code:

f + geom_violin(fill = "Light Blue")

- *fill = "Light Blue":* Sets the color of the 'violins' to light blue

One Step Further

Try New Things:

The functions mentioned in this guide are not comprehensive by any means. GGPlot objects also contain a long list of parameters to create detailed and high-level graphs.

All GGPlot2 functions can be found in R documentation. To find the complete documentation for any function, type “?[function]” into the console tab.