15.3.7

Add Layers to Plots

Jeremy has mastered the boxplot and heat map. What about adding additional plots to his visualizations?

Often when we're building visualizations for data analysis, we'll want to produce layered plots or combine very similar plots into a single visualization (also referred to as faceting, which we'll cover later on).

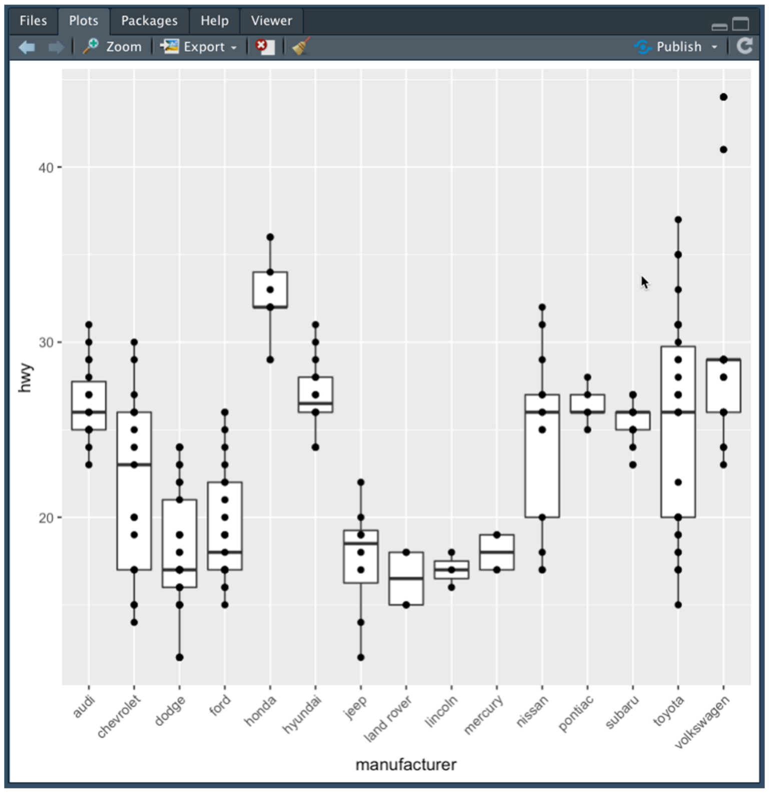
There are two types of plot layers:

1. Layering additional plots that use the **same variables and input data** as the original plot
2. Layering of additional plots that use **different but complementary data** to the original plot

We can add additional plots to our visualization by adding additional `geom` functions to our plotting statement. Layering plots that share input variables can be beneficial when you want to add context to your initial visualization.

For example, to recreate our previous boxplot example comparing the highway fuel efficiency across manufacturers, add our data points using the `geom_point()` function:

```
> plt <- ggplot(mpg,aes(x=manufacturer,y=hwy)) #import dataset into ggplot2
> plt + geom_boxplot() + #add boxplot
> theme(axis.text.x=element_text(angle=45,hjust=1)) + #rotate x-axis labels 45 degrees
> geom_point() #overlay scatter plot on top
```



By layering our data points on top of our boxplot, we can see the general distribution of values within each box as well as the number of data points. This new information can provide the reader better context when comparing two manufacturers with similarly shaped boxplots.

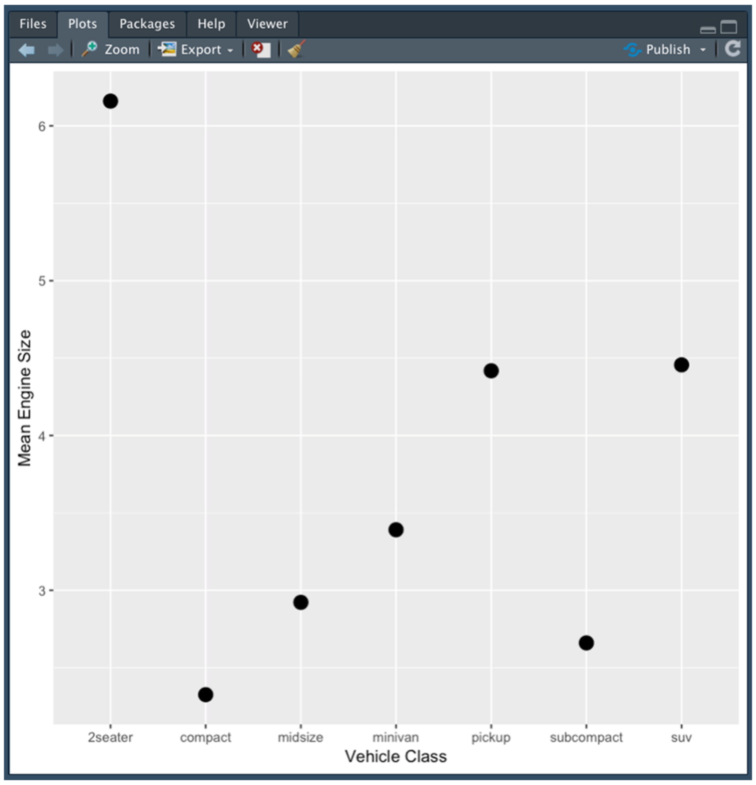
Although layering plots with visualizations using the same input variables is a more common approach, there may be instances when we would want to add additional plotting layers with new and complementary data.

For example, what if we want to compare average engine size for each vehicle class? In this case, we would supply our new data and variables directly to our new `geom` function using the optional mapping and data arguments.

The **mapping argument** functions exactly the same as our `ggplot()` function, where our mapping argument uses the `aes()` function to identify the variables to use. Additionally, the **data argument** can be used to provide a new input data structure; otherwise, the mapping function will reference the data structure provided in the `ggplot` object.

Our R code would be as follows:

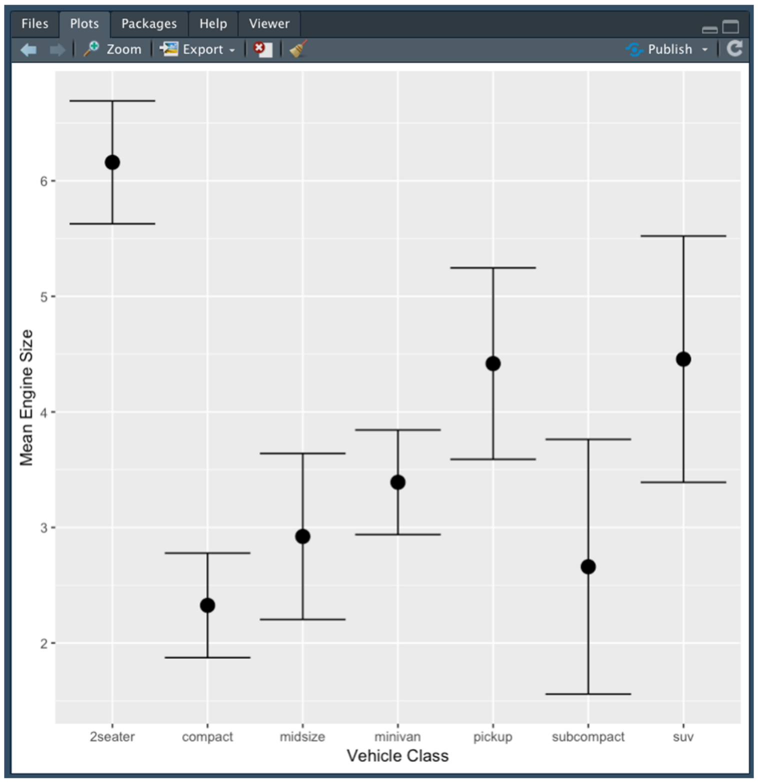
```
> mpg_summary <- mpg %>% group_by(class) %>% summarize(Mean_Engine=mean(displ), .groups = 'keep') #create sum
> plt <- ggplot(mpg_summary,aes(x=class,y=Mean_Engine)) #import dataset into ggplot2
> plt + geom_point(size=4) + labs(x="Vehicle Class",y="Mean Engine Size") #add scatter plot
```



Retake

Although this plot sufficiently visualizes the means, it's critical that we provide context around the standard deviation of the engine size for each vehicle class. If we compute the standard deviations in our dplyr `summarize()` function, we can layer the upper and lower standard deviation boundaries to our visualization using the `geom_errorbar()` function:

```
> mpg_summary <- mpg %>% group_by(class) %>% summarize(Mean_Engine=mean(displ),SD_Engine=sd(displ), .groups =
> plt <- ggplot(mpg_summary,aes(x=class,y=Mean_Engine)) #import dataset into ggplot2
> plt + geom_point(size=4) + labs(x="Vehicle Class",y="Mean Engine Size") + #add scatter plot with labels
> geom_errorbar(aes(ymin=Mean_Engine-SD_Engine,ymax=Mean_Engine+SD_Engine)) #overlay with error bars
```



Layering plots can be very helpful visualizing wide-format data or summary data when there are multiple variables and metrics used to describe a single subject. As the number of subjects increases, or if the input data is in a long format, layering might not be as effective.

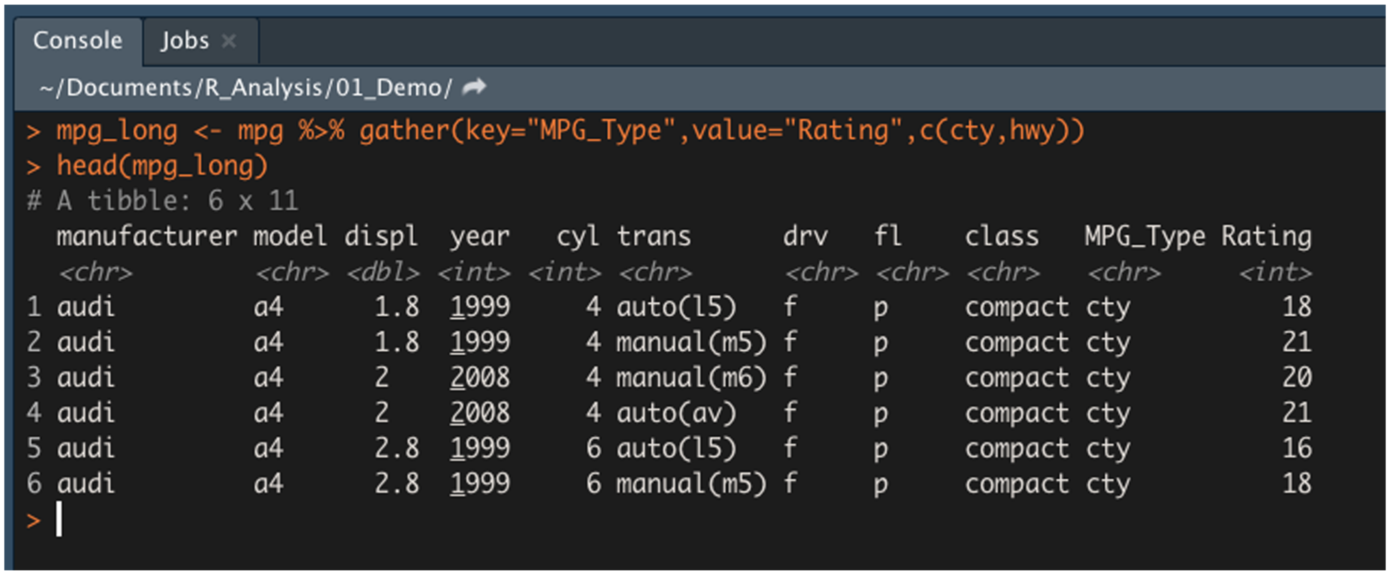
CAUTION

Not all visualizations will benefit from layering plots. Before adding layers of information, ask yourself if the new layer will add context without distracting or taking away from the original plot.

Often when our data is in a long format, we want to avoid visualizing all data within a single plot. Rather, we want to plot all our measurements but keep each level (or category) of our grouping variable separate. This process of separating out plots for each level is known as faceting in ggplot2.

Faceting is performed by adding a `facet()` function to the end of our plotting statement. Consider, if instead of the wide format, our mpg dataset was obtained where city and highway fuel efficiency data was provided in a long format:

```
> mpg_long <- mpg %>% gather(key="MPG_Type",value="Rating",c(cty,hwy)) #convert to long format
> head(mpg_long)
```

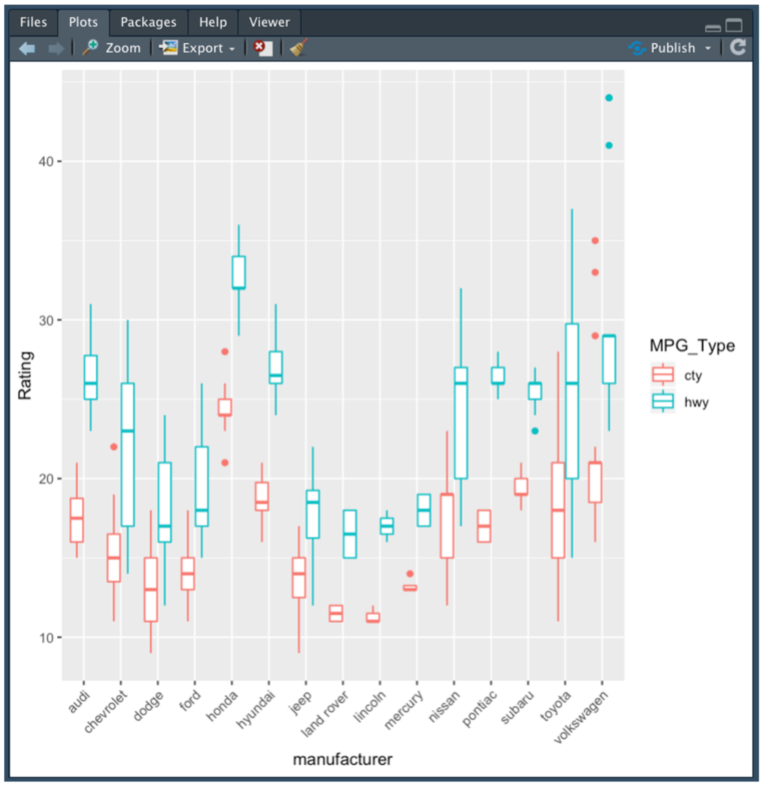


```
> mpg_long <- mpg %>% gather(key="MPG_Type",value="Rating",c(cty,hwy))
> head(mpg_long)
# A tibble: 6 x 11
  manufacturer model displ  year   cyl trans      drv   fl    class   MPG_Type Rating
  <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <chr> <chr>   <chr>     <int>
1 audi         a4      1.8  1999     4 auto(l5)   f     p     compact cty          18
2 audi         a4      1.8  1999     4 manual(m5) f     p     compact cty          21
3 audi         a4      2    2008     4 manual(m6) f     p     compact cty          20
4 audi         a4      2    2008     4 auto(av)   f     p     compact cty          21
5 audi         a4      2.8  1999     6 auto(l5)   f     p     compact cty          16
6 audi         a4      2.8  1999     6 manual(m5) f     p     compact cty          18
> 
```

Retake

If we want to visualize the different vehicle fuel efficiency ratings by manufacturer, our R code would be as follows:

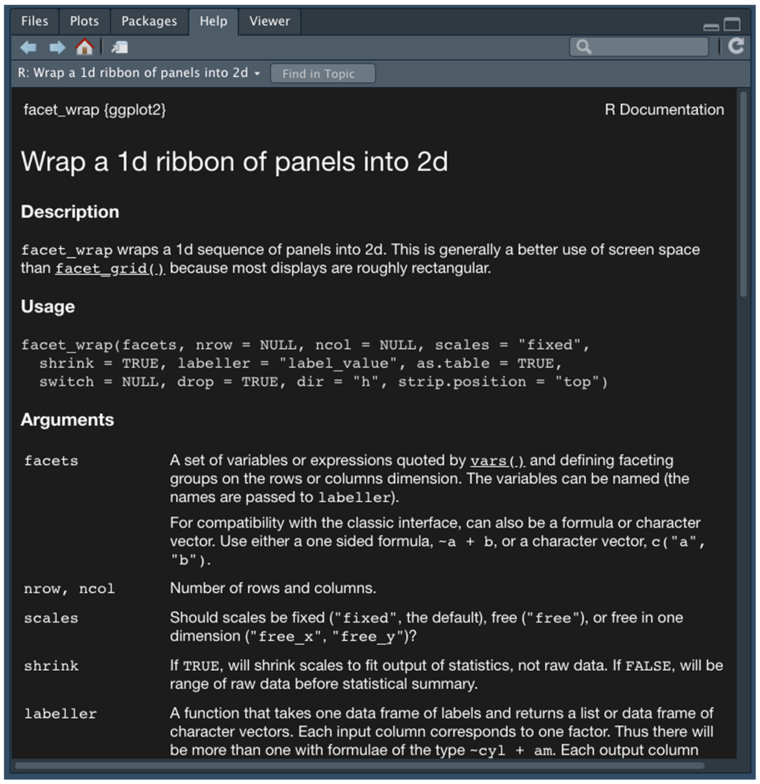
```
> plt <- ggplot(mpg_long,aes(x=manufacturer,y=Rating,color=MPG_Type)) #import dataset into ggplot2
> plt + geom_boxplot() + theme(axis.text.x=element_text(angle=45,hjust=1)) #add boxplot with labels rotated 4
```



The produced boxplot is optimal for comparing the city versus highway fuel efficiency for each manufacturer, but it is more difficult to compare all of the city fuel efficiency across manufacturers. One solution would be to facet the different types of fuel efficiency within the visualization using the `facet_wrap()` function.

Type the following code into the R console to look at the `facet_wrap()` documentation in the Help pane:

```
> ?facet_wrap()
```



facet_wrap {ggplot2} R Documentation

Wrap a 1d ribbon of panels into 2d

Description

facet_wrap wraps a 1d sequence of panels into 2d. This is generally a better use of screen space than facet_grid() because most displays are roughly rectangular.

Usage

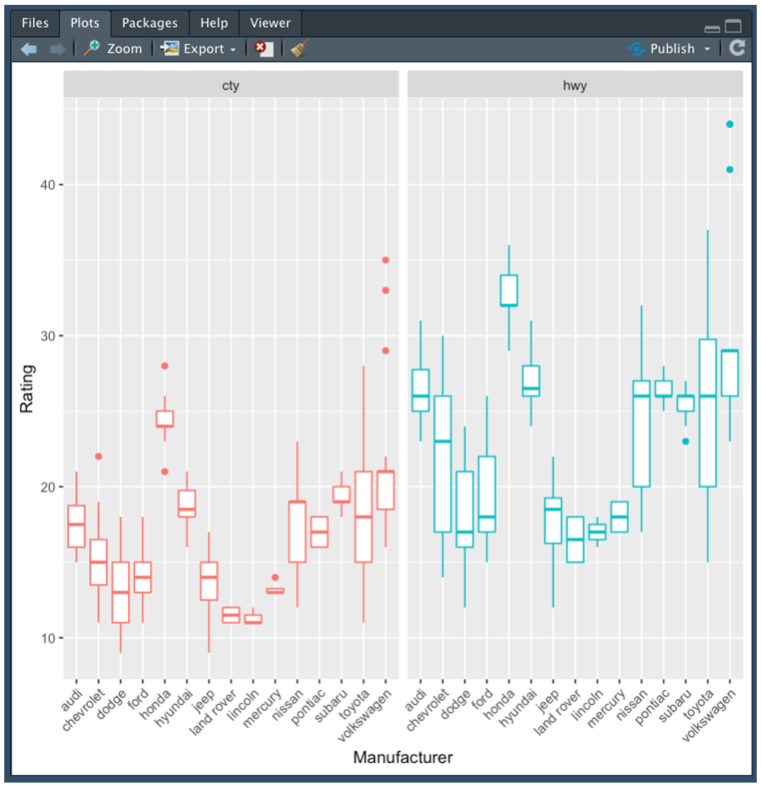
```
facet_wrap(facets, nrow = NULL, ncol = NULL, scales = "fixed",
  shrink = TRUE, labeller = "label_value", as.table = TRUE,
  switch = NULL, drop = TRUE, dir = "h", strip.position = "top")
```

Arguments

facets	A set of variables or expressions quoted by vars() and defining faceting groups on the rows or columns dimension. The variables can be named (the names are passed to labeller). For compatibility with the classic interface, can also be a formula or character vector. Use either a one sided formula, ~a + b, or a character vector, c("a", "b").
nrow, ncol	Number of rows and columns.
scales	Should scales be fixed ("fixed", the default), free ("free"), or free in one dimension ("free_x", "free_y")?
shrink	If TRUE, will shrink scales to fit output of statistics, not raw data. If FALSE, will be range of raw data before statistical summary.
labeller	A function that takes one data frame of labels and returns a list or data frame of character vectors. Each input column corresponds to one factor. Thus there will be more than one with formulae of the type ~cyl + am. Each output column

Similar to any of ggplot2's `geom` functions, the `facet_wrap()` function has many optional variables to tweak the direction and type of faceting. However, the most basic use cases for faceting only require us to provide the annotation for the **facets** argument. The facets argument expects a list of grouping variables to facet by using the `vars()` function. Therefore, to facet our previous example by the fuel-efficiency type, our R code could be as follows:

```
> plt <- ggplot(mpg_long,aes(x=manufacturer,y=Rating,color=MPG_Type)) #import dataset into ggplot2
> plt + geom_boxplot() + facet_wrap(vars(MPG_Type)) + #create multiple boxplots, one for each MPG type
> theme(axis.text.x=element_text(angle=45,hjust=1),legend.position = "none") + xlab("Manufacturer") #rotate x
```



By faceting our boxplots by fuel-efficiency type, it's easier to make comparisons across manufacturers. In this example, we faceted two levels/groups, but more complicated long-format datasets may contain measurements for multiple levels. Using faceting can help make data exploration of these complex datasets easier or can help isolate factors of interest for our audience.

CAUTION

Although there is no hard limit to the number of faceted plots that can be generated, too many faceted plots can render a visualization useless. Generally speaking, the more axis ticks you need to convey your data, the fewer facets you should use.

SKILL DRILL

1. Create one or two additional plots using a different variable for the `facet_wrap()`.
2. Create another plot using two or more variables for the `facet_wrap()`. With this data, does adding more variables make the chart easier or harder to understand?

NOTE

> There are many ways to implement the `facet_wrap()` function, and the documentation can be fairly involved. Thankfully, there are plenty of very simple examples online that can help you customize your faceting.

Now that we have practiced using R to load in a dataset, perform transformation functions, and visualize data, we are ready to analyze our datasets using R.