



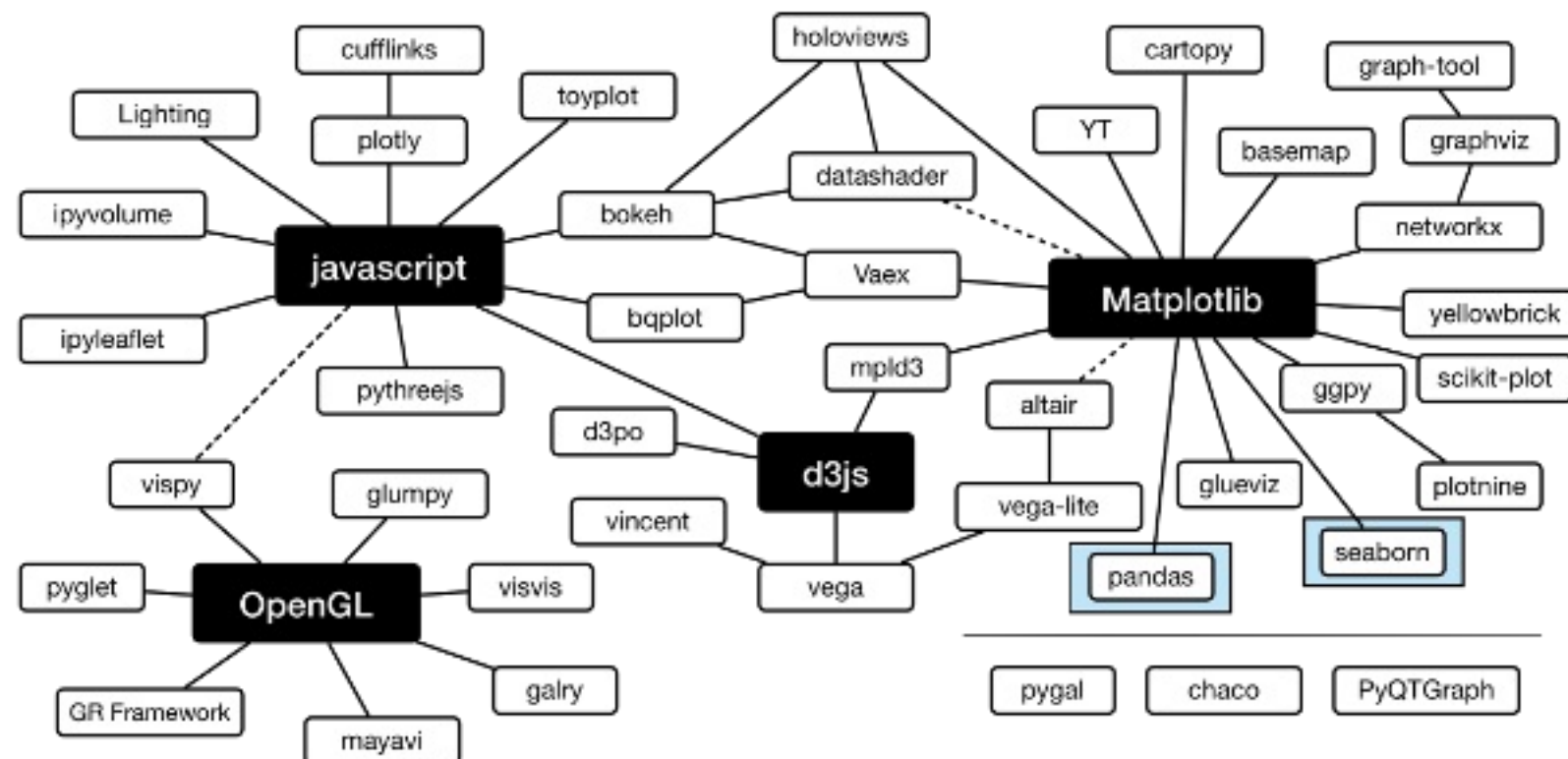
DATA VISUALIZATION WITH SEABORN

# Introduction to Seaborn

Chris Moffitt  
Instructor

# Python Visualization Landscape

- The python visualization landscape is complex and can be overwhelming





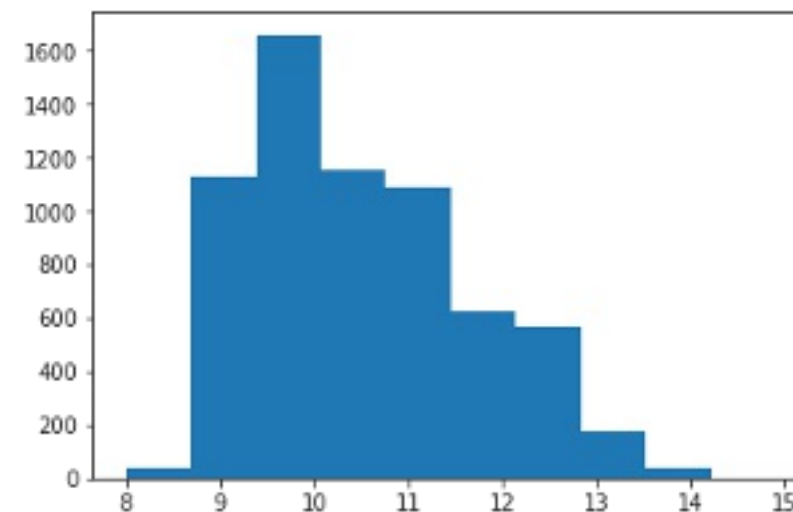
# matplotlib

- matplotlib provides the raw building blocks for Seaborn's visualizations
- It can also be used on its own to plot data

```
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv("wines.csv")

fig, ax = plt.subplots()
ax.hist(df['alcohol'])
```



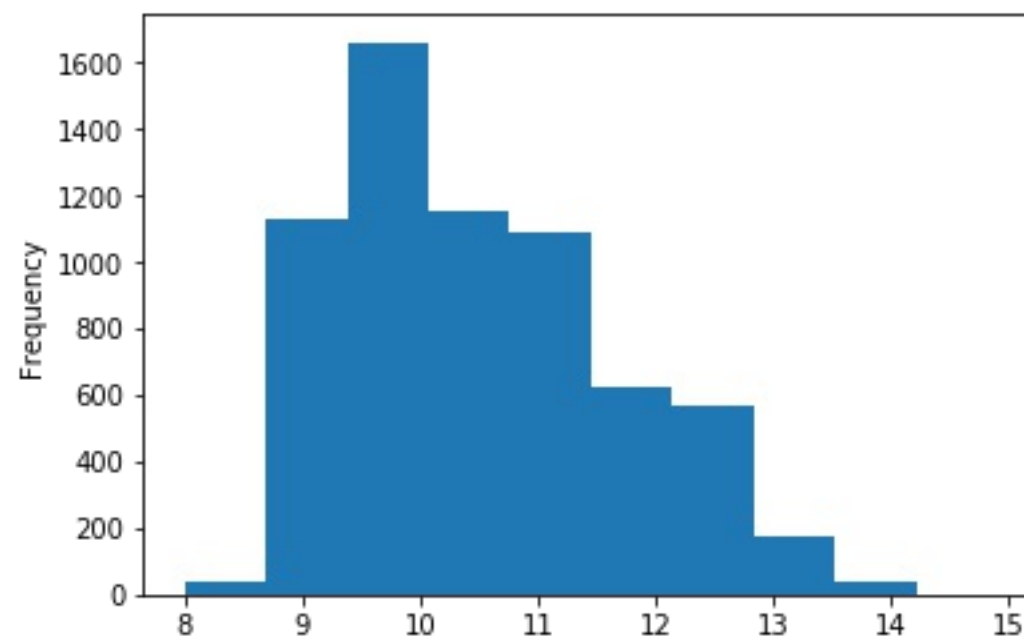


# pandas

- pandas is a foundational library for analyzing data
- It also supports basic plotting capability

```
import pandas as pd

df = pd.read_csv("wines.csv")
df['alcohol'].plot.hist()
```

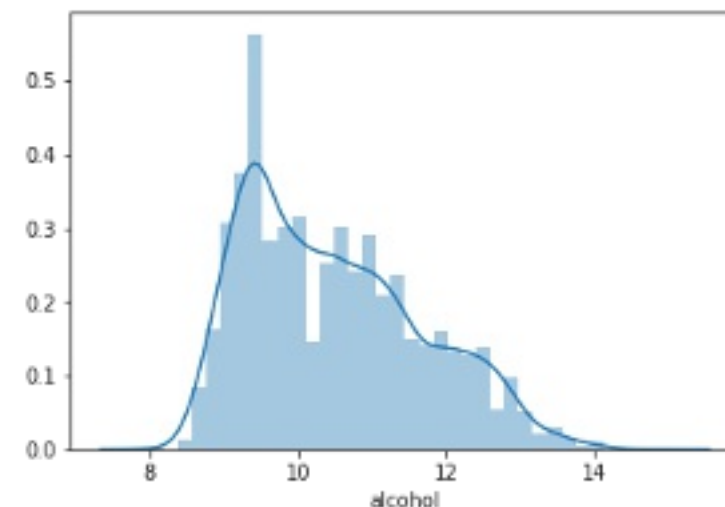




# Seaborn

- Seaborn supports complex visualizations of data
- It is built on matplotlib and works best with pandas' dataframes
- The distplot is similar to the histogram shown in previous examples
- By default, generates a Gaussian Kernel Density Estimate (KDE)

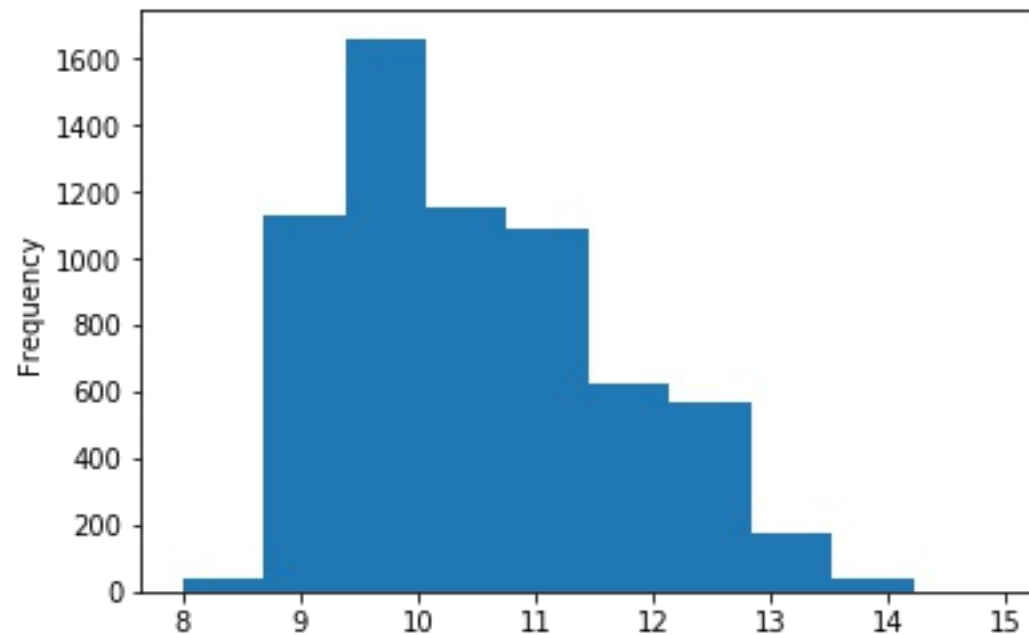
```
import seaborn as sns  
  
sns.distplot(df['alcohol'])
```



# Histogram vs. Distplot

- Pandas histogram

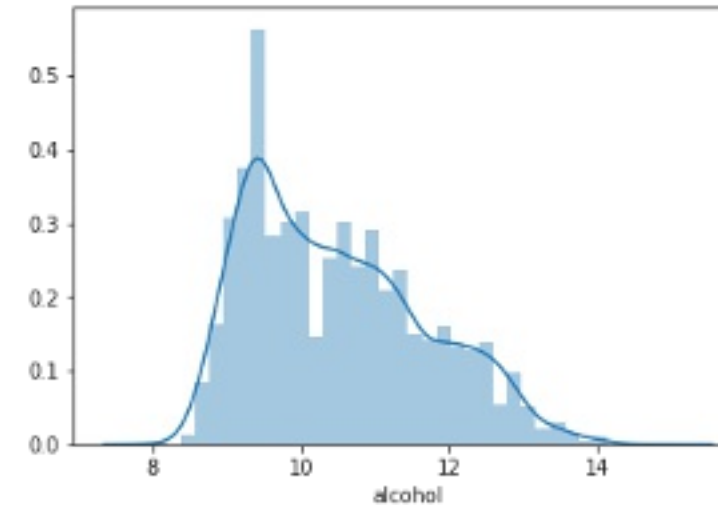
```
df['alcohol'].plot.hist()
```



- Actual frequency of observations
- No automatic labels
- Wide bins

- Seaborn distplot

```
sns.distplot(df['alcohol'])
```



- Automatic label on x axis
- Muted color palette
- KDE plot
- Narrow bins



## DATA VISUALIZATION WITH SEABORN

**Let's practice!**



DATA VISUALIZATION WITH SEABORN

# Using the distribution plot

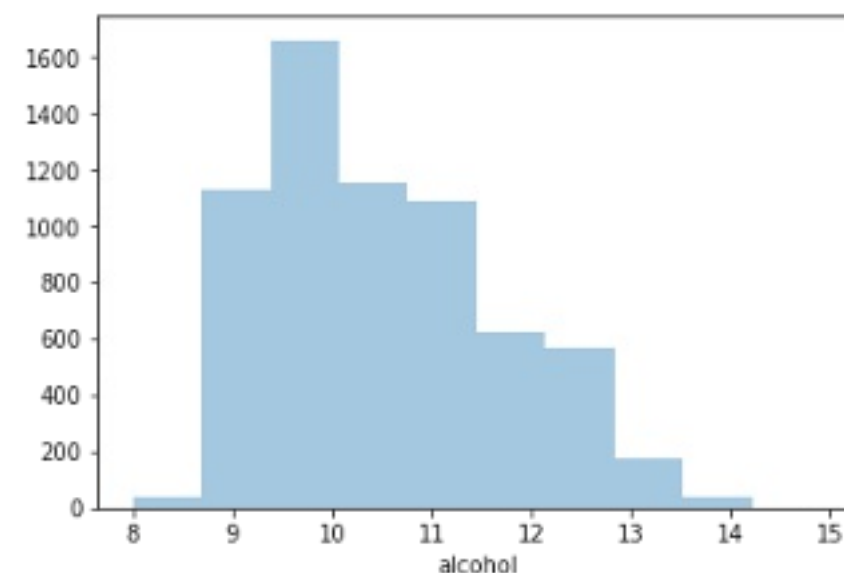




# Creating a histogram

- Distplot function has multiple optional arguments
- In order to plot a simple histogram, you can disable the kde and specify the number of bins to use

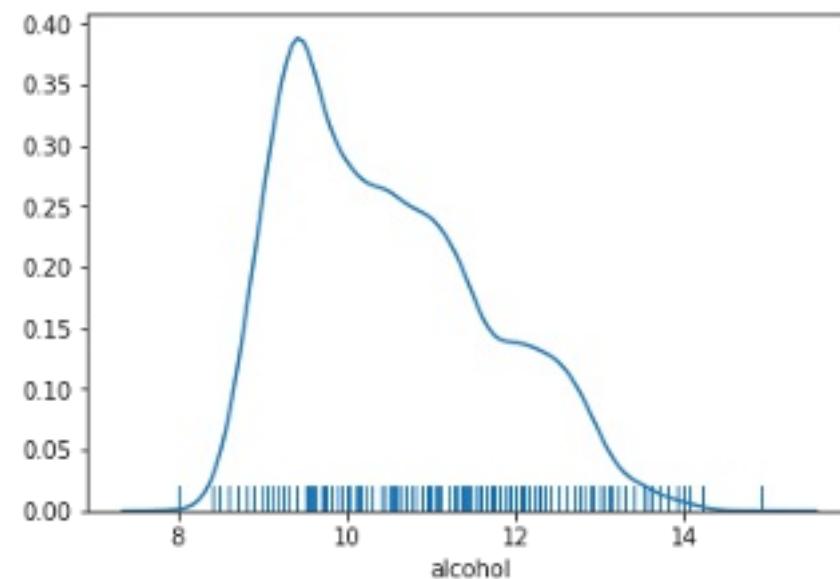
```
sns.distplot(df['alcohol'],  
             kde=False,  
             bins=10)
```



# Alternative data distributions

- A rug plot is an alternative way to view the distribution of data
- A kde curve and rug plot can be combined

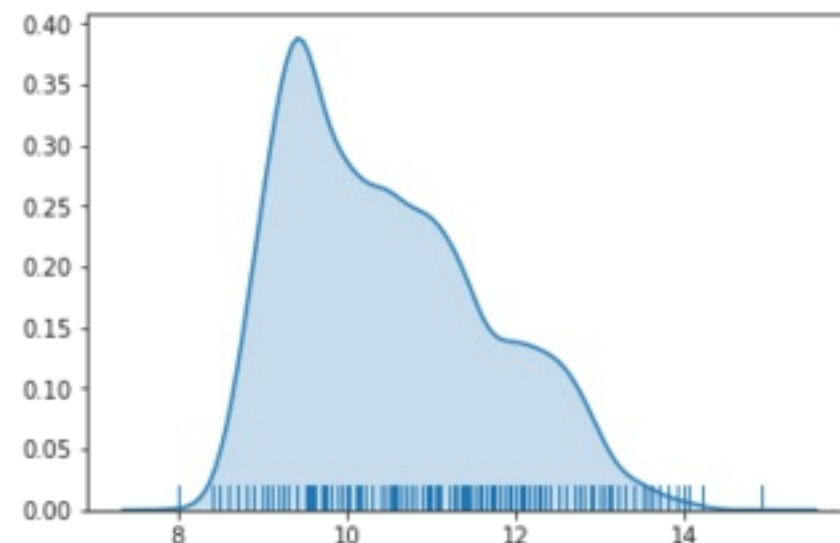
```
sns.distplot(df_wines['alcohol'],  
             hist=False,  
             rug=True)
```



# Further Customizations

- The `distplot` function uses several functions including `kdeplot` and `rugplot`
- It is possible to further customize a plot by passing arguments to the underlying function

```
sns.distplot(df_wines['alcohol'],  
             hist=False,  
             rug=True,  
             kde_kws={'shade':True})
```





## DATA VISUALIZATION WITH SEABORN

**Let's practice!**



DATA VISUALIZATION WITH SEABORN

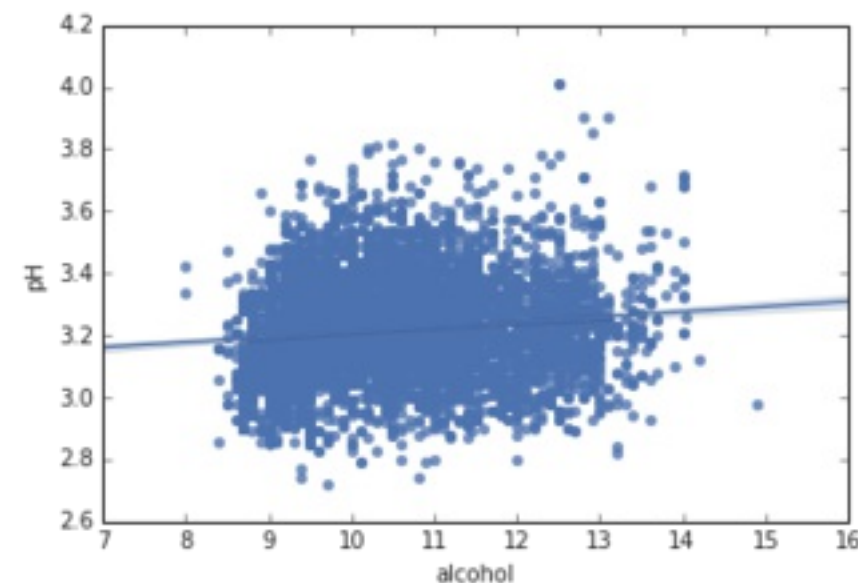
# Regression Plots in Seaborn

Chris Moffitt  
Instructor

# Introduction to regplot

- The regplot function generates a scatter plot with a regression line
- Usage is similar to the distplot
- The data and x and y variables must be defined

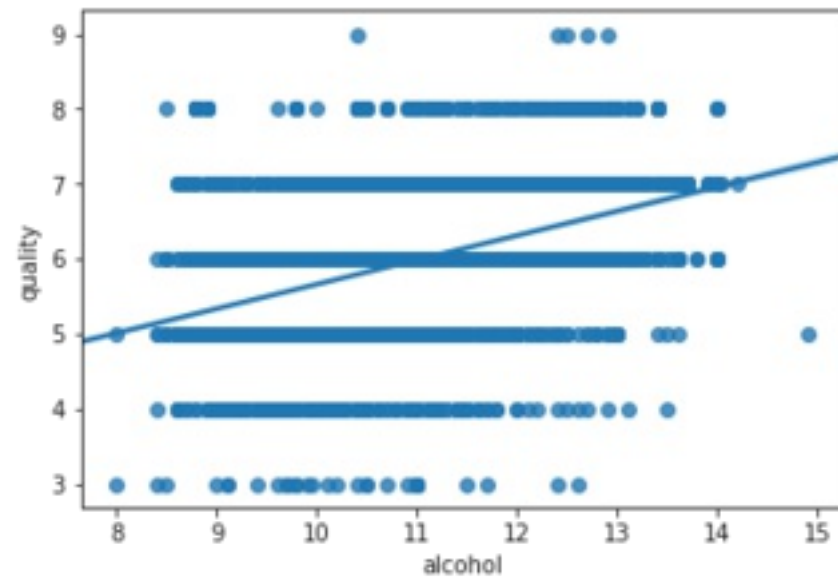
```
sns.regplot(x="alcohol",  
            y="pH",  
            data=df)
```



# Implot() builds on top of the base regplot()

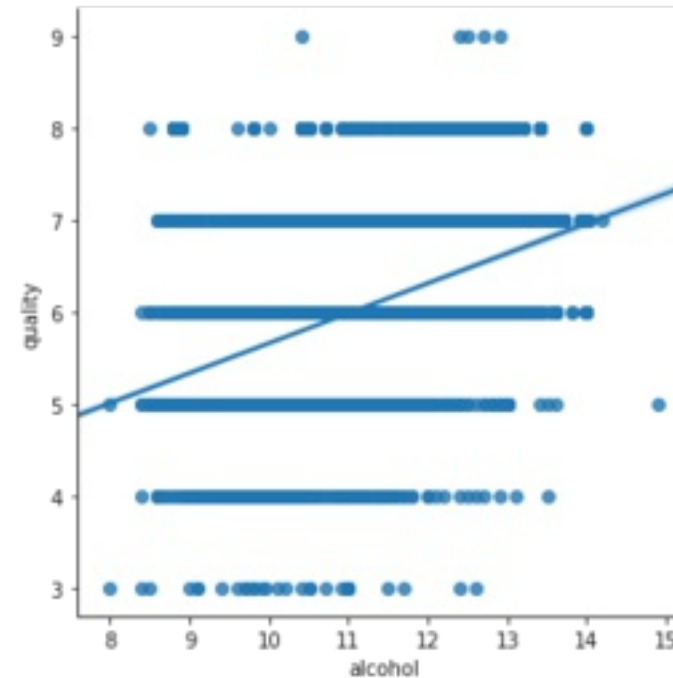
- regplot - low level

```
sns.regplot(x="alcohol",
            y="quality",
            data=df)
```



- Implot - high level

```
sns.lmplot(x="alcohol",
           y="quality",
           data=df)
```

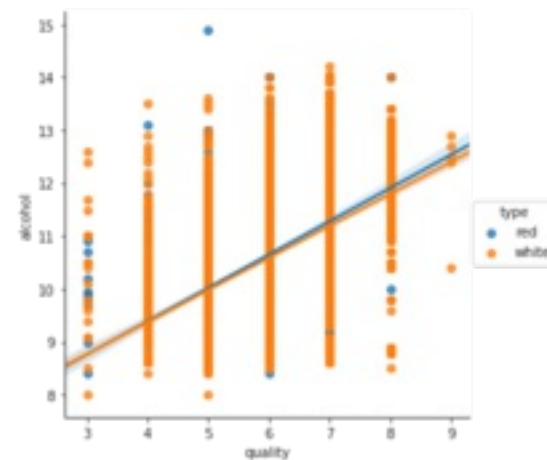




# Implot facetting

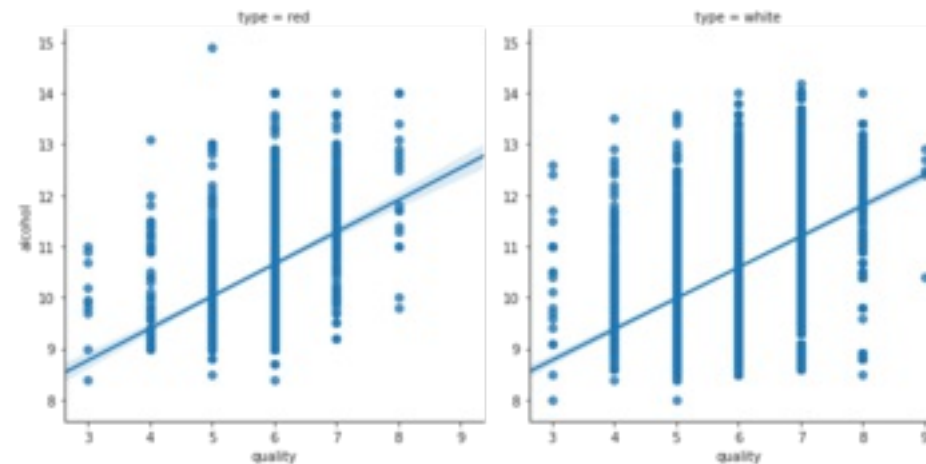
- Organize data by colors (hue)

```
sns.lmplot(x="quality",  
           y="alcohol",  
           data=df,  
           hue="type")
```



- Organize data by columns (col)

```
sns.lmplot(x="quality",  
           y="alcohol",  
           data=df,  
           col="type")
```







## DATA VISUALIZATION WITH SEABORN

**Let's practice!**



DATA VISUALIZATION WITH SEABORN

# Using Seaborn Styles

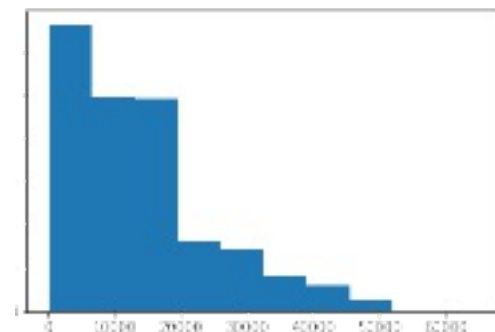
Chris Moffitt  
Instructor

# Setting Styles

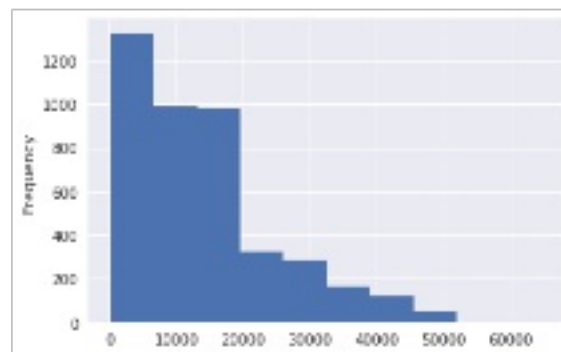
- Seaborn has default configurations that can be applied with `sns.set()`
- These styles can override matplotlib and pandas plots as well

```
sns.set()  
df['Tuition'].plot.hist()
```

Pandas histogram



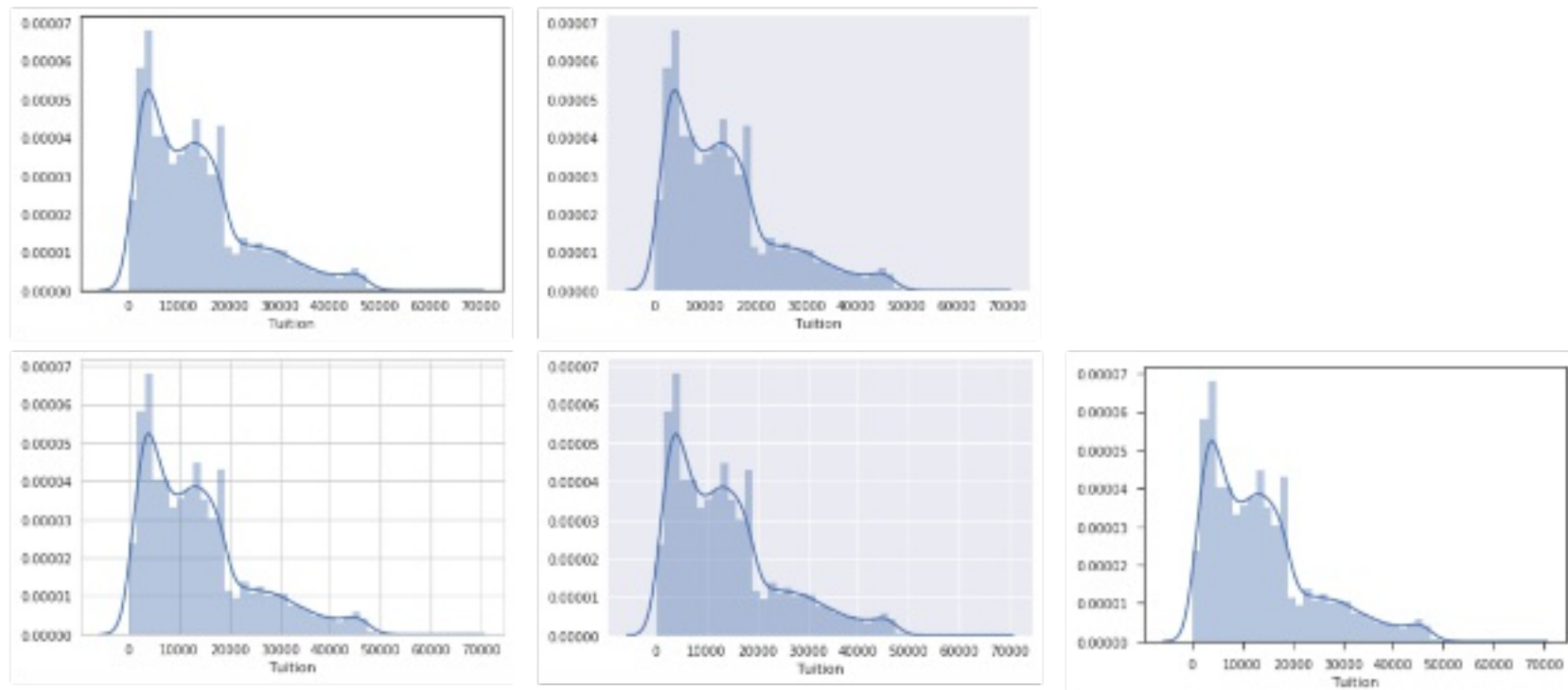
default



Seaborn style

# Theme examples with sns.set\_style()

```
for style in ['white', 'dark', 'whitegrid', 'darkgrid', 'ticks']:  
    sns.set_style(style)  
    sns.distplot(df['Tuition'])  
    plt.show()
```

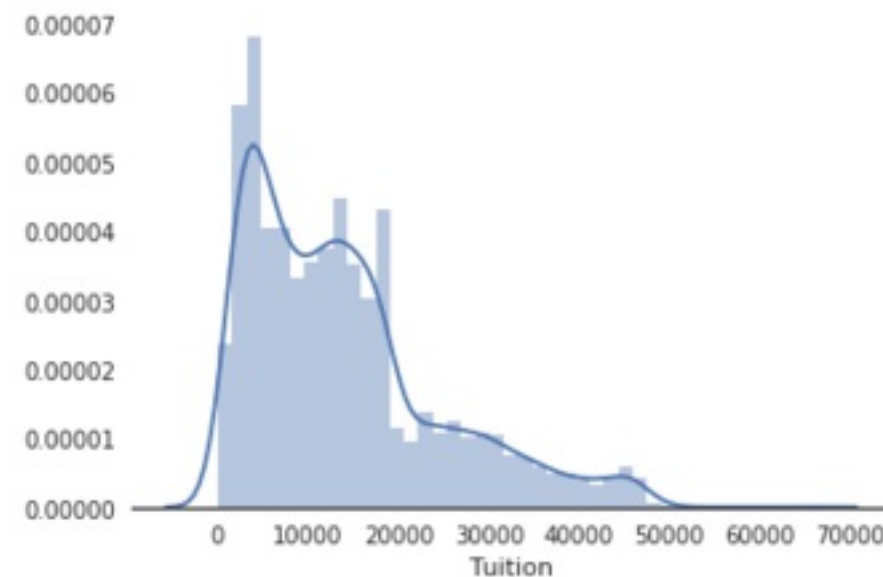




# Removing axes with despine()

- Sometimes plots are improved by removing elements
- Seaborn contains a shortcut for removing the spines of a plot

```
sns.set_style('white')  
sns.distplot(df['Tuition'])  
sns.despine(left=True)
```





## DATA VISUALIZATION WITH SEABORN

**Let's practice!**



DATA VISUALIZATION WITH SEABORN

# Colors in Seaborn

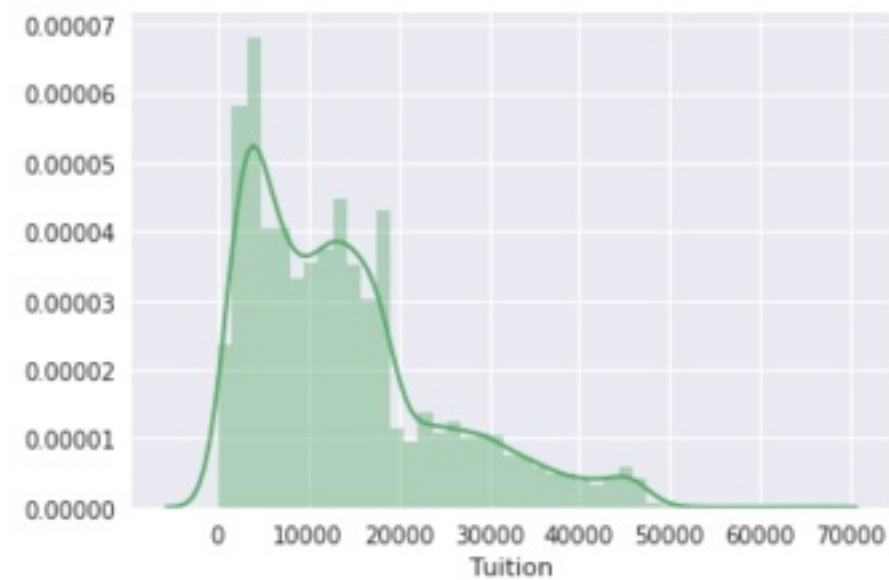
Chris Moffitt  
Instructor



# Defining a color for a plot

- Seaborn supports assigning colors to plots using matplotlib color codes

```
sns.set(color_codes=True)  
sns.distplot(df['Tuition'], color='g')
```



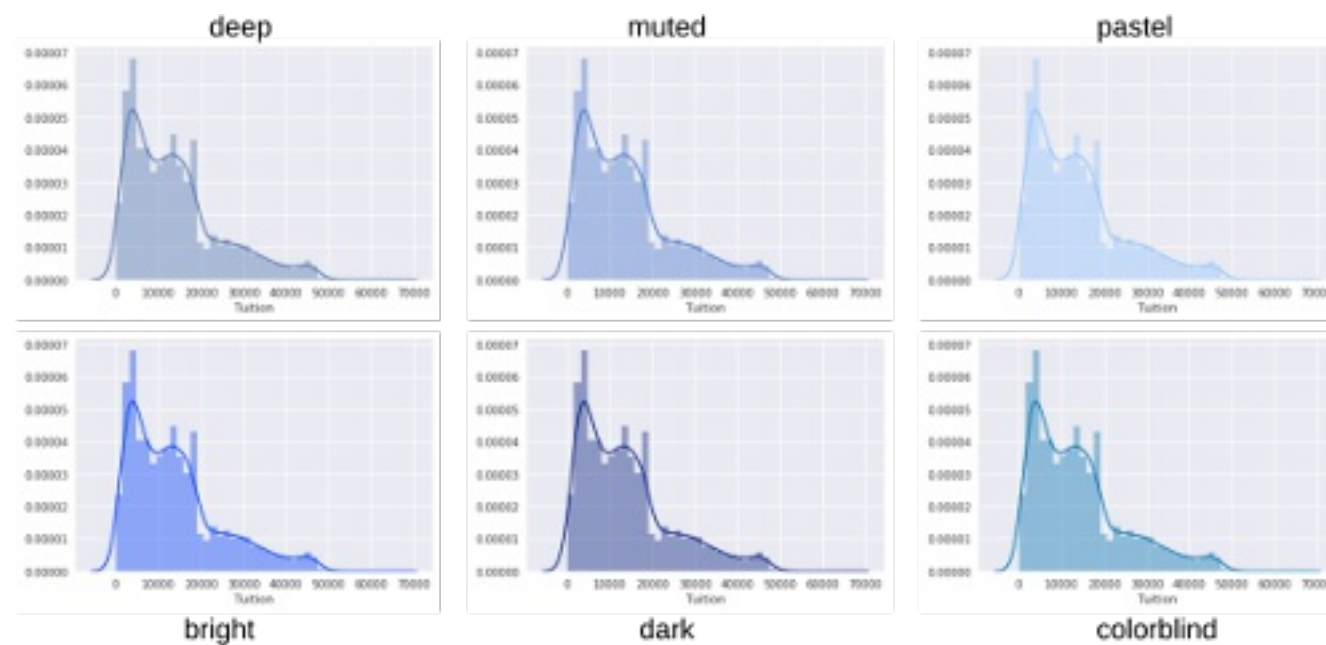




# Palettes

- Seaborn uses the `set_palette()` function to define a palette

```
for p in sns.palettes.SEABORN_PALETTES:  
    sns.set_palette(p)  
    sns.distplot(df['Tuition'])
```



# Displaying Palettes

- `sns.palplot()` function displays a palette
- `sns.color_palette()` returns the current palette

```
for p in sns.palettes.SEABORN_PALETTES:
    sns.set_palette(p)
    sns.palplot(sns.color_palette())
    plt.show()
```





# Defining Custom Palettes

- Circular colors = when the data is not ordered

```
sns.palplot(sns.color_palette("Paired", 12))
```



- Sequential colors = when the data has a consistent range from high to low

```
sns.palplot(sns.color_palette("Blues", 12))
```



- Diverging colors = when both the low and high values are interesting

```
sns.palplot(sns.color_palette("BrBG", 12))
```





## DATA VISUALIZATION WITH SEABORN

**Let's practice!**



DATA VISUALIZATION WITH SEABORN

# Customizing with matplotlib

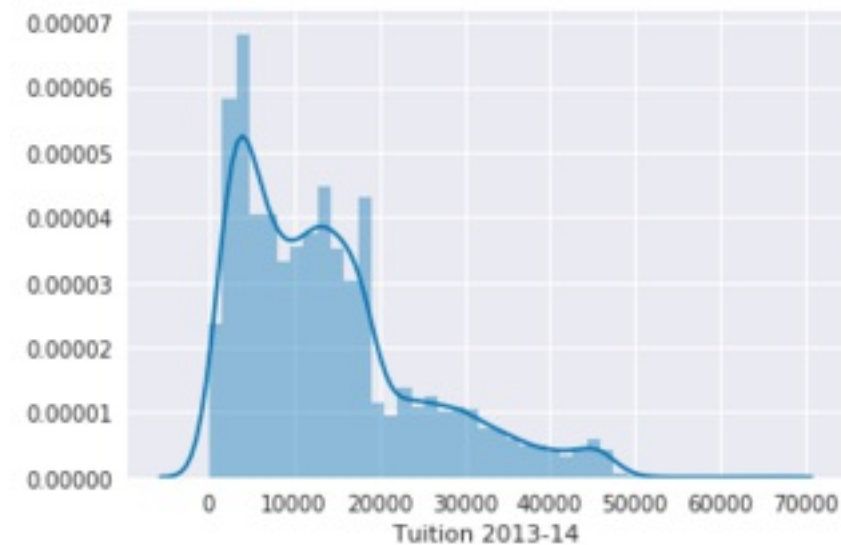
Chris Moffitt  
Instructor



# matplotlib Axes

- Most customization available through matplotlib Axes objects
- Axes can be passed to seaborn functions

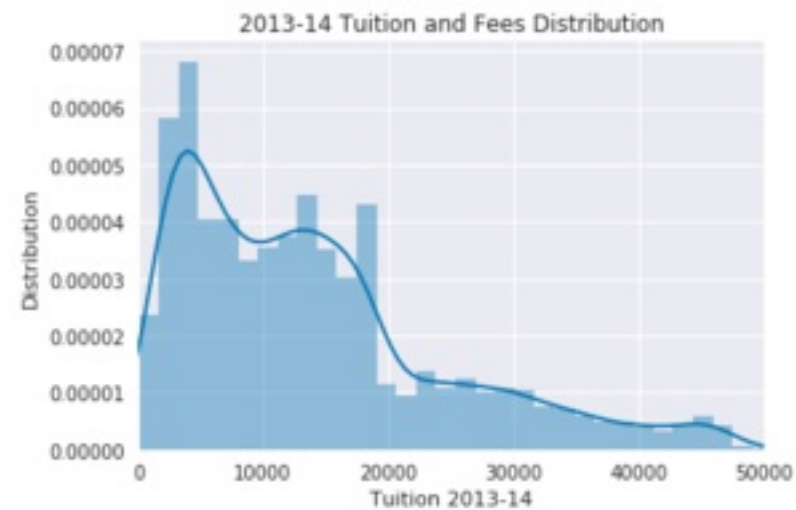
```
fig, ax = plt.subplots()
sns.distplot(df['Tuition'], ax=ax)
ax.set(xlabel="Tuition 2013-14")
```



# Further Customizations

- The axes object supports many common customizations

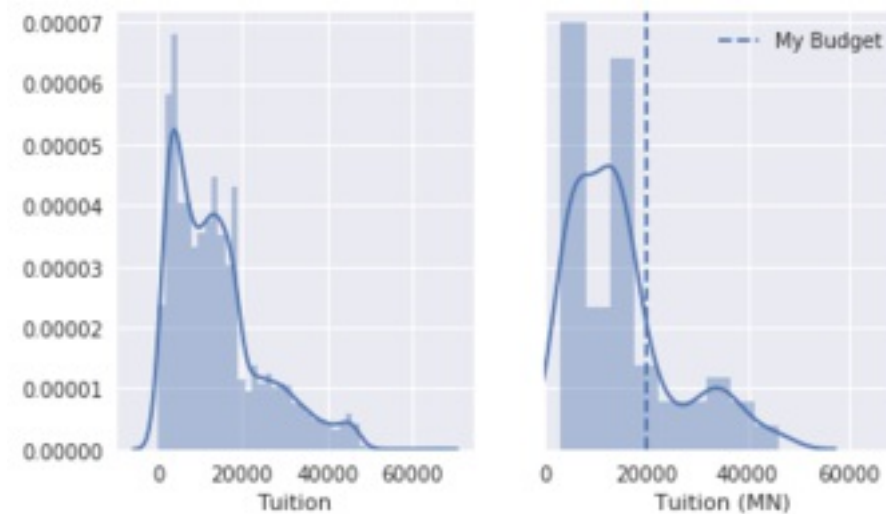
```
fig, ax = plt.subplots()
sns.distplot(df['Tuition'], ax=ax)
ax.set(xlabel="Tuition 2013-14",
       ylabel="Distribution", xlim=(0, 50000),
       title="2013-14 Tuition and Fees Distribution")
```



# Combining Plots

- It is possible to combine and configure multiple plots

```
fig, (ax0, ax1) = plt.subplots(nrows=1,ncols=2,
                               sharey=True, figsize=(7,4))
sns.distplot(df['Tuition'], ax=ax0)
sns.distplot(df.query('State == "MN"')['Tuition'], ax=ax1)
ax1.set(xlabel="Tuition (MN)", xlim=(0, 70000))
ax1.axvline(x=20000, label='My Budget', linestyle='--')
ax1.legend()
```







## DATA VISUALIZATION WITH SEABORN

**Let's practice!**



DATA VISUALIZATION WITH SEABORN

# Categorical Plot Types

Chris Moffitt  
Instructor

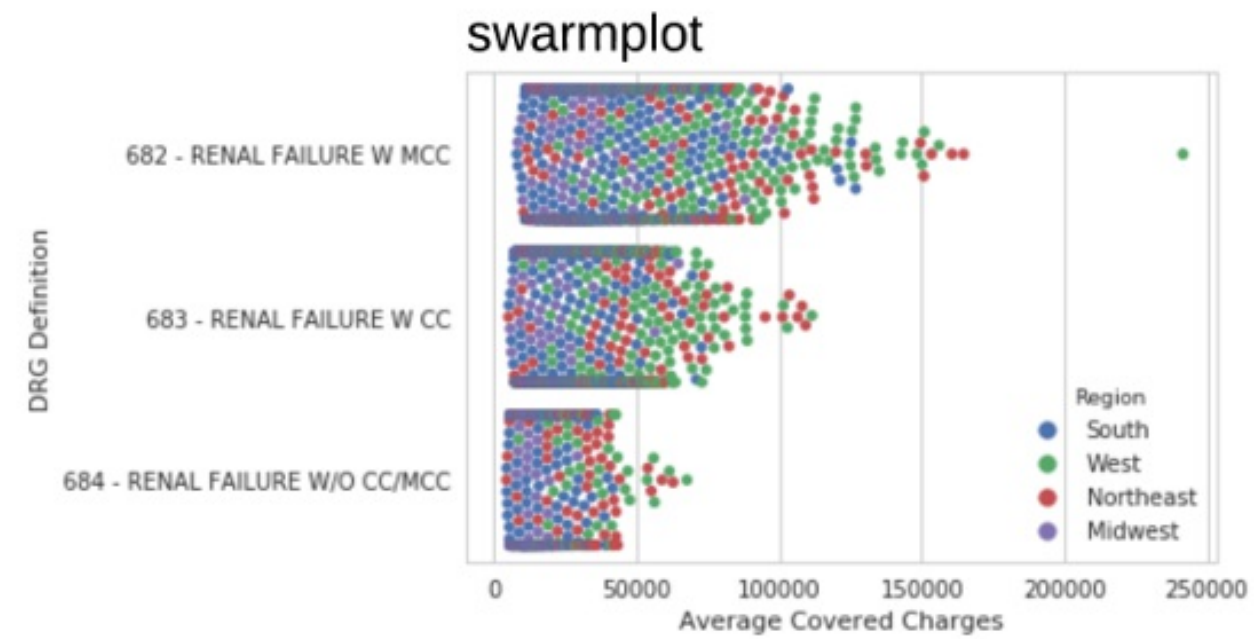
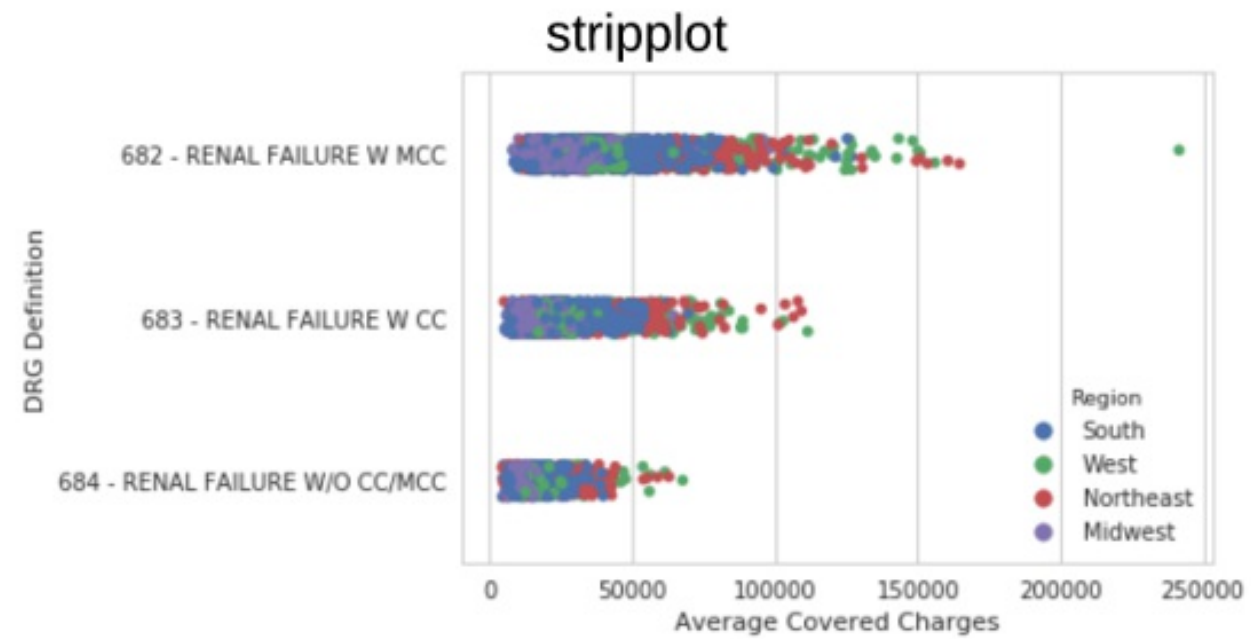


# Categorical Data

- Data which takes on a limited and fixed number of values
- Normally combined with numeric data
- Examples include:
  - Geography (country, state, region)
  - Gender
  - Ethnicity
  - Blood type
  - Eye color

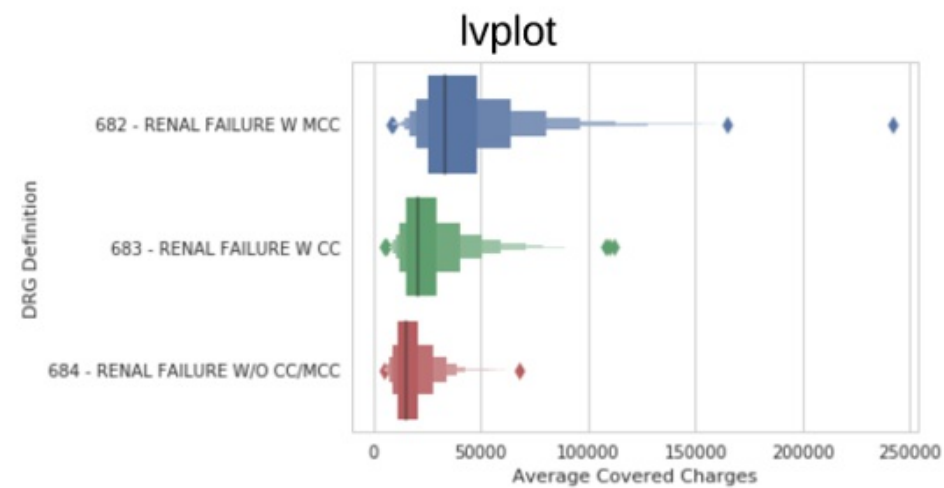
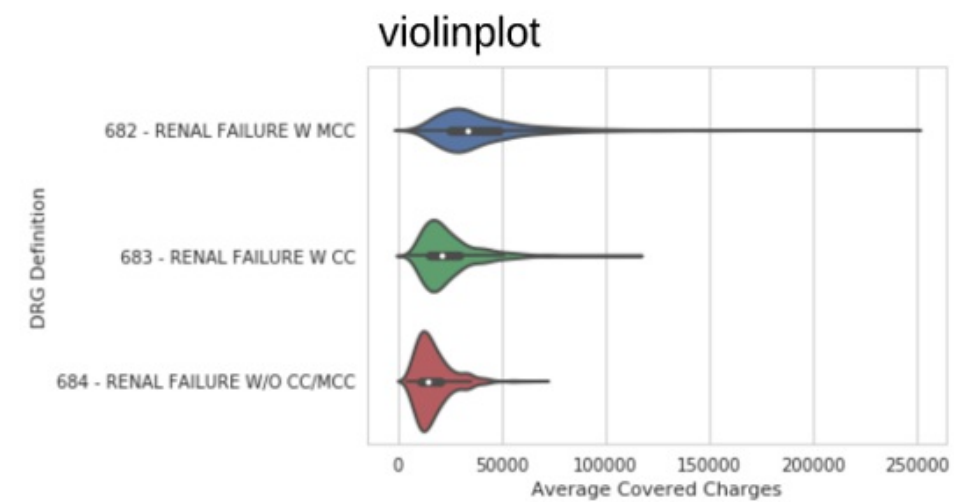
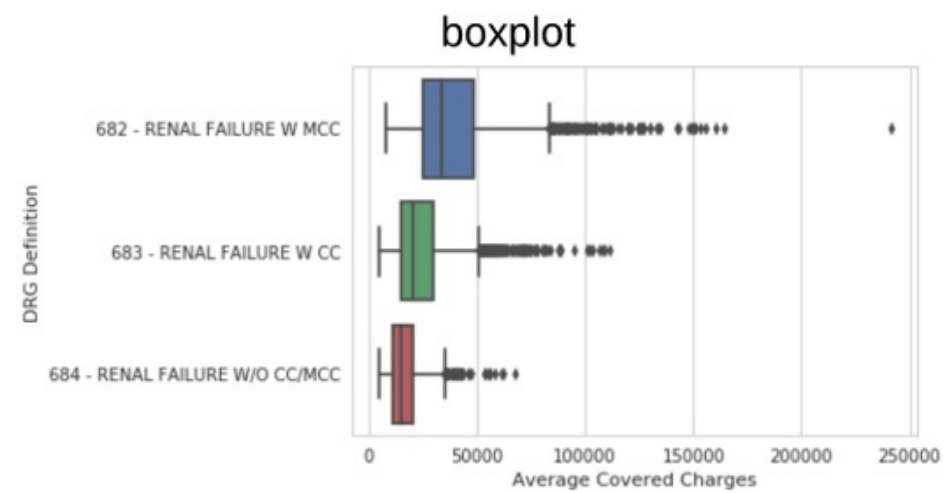


# Plot types - show each observation



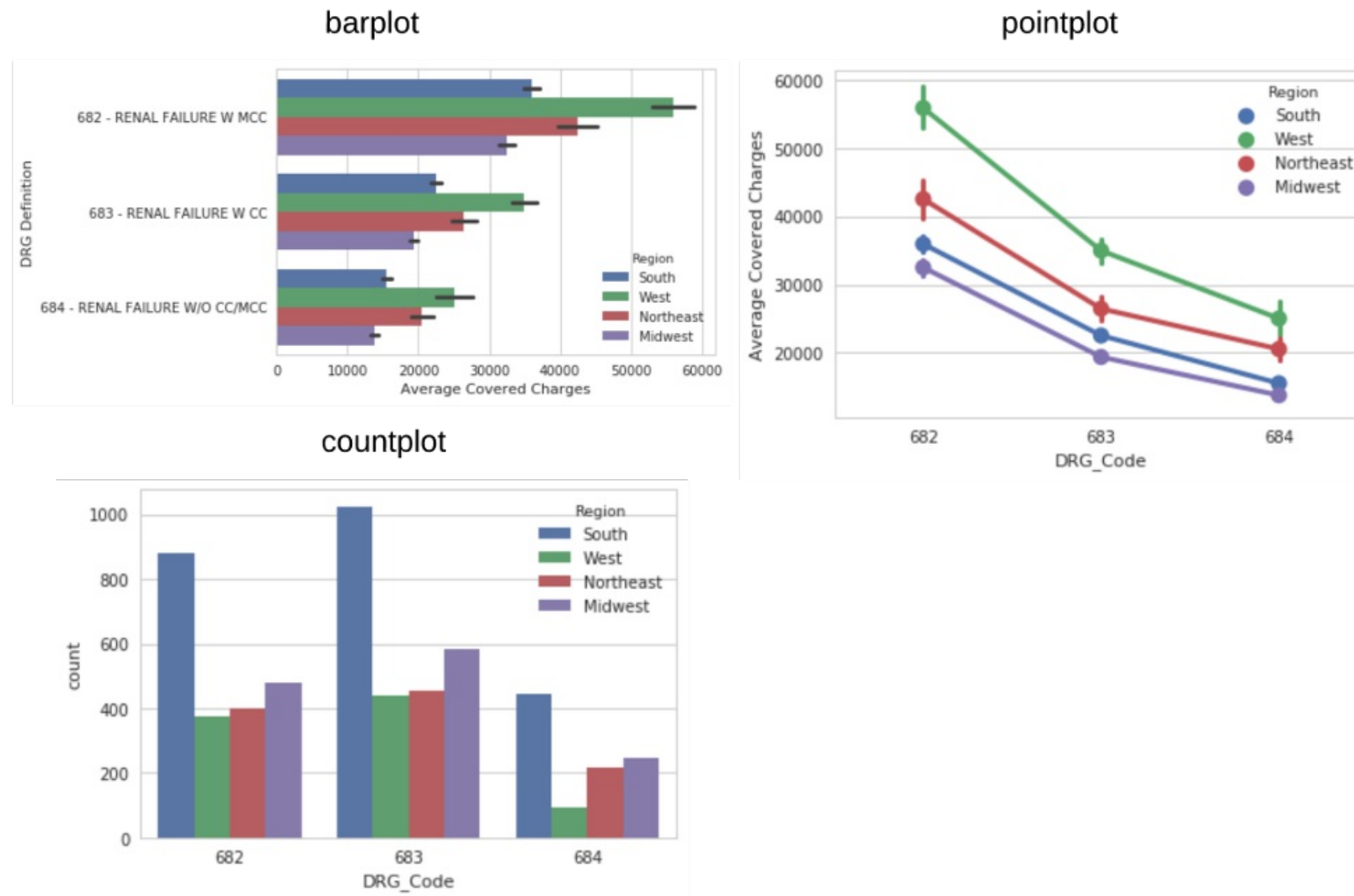


# Plot types - abstract representations



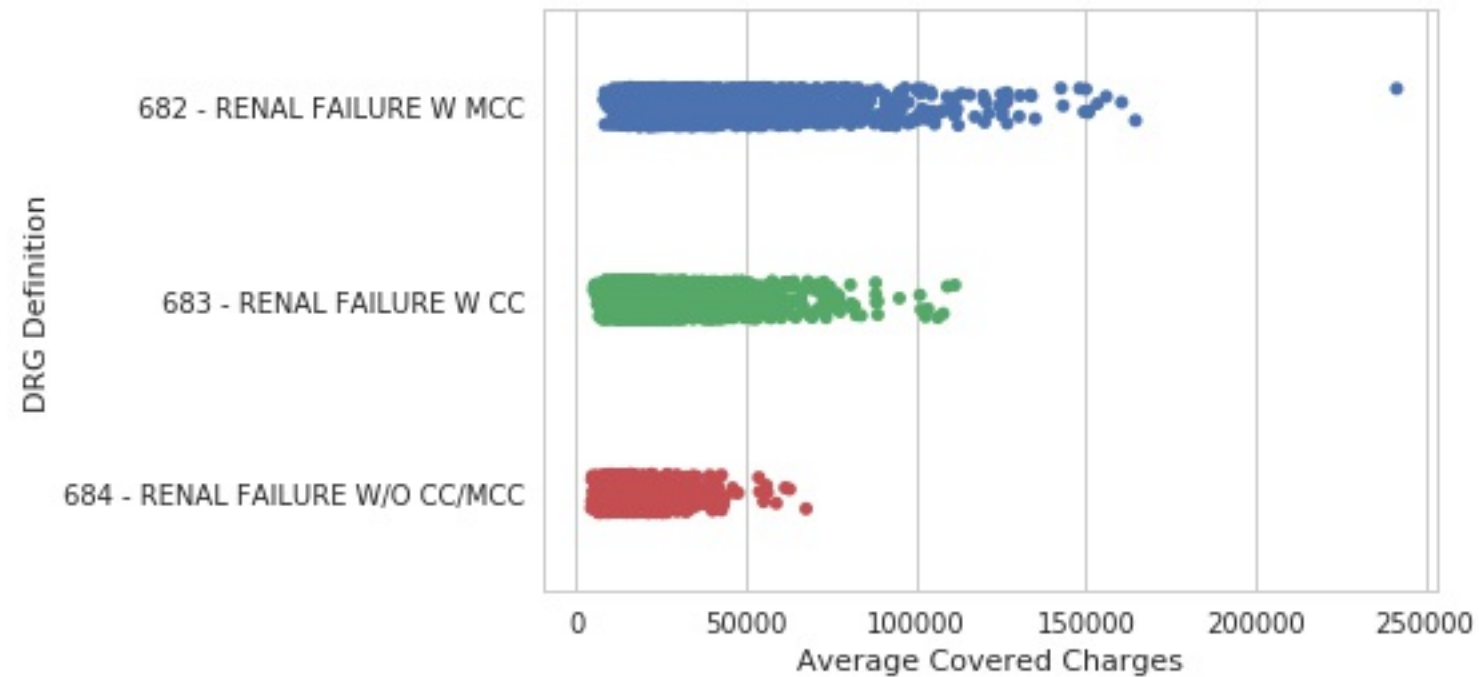


# Plot types - statistical estimates



# Plots of each observation - stripplot

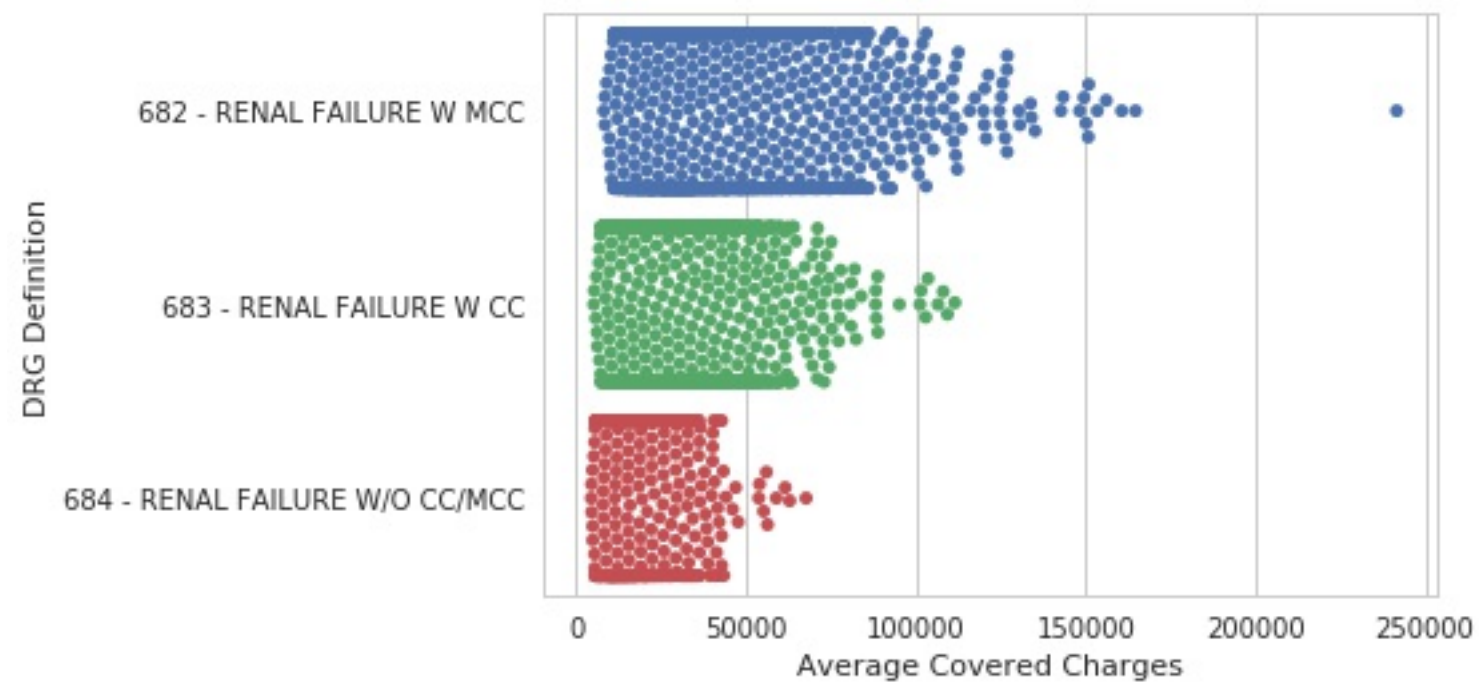
```
sns.stripplot(data=df, y="DRG Definition",  
              x="Average Covered Charges",  
              jitter=True)
```





# Plots of each observation - swarmplot

```
sns.swarmplotplot(data=df, y="DRG Definition",  
                  x="Average Covered Charges")
```

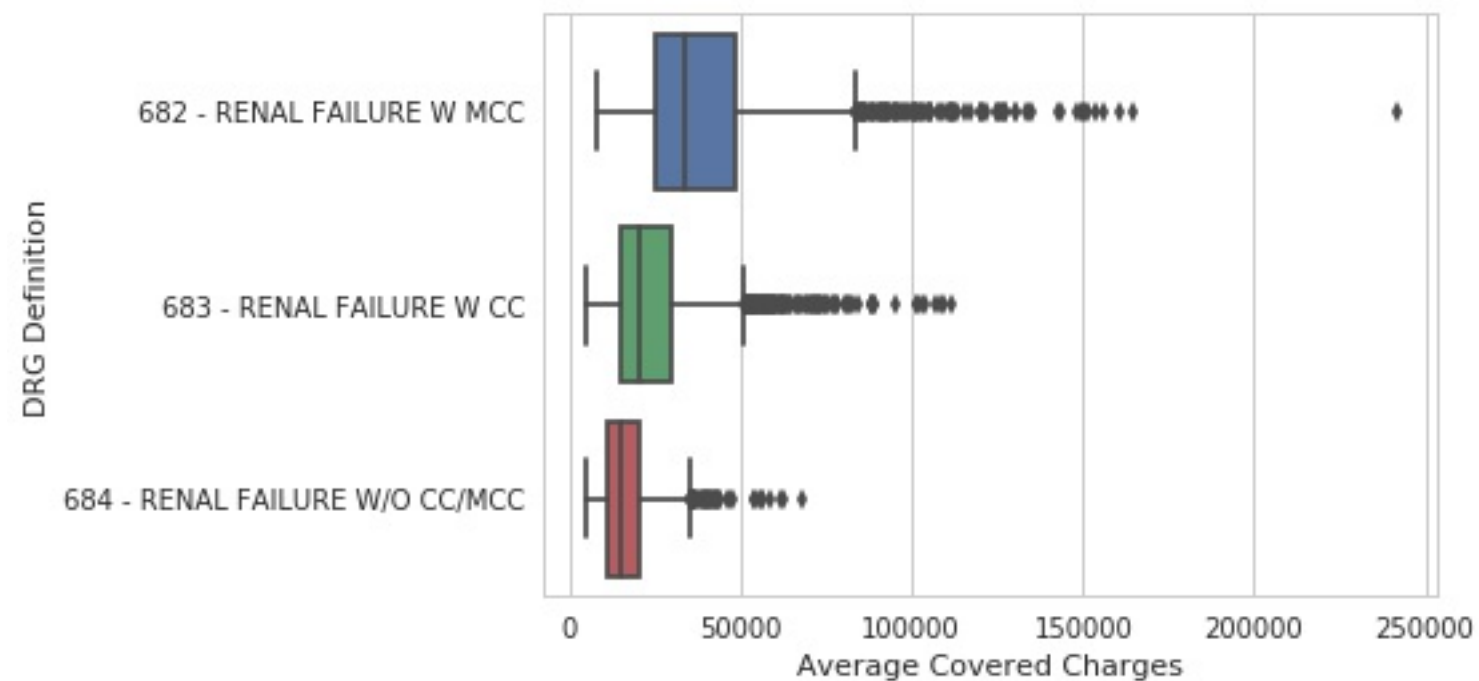






# Abstract representations - boxplot

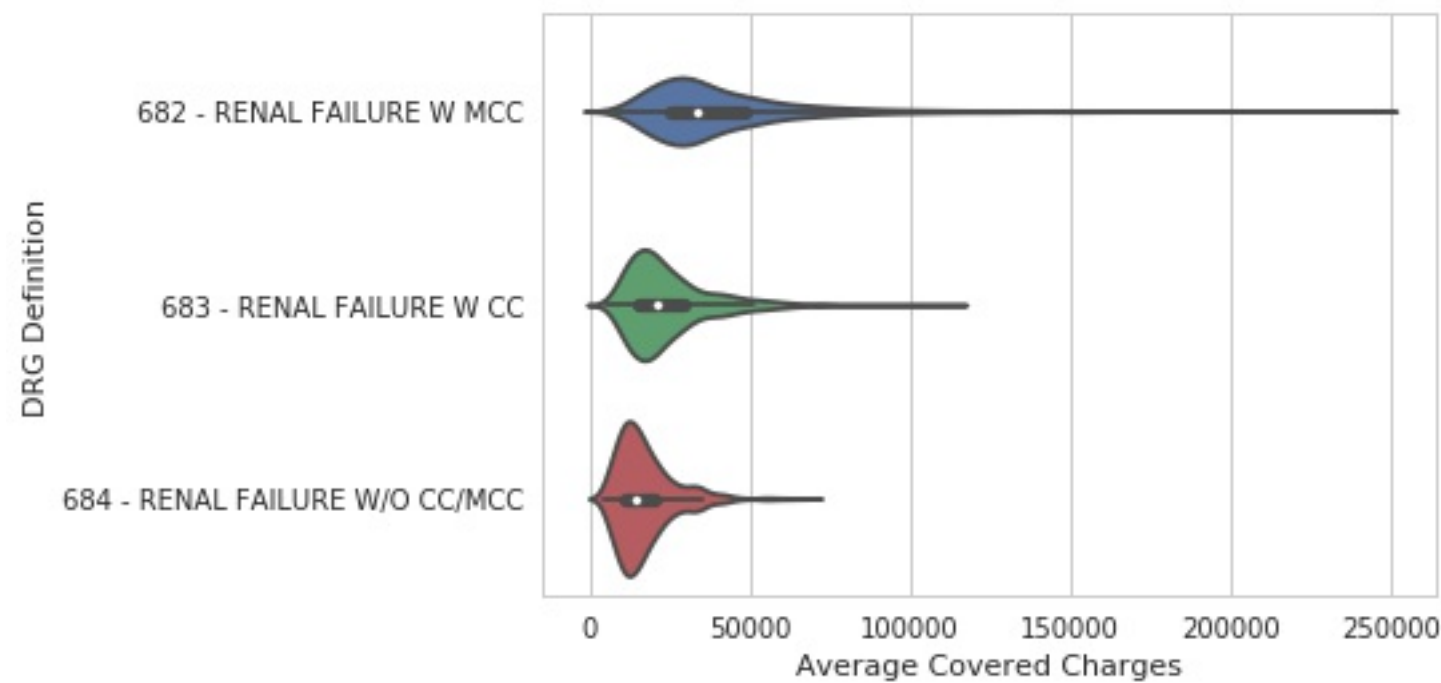
```
sns.boxplot(data=df, y="DRG Definition",  
            x="Average Covered Charges")
```





# Abstract representation - violinplot

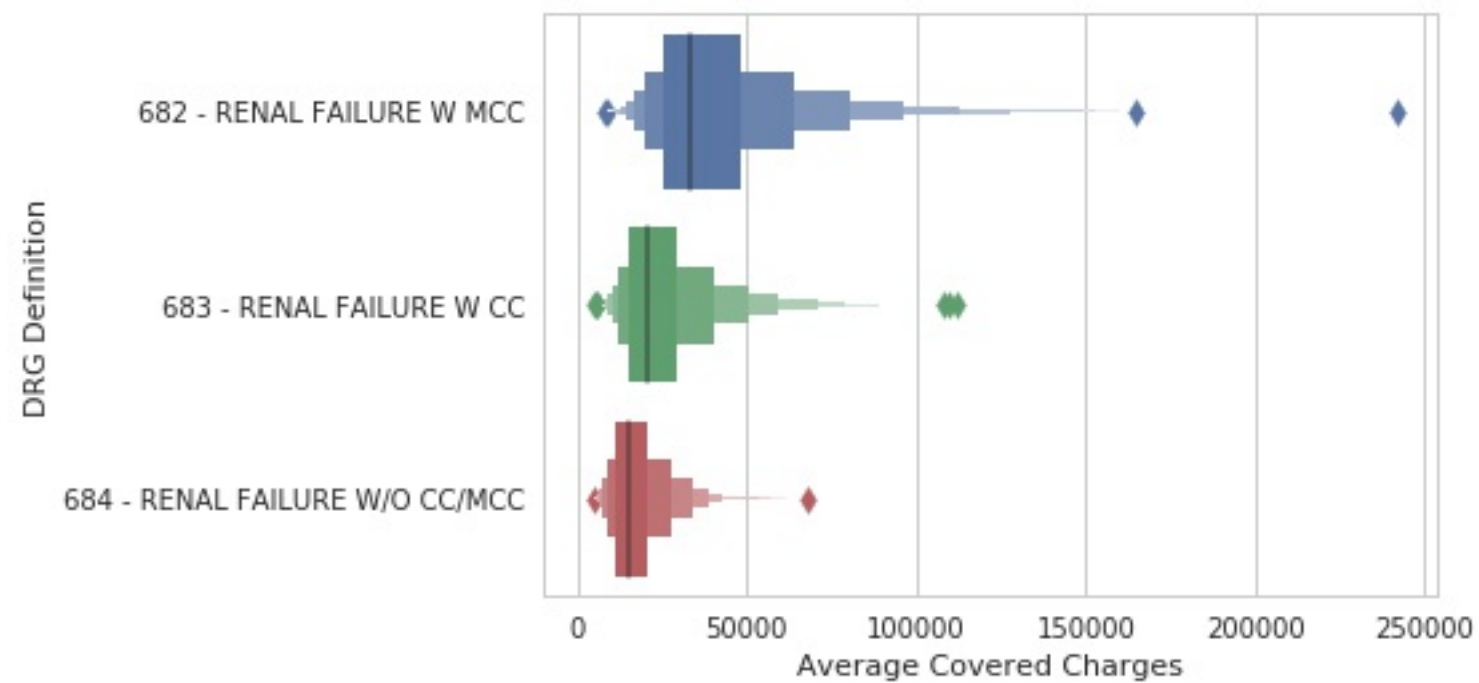
```
sns.violinplot(data=df, y="DRG Definition",  
               x="Average Covered Charges")
```





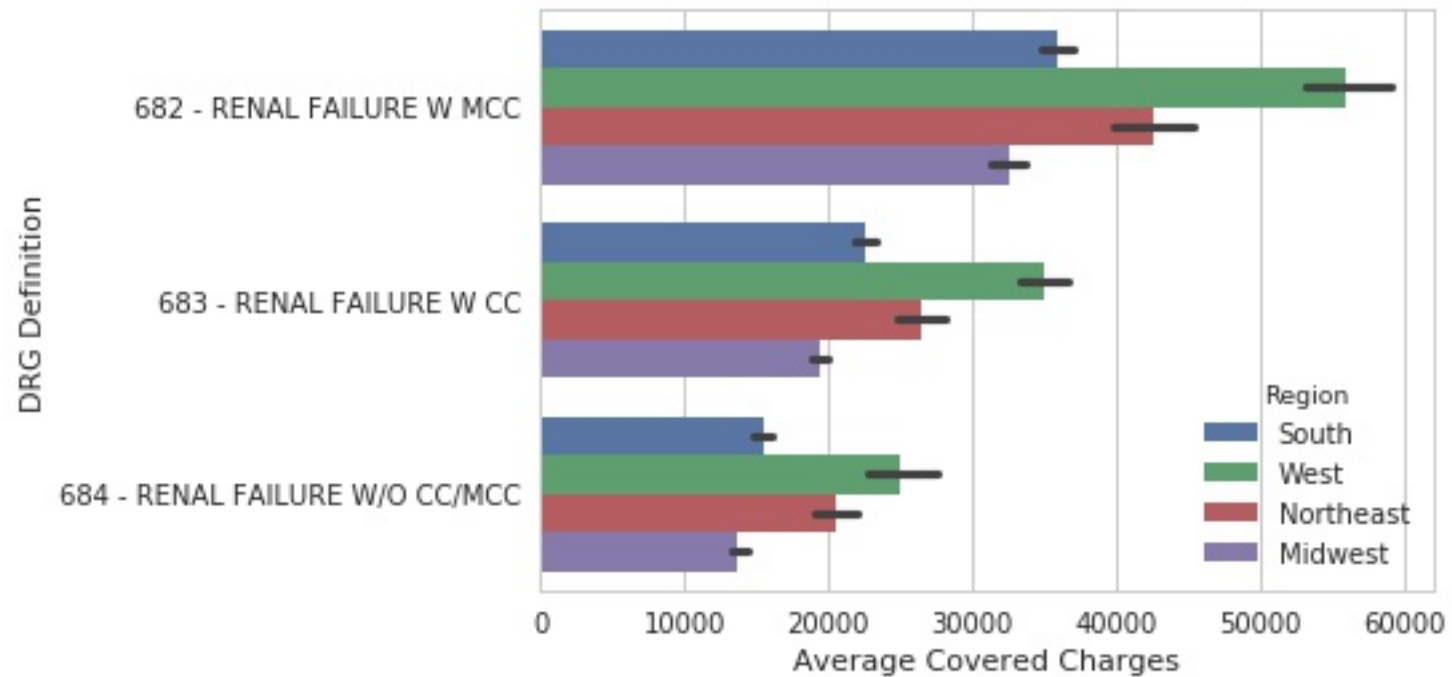
# Abstract representation - lvplot

```
sns.lvplot(data=df, y="DRG Definition",  
           x="Average Covered Charges")
```



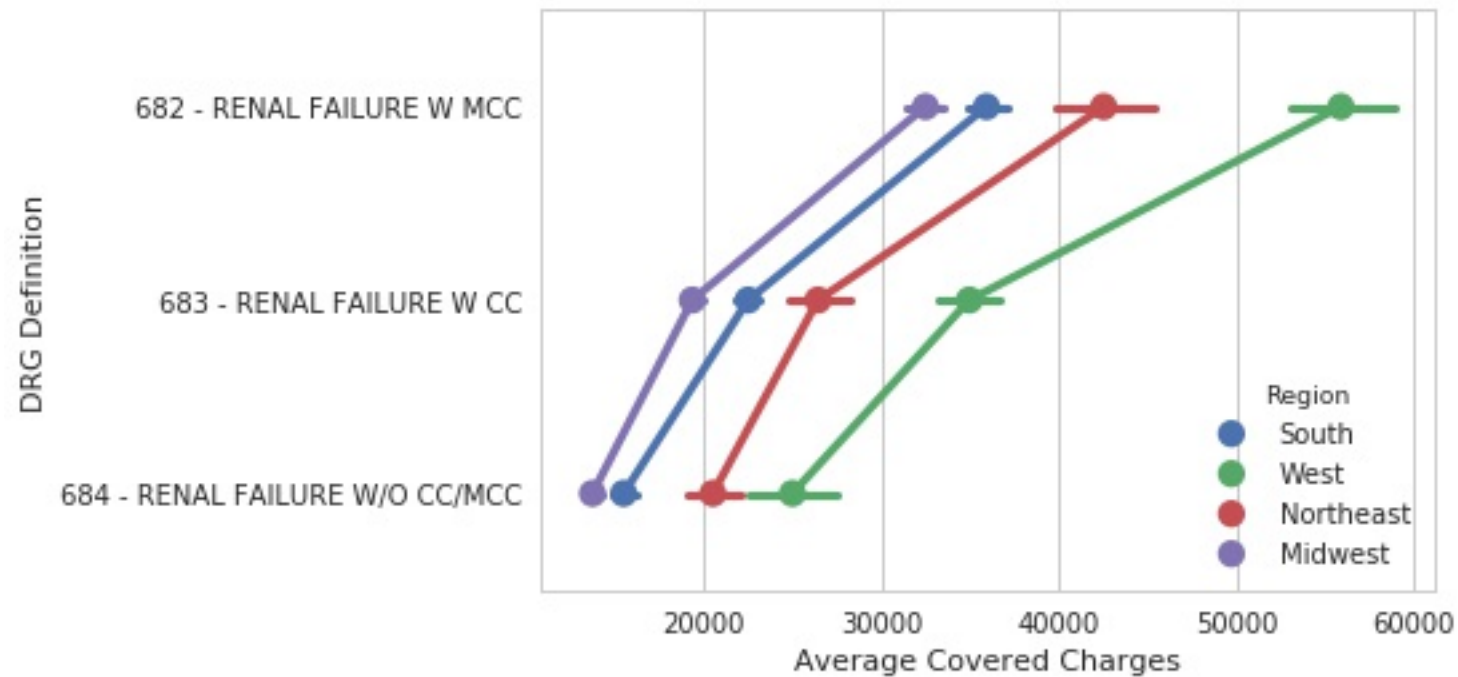
# Statistical estimates - barplot

```
sns.barplot(data=df, y="DRG Definition",  
            x="Average Covered Charges",  
            hue="Region")
```



# Statistical estimates - pointplot

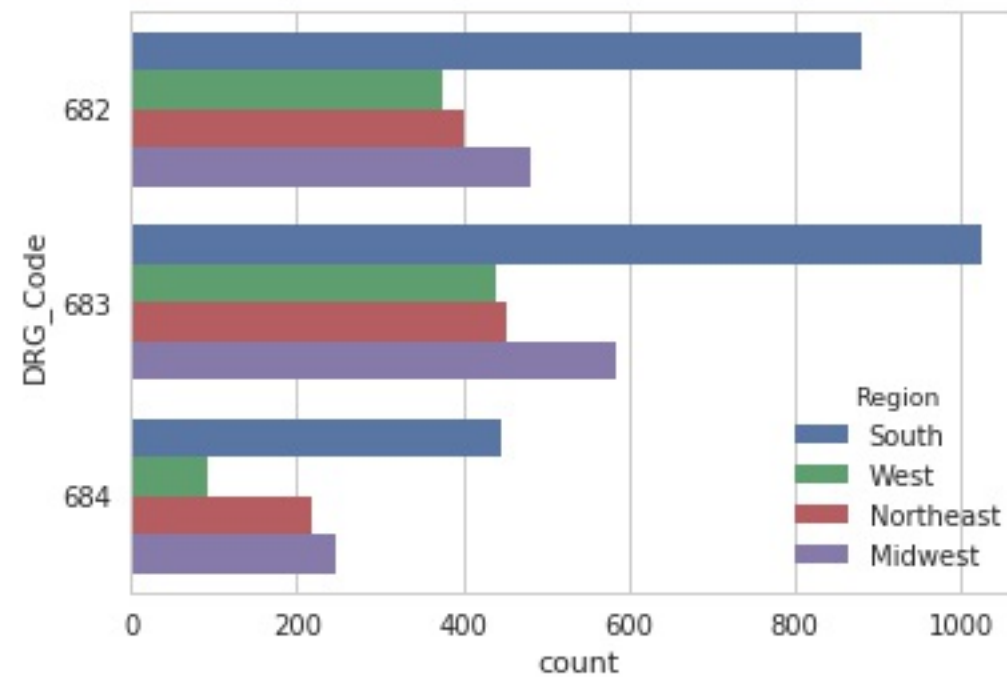
```
sns.pointplot(data=df, y="DRG Definition",  
              x="Average Covered Charges",  
              hue="Region")
```





# Statistical estimates - countplot

```
sns.countplot(data=df, y="DRG_Code", hue="Region")
```





## DATA VISUALIZATION WITH SEABORN

**Let's practice!**



DATA VISUALIZATION WITH SEABORN

# Regression Plots

Chris Moffitt  
Instructor



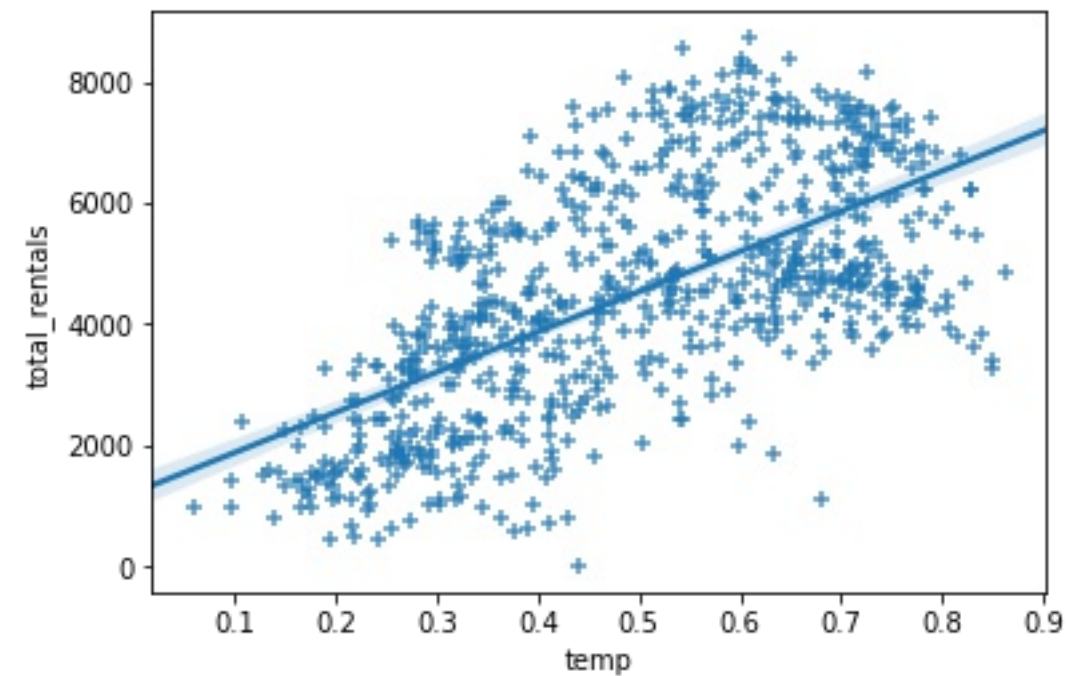


# Bicycle Dataset

- Aggregated bicycle sharing data in Washington DC
- Data includes:
  - Rental amounts
  - Weather information
  - Calendar information
- Can we predict rental amounts?

# Plotting with regplot

```
sns.regplot(data=df, x='temp', y='total_rentals', marker='+')
```

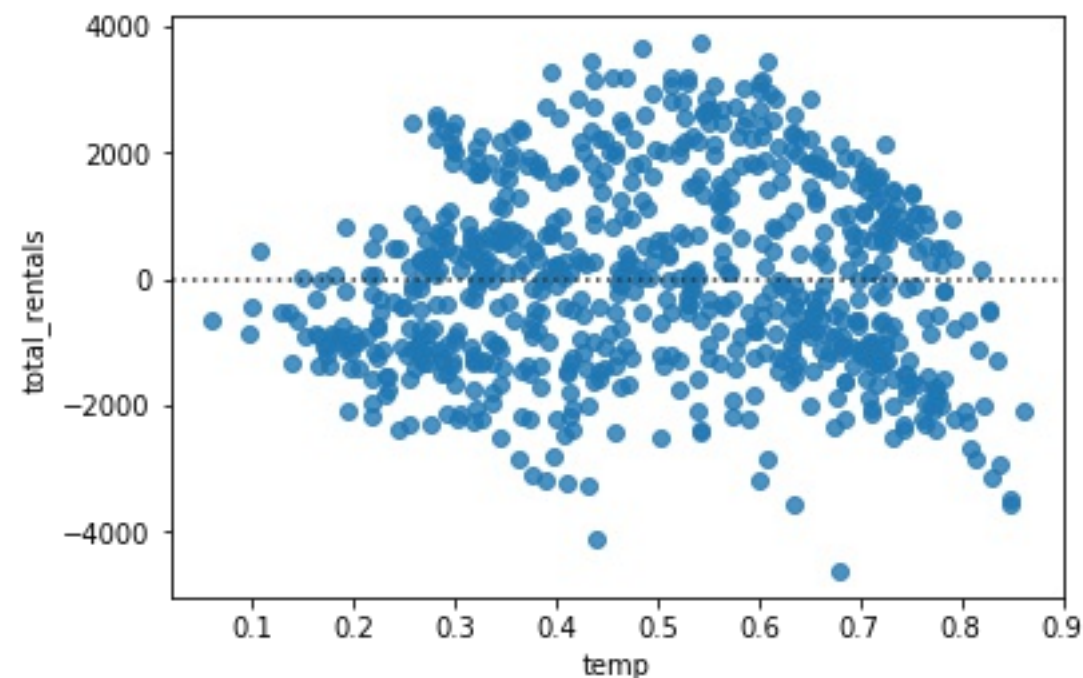




# Evaluating regression with residplot

- A residual plot is useful for evaluating the fit of a model
- Seaborn supports through residplot function

```
sns.residplot(data=df, x='temp', y='total_rentals')
```

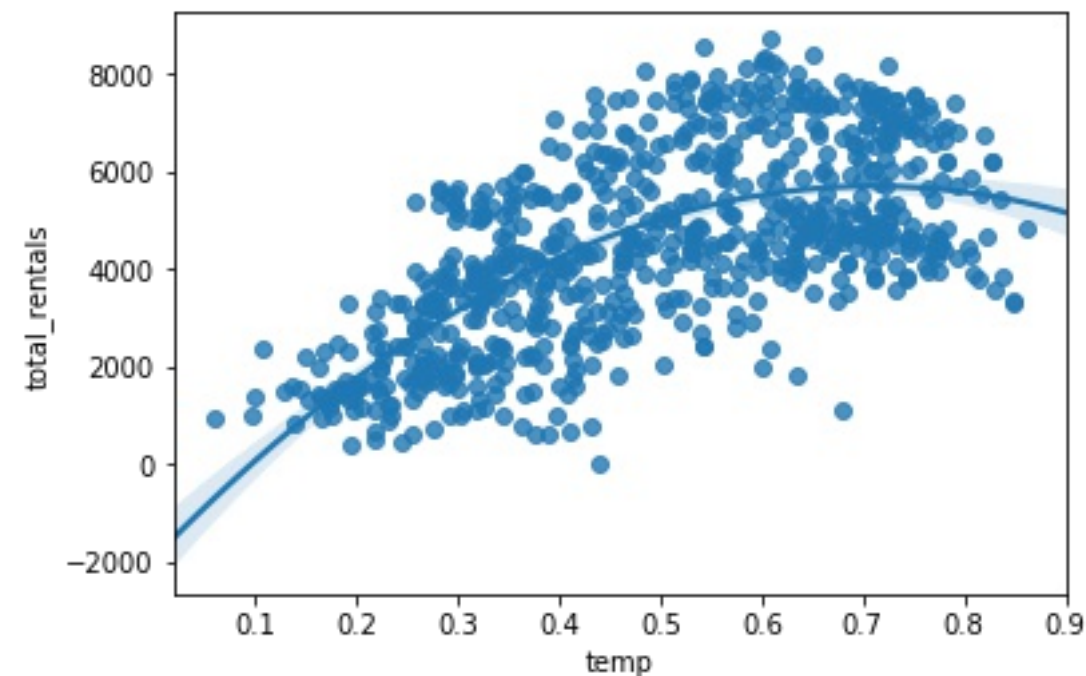




# Polynomial regression

- Seaborn supports polynomial regression using the order parameter

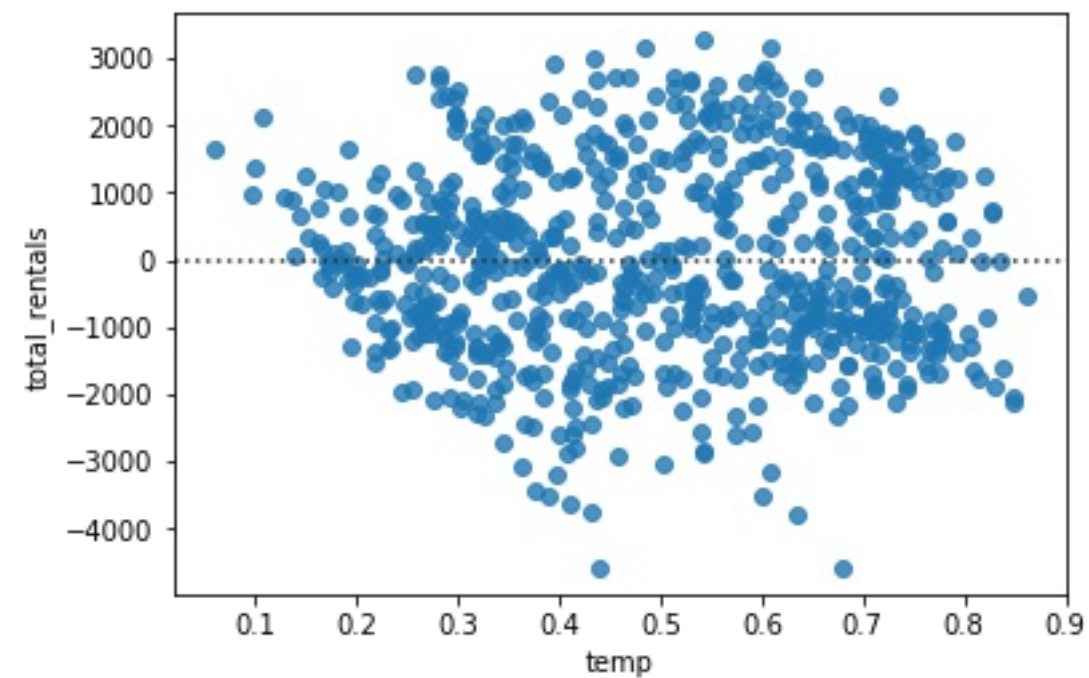
```
sns.regplot(data=df, x='temp', y='total_rentals', order=2)
```





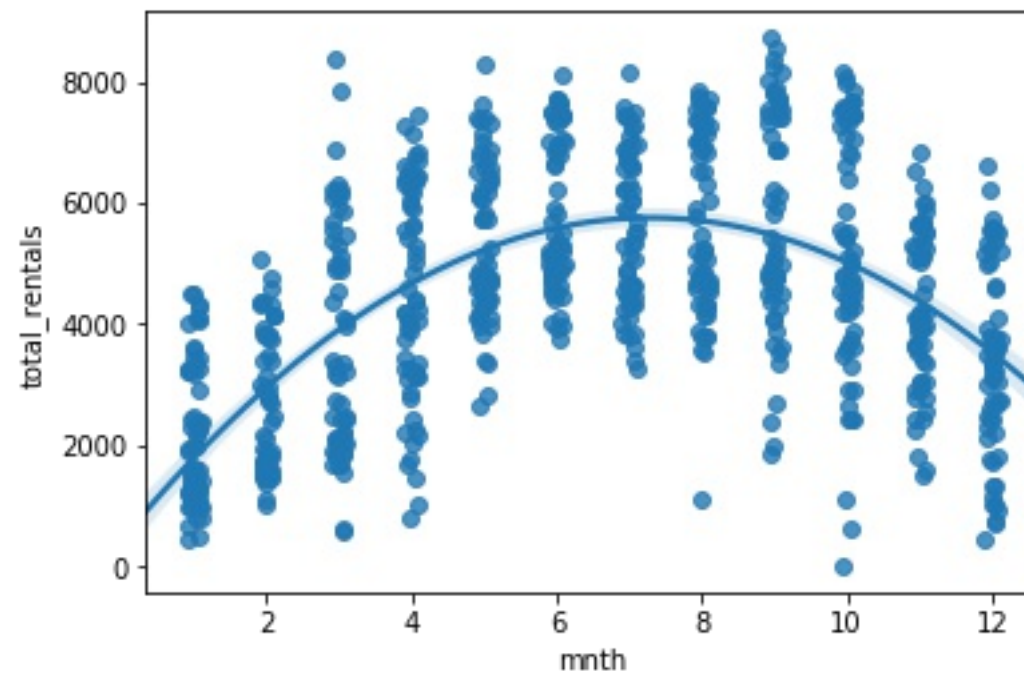
# residplot with polynomial regression

```
sns.residplot(data=df, x='temp', y='total_rentals', order=2)
```



# Categorical values

```
sns.regplot(data=df, x='mnth', y='total_rentals', x_jitter=.1, order=2)
```

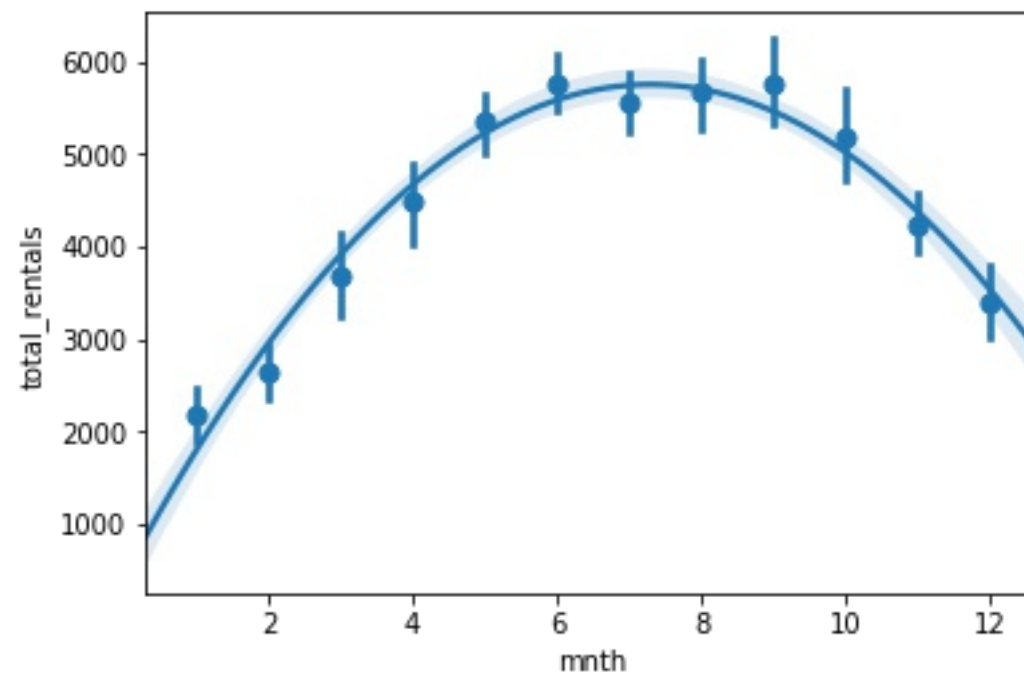




# Estimators

- In some cases, an `x_estimator` can be useful for highlighting trends

```
sns.regplot(data=df, x='mnth', y='total_rentals',  
            x_estimator=np.mean, order=2)
```

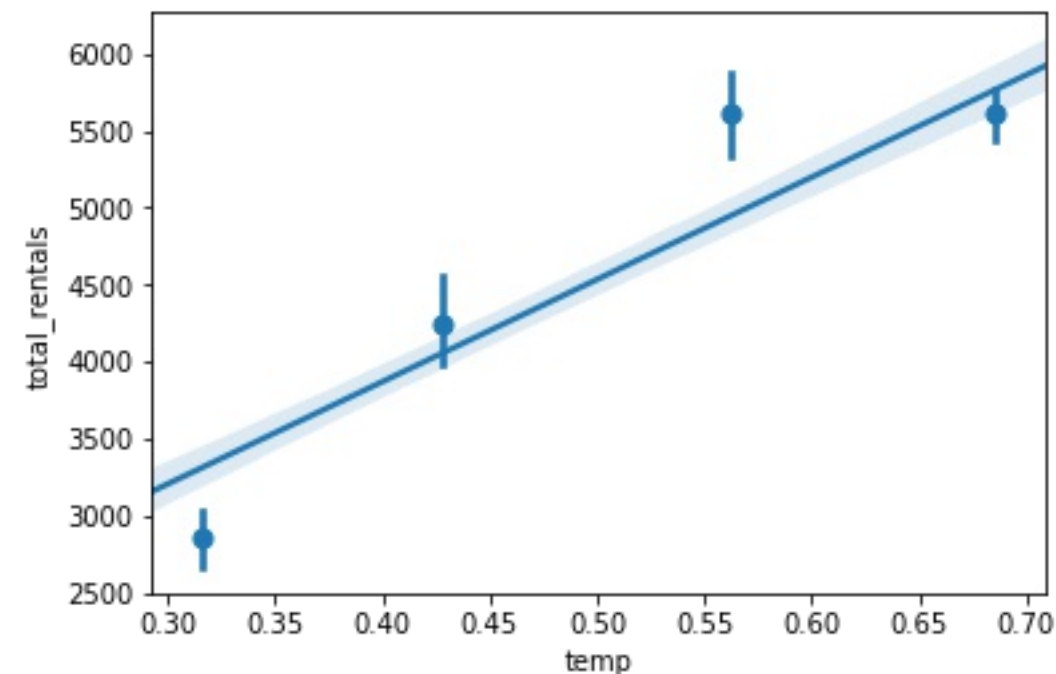




# Binning the data

- `x_bins` can be used to divide the data into discrete bins
- The regression line is still fit against all the data

```
sns.regplot(data=df, x='temp', y='total_rentals', x_bins=4)
```







## DATA VISUALIZATION WITH SEABORN

**Let's practice!**



DATA VISUALIZATION WITH SEABORN

# Matrix Plots

Chris Moffitt  
Instructor

# Getting data in the right format

- Seaborn's heatmap() function requires data to be in a grid format
- pandas crosstab() is frequently used to manipulate the data

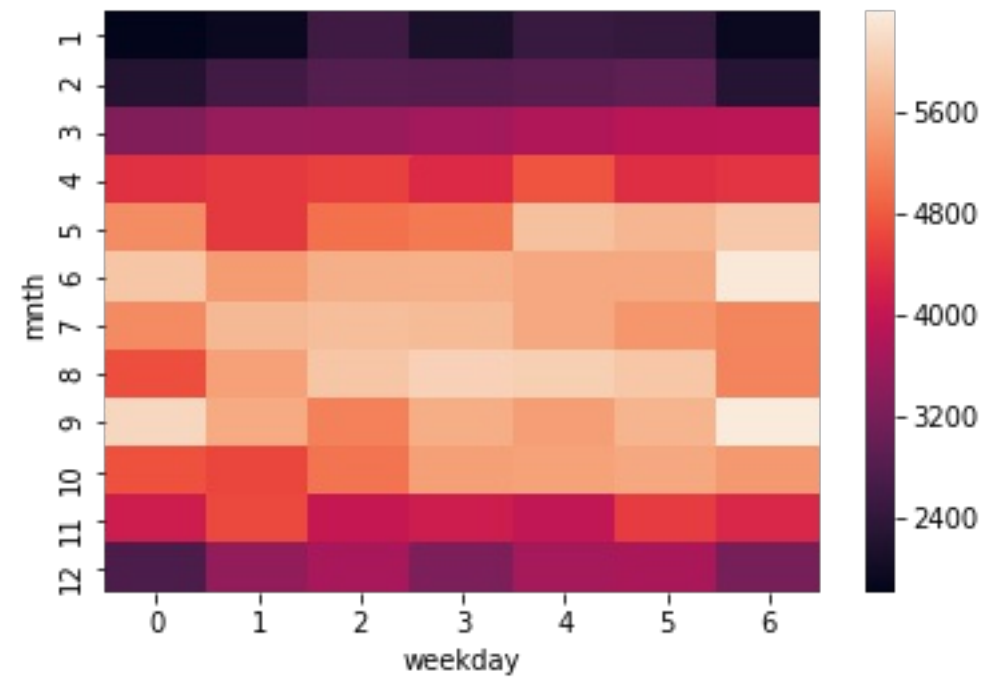
```
pd.crosstab(df["mnth"], df["weekday"], values=df["total_rentals"],  
            aggfunc='mean').round(0)
```

| weekday | 0      | 1      | 2      | 3      | 4      | 5      | 6      |
|---------|--------|--------|--------|--------|--------|--------|--------|
| mnth    |        |        |        |        |        |        |        |
| 1       | 1816.0 | 1927.0 | 2568.0 | 2139.0 | 2513.0 | 2446.0 | 1957.0 |
| 2       | 2248.0 | 2604.0 | 2824.0 | 2813.0 | 2878.0 | 2933.0 | 2266.0 |
| 3       | 3301.0 | 3546.0 | 3574.0 | 3670.0 | 3817.0 | 3926.0 | 3939.0 |
| 4       | 4417.0 | 4516.0 | 4556.0 | 4331.0 | 4764.0 | 4387.0 | 4446.0 |
| 5       | 5320.0 | 4512.0 | 5025.0 | 5119.0 | 5893.0 | 5751.0 | 5978.0 |
| 6       | 5940.0 | 5478.0 | 5681.0 | 5701.0 | 5622.0 | 5616.0 | 6344.0 |
| 7       | 5298.0 | 5792.0 | 5844.0 | 5814.0 | 5624.0 | 5406.0 | 5232.0 |
| 8       | 4703.0 | 5518.0 | 5930.0 | 6077.0 | 6038.0 | 5958.0 | 5224.0 |
| 9       | 6160.0 | 5637.0 | 5184.0 | 5668.0 | 5486.0 | 5747.0 | 6394.0 |
| 10      | 4735.0 | 4632.0 | 5065.0 | 5505.0 | 5537.0 | 5623.0 | 5445.0 |
| 11      | 4126.0 | 4658.0 | 4040.0 | 4136.0 | 3994.0 | 4524.0 | 4288.0 |
| 12      | 2740.0 | 3498.0 | 3713.0 | 3270.0 | 3711.0 | 3742.0 | 3195.0 |



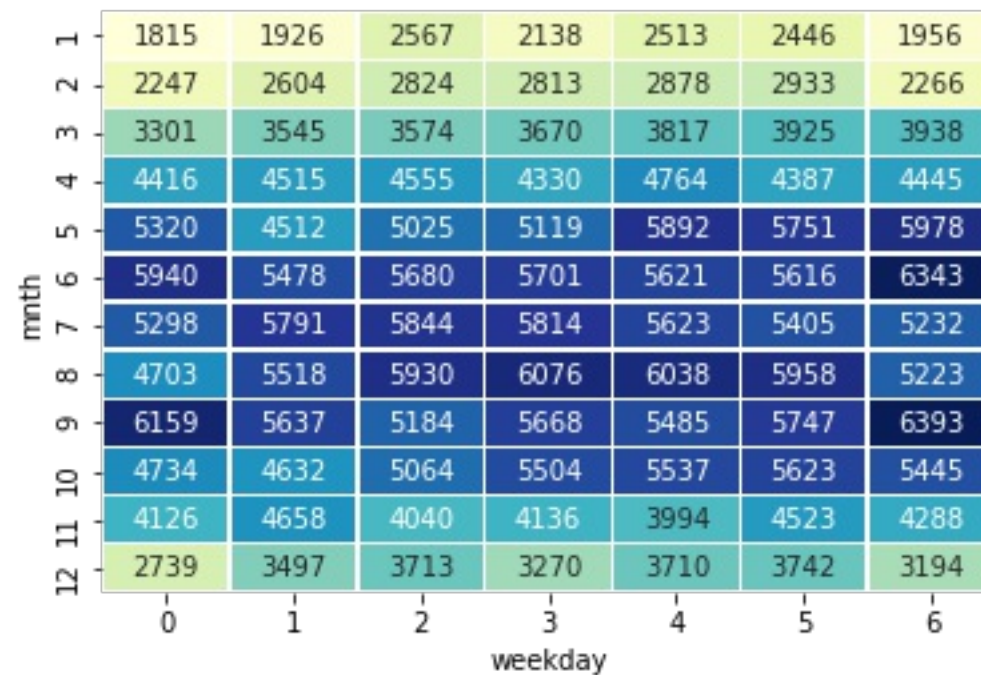
# Build a heatmap

```
sns.heatmap(pd.crosstab(df["mnth"], df["weekday"],  
                        values=df["total_rentals"], aggfunc='mean'))
```



# Customize a heatmap

```
sns.heatmap(df_crosstab, annot=True, fmt="d",  
            cmap="YlGnBu", cbar=False, linewidths=.5)
```

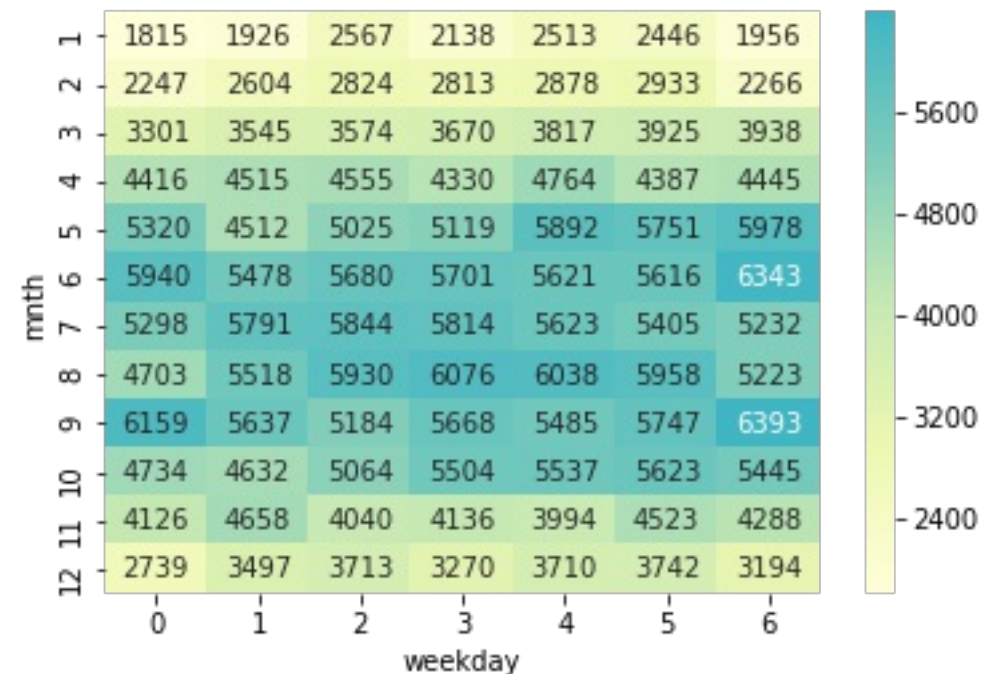




# Centering a heatmap

- Seaborn support centering the heatmap colors on a specific value

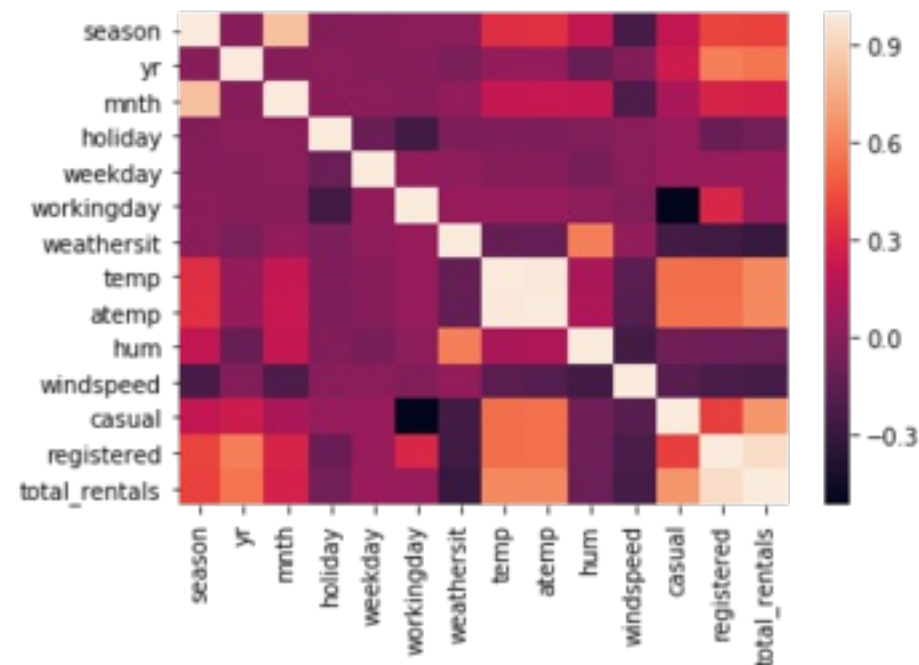
```
sns.heatmap(df_crosstab, annot=True, fmt="d", cmap="YlGnBu", cbar=True,  
            center=df_crosstab.loc[9, 6])
```



# Plotting a correlation matrix

- Pandas corr function calculates correlations between columns in a dataframe
- The output can be converted to a heatmap with seaborn

```
sns.heatmap(df.corr())
```





## DATA VISUALIZATION WITH SEABORN

**Let's practice!**

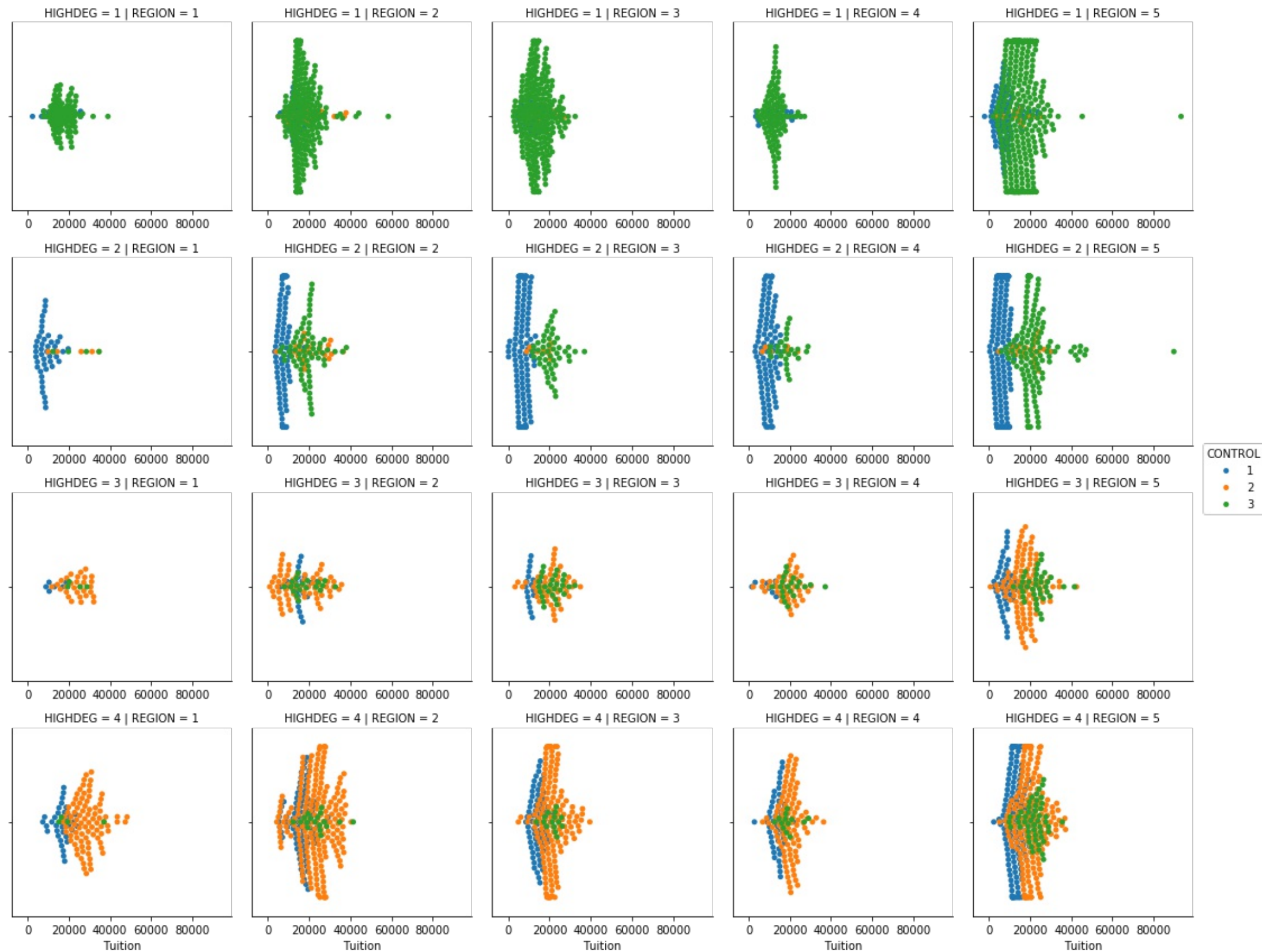




DATA VISUALIZATION WITH SEABORN

# Using FacetGrid, factorplot and Implot

Chris Moffitt  
Instructor





# Tidy data

- Seaborn's grid plots require data in "tidy format"
- One observation per row of data

|   | INSTNM                              | OPEID   | REGION | SAT_AVG_ALL | PCTPELL | PCTFLOAN | ADM_RATE_ALL | UG      | AVGFACXSAL | COMPL_RPY_5YR_RT | DEBT_MDN |
|---|-------------------------------------|---------|--------|-------------|---------|----------|--------------|---------|------------|------------------|----------|
| 0 | Alabama A & M University            | 100200  | 5      | 850.0       | 0.7249  | 0.8159   | 0.653841     | 4380.0  | 7017.0     | 0.477631579      | 14600    |
| 1 | University of Alabama at Birmingham | 105200  | 5      | 1147.0      | 0.3505  | 0.5218   | 0.604275     | 10331.0 | 10221.0    | 0.673230442      | 14250    |
| 2 | Amridge University                  | 2503400 | 5      | NaN         | 0.7455  | 0.8781   | NaN          | 98.0    | 3217.0     | 0.636363636      | 11082    |
| 3 | University of Alabama in Huntsville | 105500  | 5      | 1221.0      | 0.3179  | 0.4589   | 0.811971     | 5220.0  | 9514.0     | 0.762222222      | 15000    |
| 4 | Alabama State University            | 100500  | 5      | 844.0       | 0.7567  | 0.7692   | 0.463858     | 4348.0  | 7940.0     | 0.43006993       | 15274    |



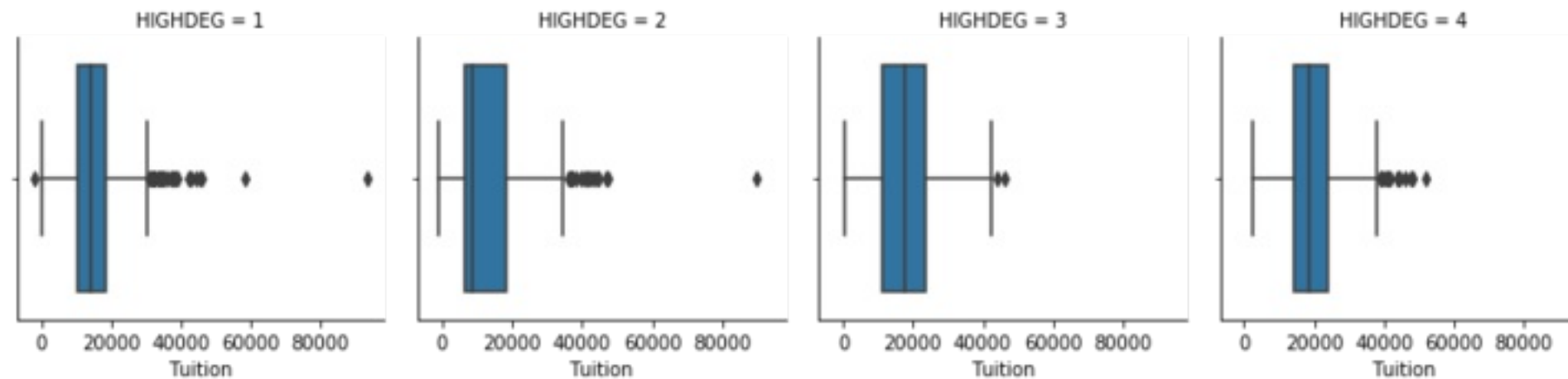
# FacetGrid

- The FacetGrid is foundational for many data aware grids
- It allows the user to control how data is distributed across columns, rows and hue
- Once a FacetGrid is created, the plot type must be mapped to the grid



# FacetGrid Categorical Example

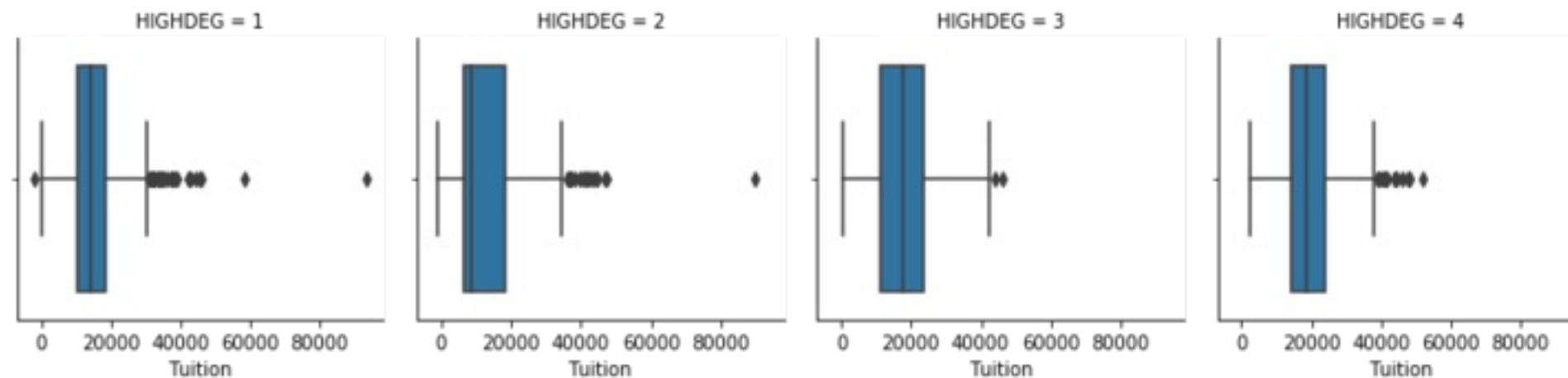
```
g = sns.FacetGrid(df, col="HIGHDEG")  
g.map(sns.boxplot, 'Tuition', order=['1', '2', '3', '4'])
```



# factorplot

- The factorplot is a simpler way to use a FacetGrid for categorical data
- Combines the faceting and mapping process into 1 function

```
sns.factorplot(x="Tuition", data=df, col="HIGHDEG", kind='box')
```

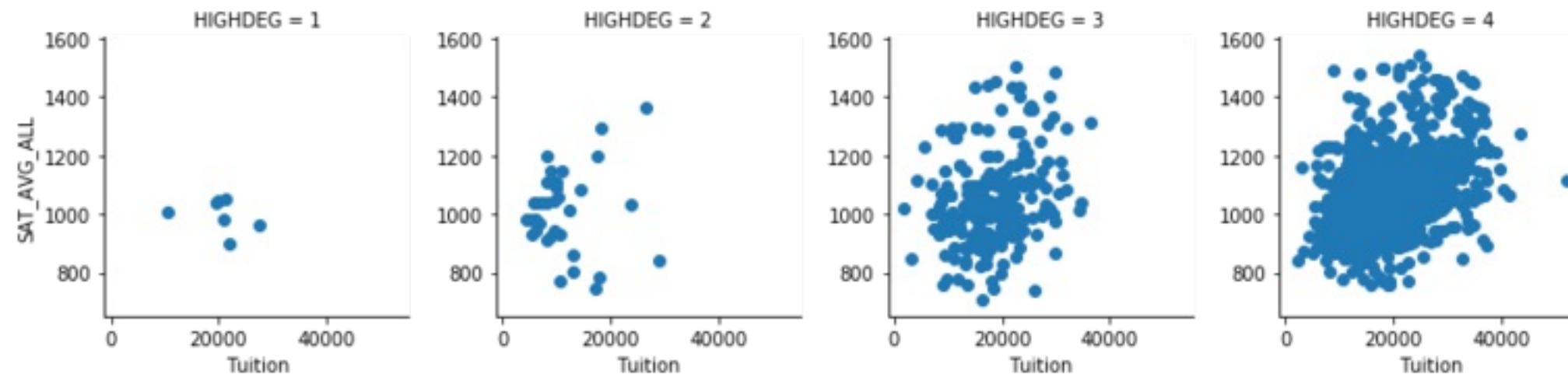




# FacetGrid for regression

- FacetGrid() can also be used for scatter or regression plots

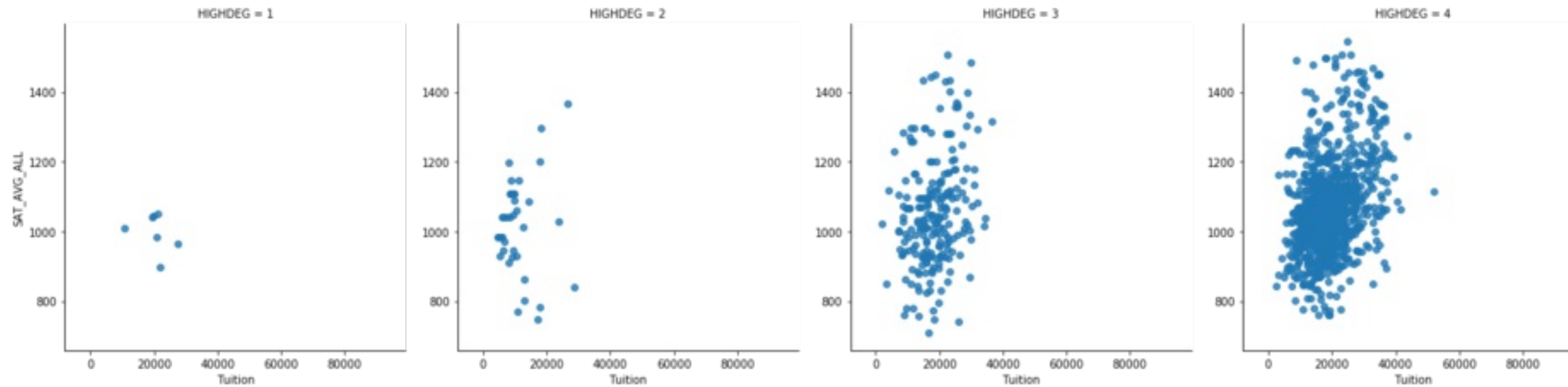
```
g = sns.FacetGrid(df, col="HIGHDEG")  
g.map(plt.scatter, 'Tuition', 'SAT_AVG_ALL')
```



# Implot

- Implot plots scatter and regression plots on a FacetGrid

```
sns.lmplot(data=df, x="Tuition", y="SAT_AVG_ALL",  
          col="HIGHDEG", fit_reg=False)
```

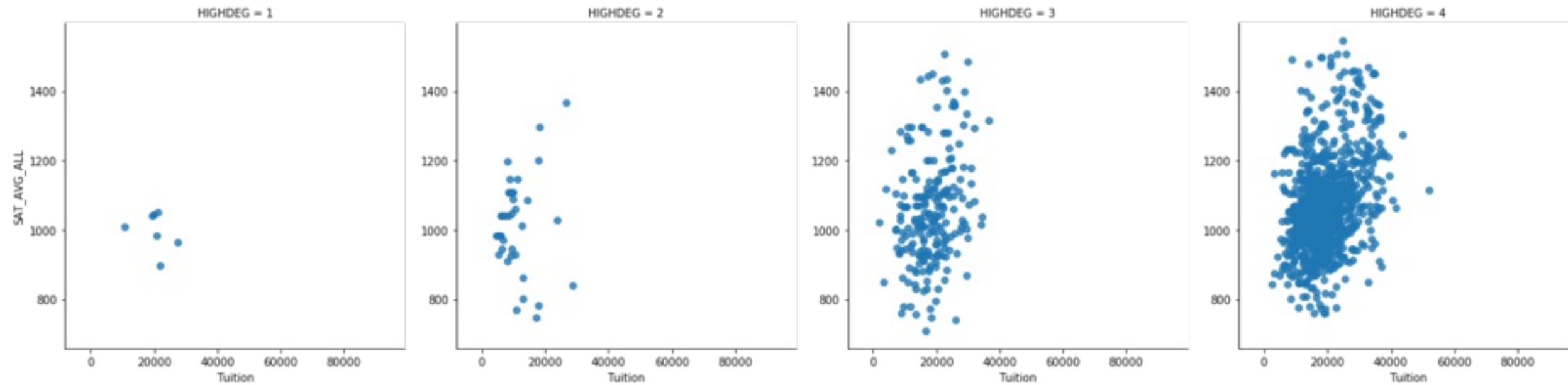






# Implot with regression

```
sns.lmplot(data=df, x="Tuition", y="SAT_AVG_ALL",  
          col="HIGHDEG", row='REGION')
```





## DATA VISUALIZATION WITH SEABORN

**Let's practice!**



DATA VISUALIZATION WITH SEABORN

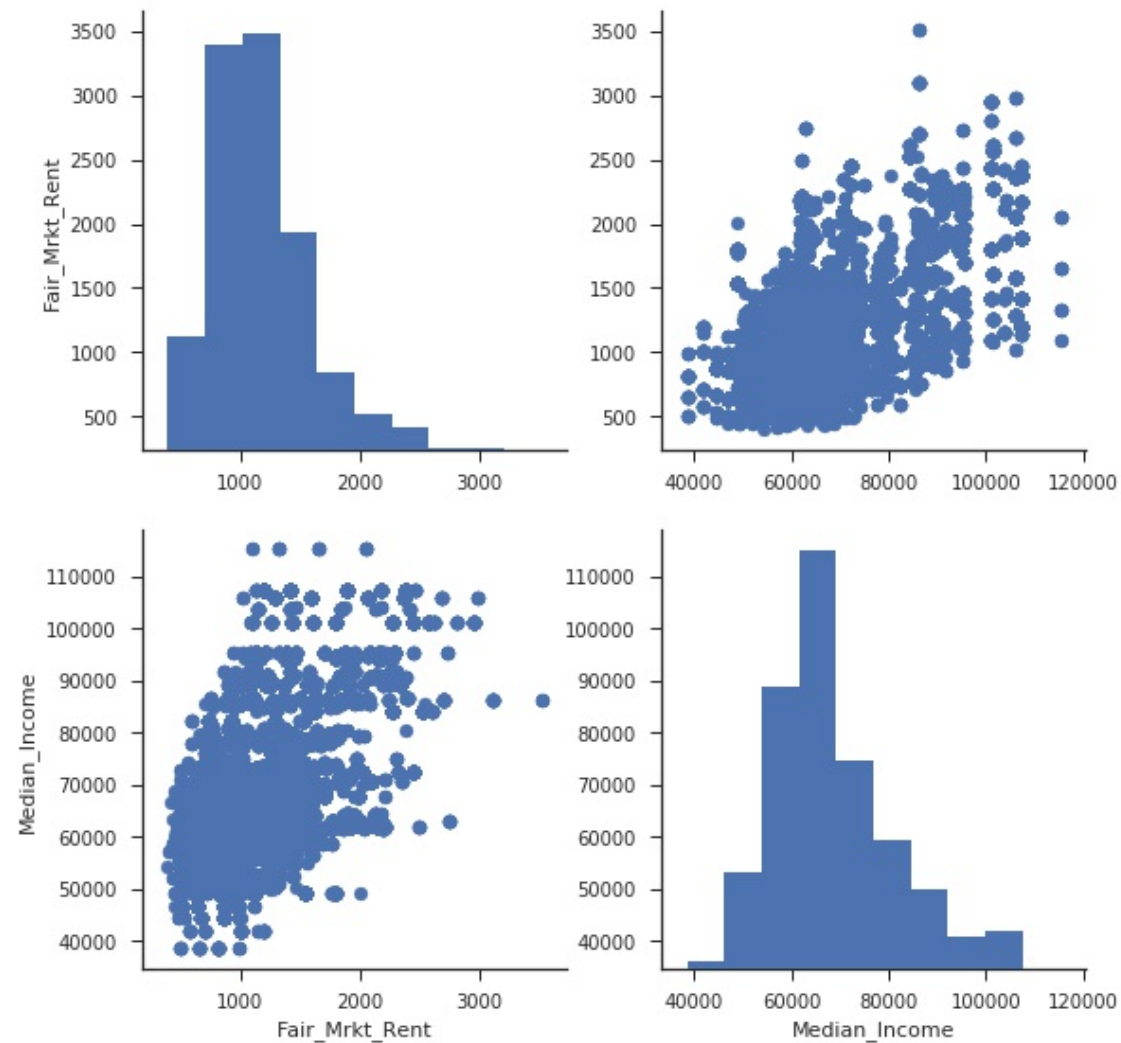
# Using PairGrid and pairplot

Chris Moffitt  
Instructor



# Pairwise relationships

- PairGrid shows pairwise relationships between data elements

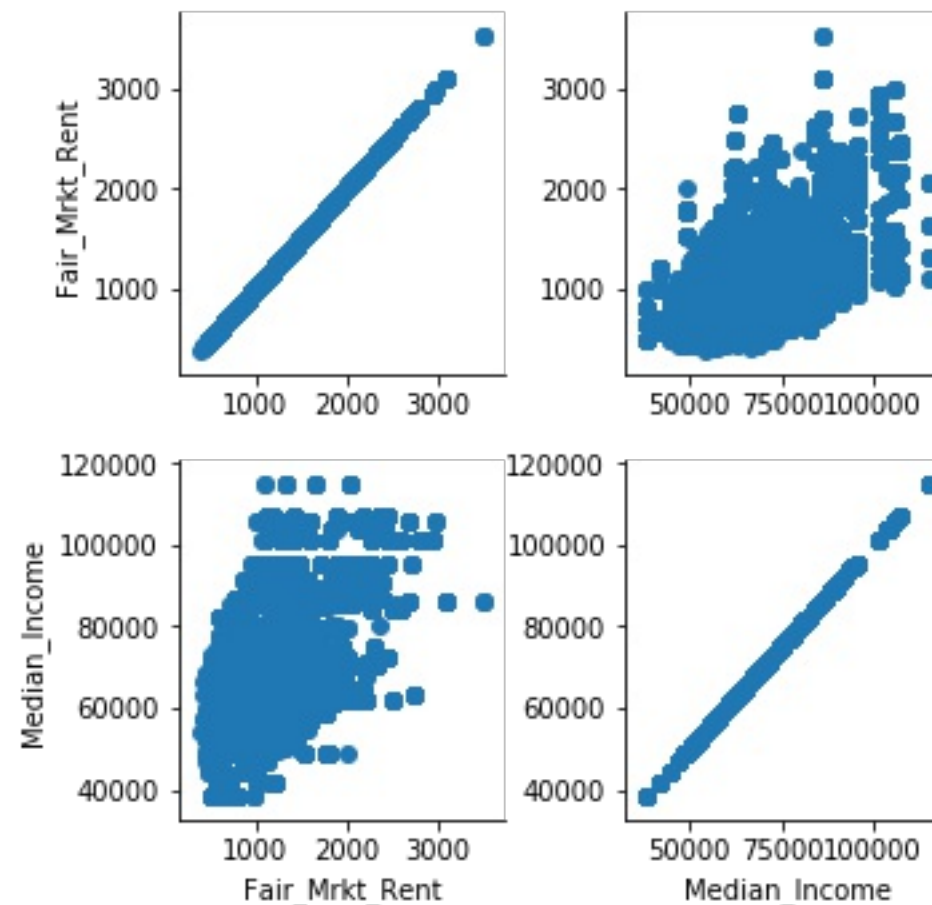




# Creating a PairGrid

- The PairGrid follows similar API to FacetGrid

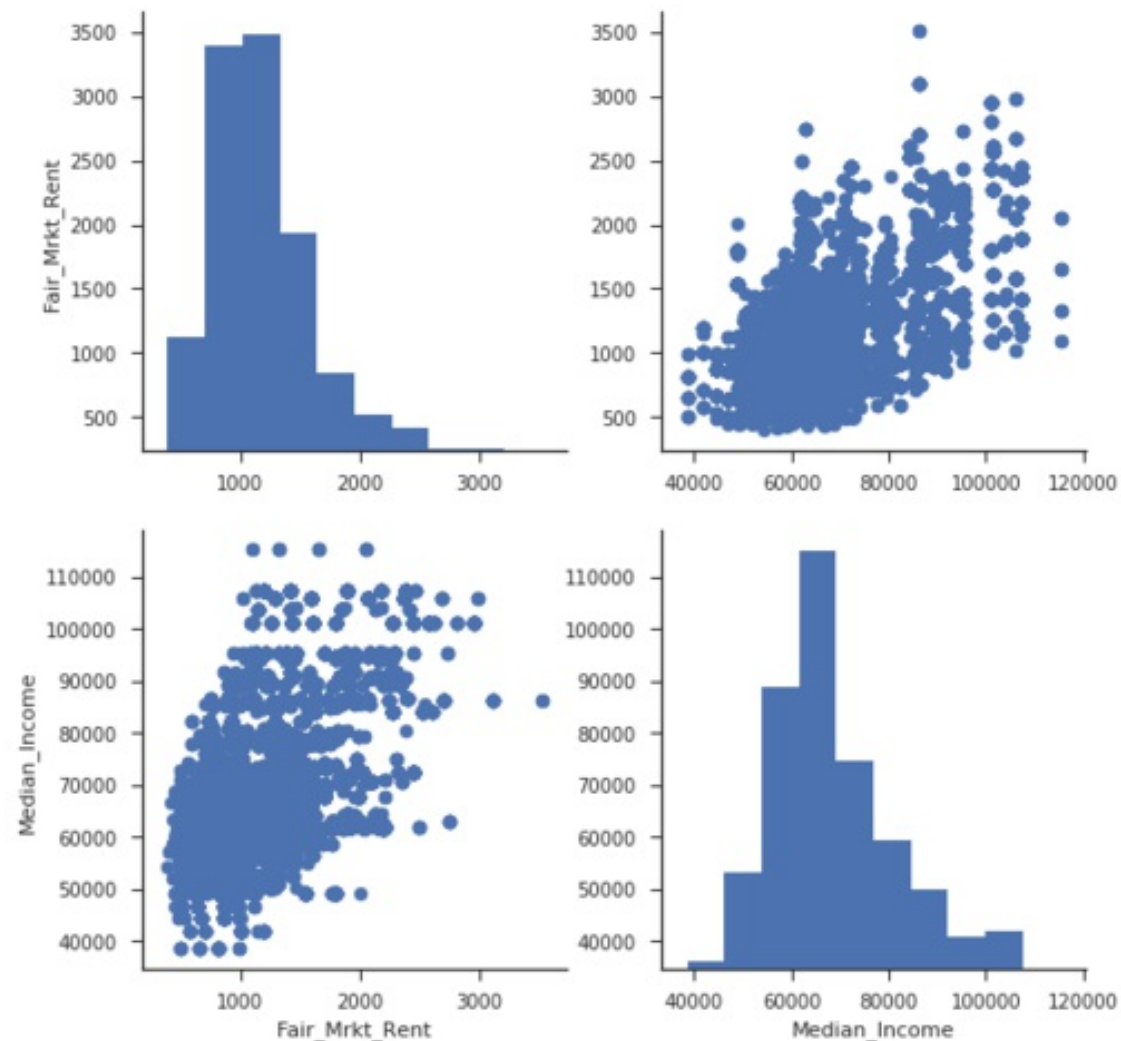
```
g = sns.PairGrid(df, vars=["Fair_Mrkt_Rent", "Median_Income"])  
g = g.map(plt.scatter)
```





# Customizing the PairGrid diagonals

```
g = sns.PairGrid(df, vars=["Fair_Mrkt_Rent", "Median_Income"])  
g = g.map_diag(plt.hist)  
g = g.map_offdiag(plt.scatter)
```

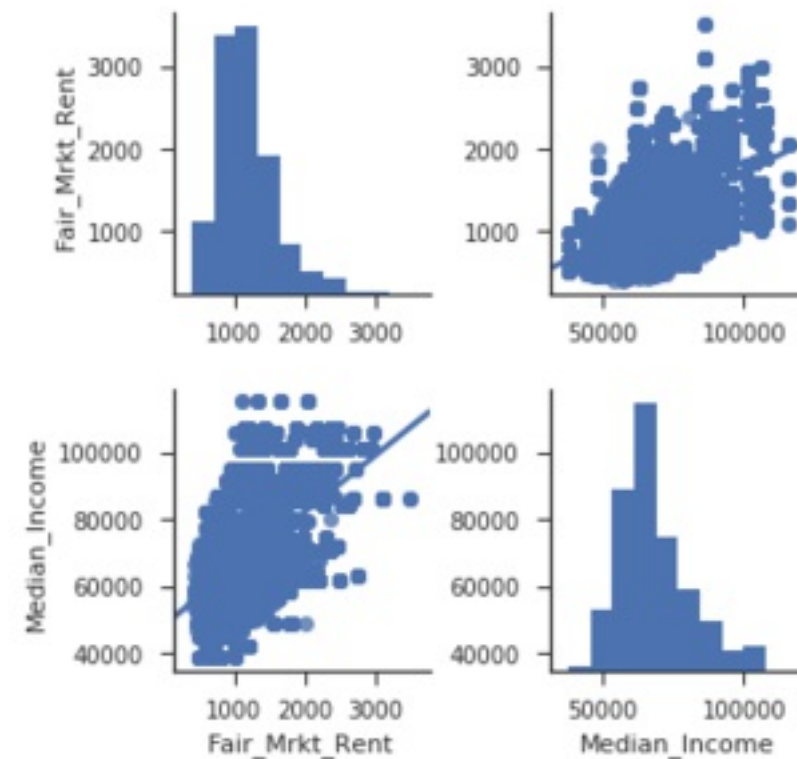




# pairplot

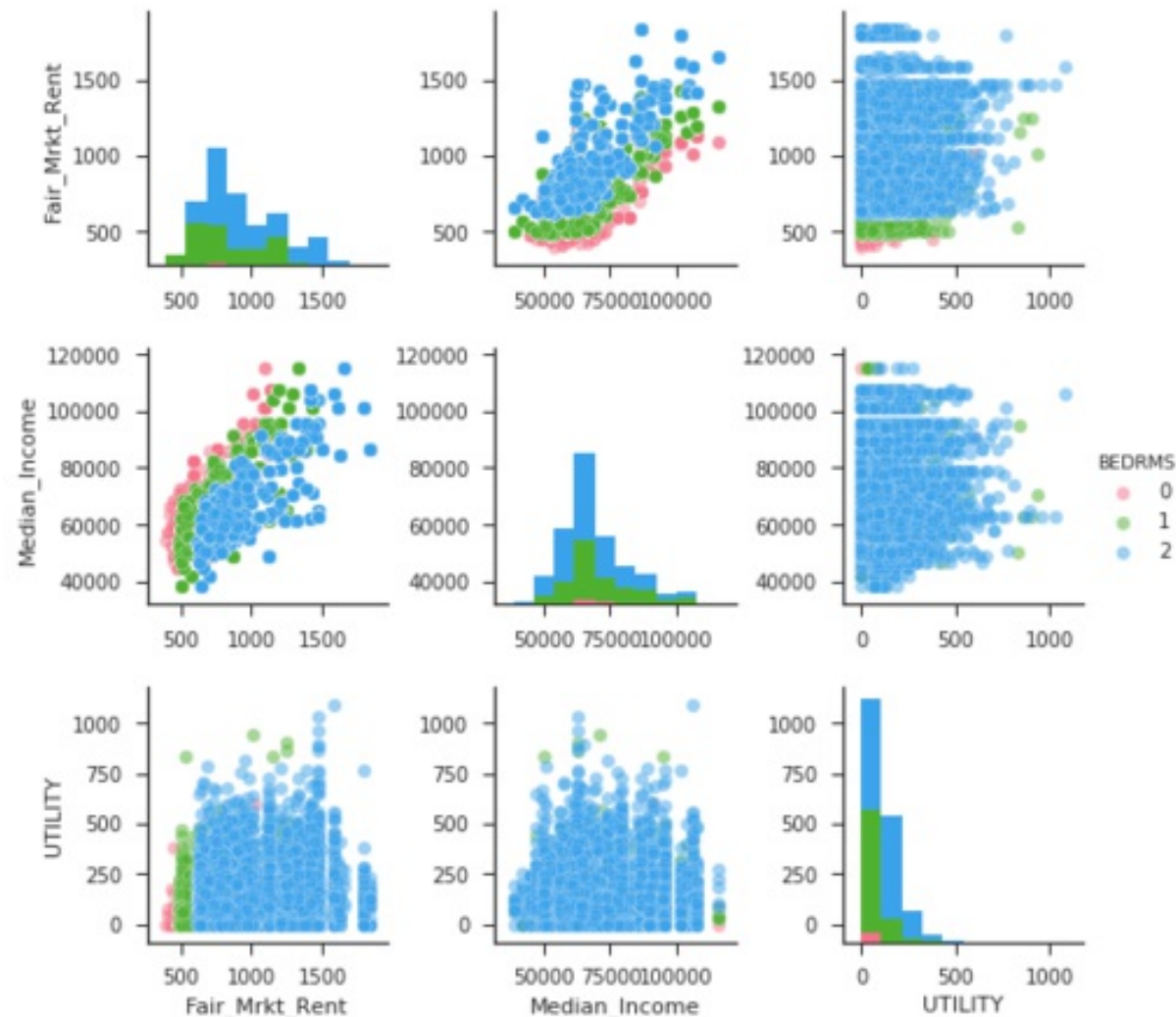
- pairplot is a shortcut for the PairGrid

```
sns.pairplot(df, vars=["Fair_Mrkt_Rent", "Median_Income"],  
              kind='reg', diag_kind='hist')
```



# Customizing a pairplot

```
sns.pairplot(df.query('BEDRMS < 3'),  
             vars=["Fair_Mrkt_Rent", "Median_Income", "UTILITY"],  
             hue='BEDRMS', palette='husl', plot_kws={'alpha': 0.5})
```







## DATA VISUALIZATION WITH SEABORN

**Let's practice!**

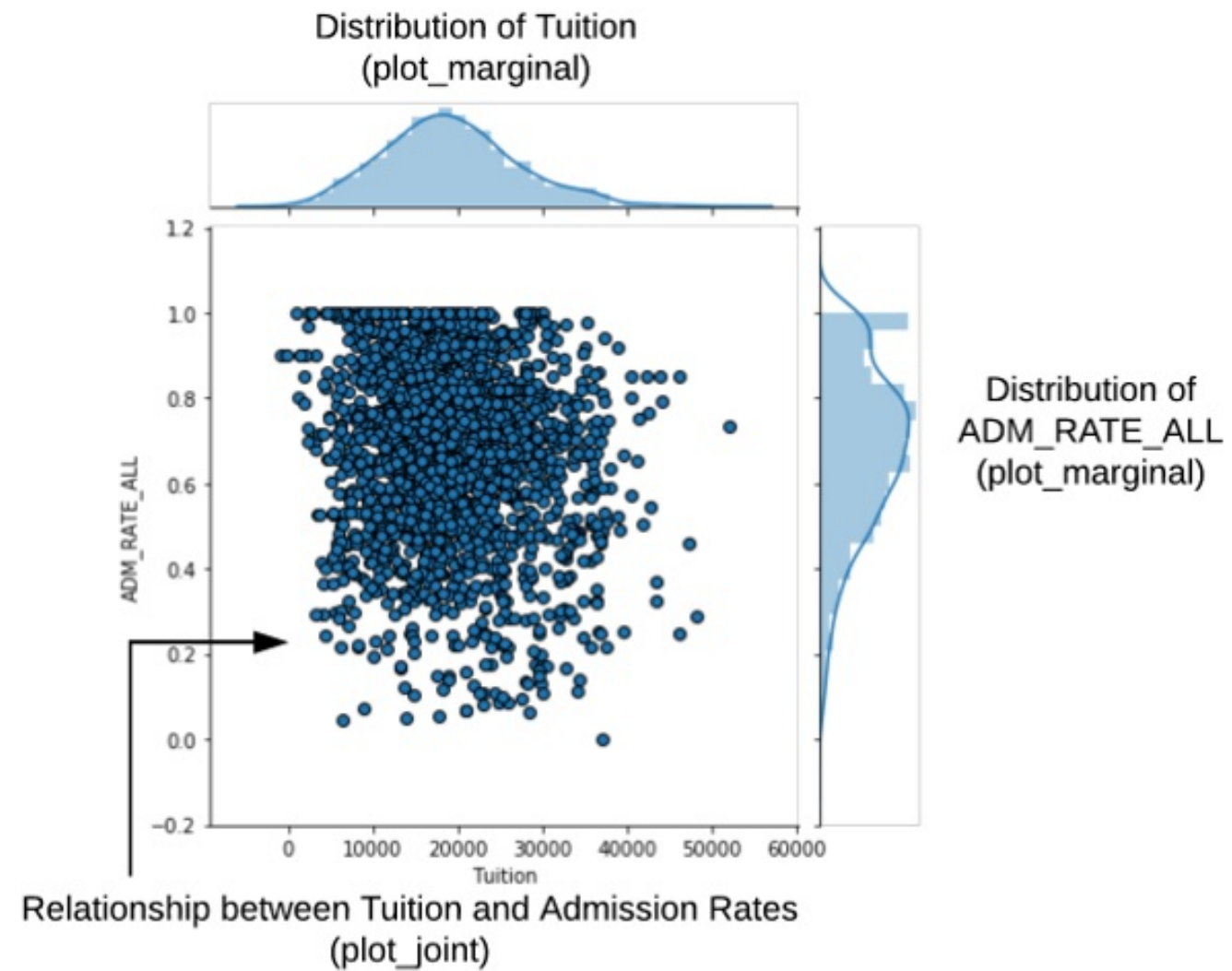


DATA VISUALIZATION WITH SEABORN

# Using JointGrid and jointplot

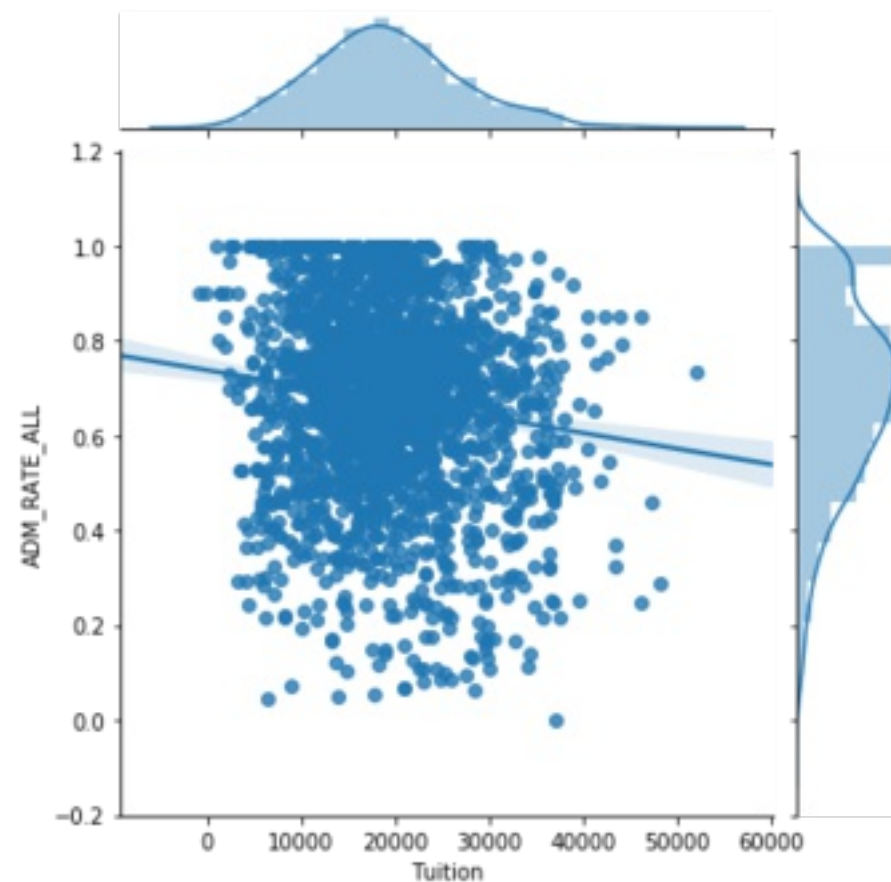
Chris Moffitt  
Instructor

# JointGrid() Overview



# Basic JointGrid

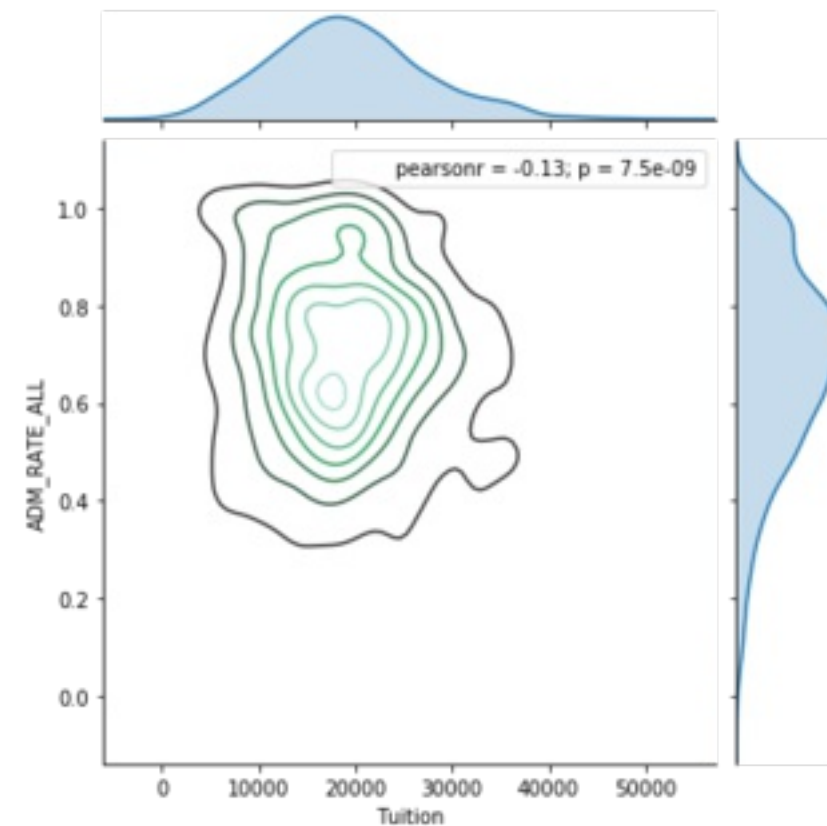
```
g = sns.JointGrid(data=df, x="Tuition", y="ADM_RATE_ALL")  
g.plot(sns.regplot, sns.distplot)
```





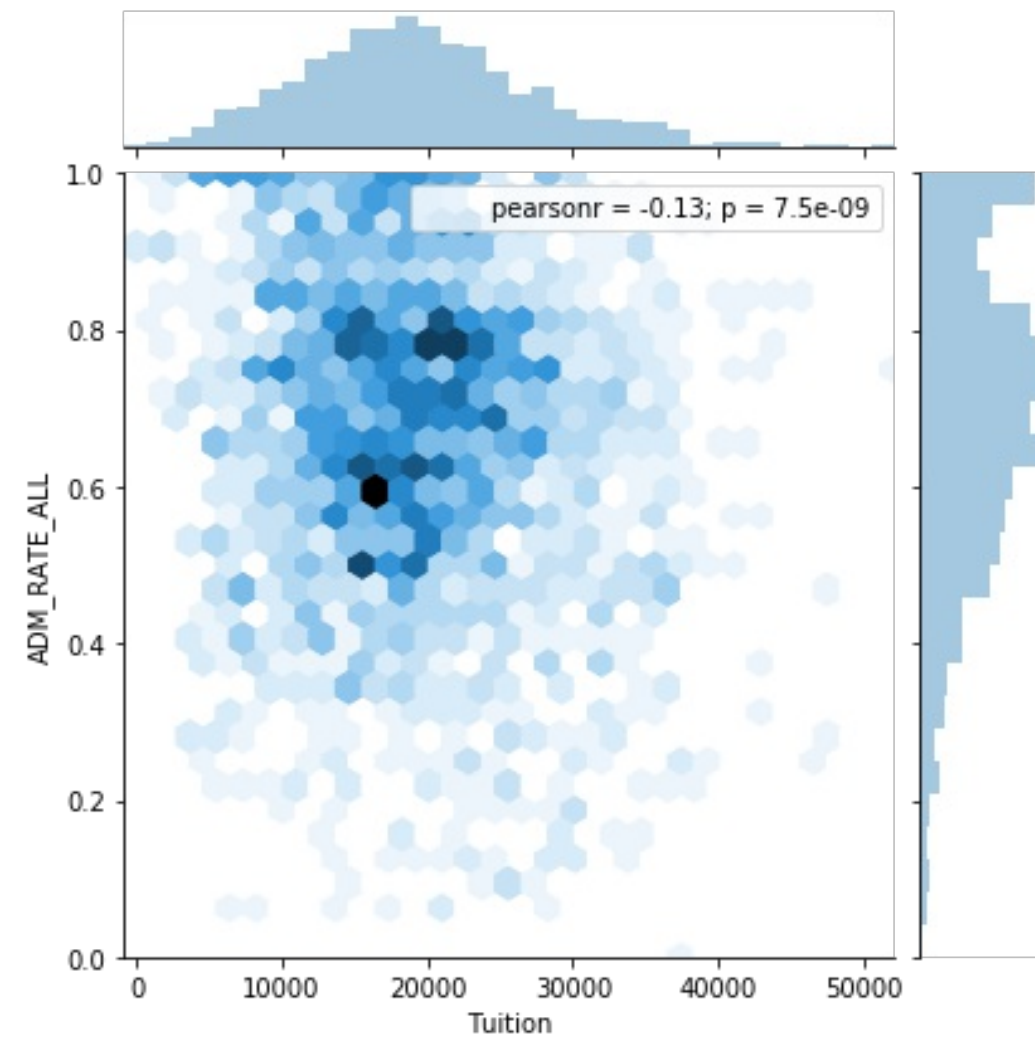
# Advanced JointGrid

```
g = sns.JointGrid(data=df, x="Tuition", y="ADM_RATE_ALL")
g = g.plot_joint(sns.kdeplot)
g = g.plot_marginals(sns.kdeplot, shade=True)
g = g.annotate(stats.pearsonr)
```



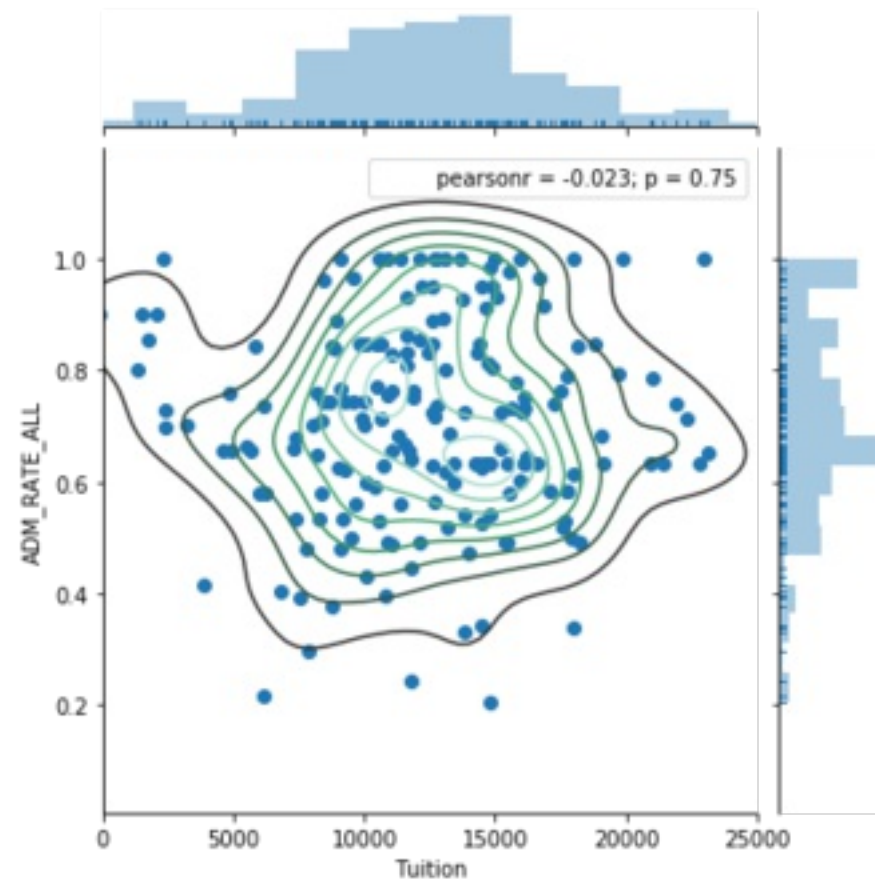
# jointplot()

```
sns.jointplot(data=df, x="Tuition", y="ADM_RATE_ALL", kind='hex')
```



# Customizing a jointplot

```
g = (sns.jointplot(x="Tuition", y="ADM_RATE_ALL", kind='scatter',  
                  xlim=(0, 25000), marginal_kws=dict(bins=15, rug=True),  
                  data=df.query('UG < 2500 & Ownership == "Public"'))  
     .plot_joint(sns.kdeplot))
```





## DATA VISUALIZATION WITH SEABORN

**Let's practice!**

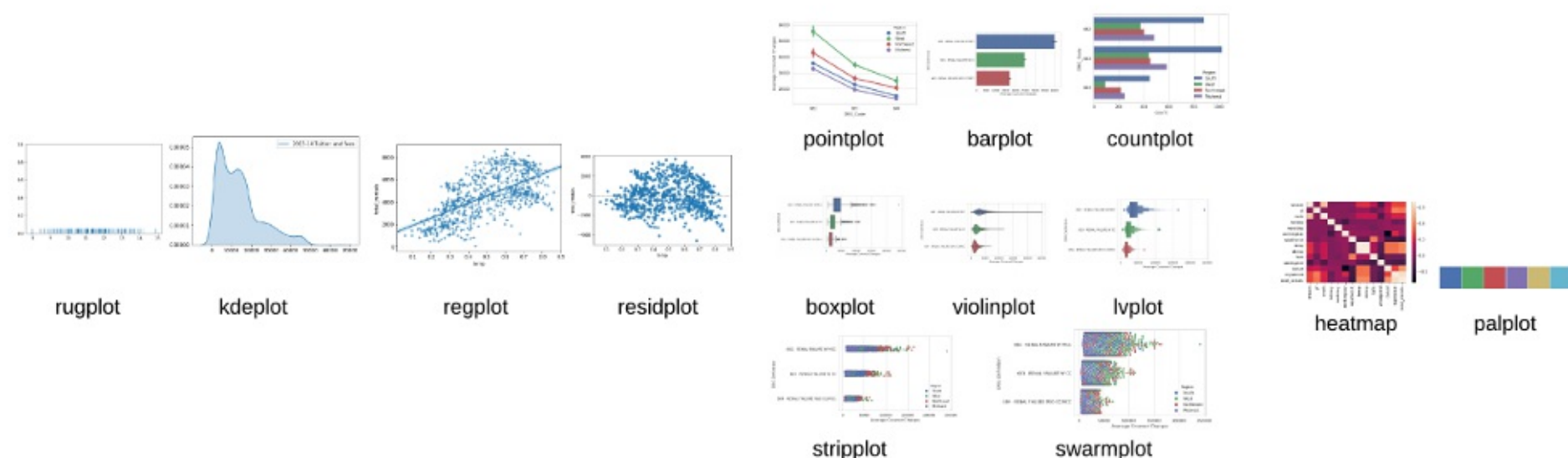
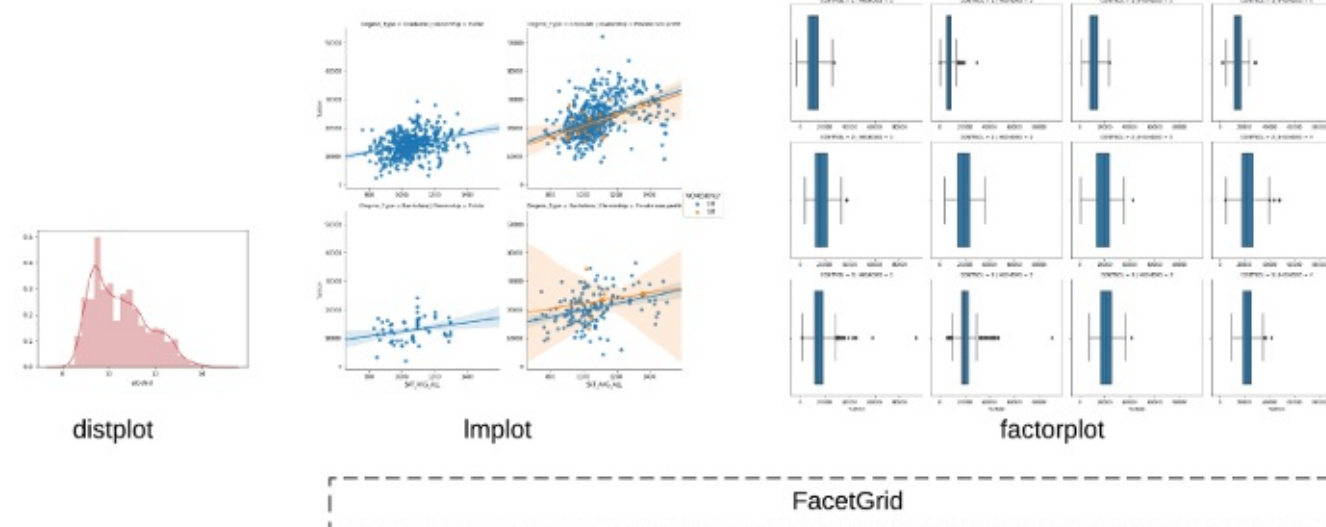
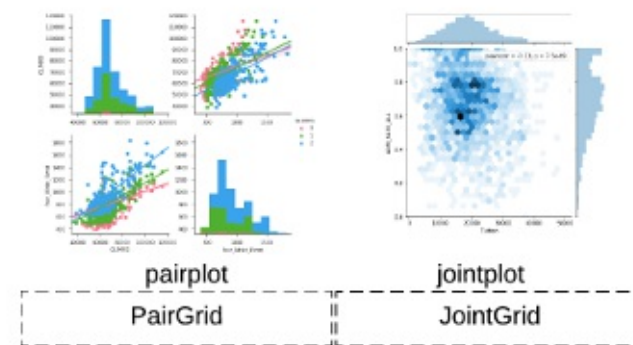




DATA VISUALIZATION WITH SEABORN

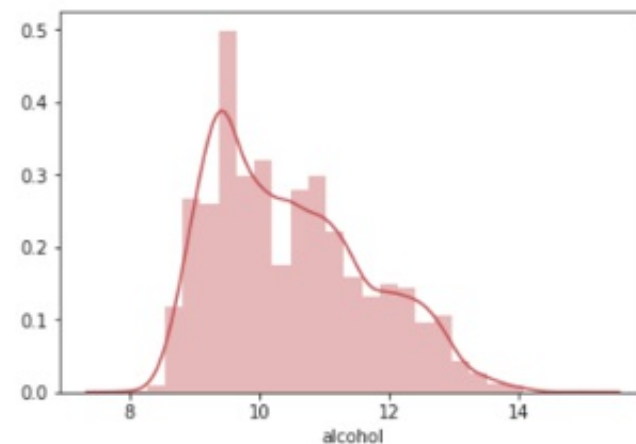
# Selecting Seaborn Plots

Chris Moffitt  
Instructor



# Univariate Distribution Analysis

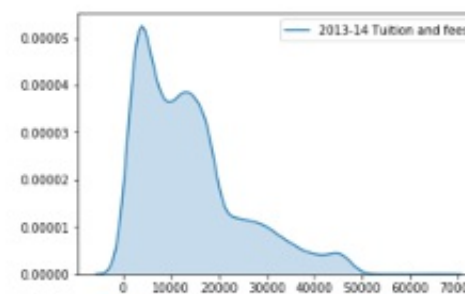
- `distplot()` is the best place to start for this analysis
- `rugplot()` and `kdeplot()` can be useful alternatives



`distplot`



`rugplot`



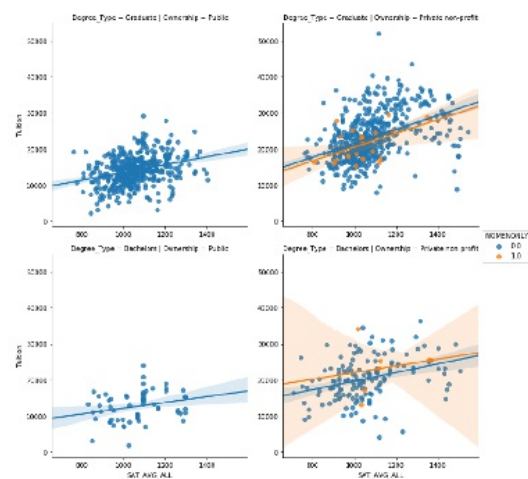
`kdeplot`

`plt.hist()`

matplotlib

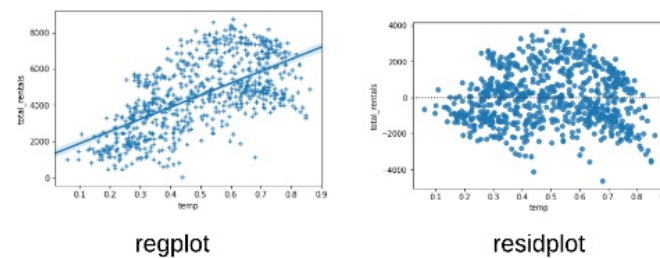
# Regression Analysis

- Implot() performs regression analysis and supports facetting



Implot

FacetGrid



regplot

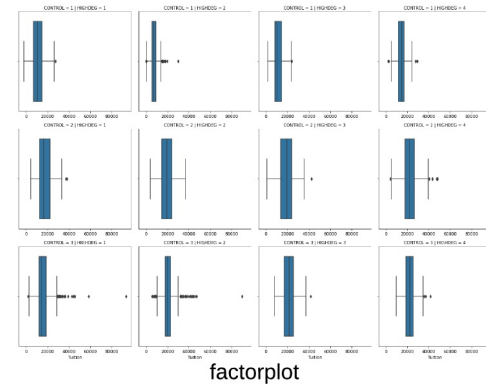
residplot

plt.scatter()

matplotlib

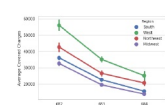
# Categorical Plots

- Explore data with the categorical plots and facet with factorplot

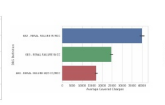


factorplot

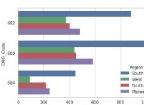
FacetGrid



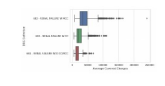
pointplot



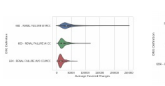
barplot



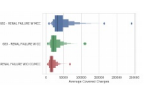
countplot



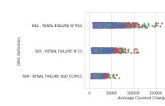
boxplot



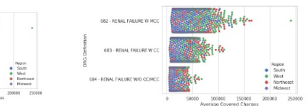
violinplot



lvplot



stripplot

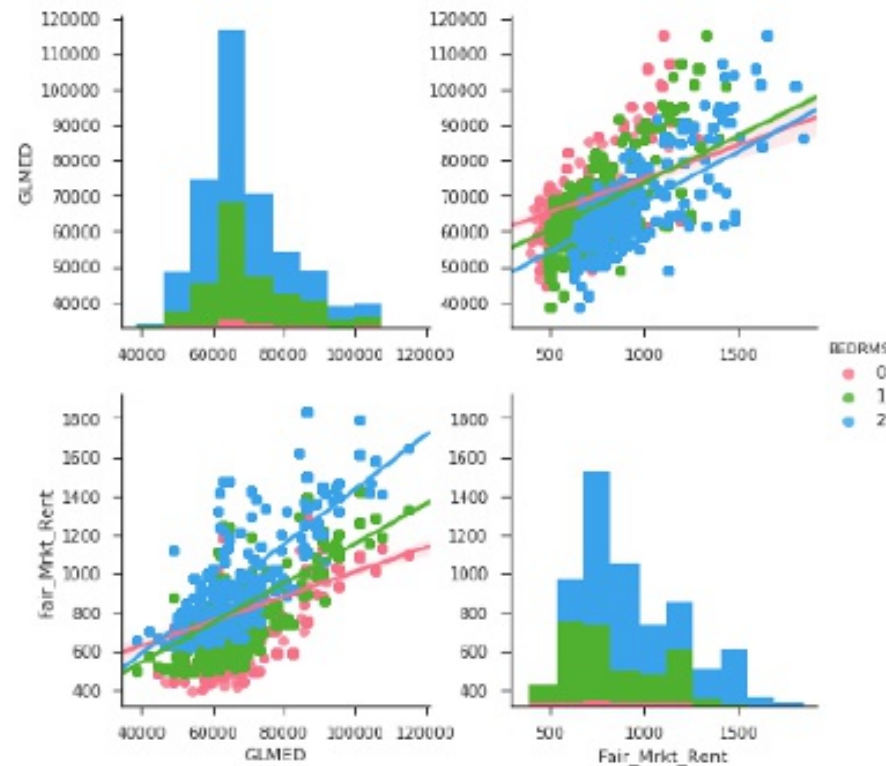


swarmplot



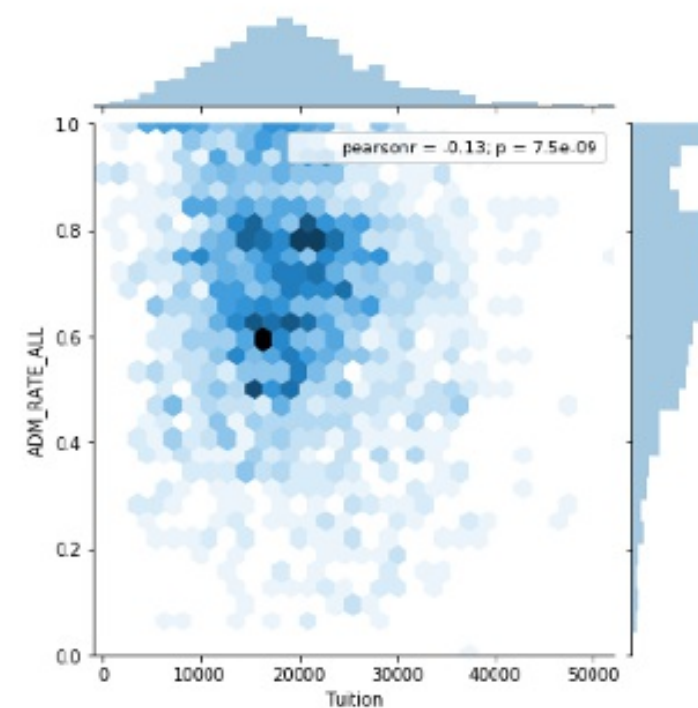
# pairplot and jointplot

- Perform regression analysis with Implot
- Analyze distributions with distplot



pairplot

PairGrid



jointplot

JointGrid



DATA VISUALIZATION WITH SEABORN

**Thank You!**