Categorical Plot Types

INTERMEDIATE DATA VISUALIZATION WITH SEABORN

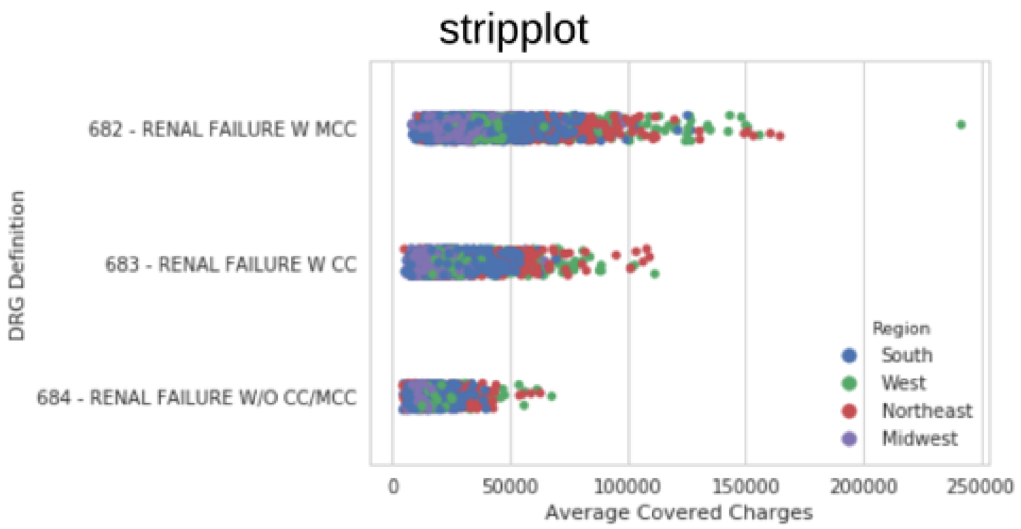
Chris Moffitt
Instructor

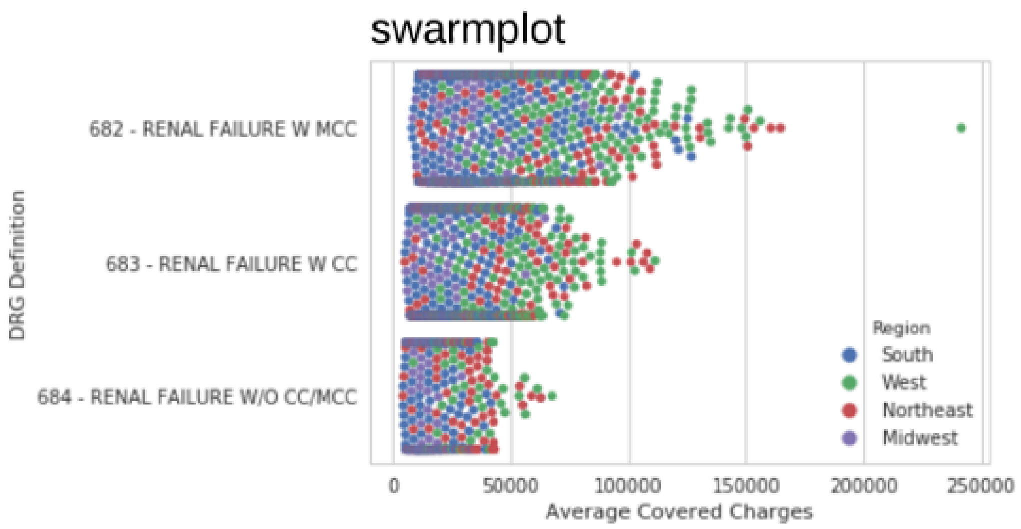
DataCamp

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

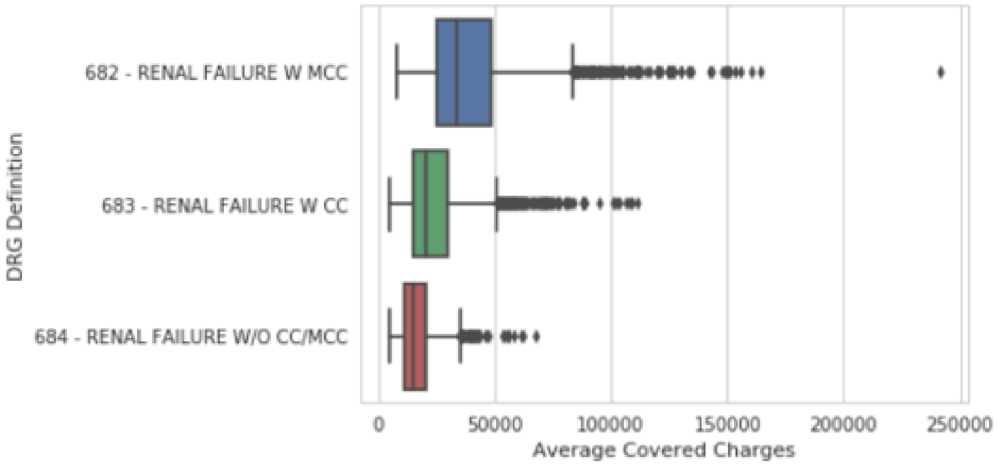
Plot types - show each observation



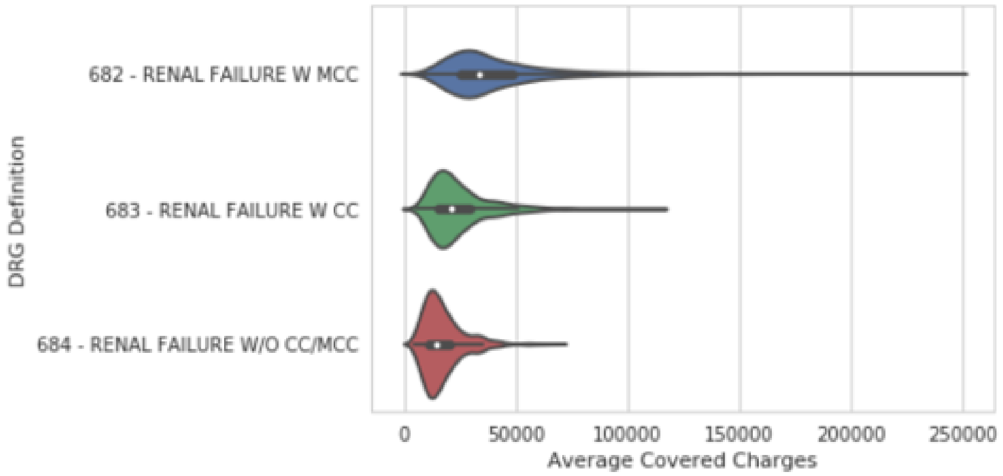


Plot types - abstract representations

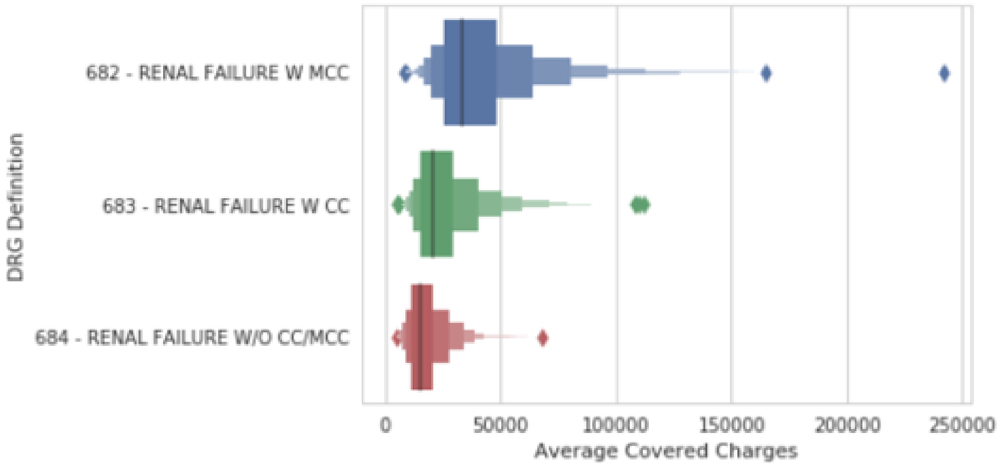
boxplot



violinplot

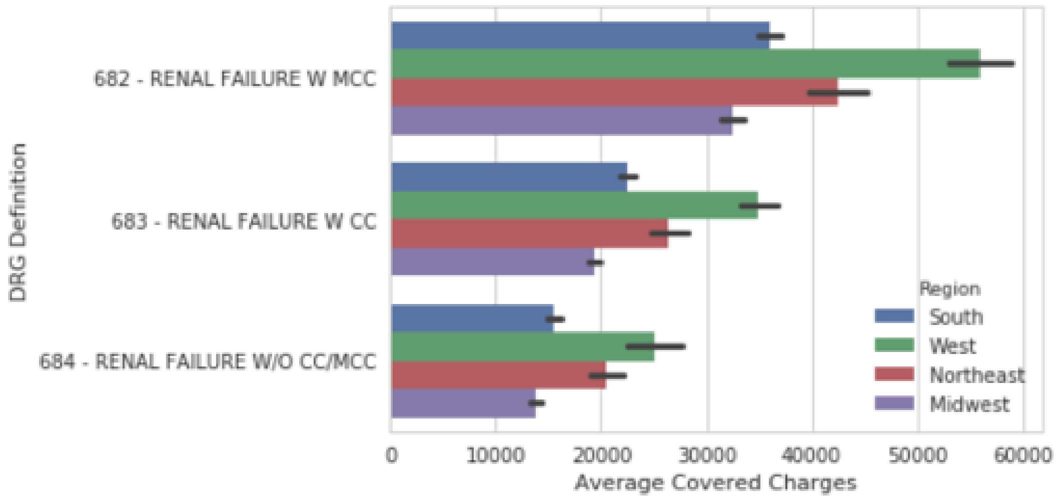


lvplot

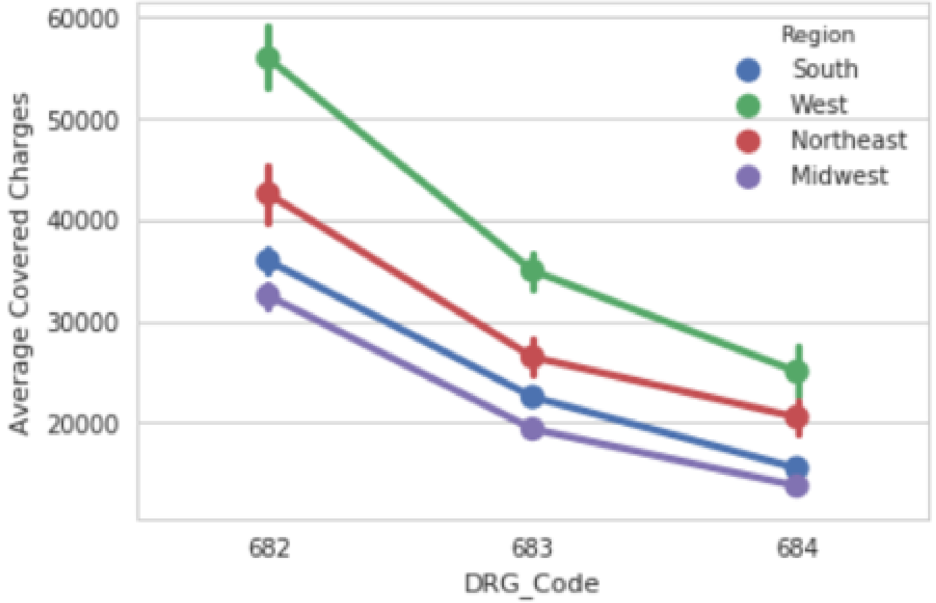


Plot types - statistical estimates

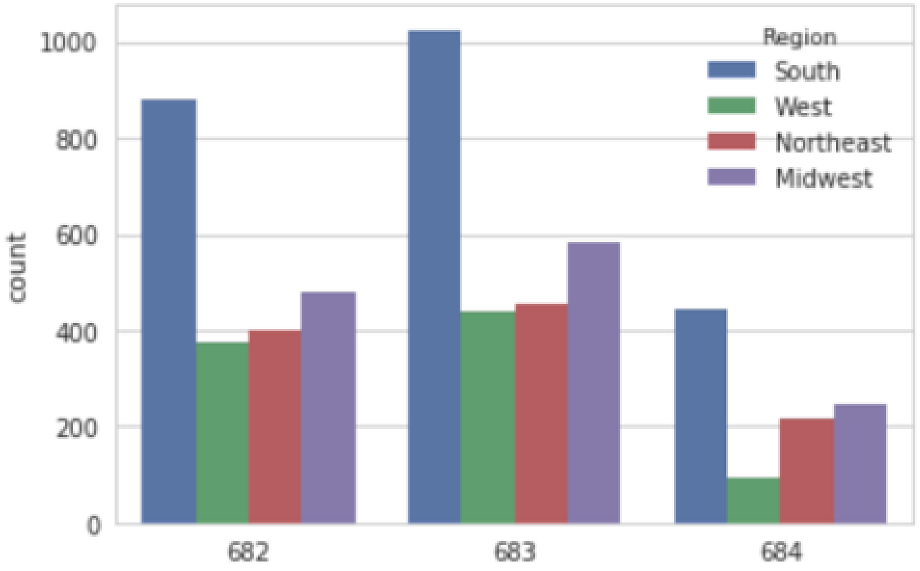
barplot



pointplot

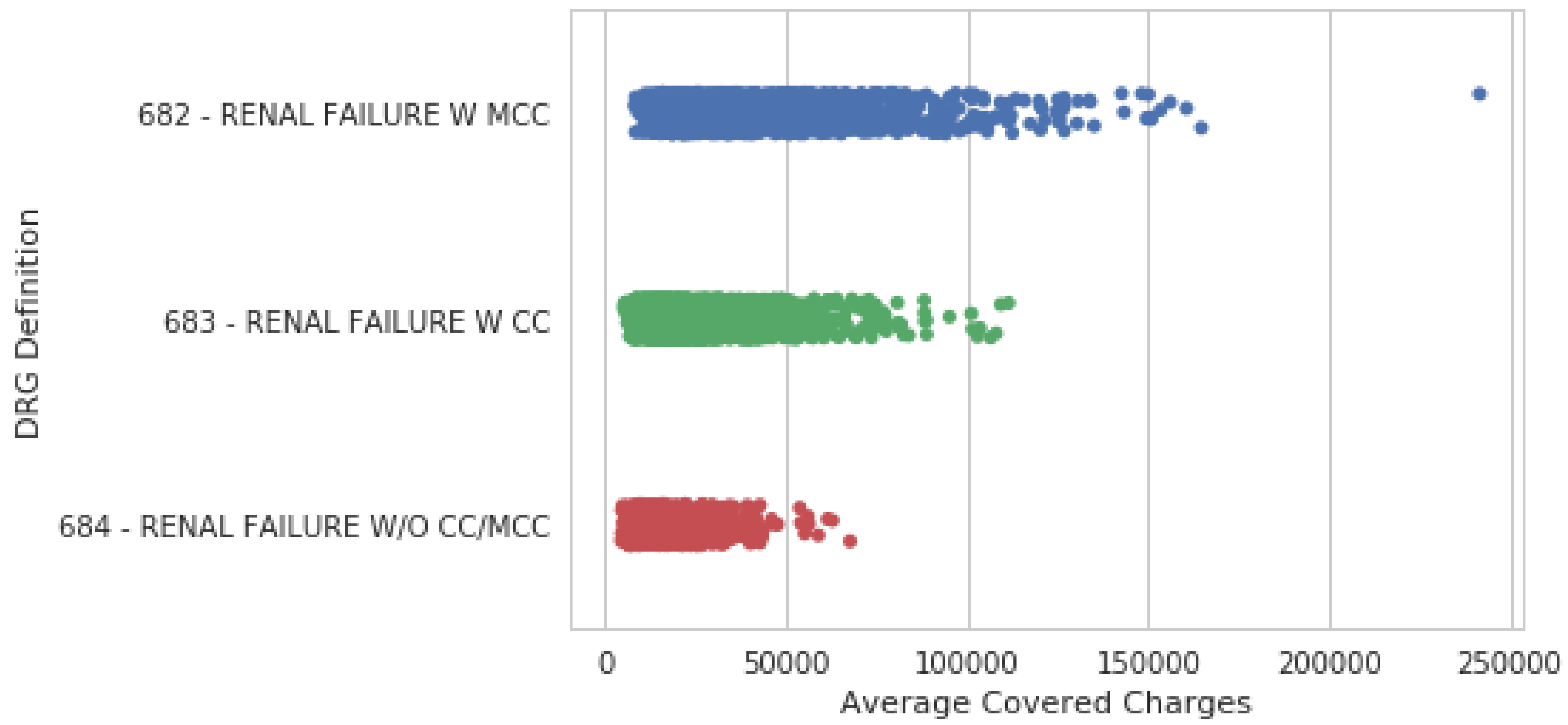


countplot



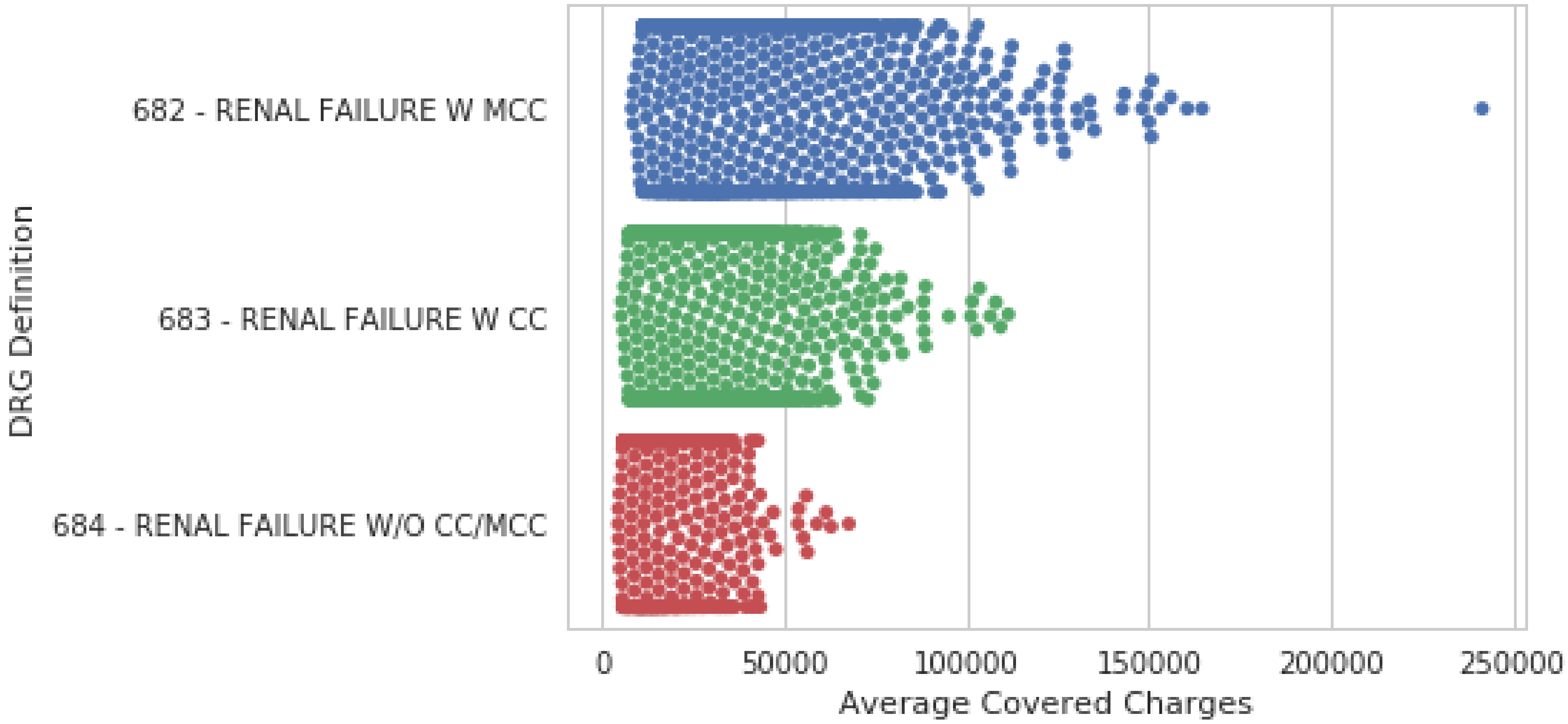
Plots of each observation - stripplot

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



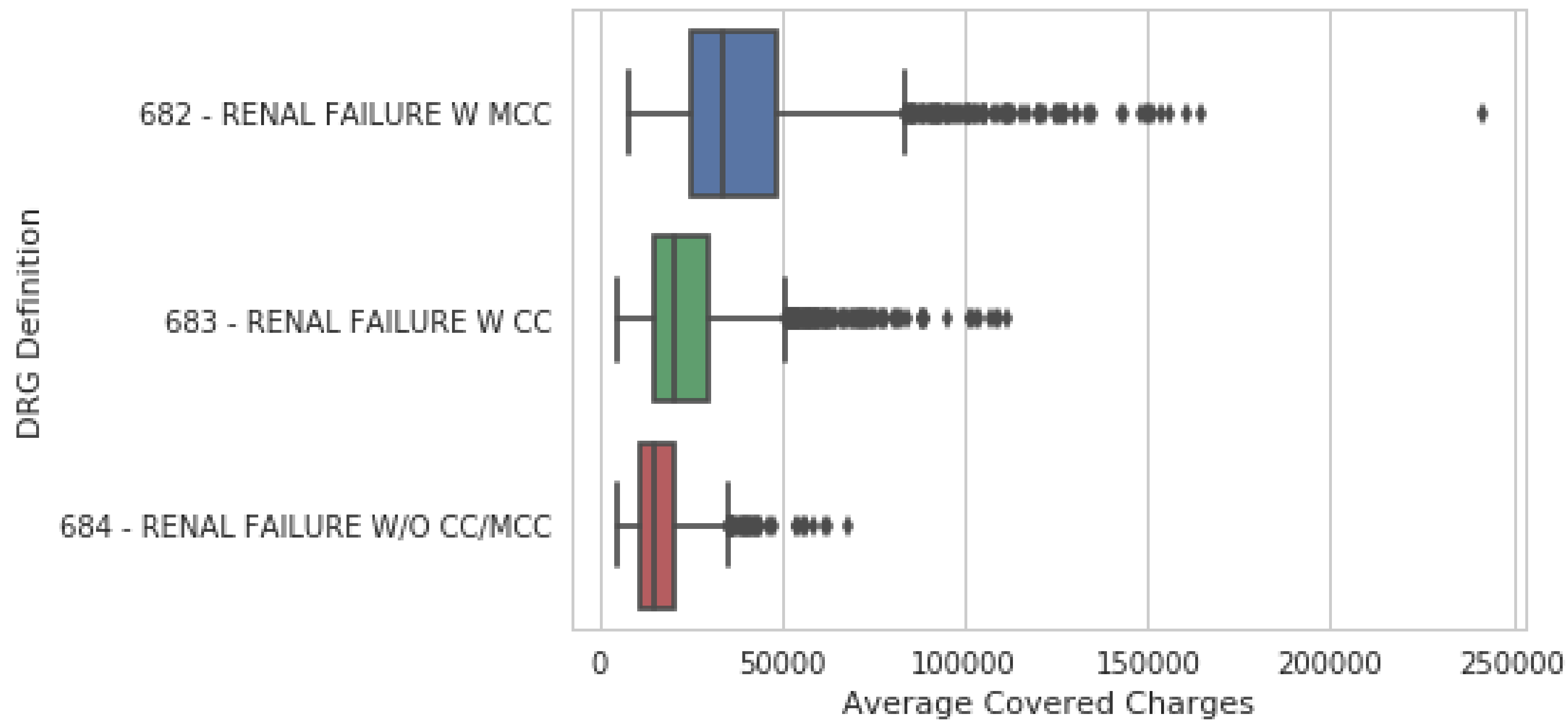
Plots of each observation - swarmplot

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



Abstract representations - boxplot

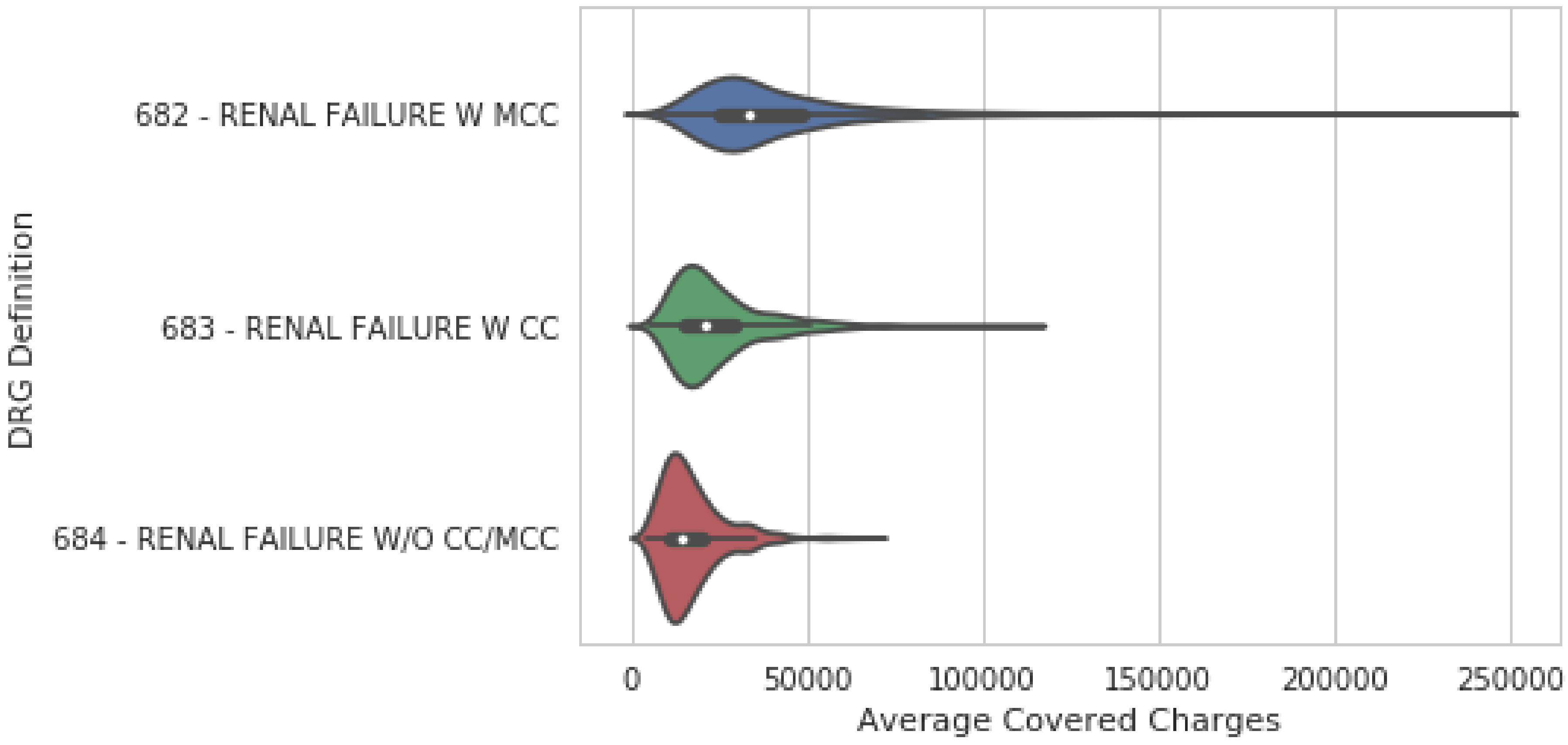
```
sns.boxplot(data=df, y="DRG Definition",
            x="Average Covered Charges")
```



Abstract representation - violinplot

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

kernel density plot + boxplot

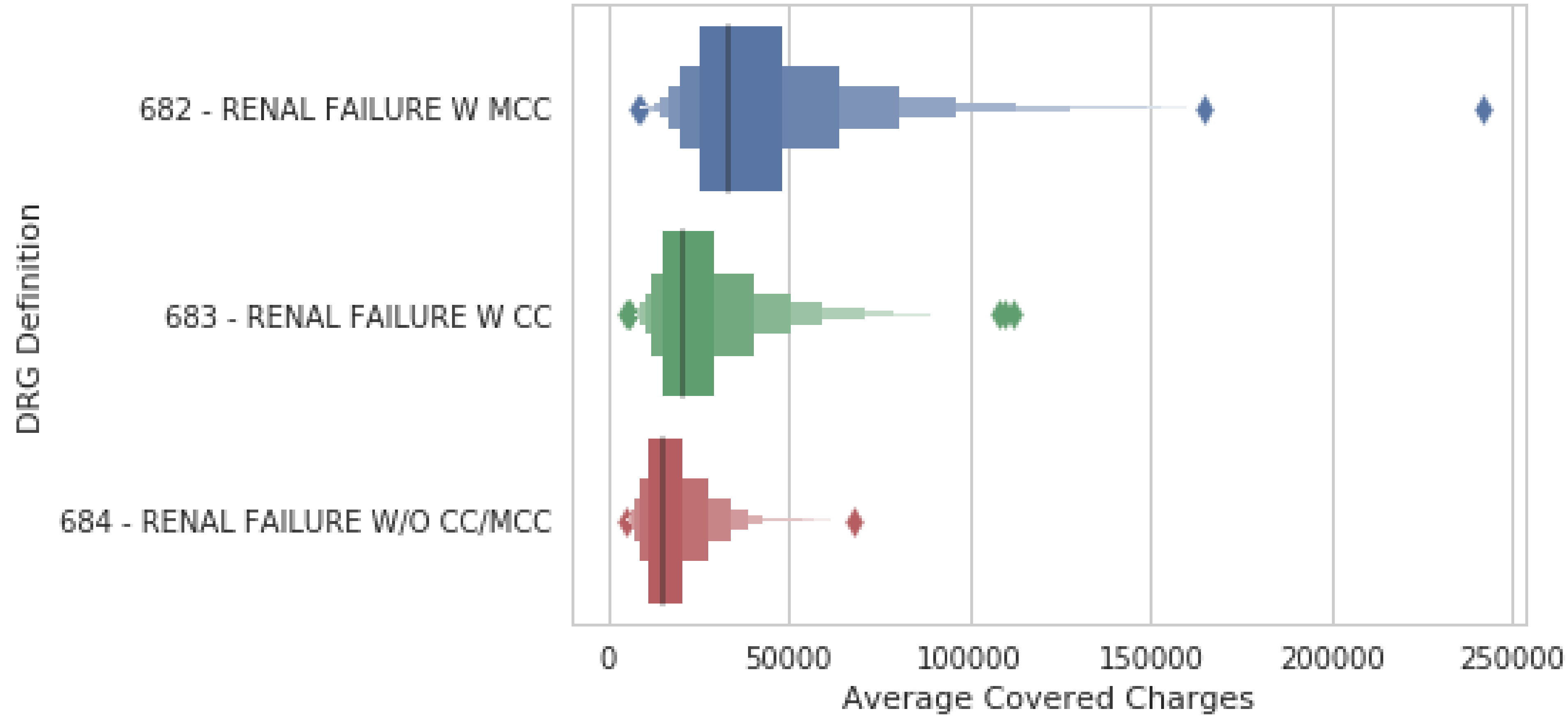


Abstract representation - lvplot

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

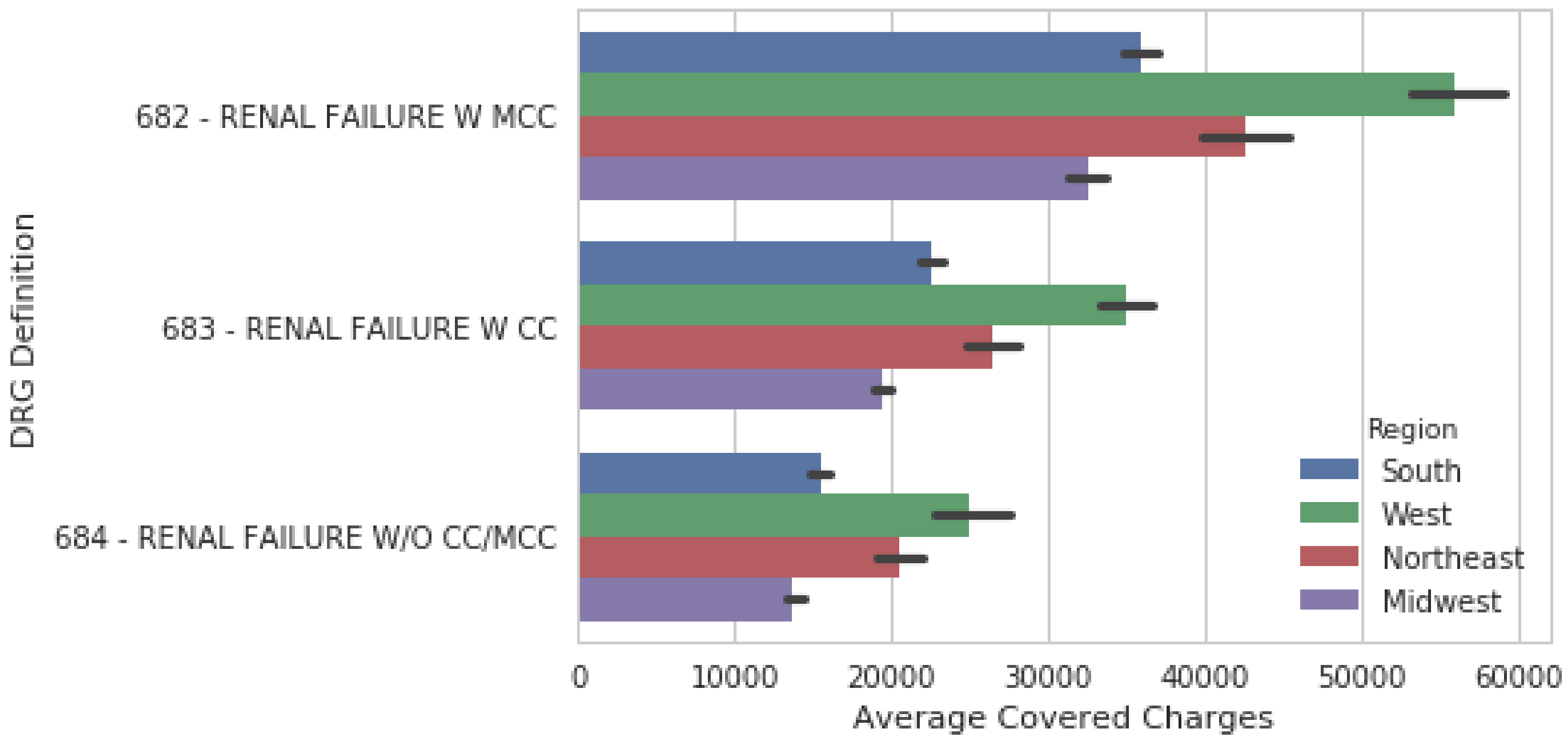
palette='Paired'

scale more effectively to large datasets



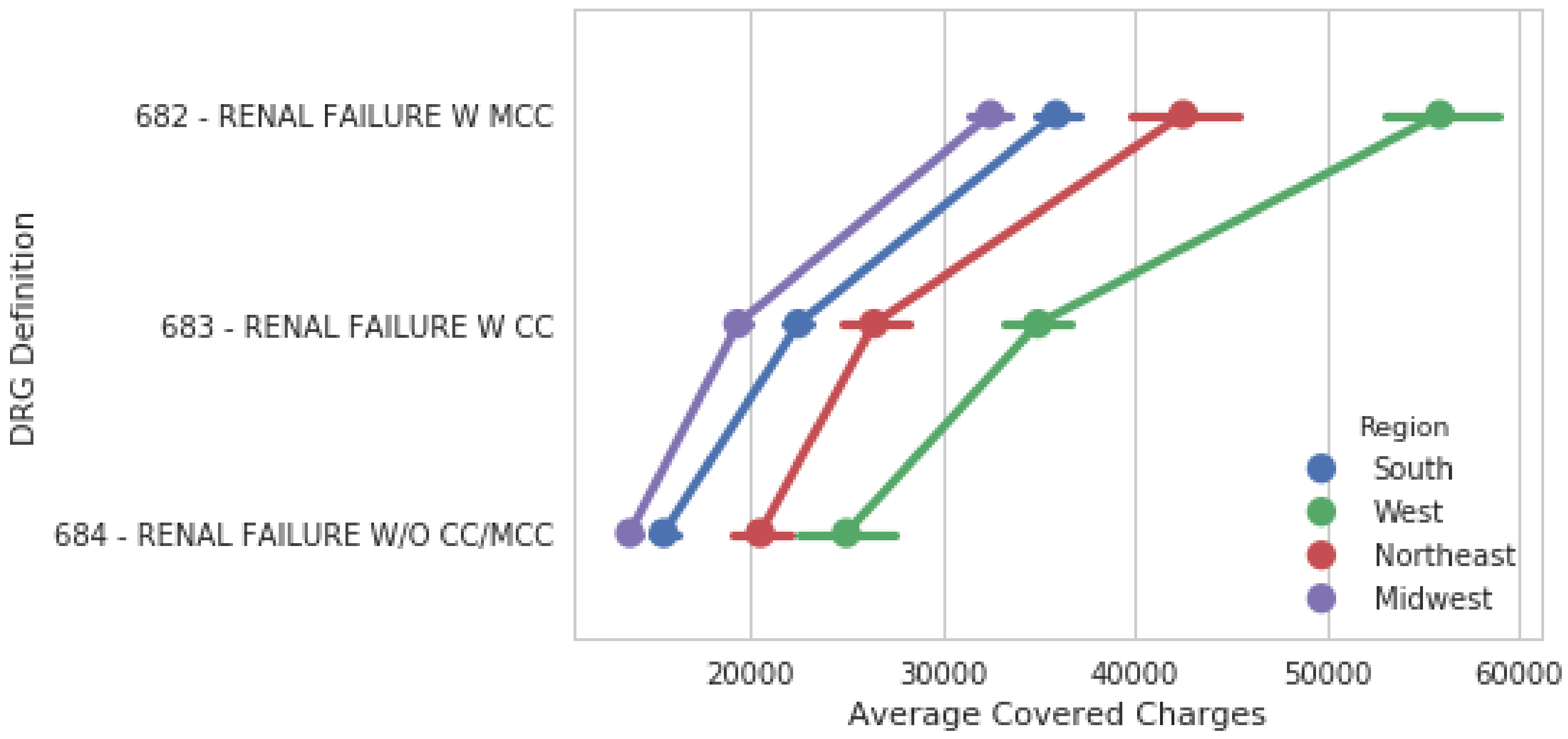
Statistical estimates - barplot

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



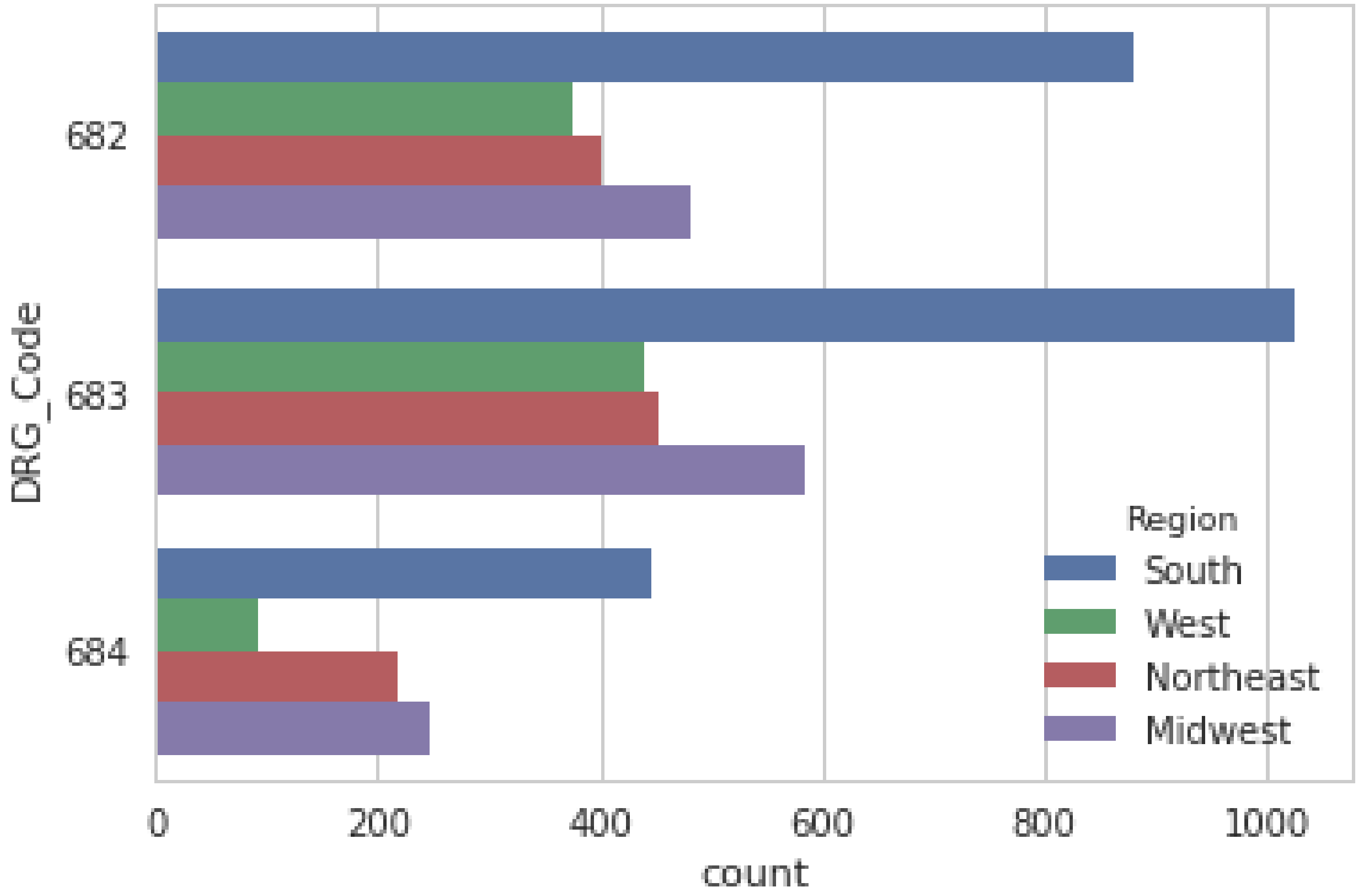
Statistical estimates - pointplot

```
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")
```



Let's practice!

INTERMEDIATE DATA VISUALIZATION WITH SEABORN

Regression Plots

INTERMEDIATE DATA VISUALIZATION WITH SEABORN

Chris Moffitt

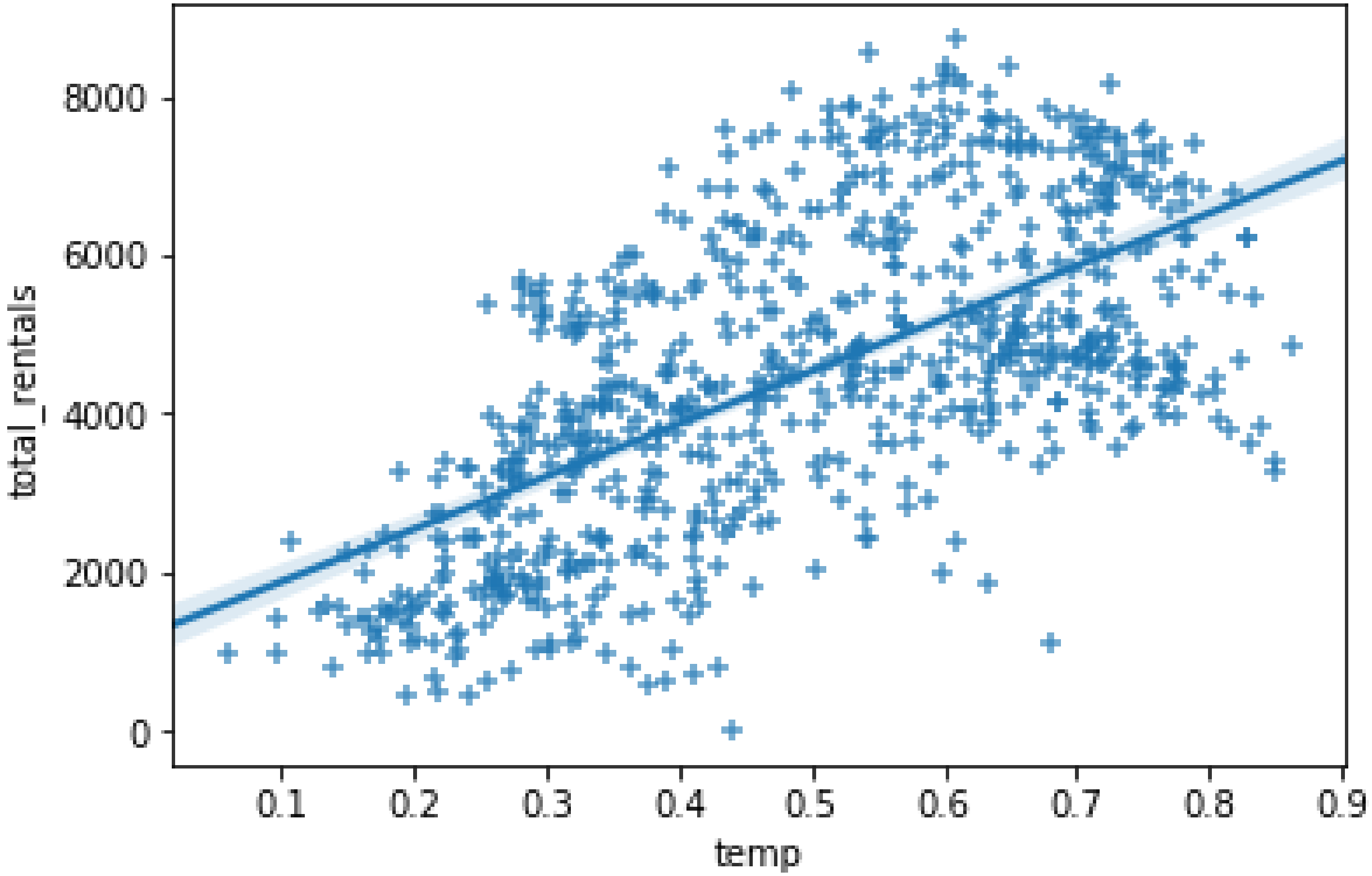
Instructor

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()

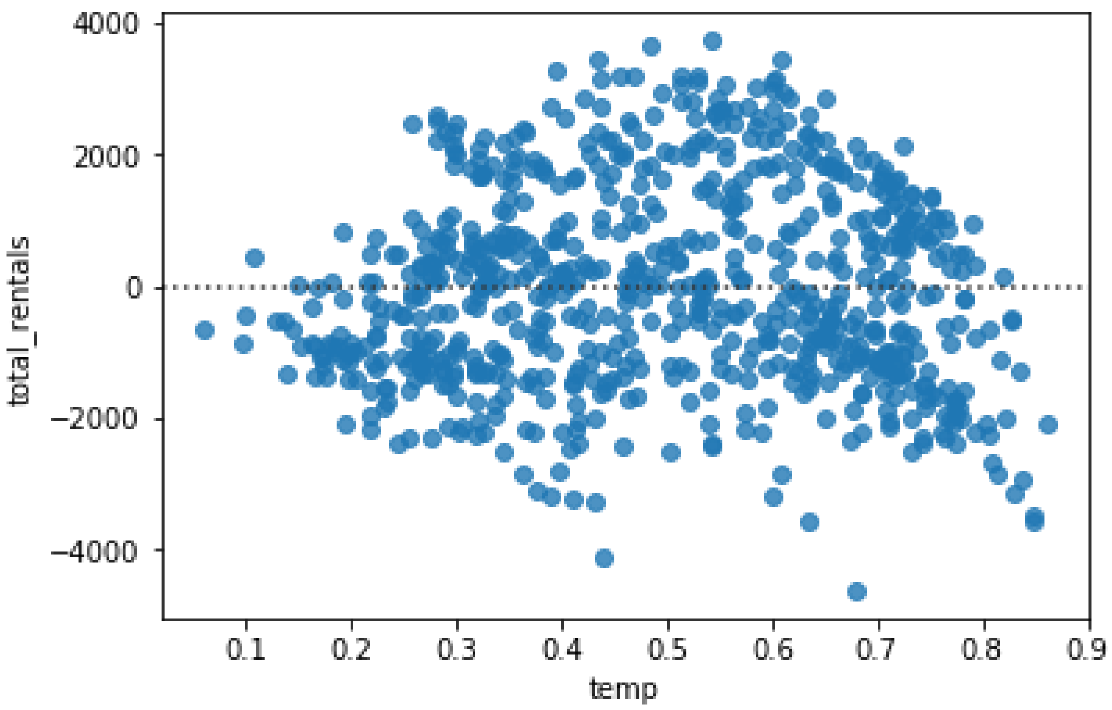
```
sns.regplot(data=df, x='temp',
            y='total_rentals', marker='+')
```



Evaluating regression with residplot()

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

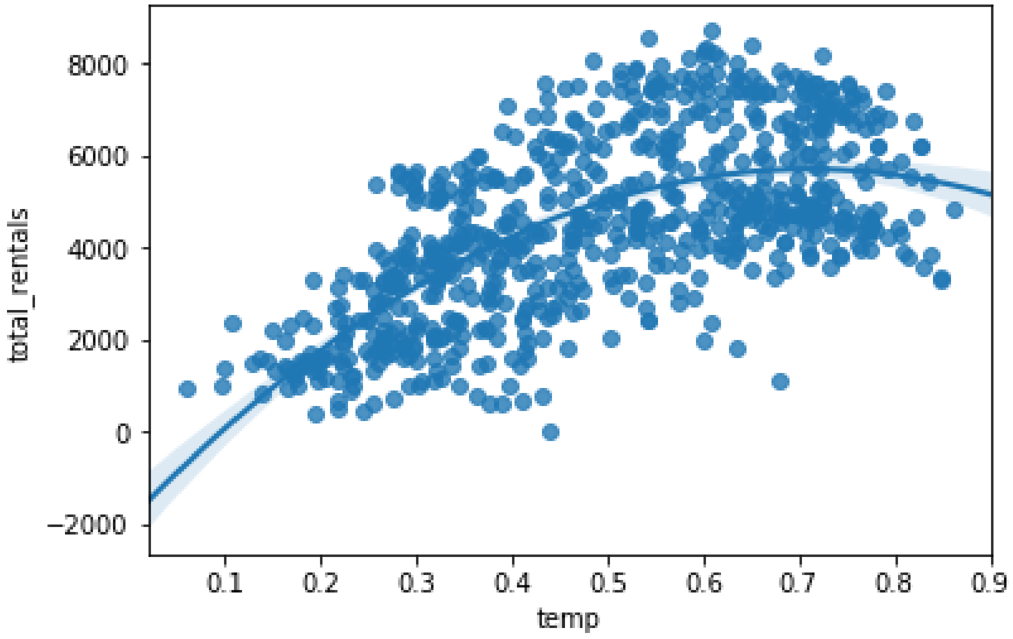
```
sns.residplot(data=df, x='temp', y='total_rentals')
```



Polynomial regression

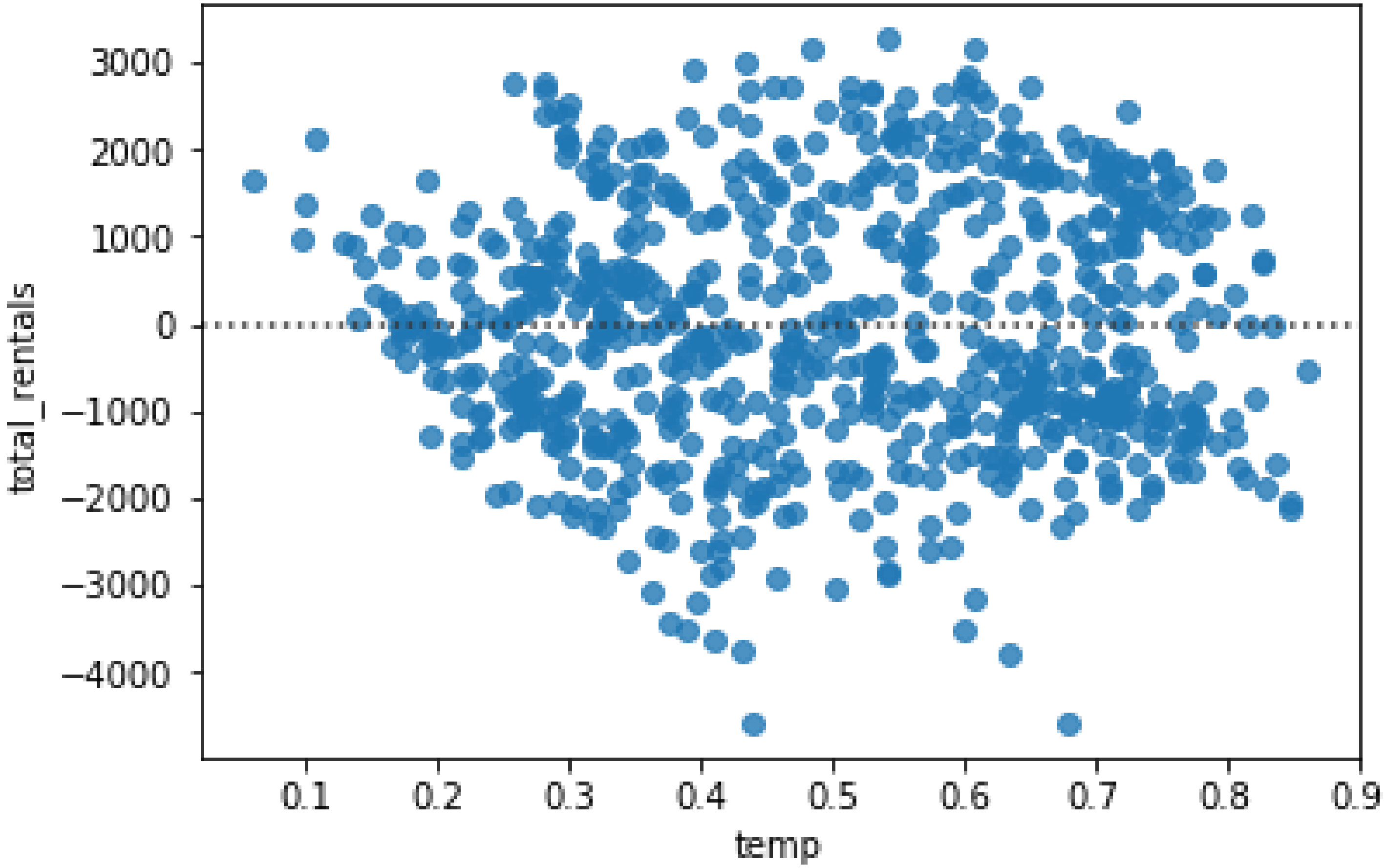
- Seaborn supports polynomial regression using the `order` parameter

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



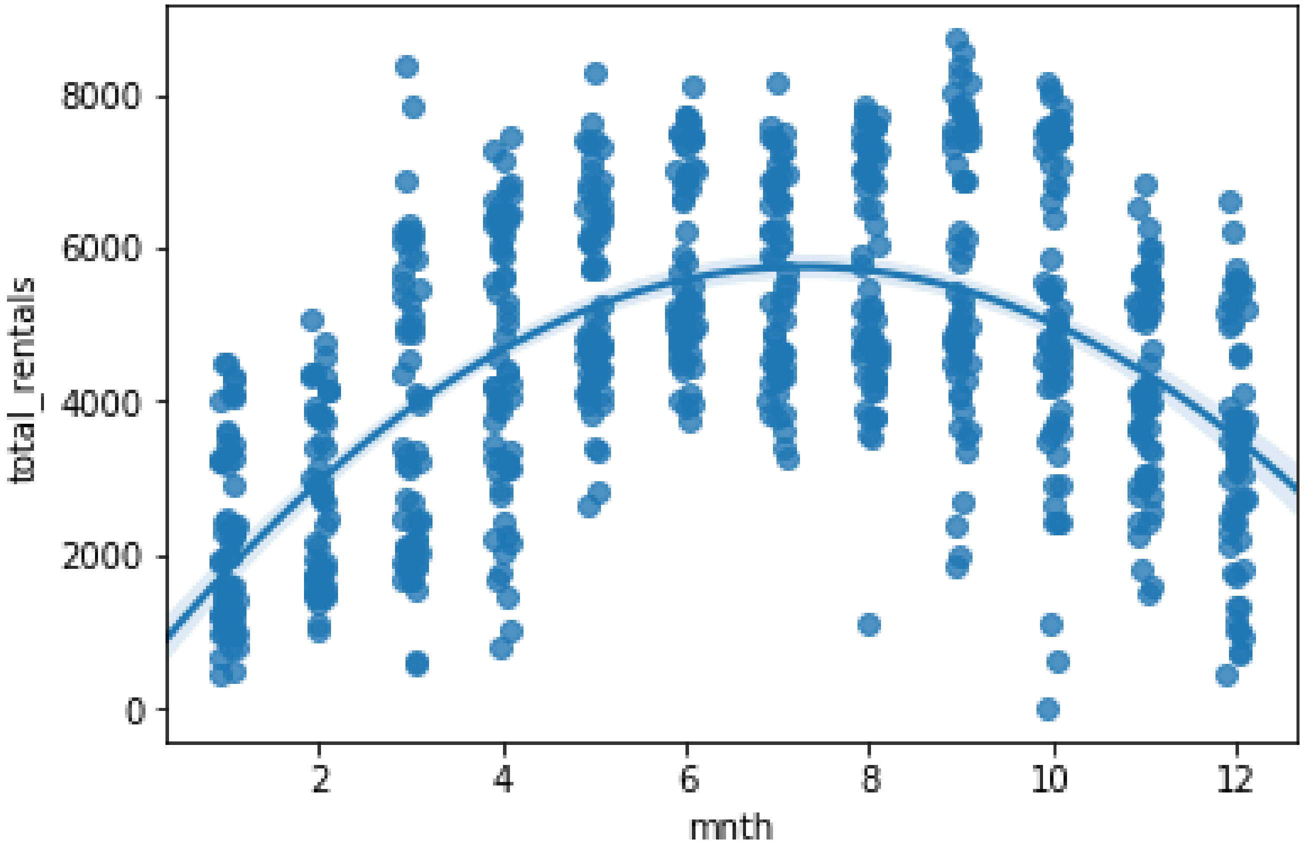
residplot with polynomial regression

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



Categorical values

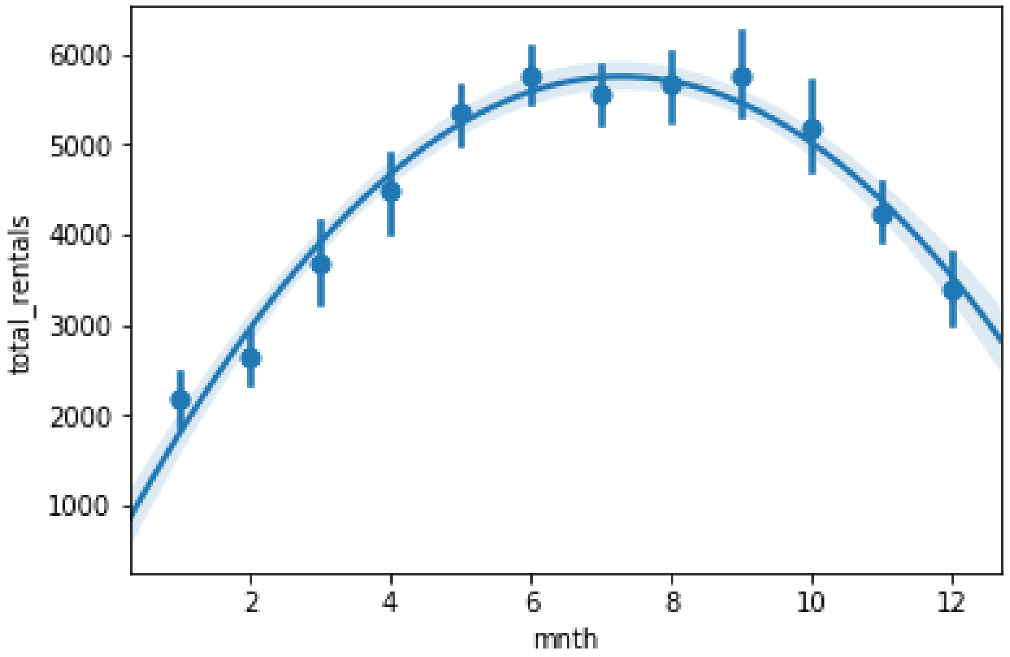
```
sns.regplot(data=df, x='mnth', y='total_rentals',
            x_jitter=.1, order=2)
```



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)
```

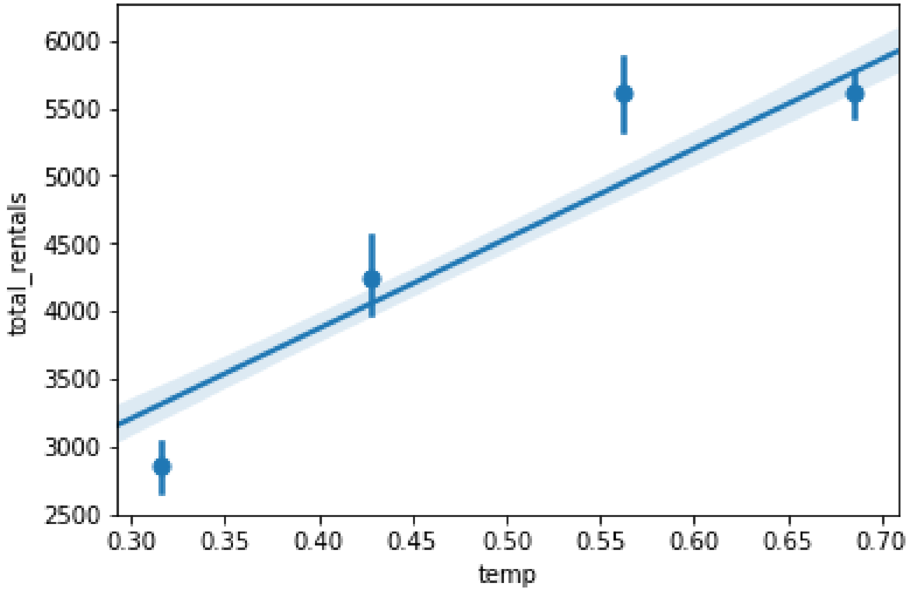


Binning the data

- `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)
```

When the data on the x axis is a continuous value, it can be useful to break it into different bins in order to get a better visualization of the changes in the data. set the fit_reg parameter to False to disable the regression line



Let's practice!

INTERMEDIATE DATA VISUALIZATION WITH SEABORN

Matrix Plots

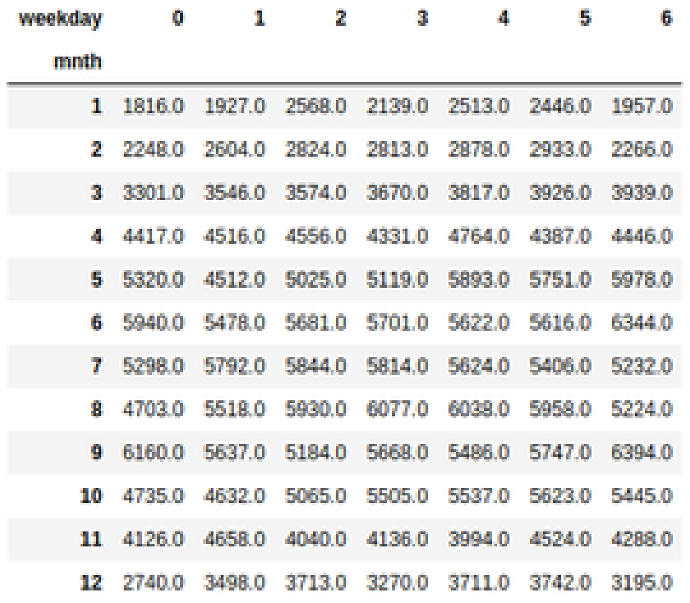
INTERMEDIATE DATA VISUALIZATION WITH SEABORN

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

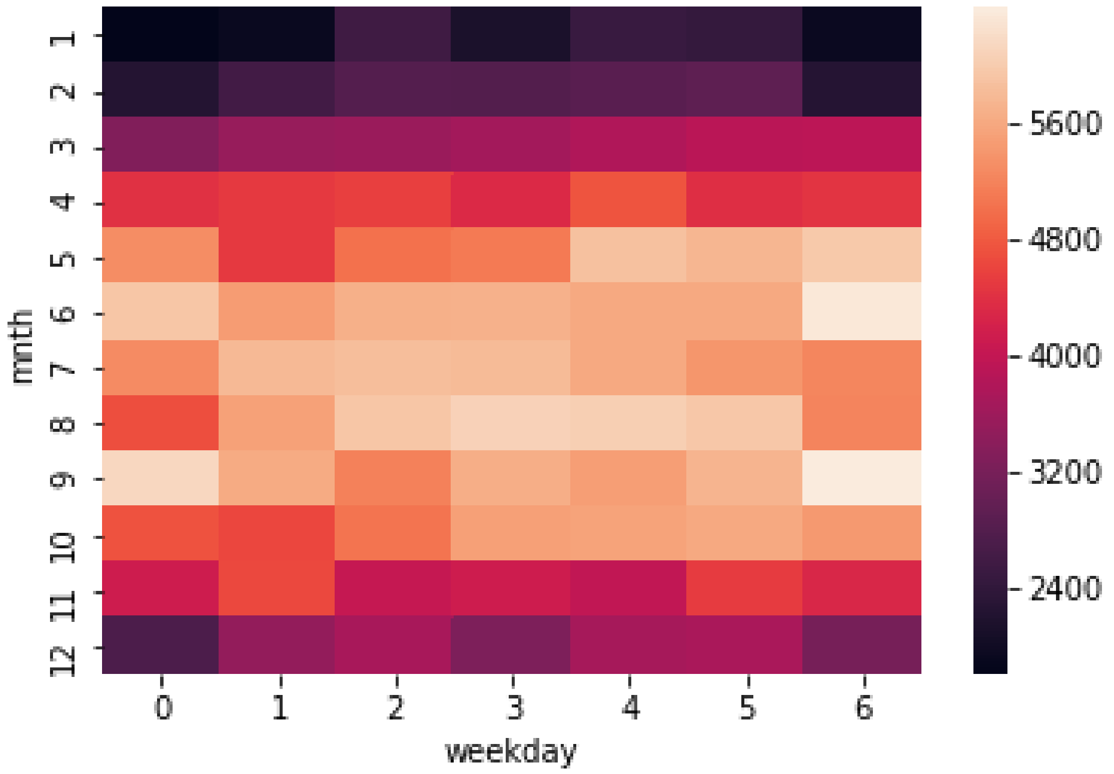
```
pd.crosstab(df["mnth"], df["weekday"],
values=df["total_rentals"],aggfunc='mean').round(0)
```



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

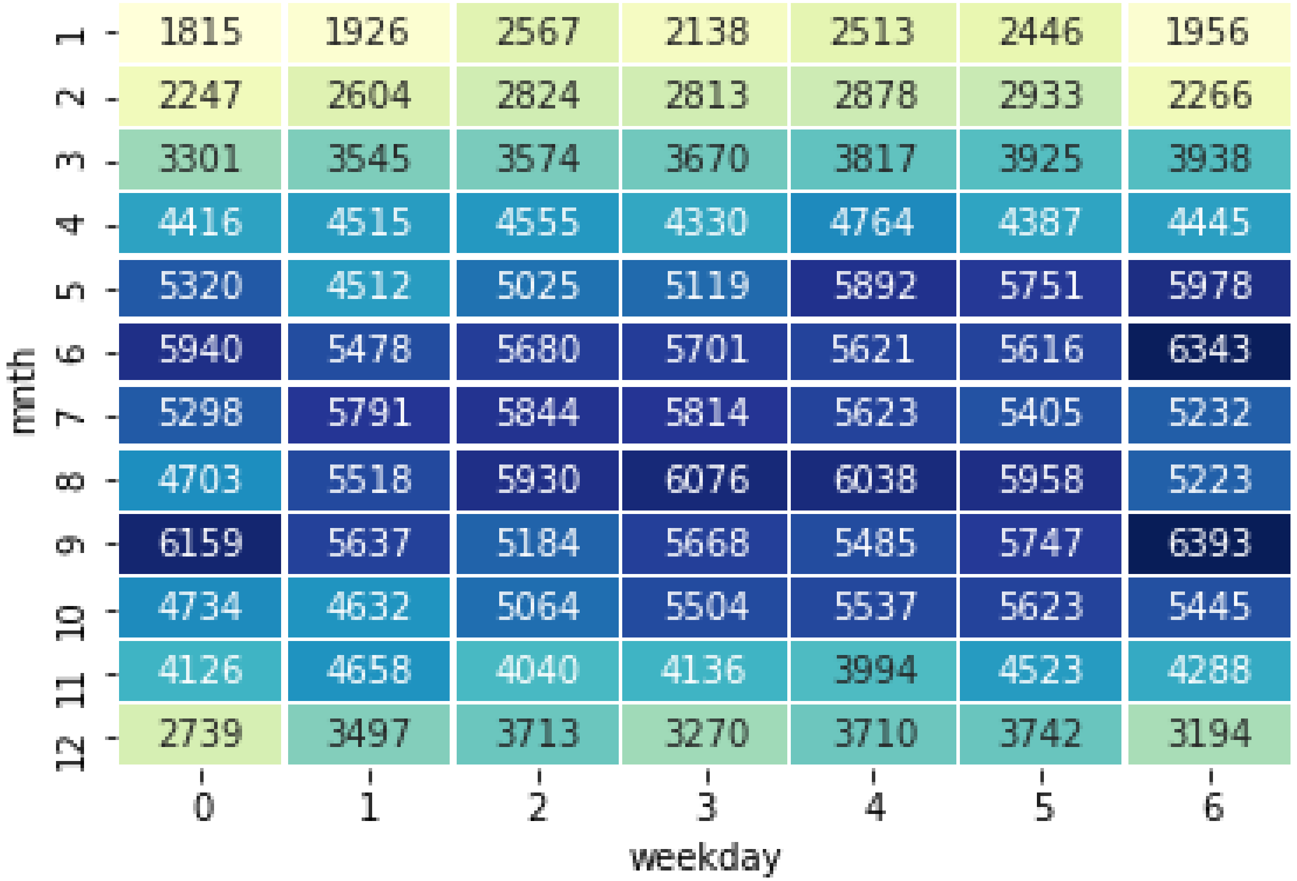
Build a heatmap

```
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)
```



mnth \ weekday	0	1	2	3	4	5	6
1	1815	1926	2567	2138	2513	2446	1956
2	2247	2604	2824	2813	2878	2933	2266
3	3301	3545	3574	3670	3817	3925	3938
4	4416	4515	4555	4330	4764	4387	4445
5	5320	4512	5025	5119	5892	5751	5978
6	5940	5478	5680	5701	5621	5616	6343
7	5298	5791	5844	5814	5623	5405	5232
8	4703	5518	5930	6076	6038	5958	5223
9	6159	5637	5184	5668	5485	5747	6393
10	4734	4632	5064	5504	5537	5623	5445
11	4126	4658	4040	4136	3994	4523	4288
12	2739	3497	3713	3270	3710	3742	3194

Centering a heatmap

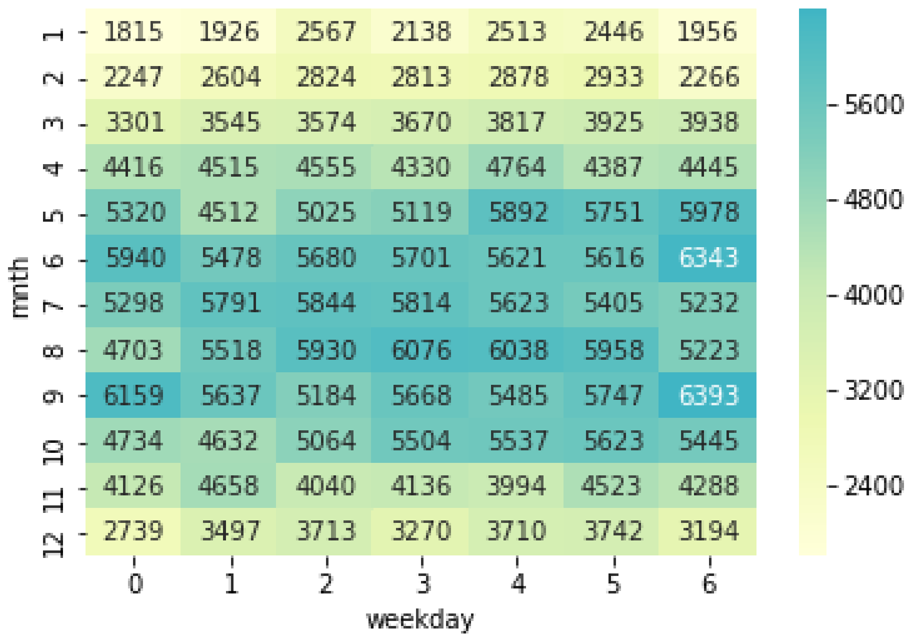
- Seaborn support centering the heatmap colors on a specific value

format: the results are displayed as integers

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

The color bar is not displayed.

put some small spacing between the cells
so that the values are simpler to view

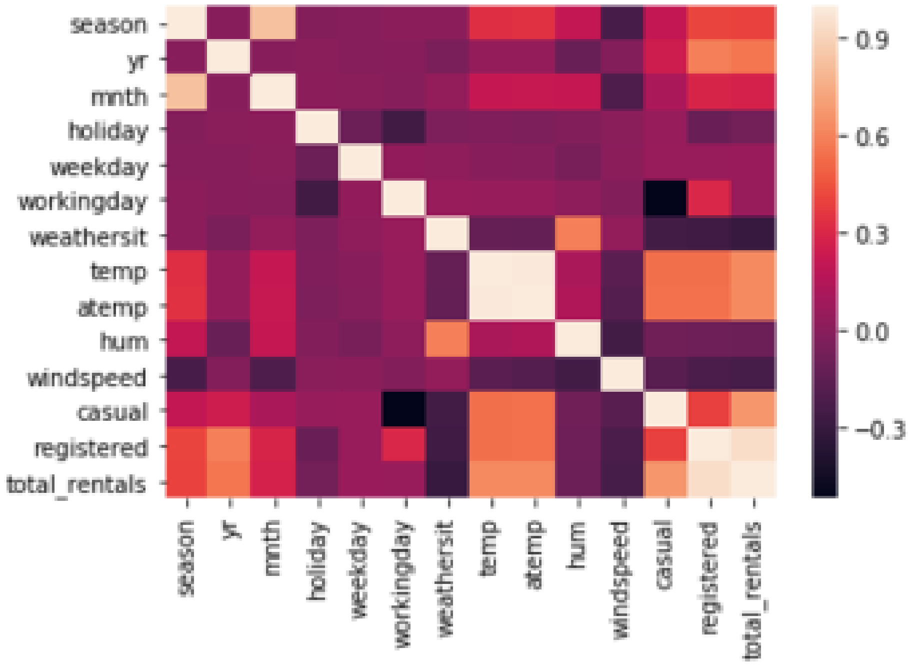


mnth	0	1	2	3	4	5	6
1	1815	1926	2567	2138	2513	2446	1956
2	2247	2604	2824	2813	2878	2933	2266
3	3301	3545	3574	3670	3817	3925	3938
4	4416	4515	4555	4330	4764	4387	4445
5	5320	4512	5025	5119	5892	5751	5978
6	5940	5478	5680	5701	5621	5616	6343
7	5298	5791	5844	5814	5623	5405	5232
8	4703	5518	5930	6076	6038	5958	5223
9	6159	5637	5184	5668	5485	5747	6393
10	4734	4632	5064	5504	5537	5623	5445
11	4126	4658	4040	4136	3994	4523	4288
12	2739	3497	3713	3270	3710	3742	3194

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())
```



Let's practice!

INTERMEDIATE DATA VISUALIZATION WITH SEABORN