Instructor 4 - ggplot & altair 2022

June 21, 2022

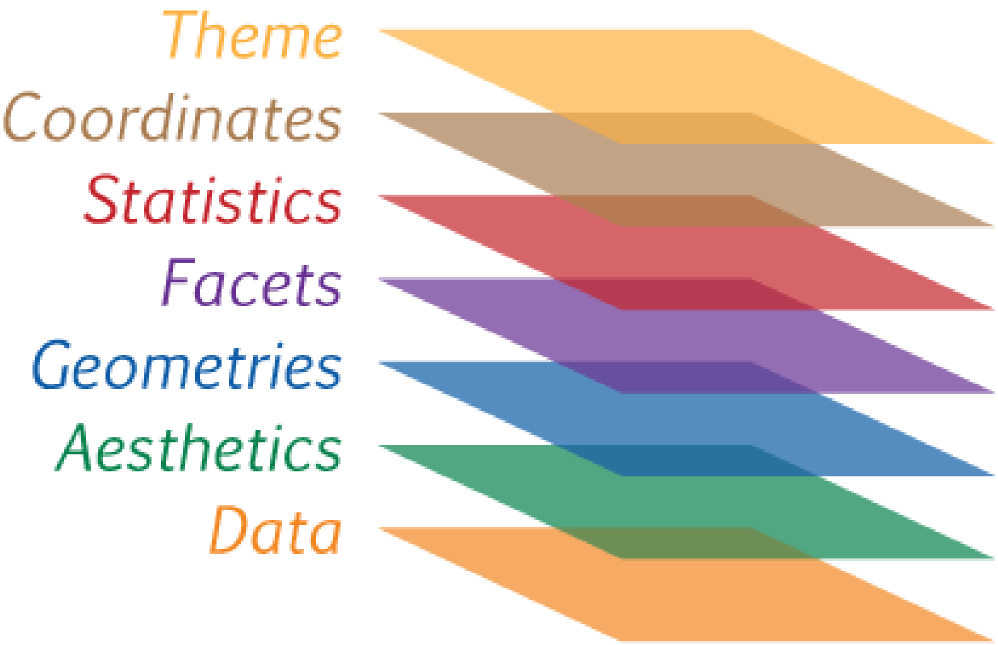
Table of Contents: ggplot & Altair

1 ggplot

1.1 The Grammar of Graphics

https://www.science-craft.com/2014/07/08/introducing-the-grammar-of-graphics-plotting-concept/

The original paper: https://vita.had.co.nz/papers/layered-grammar.pdf



1.2 Imports

1. **Data** the dataset to use when creating the plot.
2. **Aesthetics** (aes) variables used by the underlying drawing system. Variables are mapped to the x- and y-axis aesthetic variables.
3. **Geometries** objects (geoms) defines the type of geometric object to use in the drawing. You can use points, lines, bars, and many others.
4. **Facets** allow data to be divided into groups and each group is plotted on to a separate panel in the same graphic.
5. **Statistics** transformations specify computations and aggregations to be applied to the data before plotting it.
6. **Coordinates** systems map the position of objects to a 2D graphical location in the plot.
7. **Themes** allows you to control visual properties like colors, fonts, and shapes (aka non-data ink).

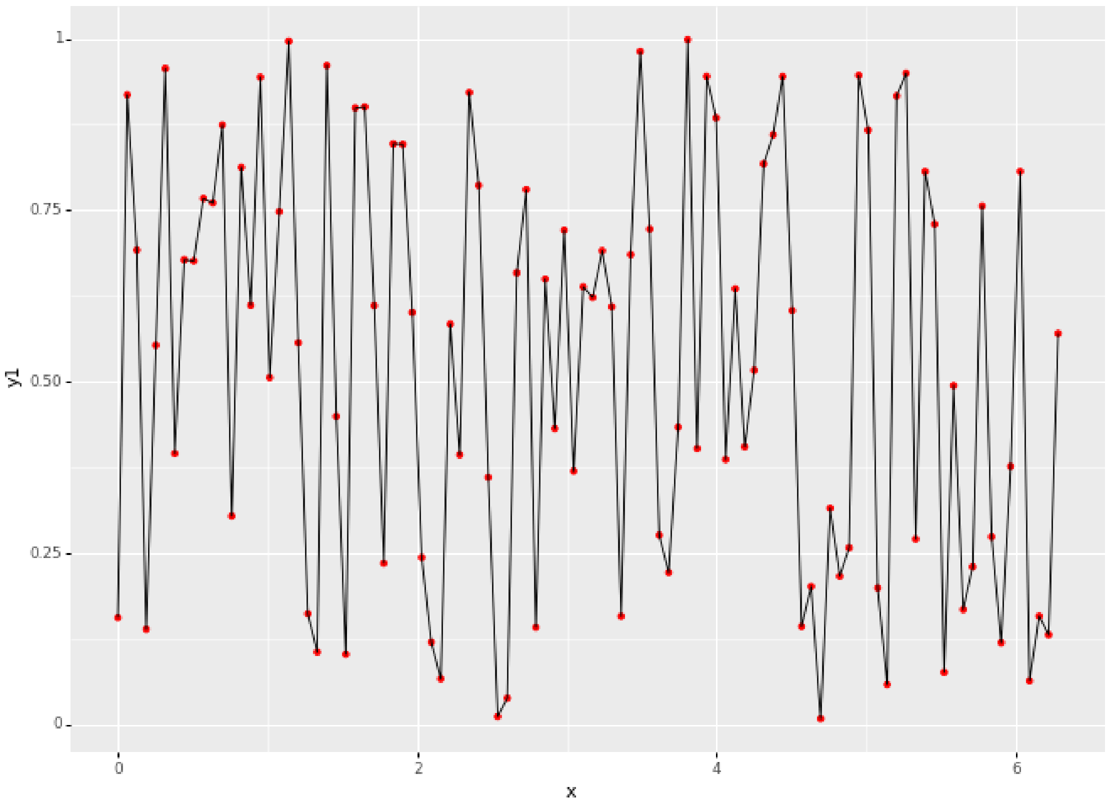
[1]:
```
#import sys
import random

from plotnine import ggplot, geom_point, aes, geom_line, geom_bar
from plotnine import stat_bin, theme, theme_538, theme_xkcd, geom_histogram #␣
 ↪on 2 lines for clarity
import pandas as pd
import numpy as np

#import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
```

```
from plotnine.data import mpg, huron, economics, diamonds
```

1.3 Sample Code

[2]:

```
dpi = 72
size_inches = (11, 8)                                       # size in inches␣
  ↪(for the plot)
size_px = int(size_inches[0]*dpi), int(size_inches[1]*dpi)  # For the canvas



n = 100
x = np.linspace(0, 2 * np.pi, n)
df = pd.DataFrame({
    'x': x,
    'y1': np.random.rand(n),
    'y2': np.sin(x),
    'y3': np.cos(x) * np.sin(x)
    })

        # change the dependent variable and color each time this method is␣
  ↪called
y = random.choice(['y1', 'y2', 'y3'])
color = random.choice(['blue', 'red', 'green'])

        # specify the plot and get the figure object
ff = (ggplot(df, aes('x', y))
    + geom_point(color=color)
    + geom_line()
    + theme(figure_size=size_inches,dpi=dpi))
fig = ff.draw()
```



1.3.1 Datasets & Aes (variables)

- diamonds
- economics
- mpg
- huron

[3]:

```
diamonds
```

[3]:

```
         carat        cut color clarity  depth  table  price     x     y     z
0         0.23      Ideal     E     SI2   61.5   55.0    326  3.95  3.98  2.43
1         0.21    Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
2         0.23       Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
3         0.29    Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
4         0.31       Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
…        …         …    …      …     …     …     …    …    …    …
53935     0.72      Ideal     D     SI1   60.8   57.0   2757  5.75  5.76  3.50
53936     0.72       Good     D     SI1   63.1   55.0   2757  5.69  5.75  3.61
53937     0.70  Very Good     D     SI1   62.8   60.0   2757  5.66  5.68  3.56
53938     0.86    Premium     H     SI2   61.0   58.0   2757  6.15  6.12  3.74
53939     0.75      Ideal     D     SI2   62.2   55.0   2757  5.83  5.87  3.64
```

```
[53940 rows x 10 columns]
```

[4]:

```
mpg
```

[4]:

```
    manufacturer   model  displ  year  cyl       trans drv  cty  hwy fl  \
0           audi      a4    1.8  1999    4    auto(l5)   f   18   29  p
1           audi      a4    1.8  1999    4  manual(m5)   f   21   29  p
2           audi      a4    2.0  2008    4  manual(m6)   f   20   31  p
3           audi      a4    2.0  2008    4    auto(av)   f   21   30  p
4           audi      a4    2.8  1999    6    auto(l5)   f   16   26  p
..           …       …     …     …   …          …   ..   …    …  ..
229   volkswagen  passat    2.0  2008    4    auto(s6)   f   19   28  p
230   volkswagen  passat    2.0  2008    4  manual(m6)   f   21   29  p
231   volkswagen  passat    2.8  1999    6    auto(l5)   f   16   26  p
232   volkswagen  passat    2.8  1999    6  manual(m5)   f   18   26  p
233   volkswagen  passat    3.6  2008    6    auto(s6)   f   17   26  p

       class
0    compact
1    compact
2    compact
3    compact
4    compact
..       …
229  midsize
230  midsize
231  midsize
232  midsize
233  midsize

[234 rows x 11 columns]
```

[5]:

```
huron
```

[5]:

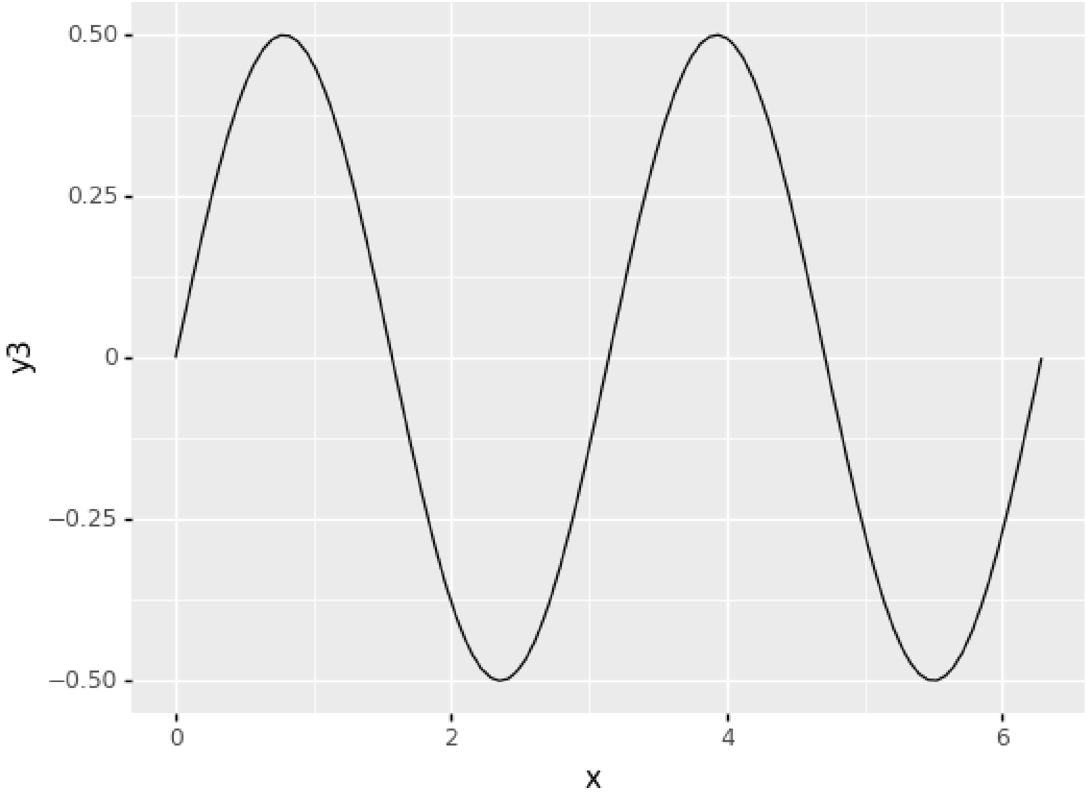
```
    year   level  decade
0   1875  580.38    1870
1   1876  581.86    1870
2   1877  580.97    1870
3   1878  580.80    1870
4   1879  579.79    1870
..   …       …     …
93  1968  578.52    1960
94  1969  579.74    1960
95  1970  579.31    1970
96  1971  579.89    1970
97  1972  579.96    1970
```

```
[98 rows x 3 columns]
```

1.3.2 Geometries

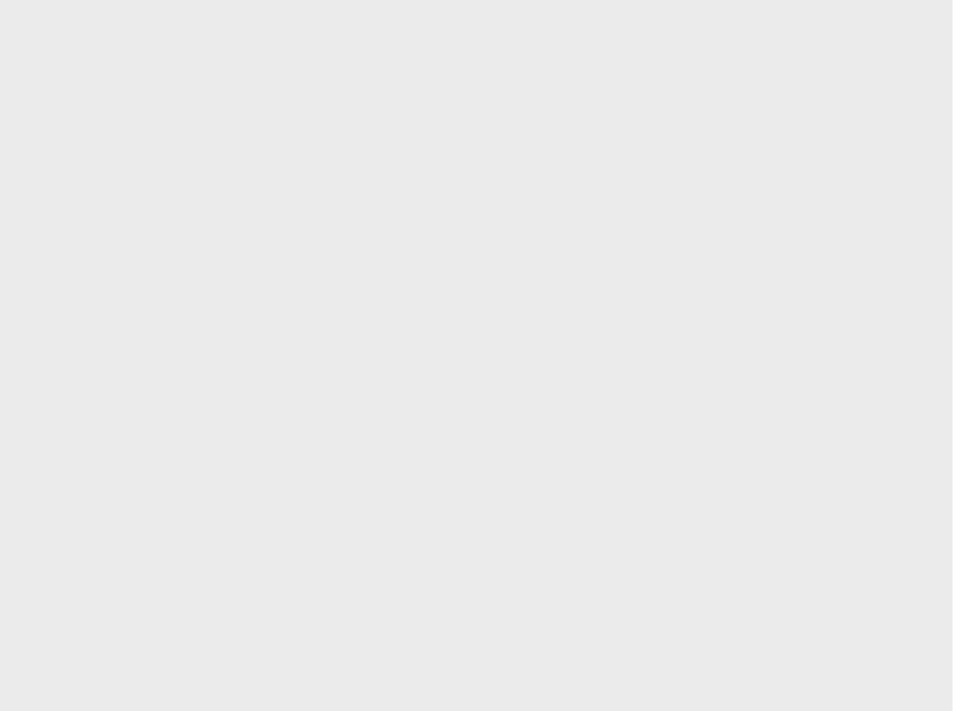
- geom_point
- geom_bar
- geom_histogram
- geom_boxplot

```
[6]: (
         ggplot(df)  # The data to use
         + aes(x="x", y="y3")  # The variables to use
         + geom_line()  # The geometric objects to use
         )
```



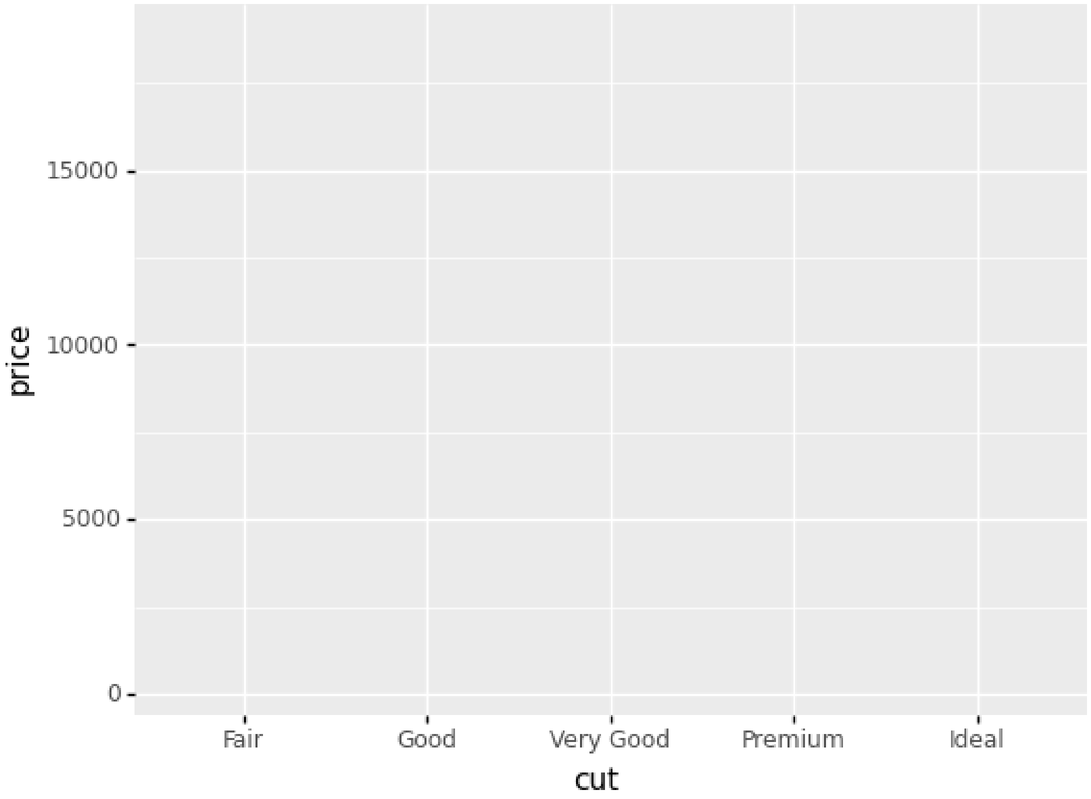
```
[6]: <ggplot: (305942001)>
```

```
[7]: # Data

     ggplot(diamonds)
```



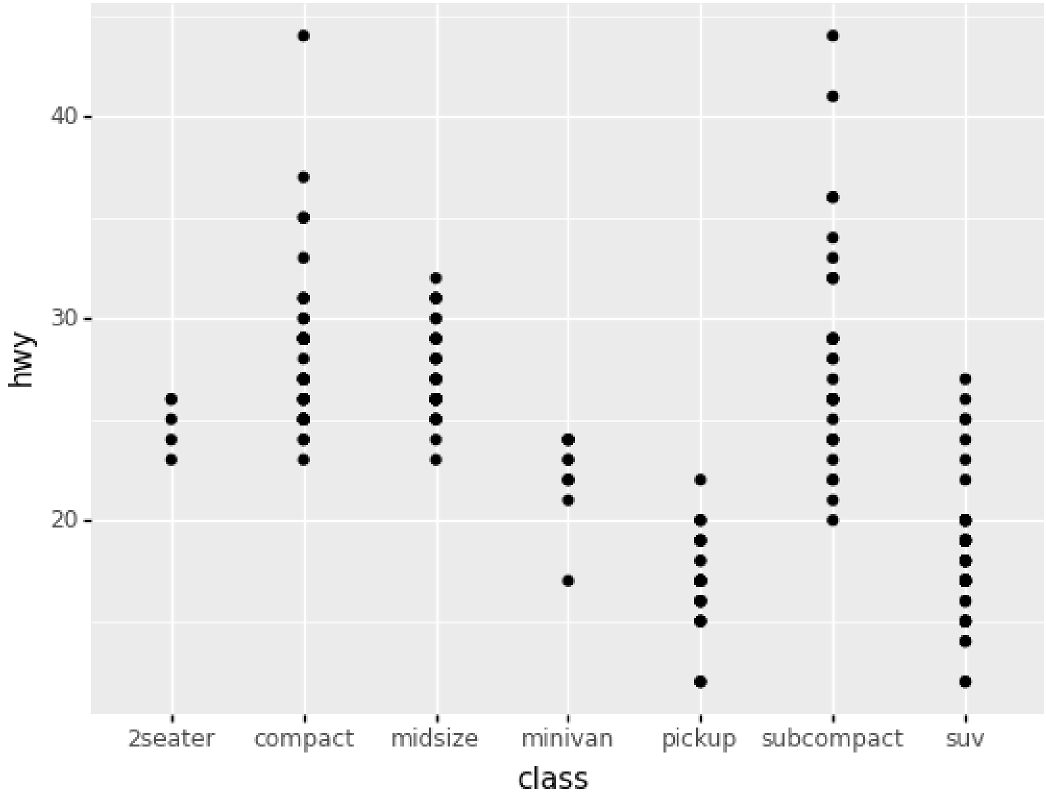
```
[7]: <ggplot: (305947320)>
```

```
[8]: # Data & aesthetics

     ggplot(diamonds) + aes(x="cut", y="price")
```



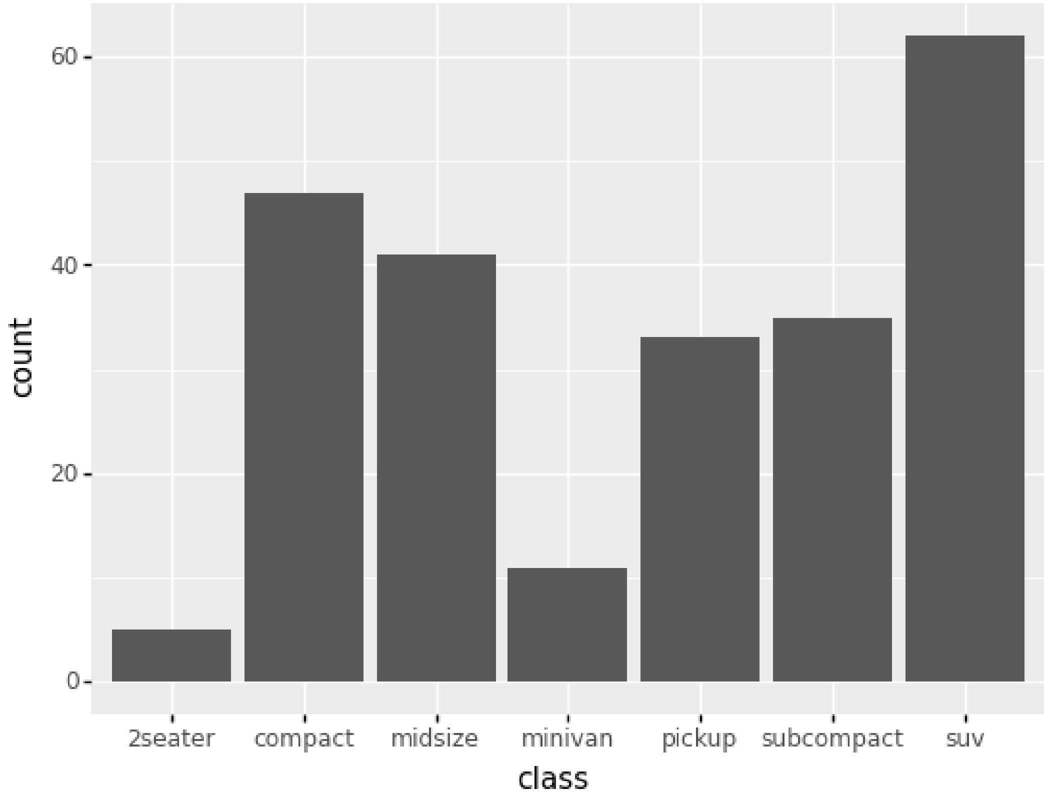
```
[8]: <ggplot: (305998731)>
```

```
[9]: # Data, aesthetics & geometries

     ggplot(mpg) + aes(x="class", y="hwy") + geom_point()
```



```
[9]: <ggplot: (306024873)>
```

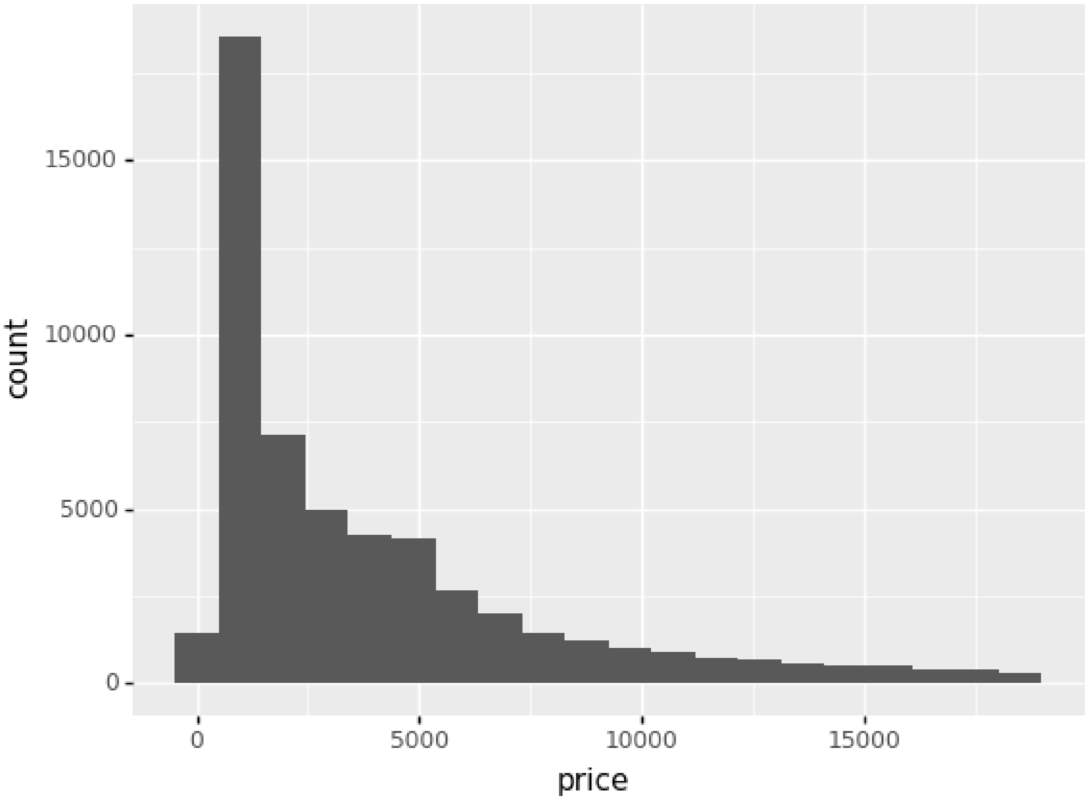
```
[10]: ggplot(mpg) + aes(x="class") + geom_bar()
```



[10]: <ggplot: (306039700)>

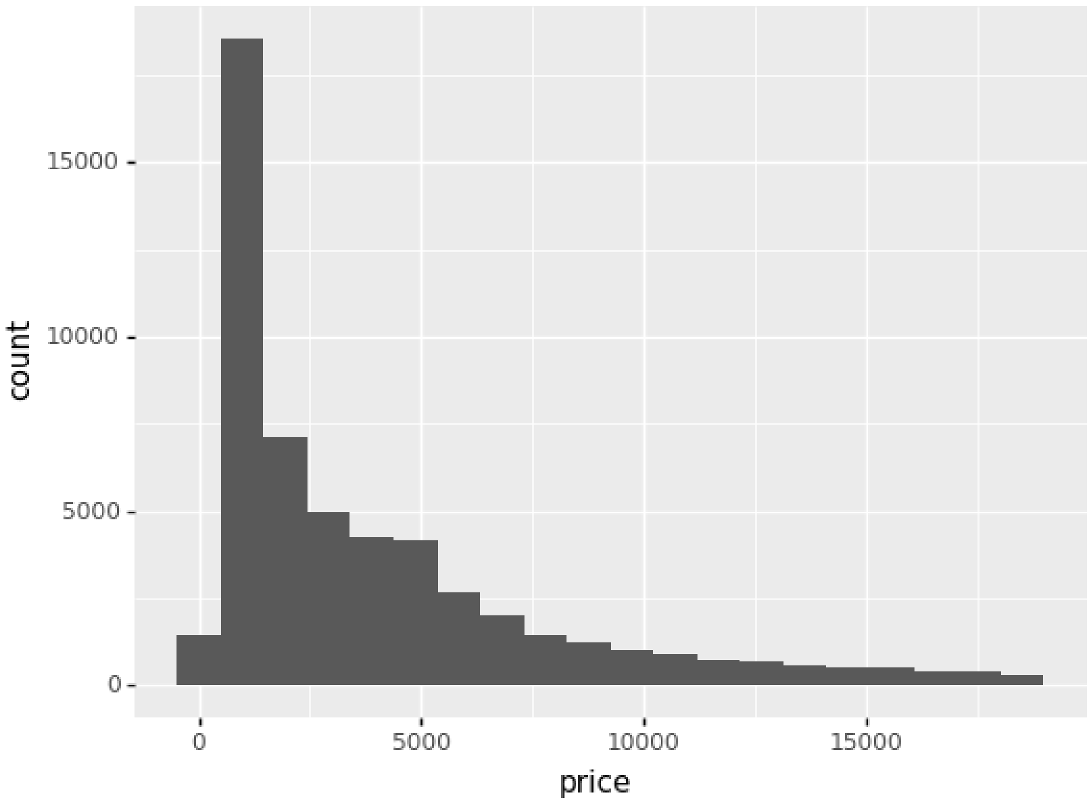
1.3.3 Adding statistics

[11]:
```
ggplot(diamonds) + aes(x="price") + stat_bin(bins=20) + geom_bar()
```



```
[11]: <ggplot: (306111338)>
```

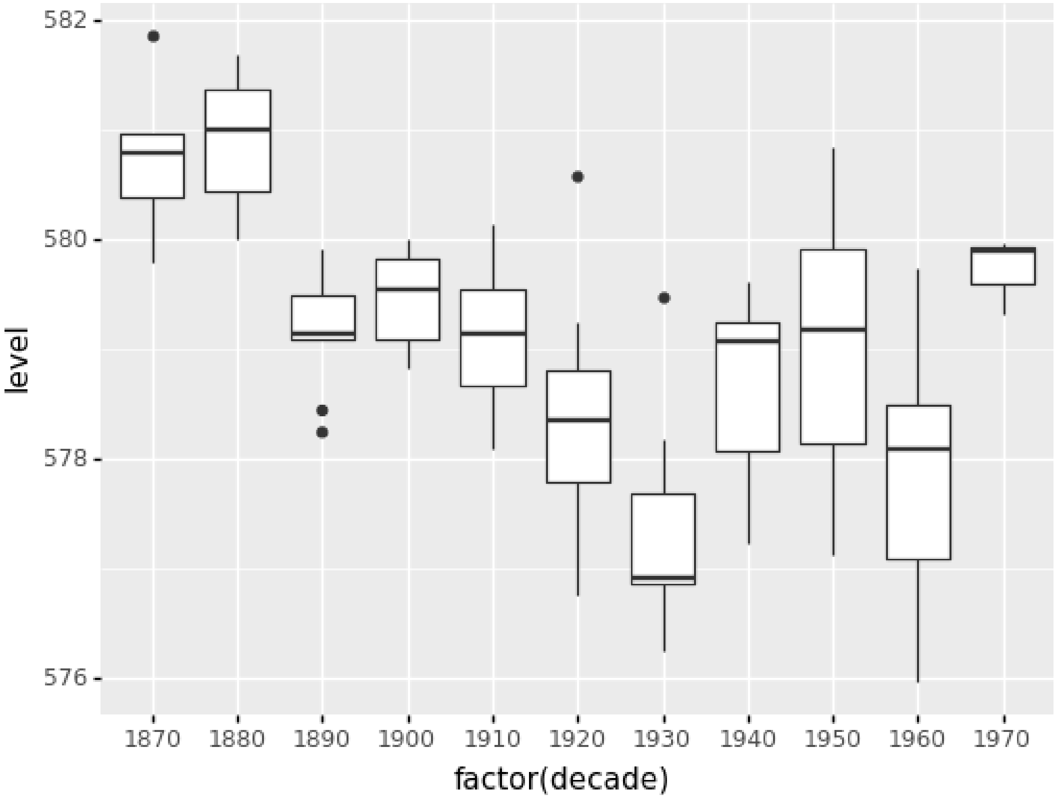
```
[12]: ggplot(diamonds) + aes(x="price") + geom_histogram(bins=20)
```



[12]: <ggplot: (311073009)>

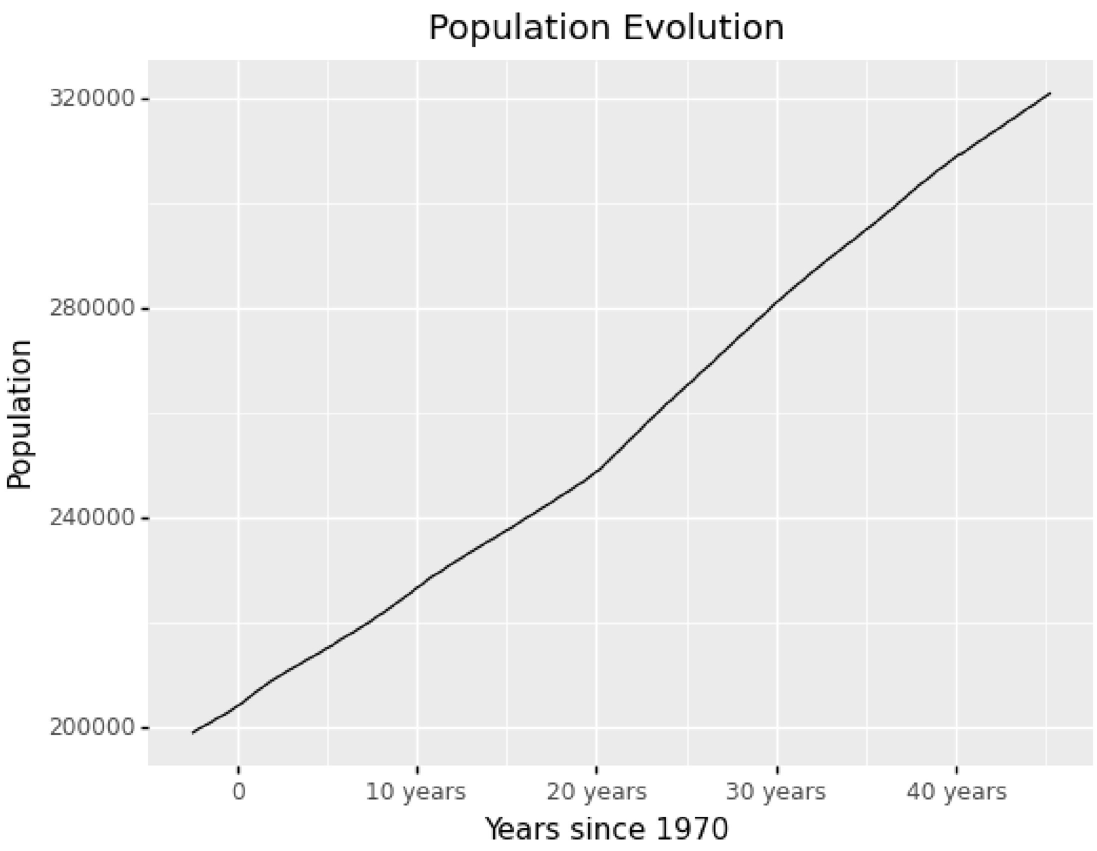
[13]:

```
# This import is here to show that additional geoms, stats, etc. can be imported when needed.

from plotnine import ggplot, aes, geom_boxplot

(
  ggplot(huron)
  + aes(x="factor(decade)", y="level")
  + geom_boxplot()
)
```



```
[13]: <ggplot: (305983594)>
```

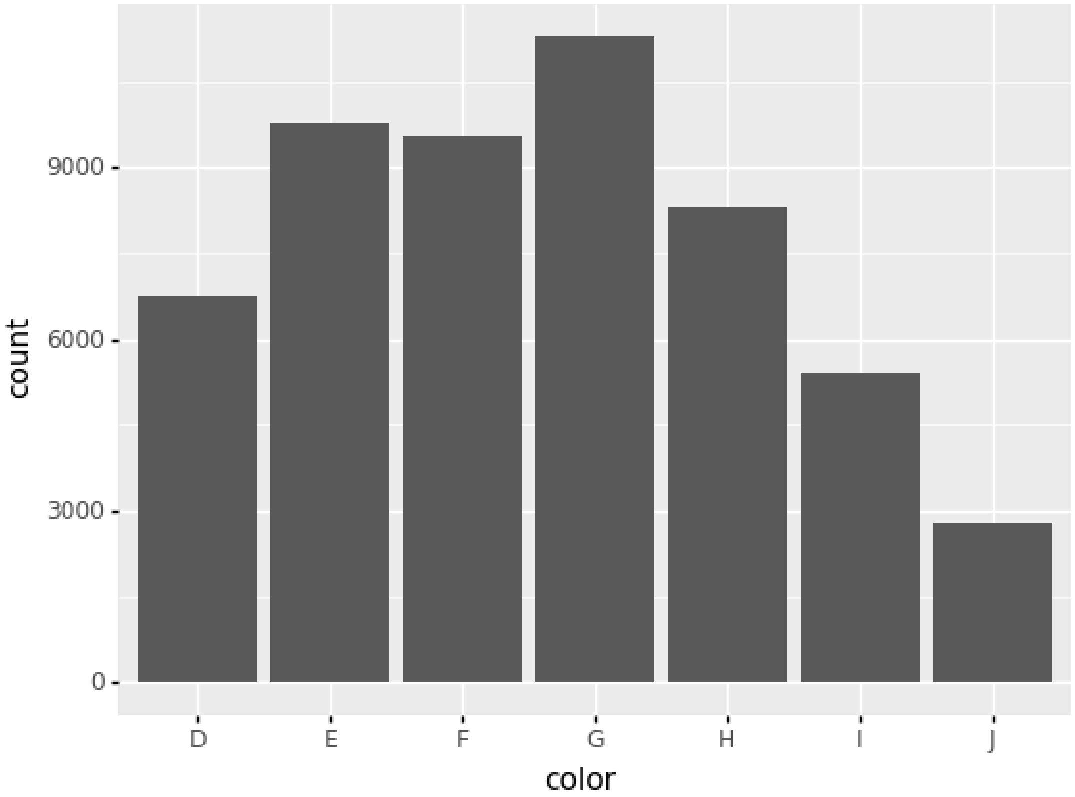
```
[14]: from plotnine import ggplot, aes, scale_x_timedelta, labs, geom_line

      (
          ggplot(economics)
          + aes(x="date", y="pop")
          + scale_x_timedelta(name="Years since 1970")
          + labs(title="Population Evolution", y="Population")
          + geom_line()
      )
```



```
[14]: <ggplot: (306011633)>
```

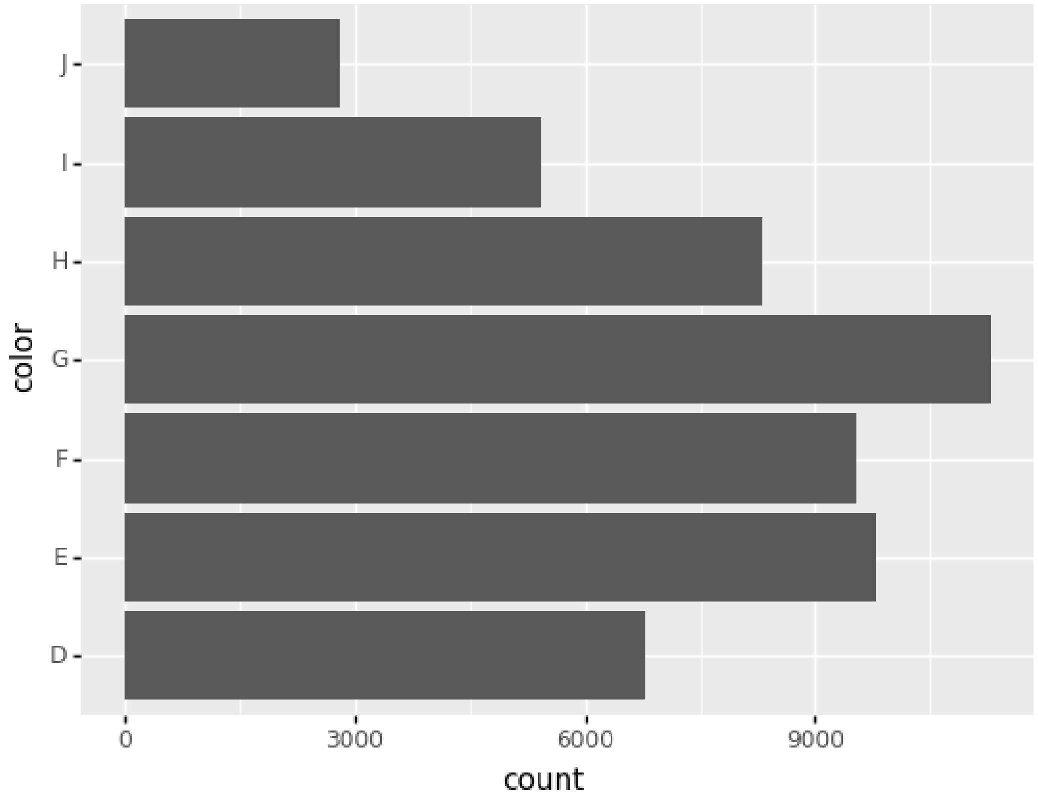
1.3.4 Coordinates

```
[15]: # Default coordinates are used.

      ggplot(diamonds) + aes(x="color") + geom_bar()
```



```
[15]: <ggplot: (305962412)>
```

```
[16]: from plotnine import ggplot, aes, geom_bar, coord_flip

      ggplot(diamonds)  + aes(x="color") + geom_bar() + coord_flip()
```



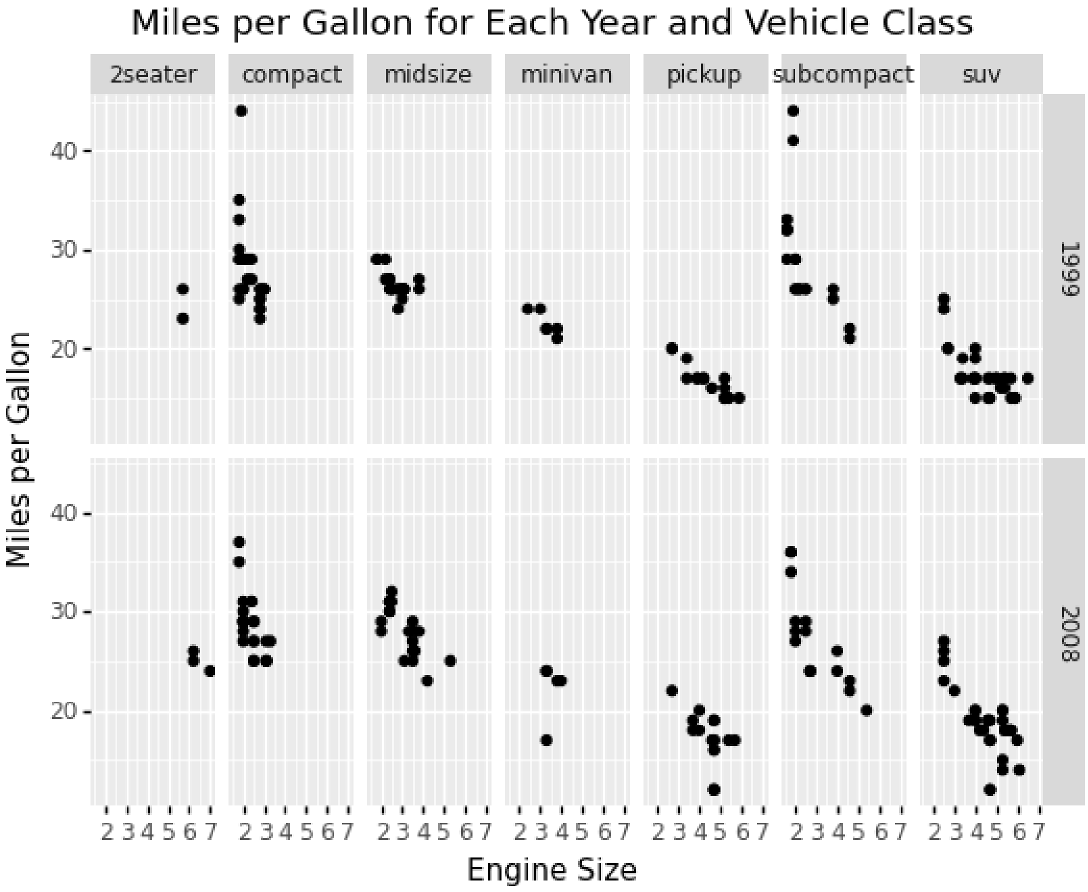
```
[16]: <ggplot: (308254332)>
```

1.3.5 Facets

```
[17]: from plotnine import facet_grid, labs

(
    ggplot(mpg)
    + facet_grid(facets="year~class")
    + aes(x="displ", y="hwy")
    + labs(
        x="Engine Size",
        y="Miles per Gallon",
        title="Miles per Gallon for Each Year and Vehicle Class",
    )
    + geom_point()

)
```

```
/Users/jamescody/opt/anaconda3/envs/CDC/lib/python3.9/site-
packages/plotnine/utils.py:371: FutureWarning: The frame.append method is
```

```
deprecated and will be removed from pandas in a future version. Use
pandas.concat instead.
```



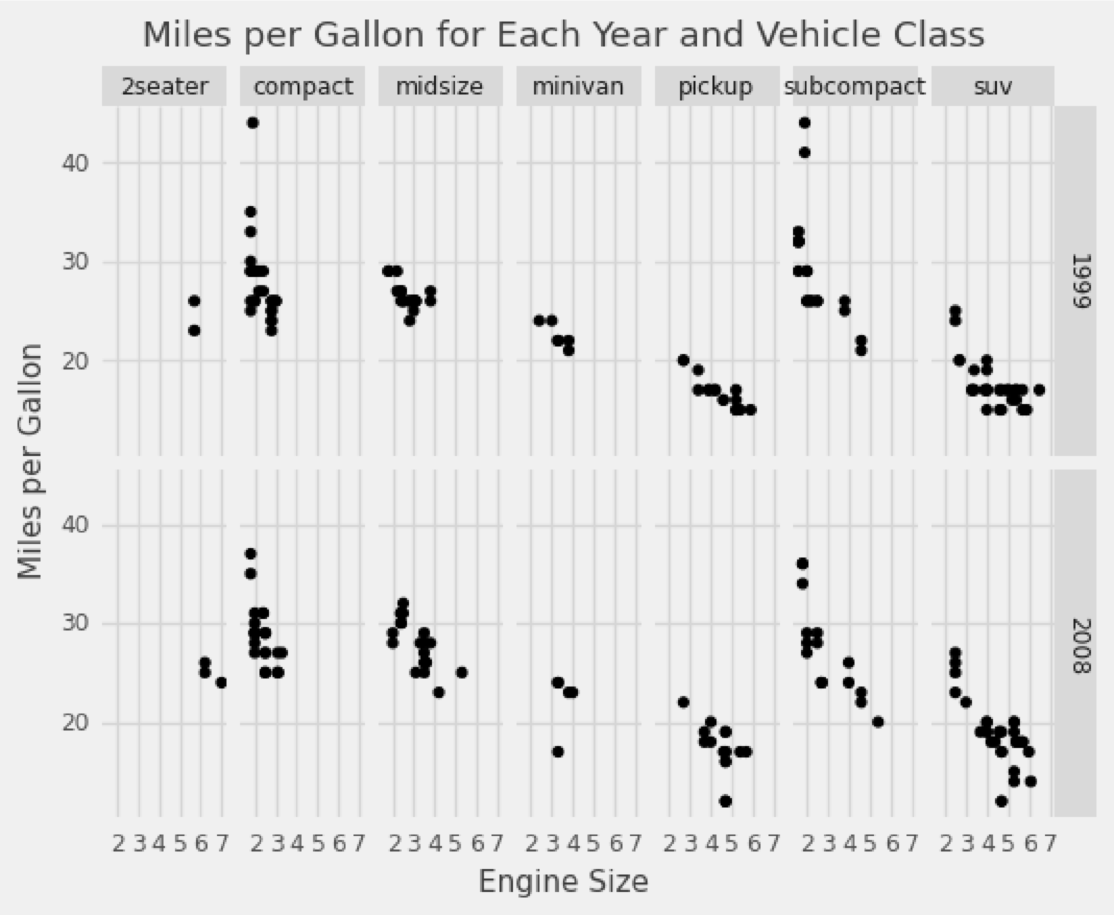
[17]: `<ggplot: (308362578)>`

1.3.6 Themes

[18]:

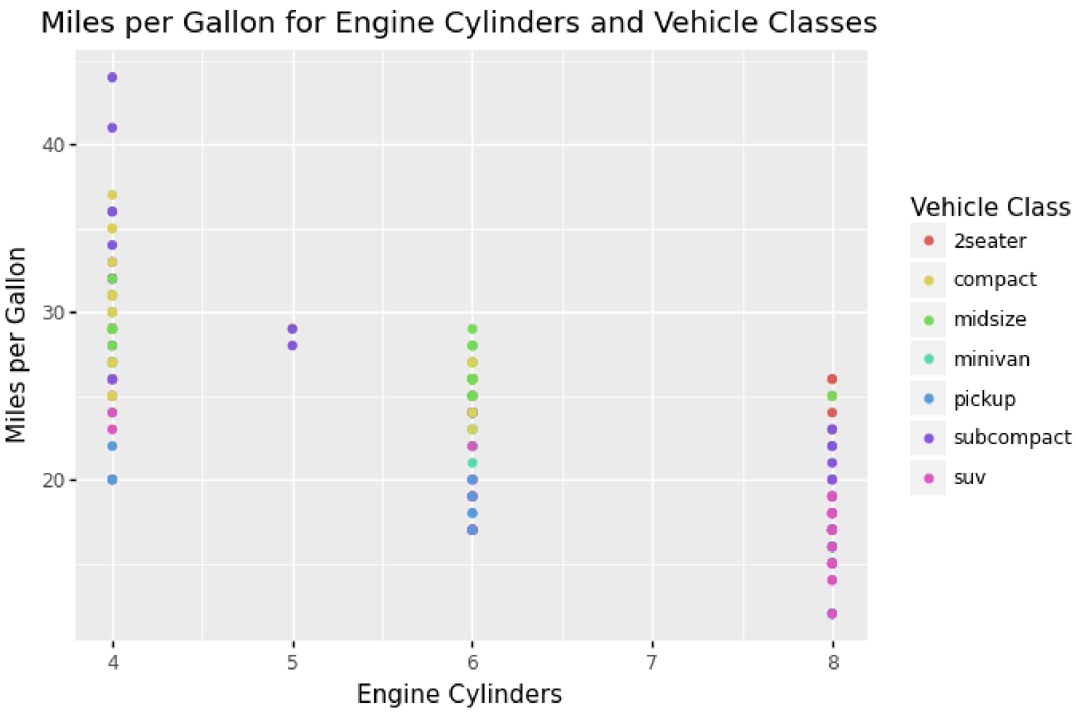
```
(
    ggplot(mpg)
    + facet_grid(facets="year~class")
    + aes(x="displ", y="hwy")
    + labs(
        x="Engine Size",
        y="Miles per Gallon",
        title="Miles per Gallon for Each Year and Vehicle Class",
    )
    + geom_point()
    + theme_538()
)
```

```
/Users/jamescody/opt/anaconda3/envs/CDC/lib/python3.9/site-
packages/plotnine/utils.py:371: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use
pandas.concat instead.
```



[18]: `<ggplot: (306056876)>`

[19]:
```
(
    ggplot(mpg)
    + aes(x="cyl", y="hwy", color="class")
    + labs(
        x="Engine Cylinders",
        y="Miles per Gallon",
        color="Vehicle Class",
        title="Miles per Gallon for Engine Cylinders and Vehicle Classes",
    )
    + geom_point()
)
```



```
[19]: <ggplot: (306969411)>
```

1.4 Saving a plot to a file

```
[20]: myPlot = ggplot(economics) + aes(x="date", y="pop") + geom_line()
      myPlot.save("myplot.png", dpi=600)
```

```
/Users/jamescody/opt/anaconda3/envs/CDC/lib/python3.9/site-
packages/plotnine/ggplot.py:719: PlotnineWarning: Saving 6.4 x 4.8 in image.
/Users/jamescody/opt/anaconda3/envs/CDC/lib/python3.9/site-
packages/plotnine/ggplot.py:722: PlotnineWarning: Filename: myplot.png
```

1.5 ggplot Exercise

1. Use the built-in dataset 'Midwest.
2. Create a simple bar chart showing the number of rows by state.
3. Create a visualization showing a plot for each state (along the y-axis). On that plot show a jitter (a version of a scatter plot) plot of the number of adults by percent professional for each country

```
[21]: # Put bar chart here
```

```
[22]: # Put state plots here
```

2 Altair

Altair is a Python library designed for statistical visualizations. It is considered a declarative API rather than the more common imperative API. The claim from Altair is that it allows developers to 'declare' what they want to do vs the imperative API in which is focused on how to do it. Altair is constructed around the use of pandas dataframes.

https://altair-viz.github.io/

```
[23]: import altair as alt
      from plotnine.data import midwest
```

```
[24]: midwest.head()
```

```
[24]:    PID     county state   area  poptotal   popdensity  popwhite  popblack  \
      0  561      ADAMS    IL  0.052     66090  1270.961540     63917      1702
      1  562  ALEXANDER    IL  0.014     10626   759.000000      7054      3496
      2  563       BOND    IL  0.022     14991   681.409091     14477       429
      3  564      BOONE    IL  0.017     30806  1812.117650     29344       127
      4  565      BROWN    IL  0.018      5836   324.222222      5264       547

         popamerindian  popasian  …  percollege  percprof  poppovertyknown  \
      0             98       249  …   19.631392  4.355859            63628
      1             19        48  …   11.243308  2.870315            10529
      2             35        16  …   17.033819  4.488572            14235
      3             46       150  …   17.278954  4.197800            30337
      4             14         5  …   14.475999  3.367680             4815

         percpovertyknown  percbelowpoverty  percchildbelowpovert  percadultpoverty  \
      0         96.274777         13.151443             18.011717         11.009776
      1         99.087145         32.244278             45.826514         27.385647
      2         94.956974         12.068844             14.036061         10.852090
      3         98.477569          7.209019             11.179536          5.536013
      4         82.505140         13.520249             13.022889         11.143211

         percelderlypoverty  inmetro category
      0           12.443812        0      AAR
      1           25.228976        0      LHR
      2           12.697410        0      AAR
      3            6.217047        1      ALU
      4           19.200000        0      AAR

      [5 rows x 28 columns]
```

```
[25]: alt.Chart(midwest).mark_bar().encode(
          alt.X('state'),
          y='count()'
      )
```

[25]:
```
alt.Chart(…)
```

[26]:
```python
IL = midwest[midwest['state'] == 'IL']
```

[27]:
```python
alt.Chart(IL).mark_point().encode(
    alt.X('percollege'), # percent college
    alt.Y('percprof')  # percent professional
)
```

[27]:
```
alt.Chart(…)
```

[28]:
```python
alt.Chart(IL).mark_point(filled=False).encode(
    alt.X('percollege'), # percent college
    alt.Y('percprof'),  # percent professional
    alt.Size('poptotal')
)
```

[28]:
```
alt.Chart(…)
```

[29]:
```python
alt.Chart(IL).mark_point(filled=True).encode(
    alt.X('percollege'), # percent college
    alt.Y('percprof'),  # percent professional
    alt.Size('poptotal'),
    alt.Color('popdensity'),
    alt.OpacityValue(0.7)
)
```

[29]:
```
alt.Chart(…)
```

[30]:
```python
alt.Chart(IL).mark_point(filled=True).encode(
    alt.X('percollege'), # percent college
    alt.Y('percprof'),  # percent professional
    alt.Size('poptotal'),
    alt.Color('popdensity'),
    alt.OpacityValue(0.7),
        tooltip = [alt.Tooltip('county'),
               alt.Tooltip('percwhite'),
               alt.Tooltip('percblack'),
               alt.Tooltip('percother')
              ]
)
```

[30]:
```
alt.Chart(…)
```

[31]:
```python
alt.Chart(IL).mark_point(filled=True).encode(
    alt.X('percollege'), # percent college
    alt.Y('percprof'),  # percent professional
    alt.Size('poptotal'),
    alt.Color('popdensity'),
    alt.OpacityValue(0.7),
        tooltip = [alt.Tooltip('county'),
                   alt.Tooltip('percwhite'),
                   alt.Tooltip('percblack'),
                   alt.Tooltip('percother')
                  ]
).interactive()
```

[31]:
```
alt.Chart(…)
```

2.1 Altair Exercise

- Modify the plot shown above.
- Use the full midwest dataset
- Instead of just circles, change the mark so that each state is represented with a different shape
- Add the state abbreviation to the tooltip
- Try to change the size of the plot

[32]:
```
# Altair exercise here.
```

[ ]:
```

```