Seaborn

January 22, 2025

1 Seaborn data visualization in Python

---

[1]:
```
# Setup
import seaborn as sns
from matplotlib import pyplot as plt
```

[2]:
```
# Let's just import 1 file and see what we get
mtcars = sns.load_dataset("mpg")
mtcars.head()
```

[2]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	usa	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	usa	ford torino

[3]:
```
# What are the formats of each column?
mtcars.dtypes
```

[3]:
```
mpg             float64
cylinders         int64
displacement    float64
horsepower      float64
weight            int64
acceleration    float64
model_year        int64
origin           object
name             object
```

```
dtype: object
```

AMAZING! Now that we have all our data, let's get to plotting!

2 Plots

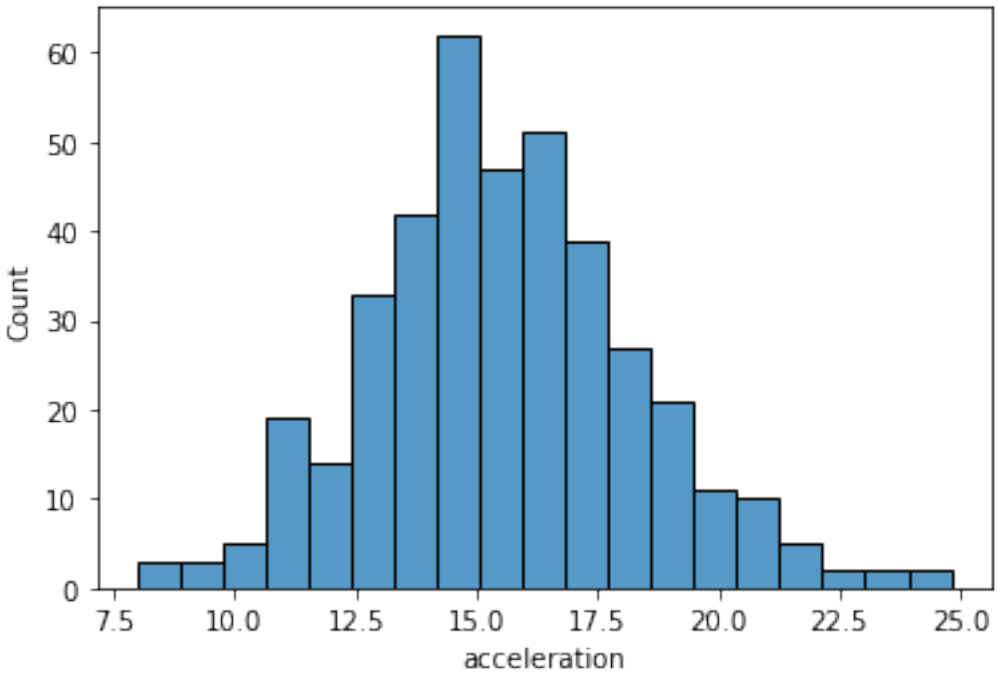
Plots are a tremendously beneficial tool to convey complicated information in a tight way that is much easier to digest than a table or a list of numbers. It allows viewers to visually compare data between groups while also demonstrating the degree of change in a way that makes sense. A detailed and easy to orient plot can make for an impressive and educational addition to any presentation, poster, and paper. Here we will go through some basic plots, how to show groups, confidence intervals, change themes and make your plots easier to read

2.1 Histograms

Histograms are a great way to look at categorical data. By default, they provide count data of on the y-axis per category on the x-axis. Histograms are a great way to see if your data is skewed or normally distributed. The most powerful tend to be ones that use their bins to the fullest. Let's check the spread of our `acceleration` data:

[4]:
```
sns.histplot(data=mtcars, x = "acceleration")
```

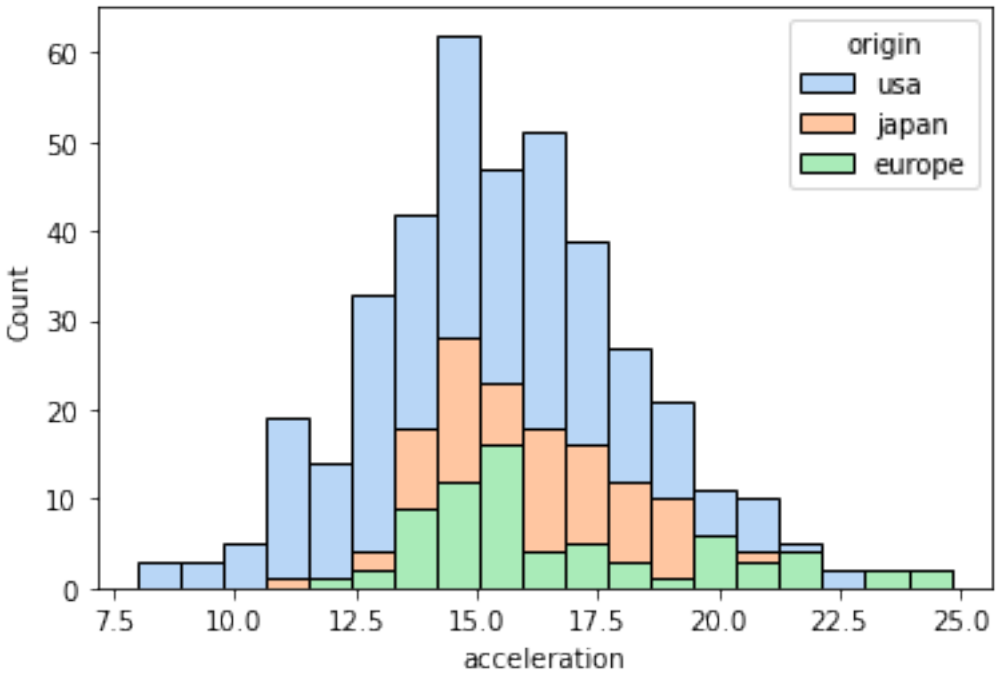
[4]:
```
<AxesSubplot:xlabel='acceleration', ylabel='Count'>
```



How much of our bars represent the origin of each car? Let's make a stacked histogram

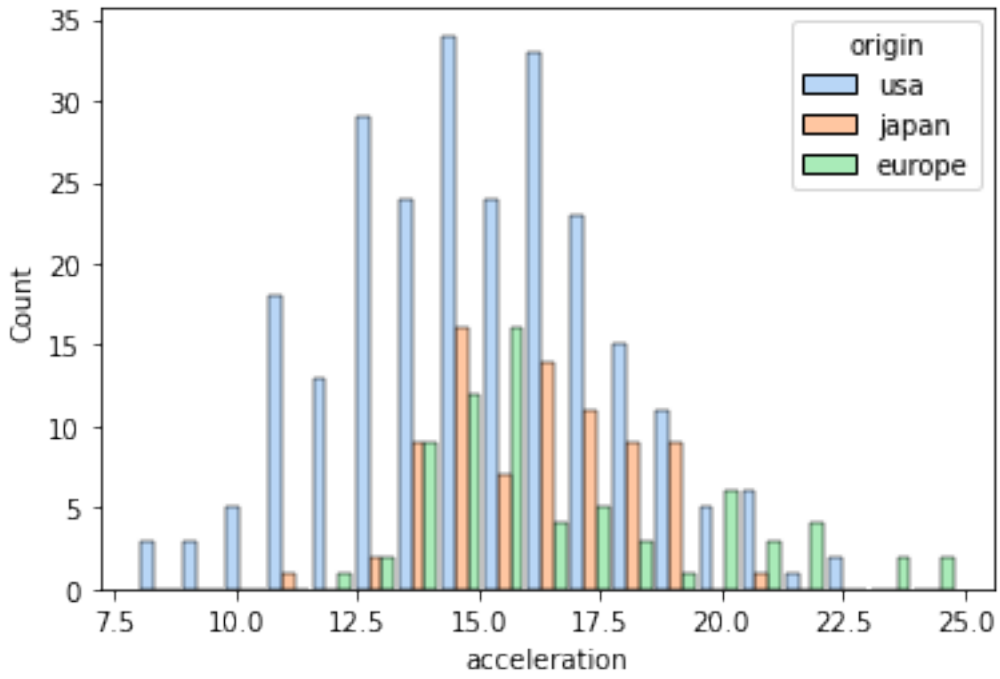
[5]:
```
sns.histplot(data=mtcars, x = "acceleration", hue = "origin", stat="count",␣
↪multiple="stack",
             kde=False,
             palette="pastel",
             element="bars", legend=True)

#Looks like the fastest cars tend to be European!
```

[5]: <AxesSubplot:xlabel='acceleration', ylabel='Count'>

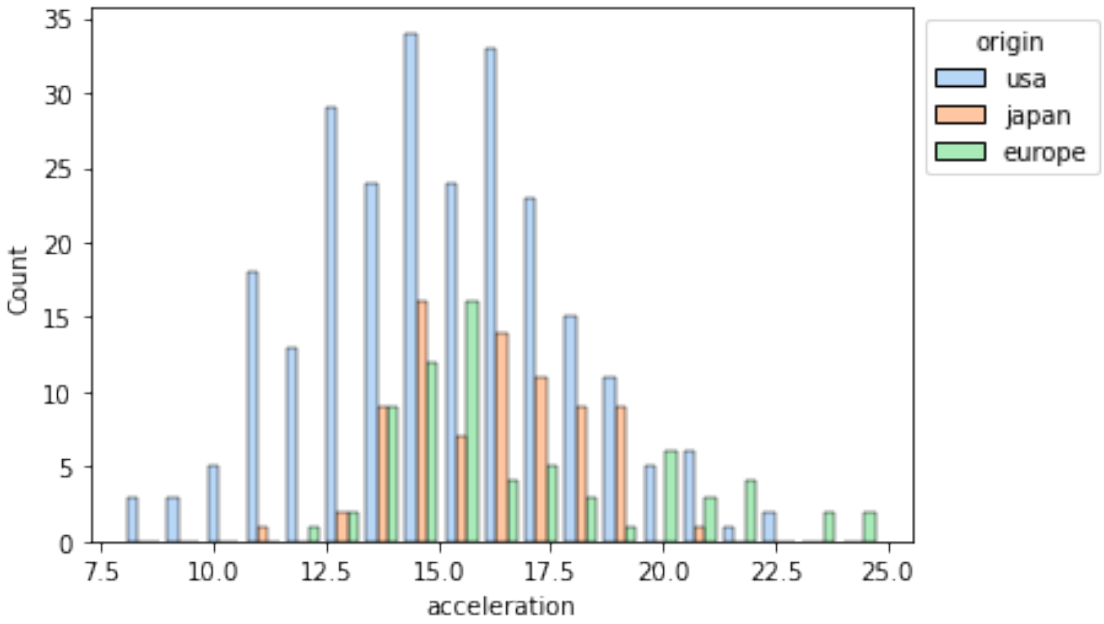


[6]:
```
# What if we wanted to make a jittered histogram instead rather than a stacked?
sns.histplot(data=mtcars, x = "acceleration", hue = "origin", stat="count",␣
↪multiple="dodge", shrink=.9, #Shrink will decrease the width of the bars
             kde=False, #This can draw a curve for us but we can turn it off␣
↪for now to keep it more clean
             palette="pastel",
             element="bars", legend=True)
```

[6]: <AxesSubplot:xlabel='acceleration', ylabel='Count'>



[7]:
```
#What if we want to move the legend outside of the graph space?
hist_plot = sns.histplot(data=mtcars, x = "acceleration", hue = "origin",␣
 ↪stat="count", multiple="dodge", shrink=.8,
             kde=False,
             palette="pastel",
             element="bars", legend=True)
# Change the position of the legend
sns.move_legend(hist_plot, "upper left", bbox_to_anchor=(1, 1))
```

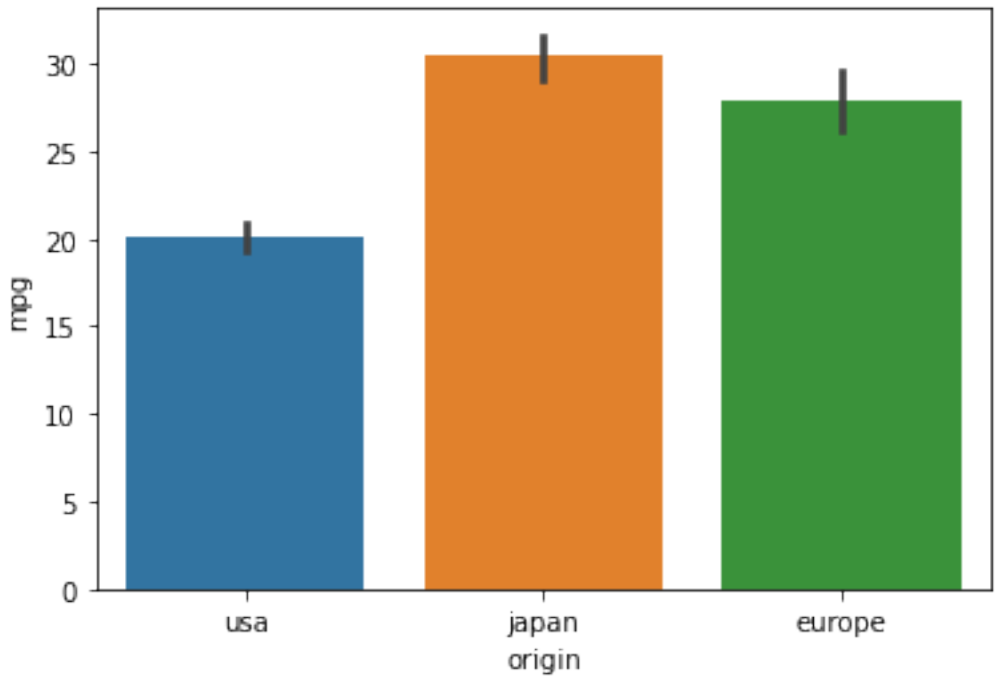


2.2 Bar charts

Bar charts are good for looking at how data is spread and can give you clues about where most of your data sits. It's a great way to connect categorical data to numeric data

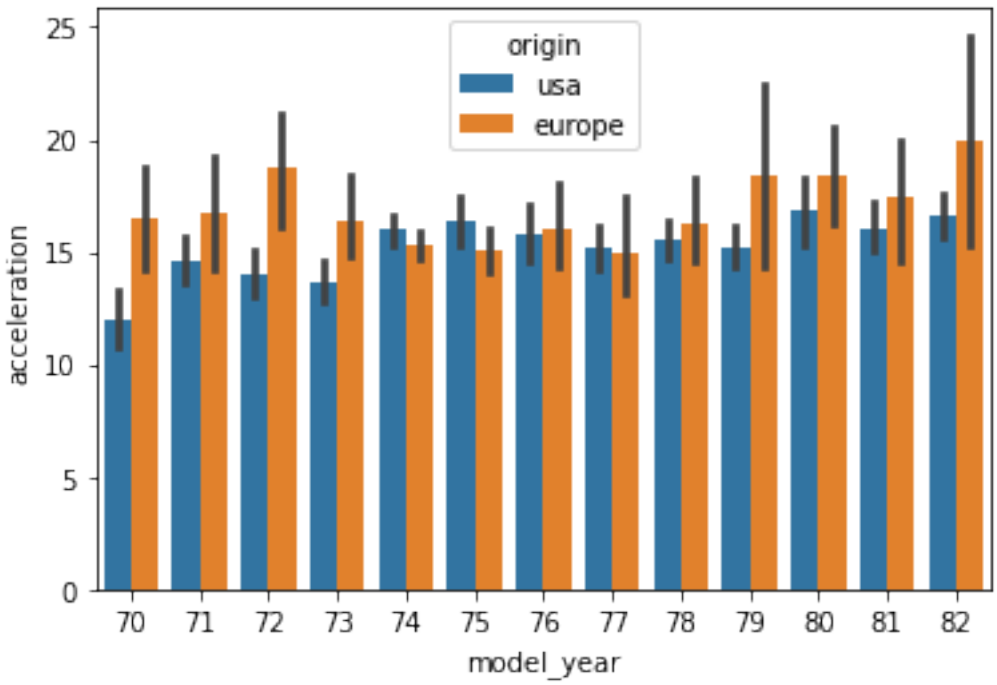
[8]:
```
# Let's look at whether the different origins make more fuel economic cars
sns.barplot(data = mtcars, y="mpg", x="origin")
```

[8]:
```
<AxesSubplot:xlabel='origin', ylabel='mpg'>
```



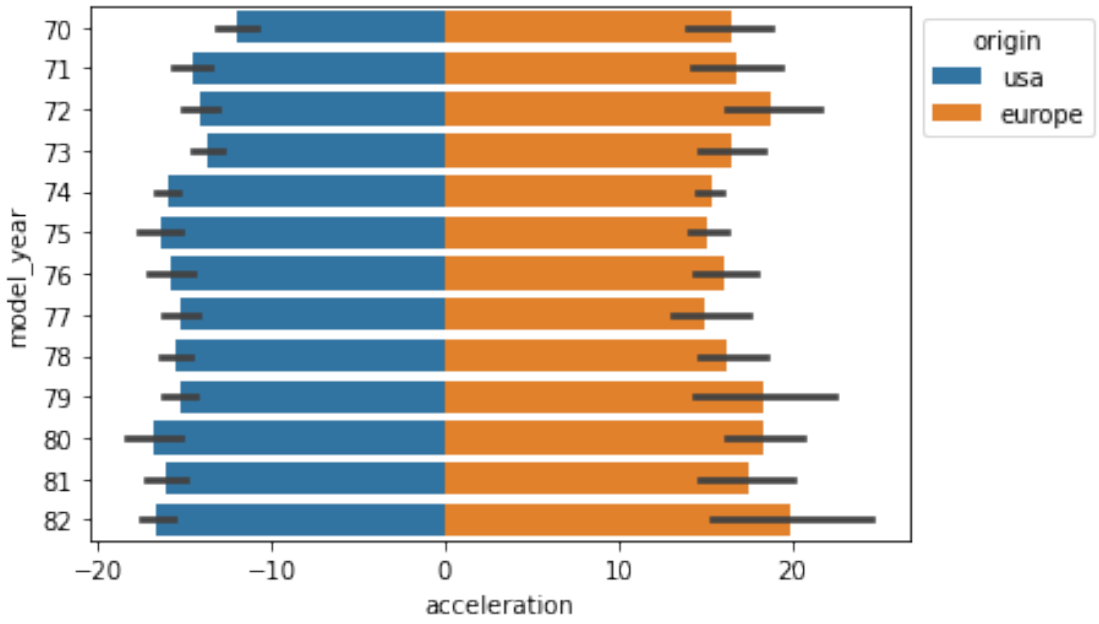
[9]:
```
#What is the acceleration for cars in usa compared to europe from 1970 to 1980?
#First let's filter our data
mtcars_subset = mtcars[(mtcars["origin"] == "usa") | (mtcars["origin"] ==␣
↪"europe") & (mtcars["model_year"] <= 1980)]
mtcars_subset.shape
mtcars_subset.head()
sns.barplot(data = mtcars_subset, y="acceleration", x="model_year", hue␣
↪="origin")
```

[9]:
```
<AxesSubplot:xlabel='model_year', ylabel='acceleration'>
```



[10]:
```
# Can we make our plot look a little more interesting?
mtcars_subset.loc[mtcars_subset["origin"] == "usa", "acceleration"] =␣
↪mtcars_subset.loc[mtcars_subset["origin"] == "usa", "acceleration"] * -1
plot = sns.barplot(data = mtcars_subset, y="model_year", x="acceleration", hue␣
↪="origin", orient = "horizontal", dodge = False)
sns.move_legend(plot, "upper left", bbox_to_anchor=(1, 1))
```

```
/Users/princess/anaconda3/lib/python3.9/site-
packages/pandas/core/indexing.py:1773: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(ilocs[0], value, pi)
```

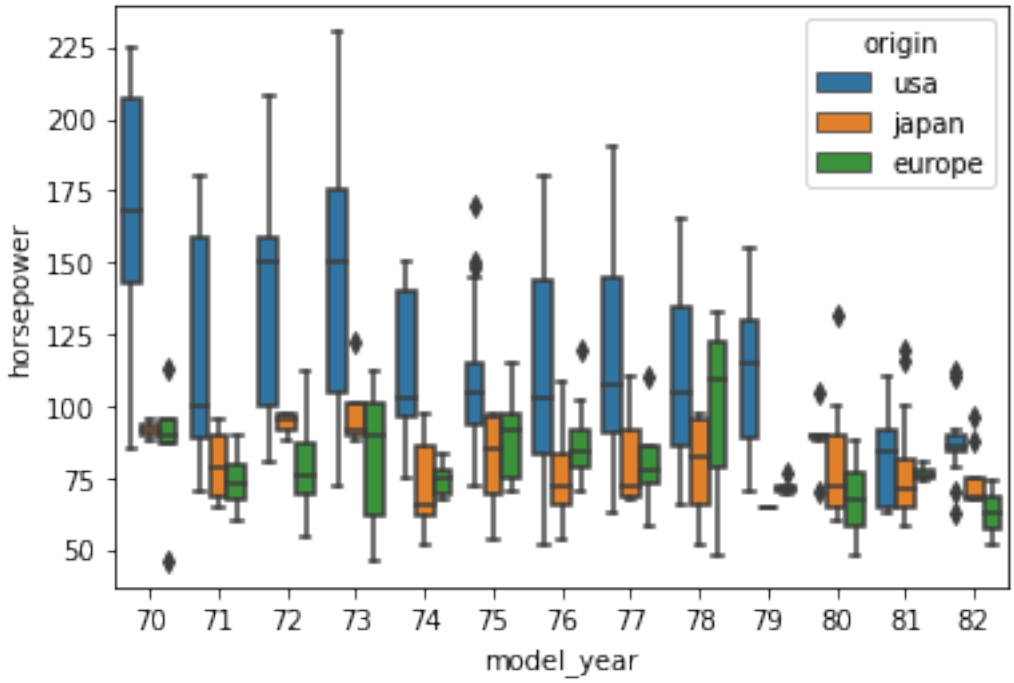


2.3 Box and whisker plots

Box and whisker plots are a great way to easily visualize the statistics of your data. It shows you the minimum, 1st quartile, median, 3rd quartile (Interquartile range), maximum and their outliers. It's a great way to see if you have very different groups

[11]:
```
# Let's see how horsepower changes over time
sns.boxplot(x="model_year",
            y="horsepower",
            hue="origin",
            data=mtcars,
            width=0.8)
```

[11]: <AxesSubplot:xlabel='model_year', ylabel='horsepower'>

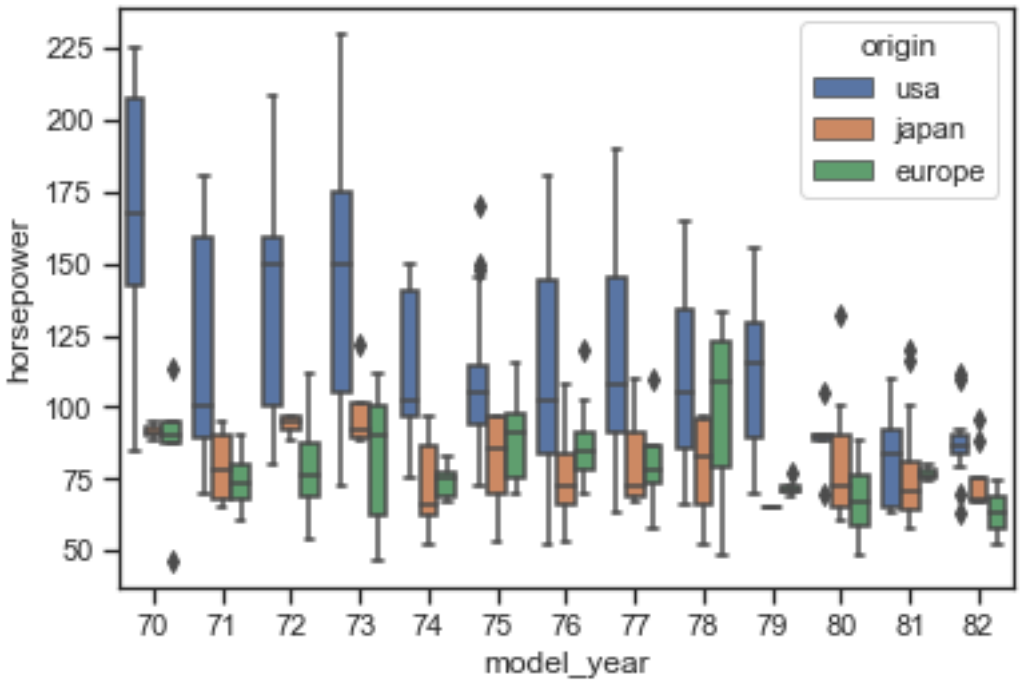


2.3.1 Short Theme and Formatting Introduction

Let's add some themes. Just like `ggplot2`, `seaborn` also has some themes that we can use. `seaborn` comes with 5 built in themes that you can set using the `.set_theme()` method: "darkgrid", "whitegrid", "dark", "white" and "ticks". Let's look at how each of them alters our plot space.

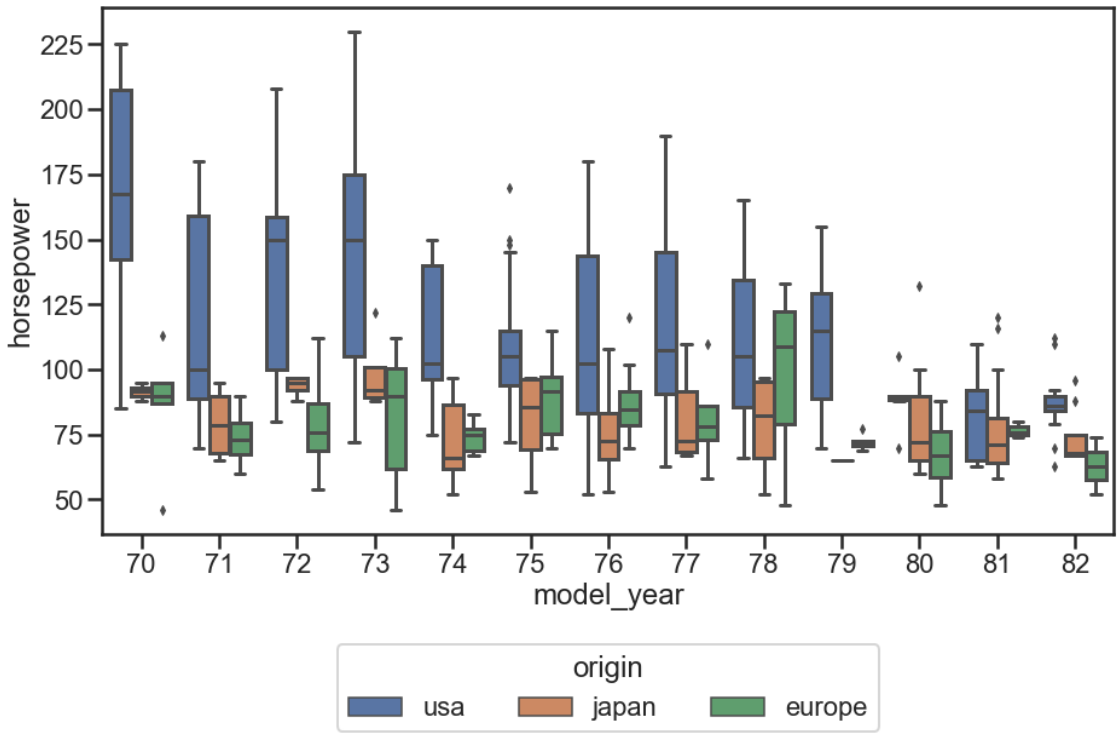
[12]:
```
sns.set_theme(style = "ticks")
sns.boxplot(x="model_year",
            y="horsepower",
            hue="origin",
            data=mtcars,
            width=0.8)
```

[12]:
```
<AxesSubplot:xlabel='model_year', ylabel='horsepower'>
```



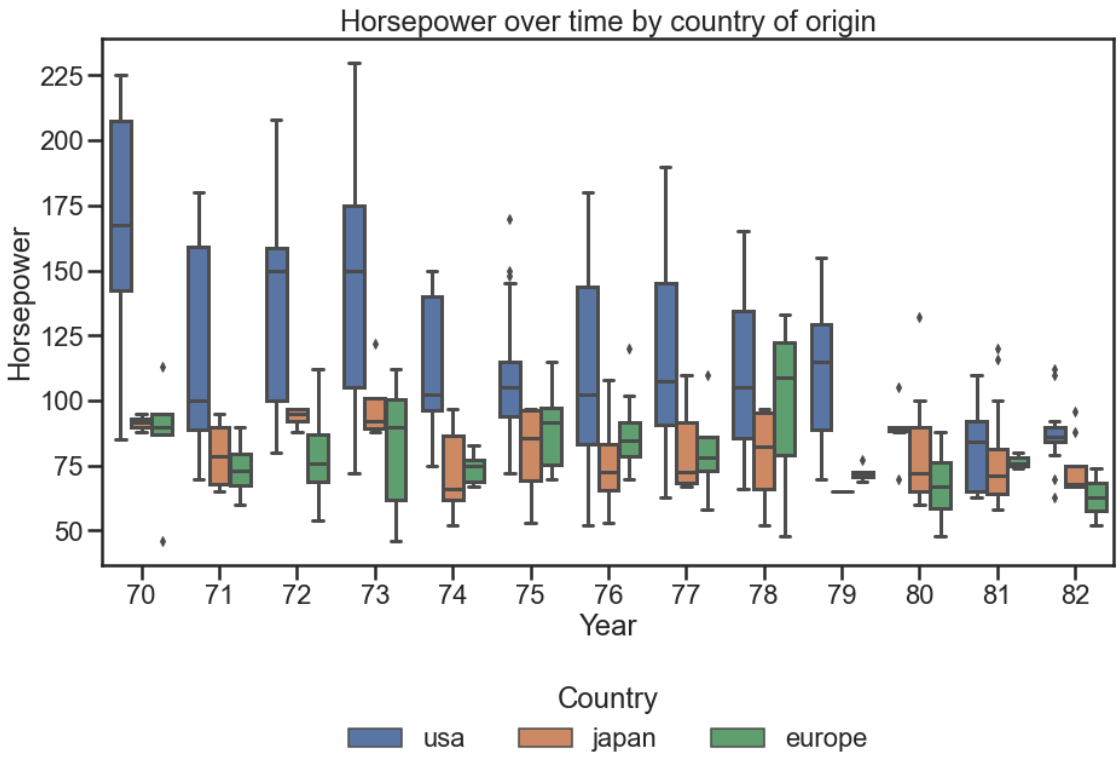
`seaborn` also has some preset layouts for us to use depending on the context that we want to use it for. Using the `.set_context()` method we can choose between paper, notebook, talk, and poster. Let's see how each looks

[13]:
```
sns.set_context("poster")
plt.figure(figsize=(15,8))
box = sns.boxplot(x="model_year",
            y="horsepower",
            hue="origin",
            data=mtcars)
sns.move_legend(box, "lower center", bbox_to_anchor=(0.5, -0.4), ncol = 3)
```



We can also change the name of our title and x and y labels. We can do this using the `.set()` method

[14]:
```
plt.figure(figsize=(15,8))
box = sns.boxplot(x="model_year",
            y="horsepower",
            hue="origin",
            data=mtcars)
box.set(xlabel="Year", ylabel = "Horsepower", title = "Horsepower over time by␣
 ↪country of origin")
plt.legend(title='Country', frameon=False)
sns.move_legend(box, "lower center", bbox_to_anchor=(0.5, -0.4), ncol = 3)
plt.show()
```

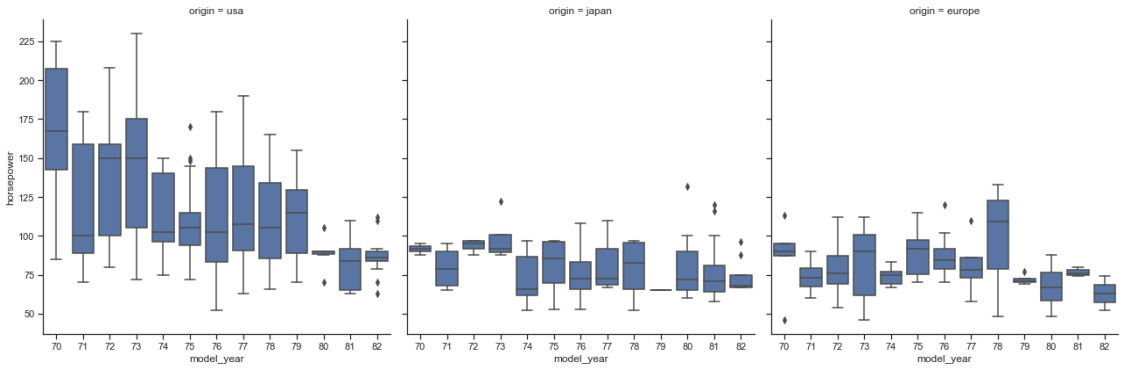


Let's go back to the basics - what if we want to look at each of the countries on a different plot? We can use a `FacetGrid`. `FacetGrids` are a great way to plot many mini plots together in one figure so that you can see how your data relates together as a whole. Let's try it out:

[15]:
```
sns.set_context("notebook")
# Set up the FacetGrid with our data and what we want the columns to be␣
 ↪separated by
plot = sns.FacetGrid(mtcars, col="origin", height = 6, aspect = 1) #make it␣
 ↪larger but maintain aspect ratio

# Map our data
plot.map(sns.boxplot, "model_year", "horsepower")
```

```
/Users/princess/anaconda3/lib/python3.9/site-packages/seaborn/axisgrid.py:670:
UserWarning: Using the boxplot function without specifying `order` is likely to
produce an incorrect plot.
  warnings.warn(warning)
```

[15]:
```
<seaborn.axisgrid.FacetGrid at 0x7fe899abe100>
```



[16]:
```
mtcars.head()
```

[16]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	\
0	18.0	8	307.0	130.0	3504	12.0	
1	15.0	8	350.0	165.0	3693	11.5	
2	18.0	8	318.0	150.0	3436	11.0	
3	16.0	8	304.0	150.0	3433	12.0	
4	17.0	8	302.0	140.0	3449	10.5	

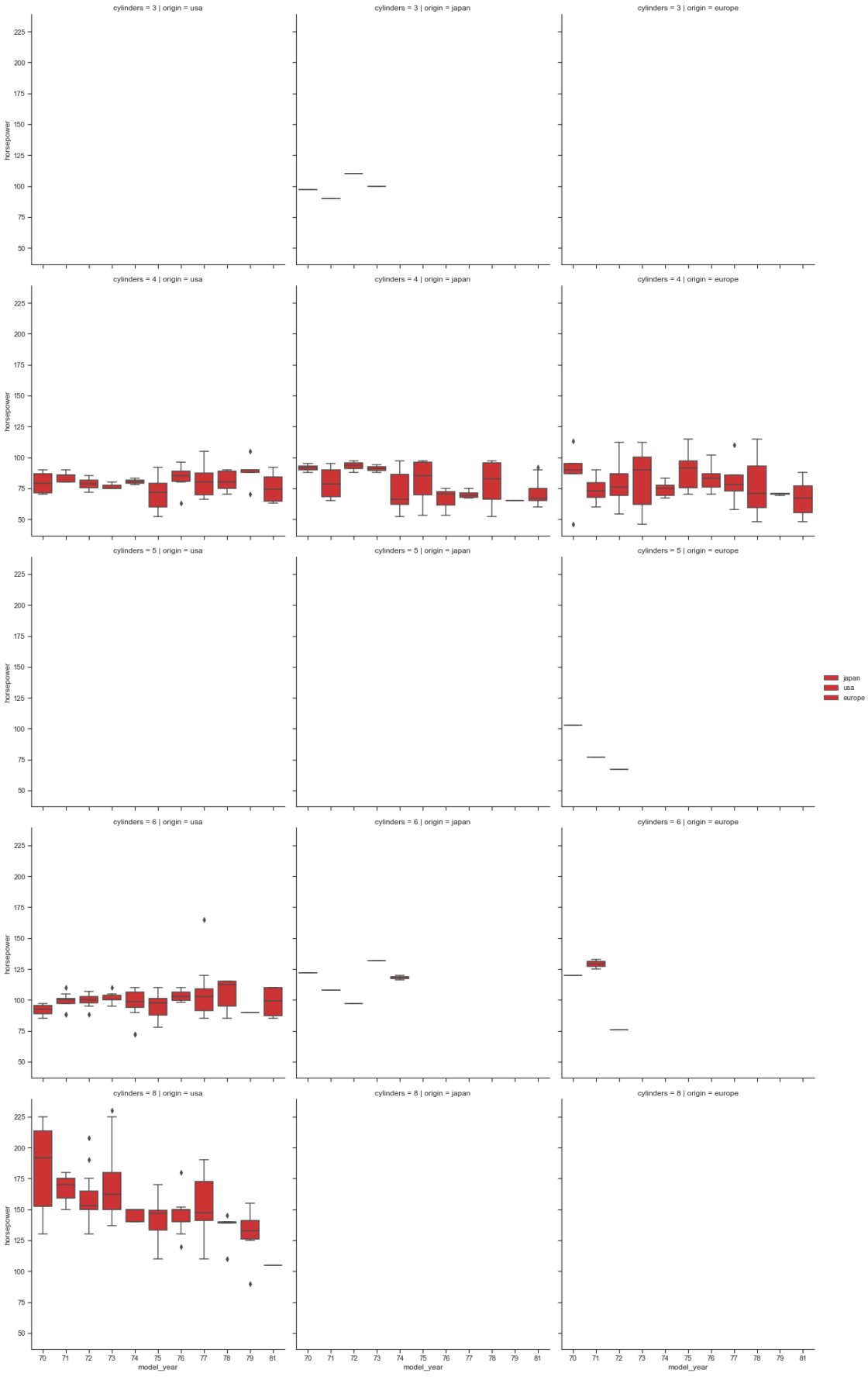
	model_year	origin	name
0	70	usa	chevrolet chevelle malibu
1	70	usa	buick skylark 320
2	70	usa	plymouth satellite
3	70	usa	amc rebel sst
4	70	usa	ford torino

[17]:
```
#Let's add another dimention. Is there a pattern with number of cylinders?
plot = sns.FacetGrid(mtcars, col="origin", row = "cylinders", height=6,␣
 ↪aspect=1)

# Map our data
plot.map(sns.boxplot, "model_year", "horsepower", "origin", palette="Set1").
 ↪add_legend()

#Ah well that's probably not ideal, but we know that our data is not evenly␣
 ↪spread!
```

```
/Users/princess/anaconda3/lib/python3.9/site-packages/seaborn/axisgrid.py:670:
UserWarning: Using the boxplot function without specifying `order` is likely to
produce an incorrect plot.
  warnings.warn(warning)
/Users/princess/anaconda3/lib/python3.9/site-packages/seaborn/axisgrid.py:675:
UserWarning: Using the boxplot function without specifying `hue_order` is likely
to produce an incorrect plot.
```

```
        warnings.warn(warning)
```

[17]: <seaborn.axisgrid.FacetGrid at 0x7fe87af43700>

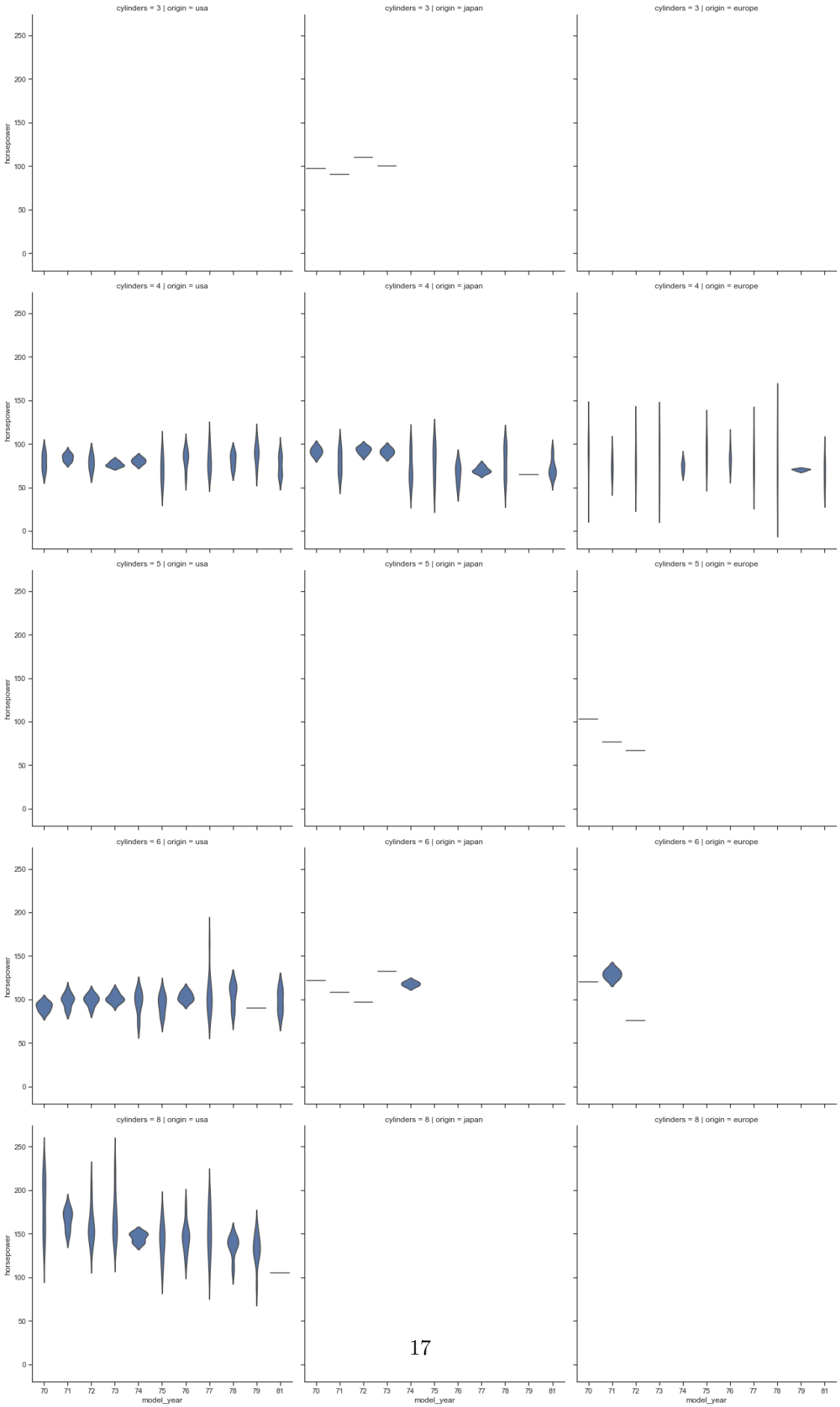

cylinders = 3 | origin = usa
cylinders = 3 | origin = japan
cylinders = 3 | origin = europe
cylinders = 4 | origin = usa
cylinders = 4 | origin = japan
cylinders = 4 | origin = europe
cylinders = 5 | origin = usa
cylinders = 5 | origin = japan
cylinders = 5 | origin = europe
cylinders = 6 | origin = usa
cylinders = 6 | origin = japan
cylinders = 6 | origin = europe
cylinders = 8 | origin = usa
cylinders = 8 | origin = japan
cylinders = 8 | origin = europe
horsepower
model_year
japan
usa
europe

Now that's some wonky looking data! Let's add some labels so that we can make sense of it and our readers can too.

[18]:
```
#Let's add another dimention. Is there a pattern with number of cylinders?
vplot = sns.FacetGrid(mtcars, col="origin", row = "cylinders", height=6,␣
 ↪aspect=1)
# Map our data
vplot.map(sns.violinplot, "model_year", "horsepower", inner = None) #inner =␣
 ↪None will turn off the inner boxplot
```

```
/Users/princess/anaconda3/lib/python3.9/site-packages/seaborn/axisgrid.py:670:
UserWarning: Using the violinplot function without specifying `order` is likely
to produce an incorrect plot.
  warnings.warn(warning)
```

[18]:
```
<seaborn.axisgrid.FacetGrid at 0x7fe8a84124c0>
```

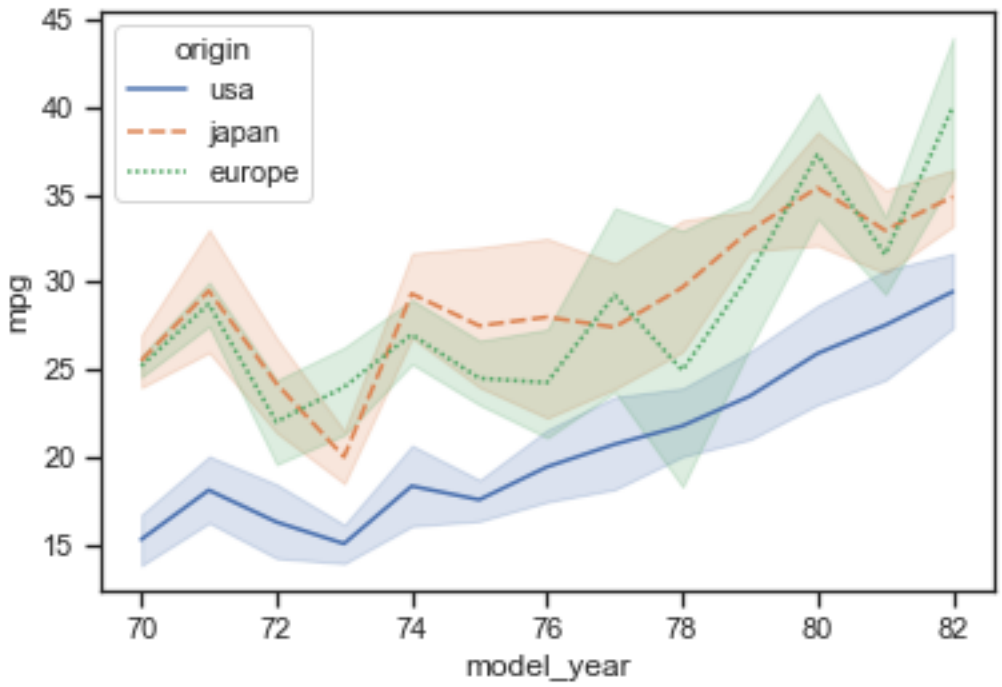

cylinders = 3 | origin = usa
cylinders = 3 | origin = japan
cylinders = 3 | origin = europe
cylinders = 4 | origin = usa
cylinders = 4 | origin = japan
cylinders = 4 | origin = europe
cylinders = 5 | origin = usa
cylinders = 5 | origin = japan
cylinders = 5 | origin = europe
cylinders = 6 | origin = usa
cylinders = 6 | origin = japan
cylinders = 6 | origin = europe
cylinders = 8 | origin = usa
cylinders = 8 | origin = japan
cylinders = 8 | origin = europe
horsepower
0
50
100
150
200
250
model_year
70
71
72
73
74
75
76
77
78
79
81

2.4 Line graphs

Line graphs are a great tool to show data that spans time. It can be similar to a scatterplot in that the raw data is plotted, however it can also provide useful information such as clear trends, especially when handling multiple groups, standard error, confidence intervals and more. Let's make some line graphs!

[19]:
```
sns.lineplot(x="model_year",
             y="mpg",
             style="origin",
              hue = "origin",
             data=mtcars)
```

[19]: <AxesSubplot:xlabel='model_year', ylabel='mpg'>



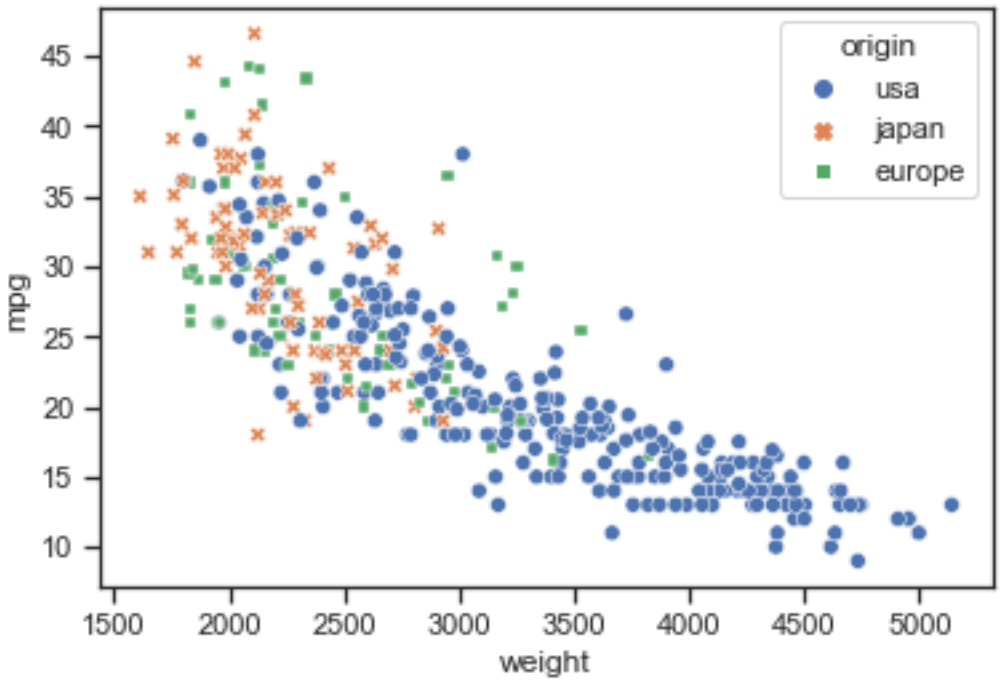
We see that we have some faint orange outline surrounding our solid lines. This is the confidence interval of all the different car models that we have.

2.5 Scatter plots

Scatterplots like lineplots are a great way to try and find the relationships between two variables. Let's try and look to see if weight is correlated with gas consumption

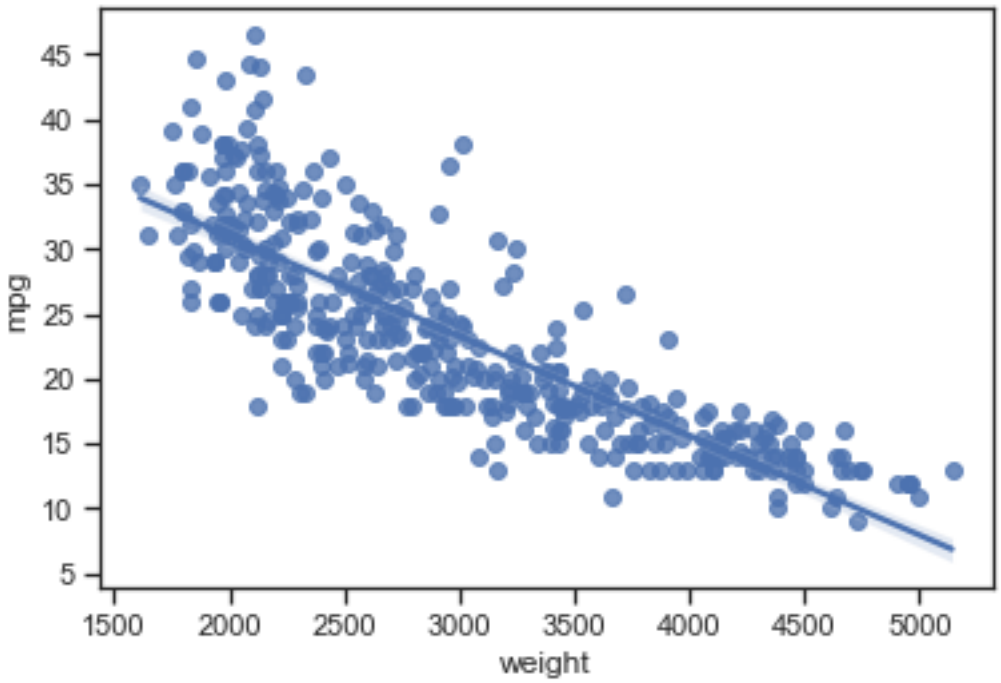
[20]:
```
# Do heavier cars take more gas?
sns.scatterplot(x="weight",
                y="mpg",
                style="origin",
                 hue = "origin",
                data=mtcars)
```

[20]: <AxesSubplot:xlabel='weight', ylabel='mpg'>



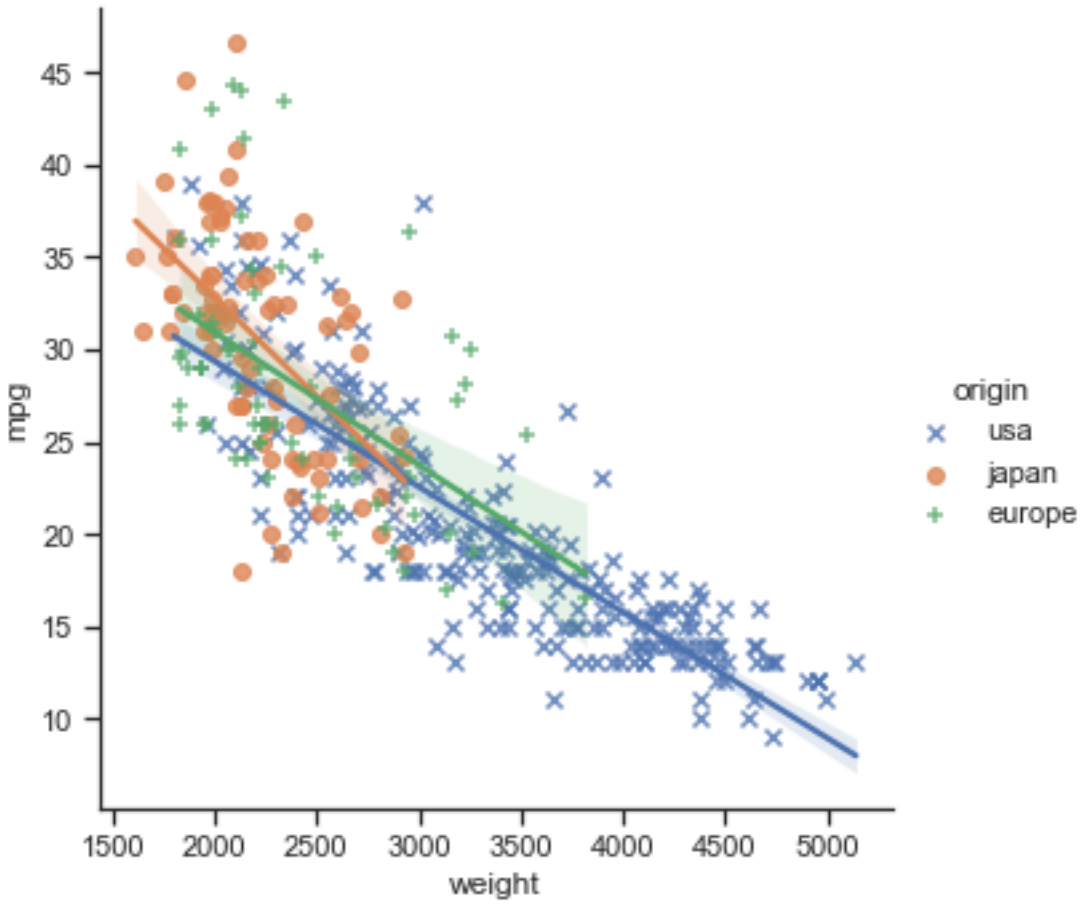
[21]:
```
# we can also make a regression plot
sns.regplot(x="weight",
                y="mpg",
                data=mtcars)
```

[21]: <AxesSubplot:xlabel='weight', ylabel='mpg'>



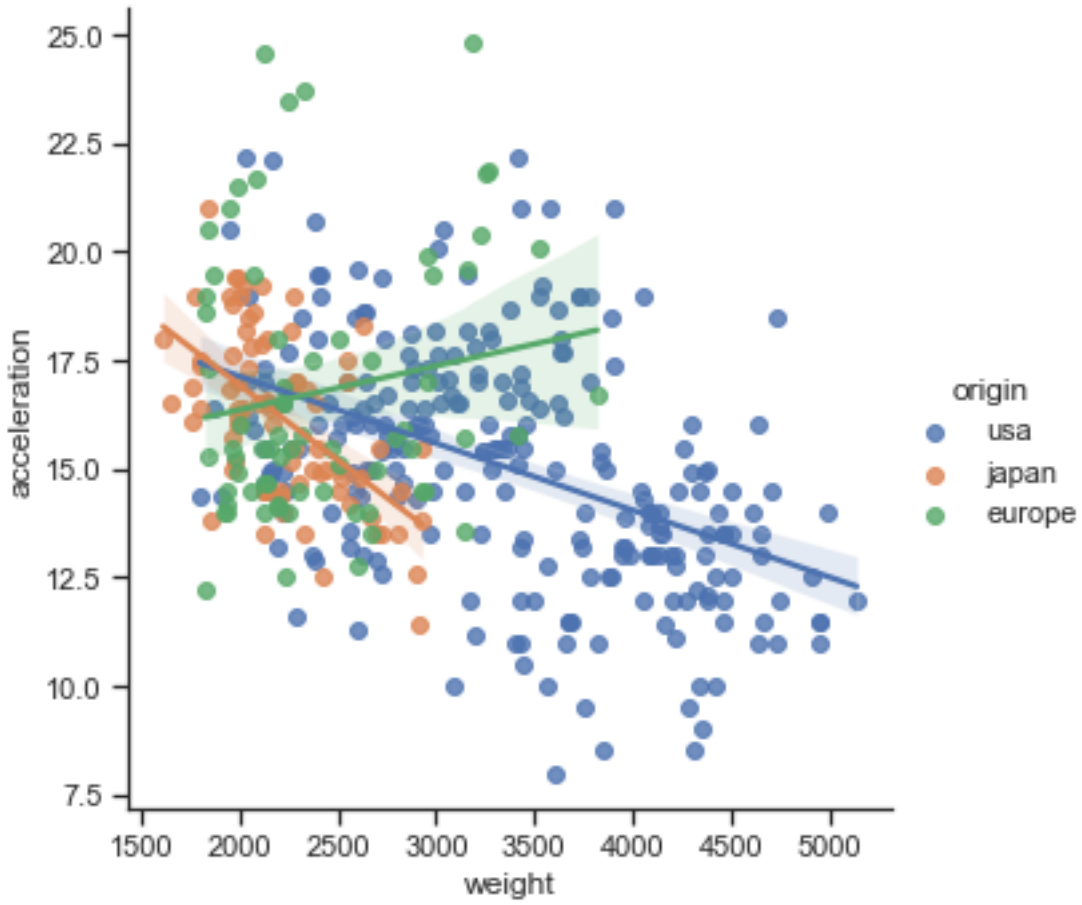
[22]:
```
#and a linear model plot
sns.lmplot(x="weight",
           y="mpg",
          hue = "origin",
           data=mtcars,
         markers = ["x", "o", "+"])
```

[22]: <seaborn.axisgrid.FacetGrid at 0x7fe86820f0d0>



[23]:
```
# Do heavier cars accelerate slower?
sns.lmplot(x="weight",
            y="acceleration",
           hue = "origin",
            data=mtcars)
```

[23]: <seaborn.axisgrid.FacetGrid at 0x7fe8681fa610>



3 Recap

In this lesson we've looked at 5 major graphs each better at representing different things:

- Histograms are a good way to visualize counts of categorical data and proportions
- Bar charts are a good way of relating categorical data and numeric data that aren't counts
- Box and Whisker plots (and violin) are good for looking at the spread of your data and potential outliers.
- Line graphs are good to show how data changes over time
- Scatterplots are great to show the relationship between two variables

While `seaborn` does not add layers like `ggplot2` might, the function structures are generally the same:

```
sns.xxxplot(data = my_data, x = x_var, y = y_var, hue = color_var, style = style_var)
```

We've also looked at various themes that come pre-packaged with `seaborn`.

There is only so much that we can cover in lesson, however the assignment will help you to further

your skills on your own.