Plotting and Visualization

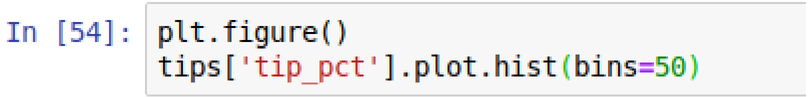
Part 5

Plotting with pandas and seaborn

Part 2

Histograms and Density Plots

- A histogram is a kind of bar plot that gives a discretized display of value frequency.
- The data points are split into discrete, evenly spaced bins, and the number of data points in each bin is plotted.
- Using the tipping data from before, we can make a histogram of tip percentages of the total bill using the `plot.hist` method on the Series:



```
In [54]: plt.figure()
         tips['tip_pct'].plot.hist(bins=50)
```

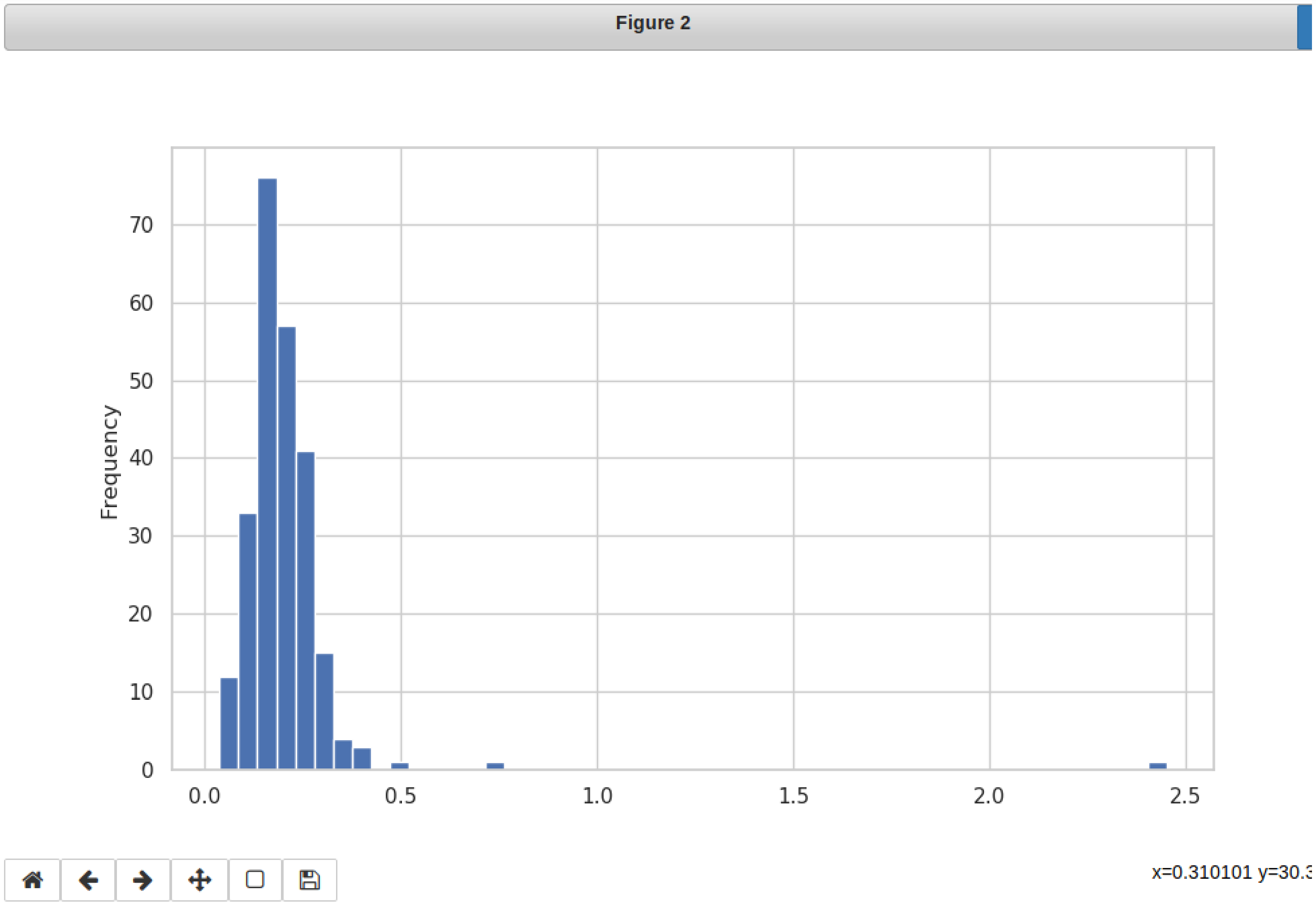
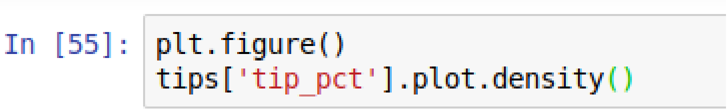


Figure 2
Frequency
70
60
50
40
30
20
10
0
0.0
0.5
1.0
1.5
2.0
2.5
x=0.310101 y=30.3

- A related plot type is a *density plot*, which is formed by computing an estimate of a continuous probability distribution that might have generated the observed data.
- The usual procedure is to approximate this distribution as a mixture of “kernels”—that is, simpler distributions like the normal distribution.
- Thus, density plots are also known as kernel density estimate (KDE) plots.
- Using `plot.kde` makes a density plot using the conventional mixture-of-normals estimate:



```
In [55]: plt.figure()
         tips['tip_pct'].plot.density()
```

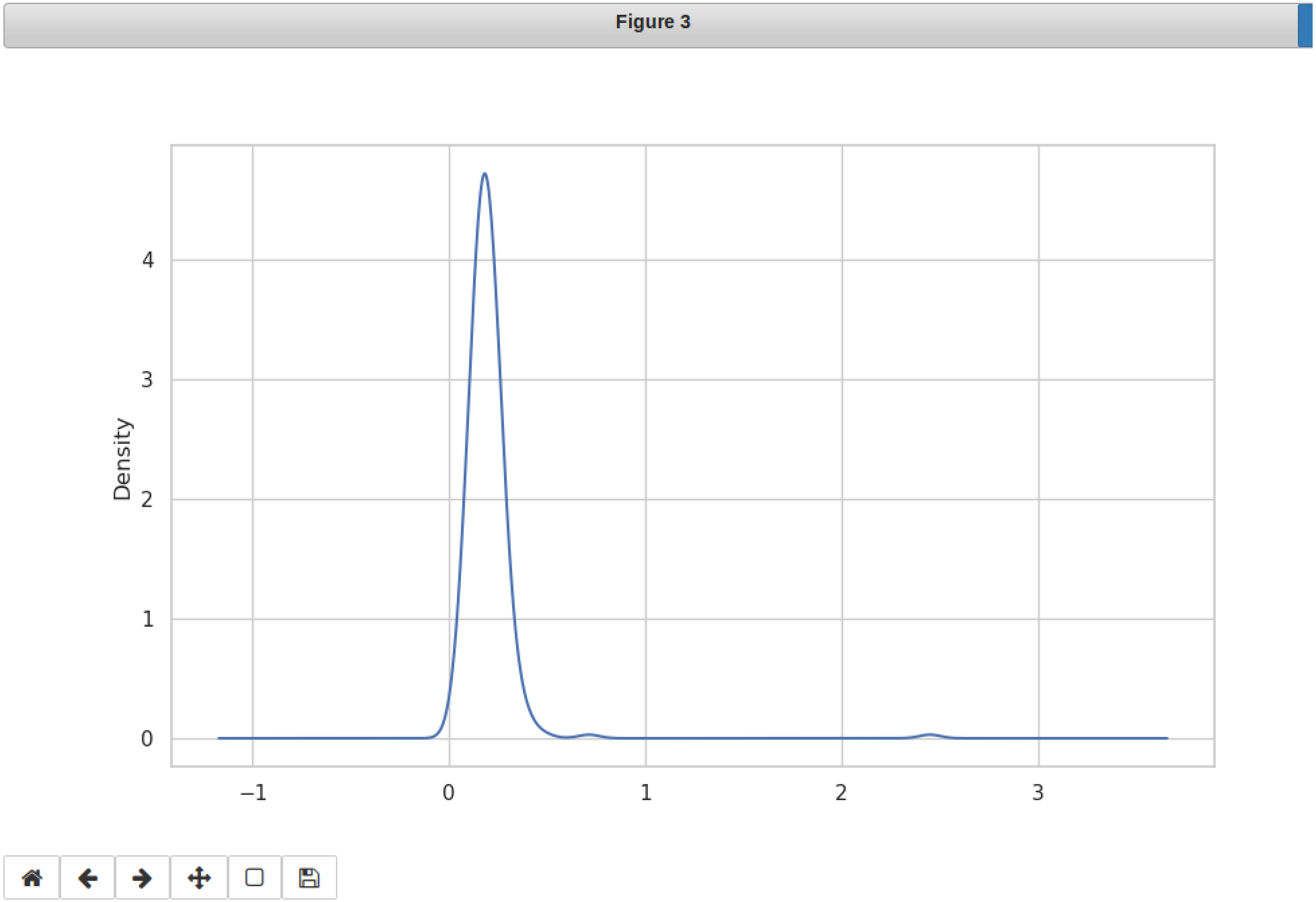
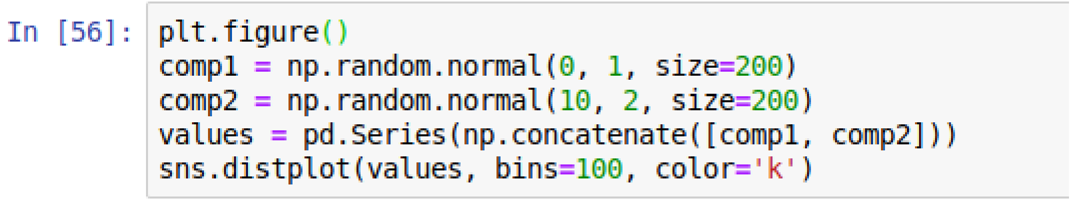


Figure 3
Density
0
1
2
3
4
−1
0
1
2
3

- Seaborn makes histograms and density plots even easier through its `distplot` method, which can plot both a histogram and a continuous density estimate simultaneously.
- As an example, consider a bimodal distribution consisting of draws from two different standard normal distributions:



```
In [56]: plt.figure()
         comp1 = np.random.normal(0, 1, size=200)
         comp2 = np.random.normal(10, 2, size=200)
         values = pd.Series(np.concatenate([comp1, comp2]))
         sns.distplot(values, bins=100, color='k')
```

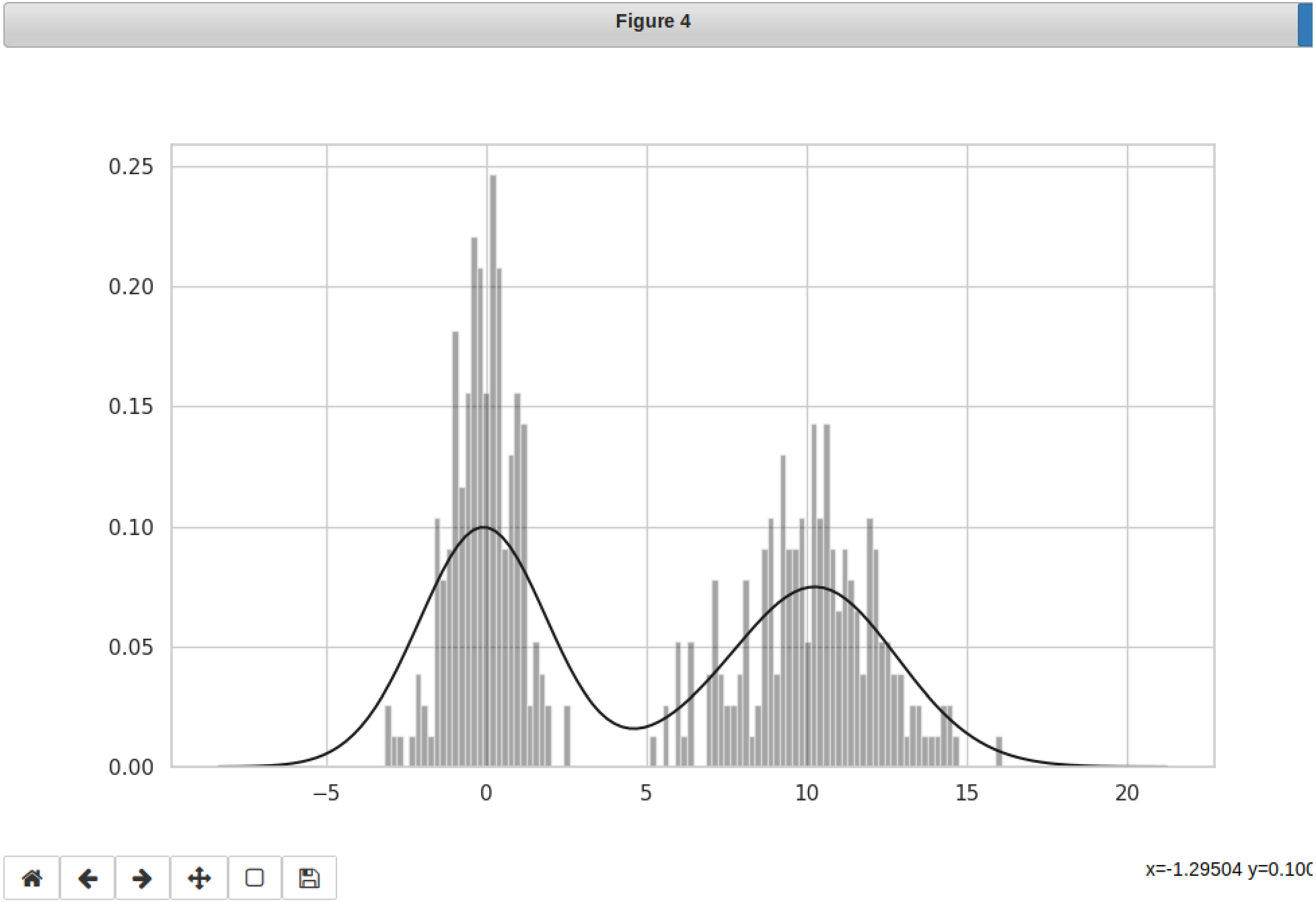
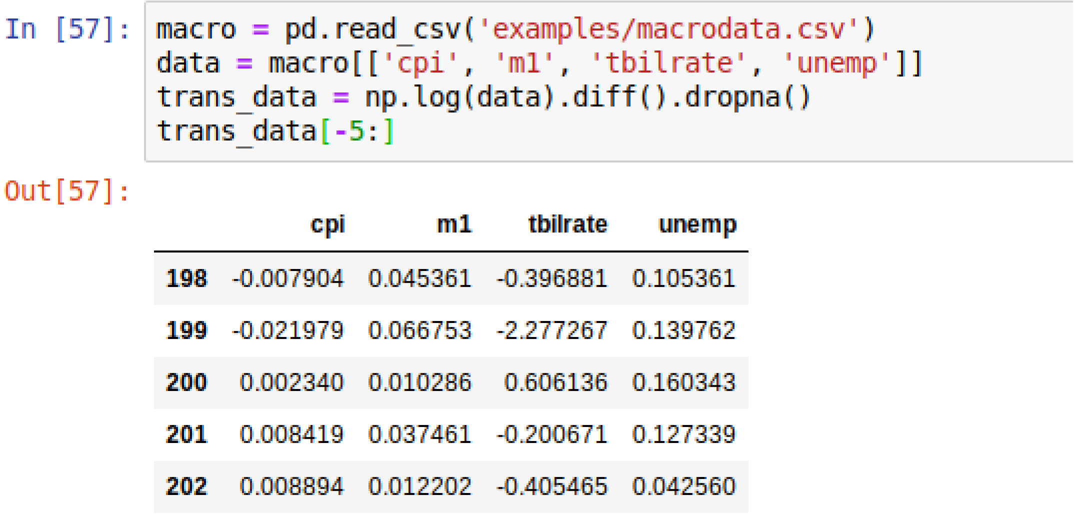


Figure 4
0.25
0.20
0.15
0.10
0.05
0.00
−5
0
5
10
15
20
x=-1.29504 y=0.10(

Scatter or Point Plots

- Point plots or scatter plots can be a useful way of examining the relationship between two one-dimensional data series.
- For example, here we load the `macrodata` dataset from the statsmodels project, select a few variables, then compute log differences:



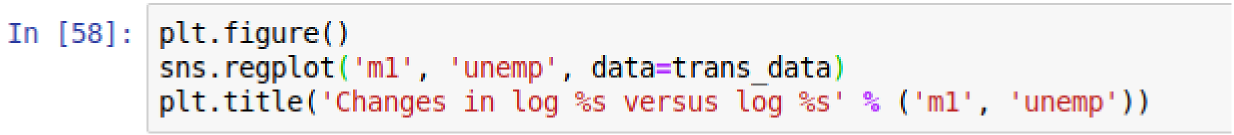
In [57]:

```
macro = pd.read_csv('examples/macrodata.csv')
data = macro[['cpi', 'm1', 'tbilrate', 'unemp']]
trans_data = np.log(data).diff().dropna()
trans_data[-5:]
```

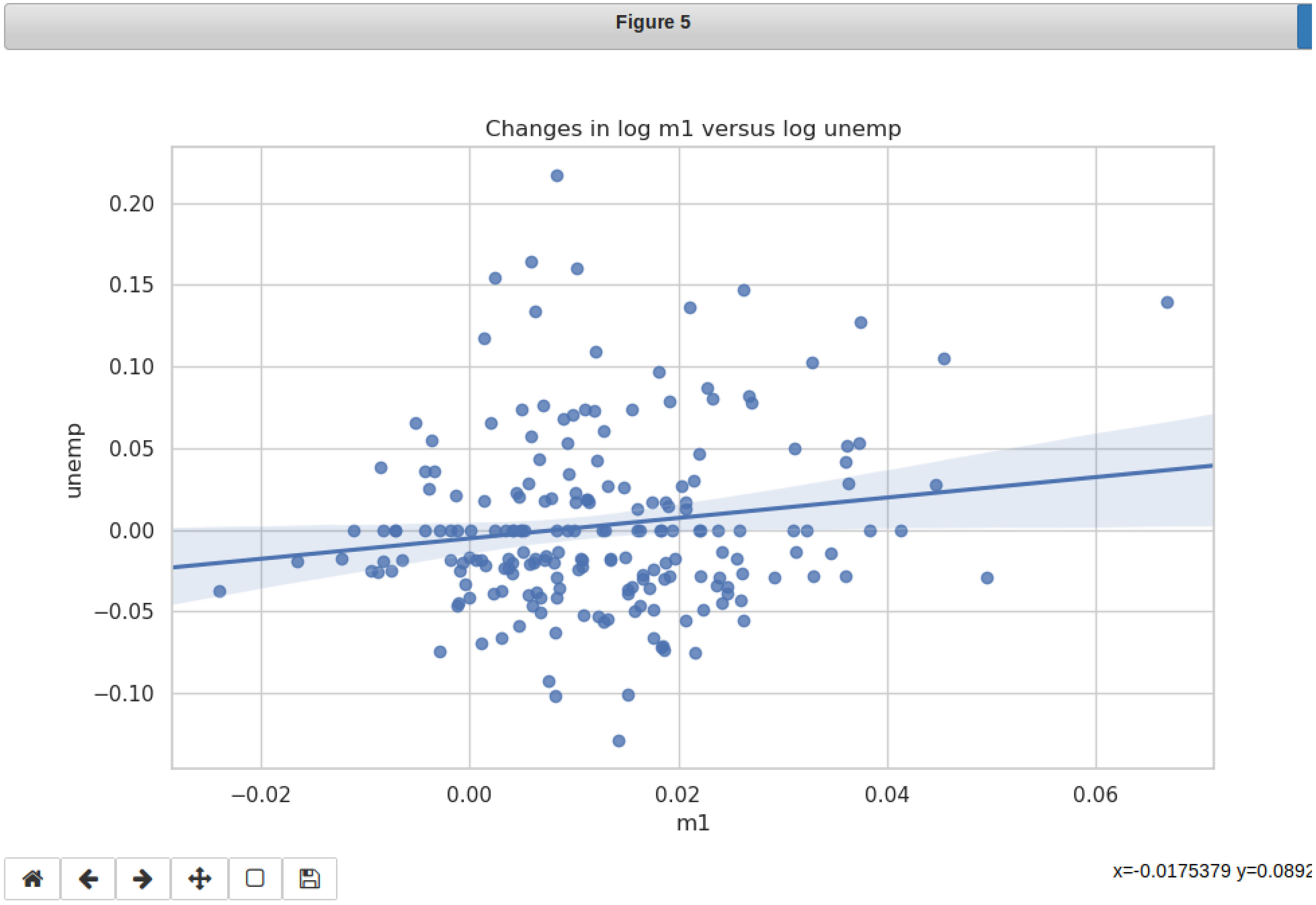
Out[57]:

	cpi	m1	tbilrate	unemp
198	-0.007904	0.045361	-0.396881	0.105361
199	-0.021979	0.066753	-2.277267	0.139762
200	0.002340	0.010286	0.606136	0.160343
201	0.008419	0.037461	-0.200671	0.127339
202	0.008894	0.012202	-0.405465	0.042560

- We can then use seaborn’s `regplot` method, which makes a scatter plot and fits a linear regression line:

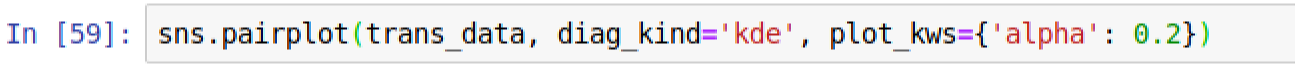


```
In [58]: plt.figure()
         sns.regplot('m1', 'unemp', data=trans_data)
         plt.title('Changes in log %s versus log %s' % ('m1', 'unemp'))
```



x=-0.0175379 y=0.0892

- In exploratory data analysis it’s helpful to be able to look at all the scatter plots among a group of variables; this is known as a *pairs* plot or *scatter plot matrix*.
- Making such a plot from scratch is a bit of work, so seaborn has a convenient `pairplot` function, which supports placing histograms or density estimates of each variable along the diagonal:



```
In [59]: sns.pairplot(trans_data, diag_kind='kde', plot_kws={'alpha': 0.2})
```

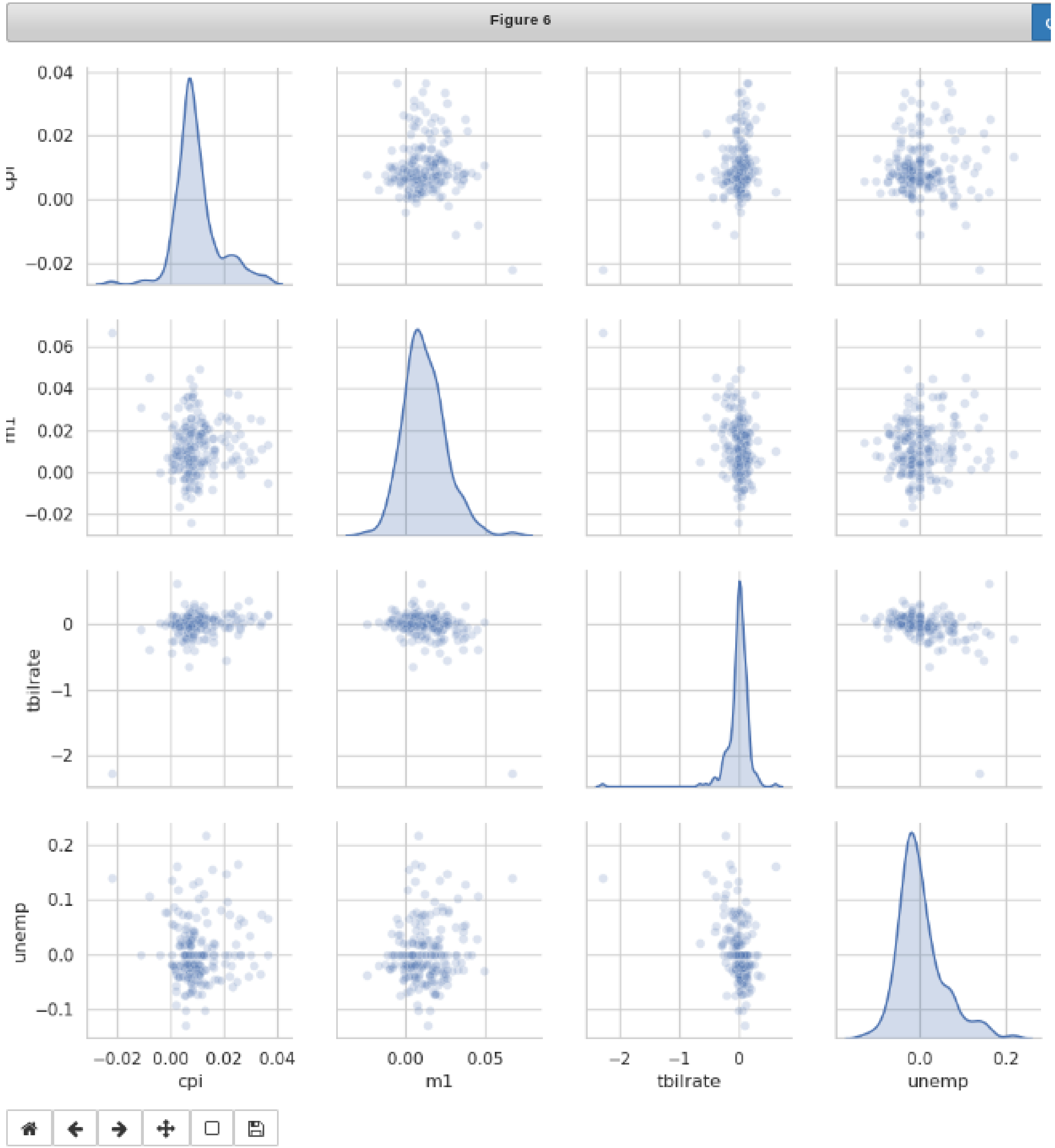
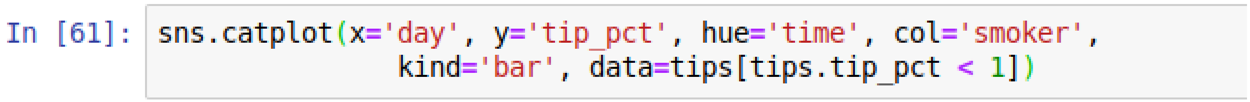


Figure 6
0.04
0.02
0.00
−0.02
cpi
0.06
0.04
0.02
0.00
−0.02
m1
0
−1
−2
tbilrate
0.2
0.1
0.0
−0.1
unemp
−0.02
0.00
0.02
0.04
cpi
0.00
0.05
m1
−2
−1
0
tbilrate
0.0
0.2
unemp

Facet Grids and Categorical Data

- What about datasets where we have additional grouping dimensions?
- One way to visualize data with many categorical variables is to use a `facet grid`.
- Seaborn has a useful built-in function `catplot` that simplifies making many kinds of faceted plots:



```
In [61]: sns.catplot(x='day', y='tip_pct', hue='time', col='smoker',
                        kind='bar', data=tips[tips.tip_pct < 1])
```

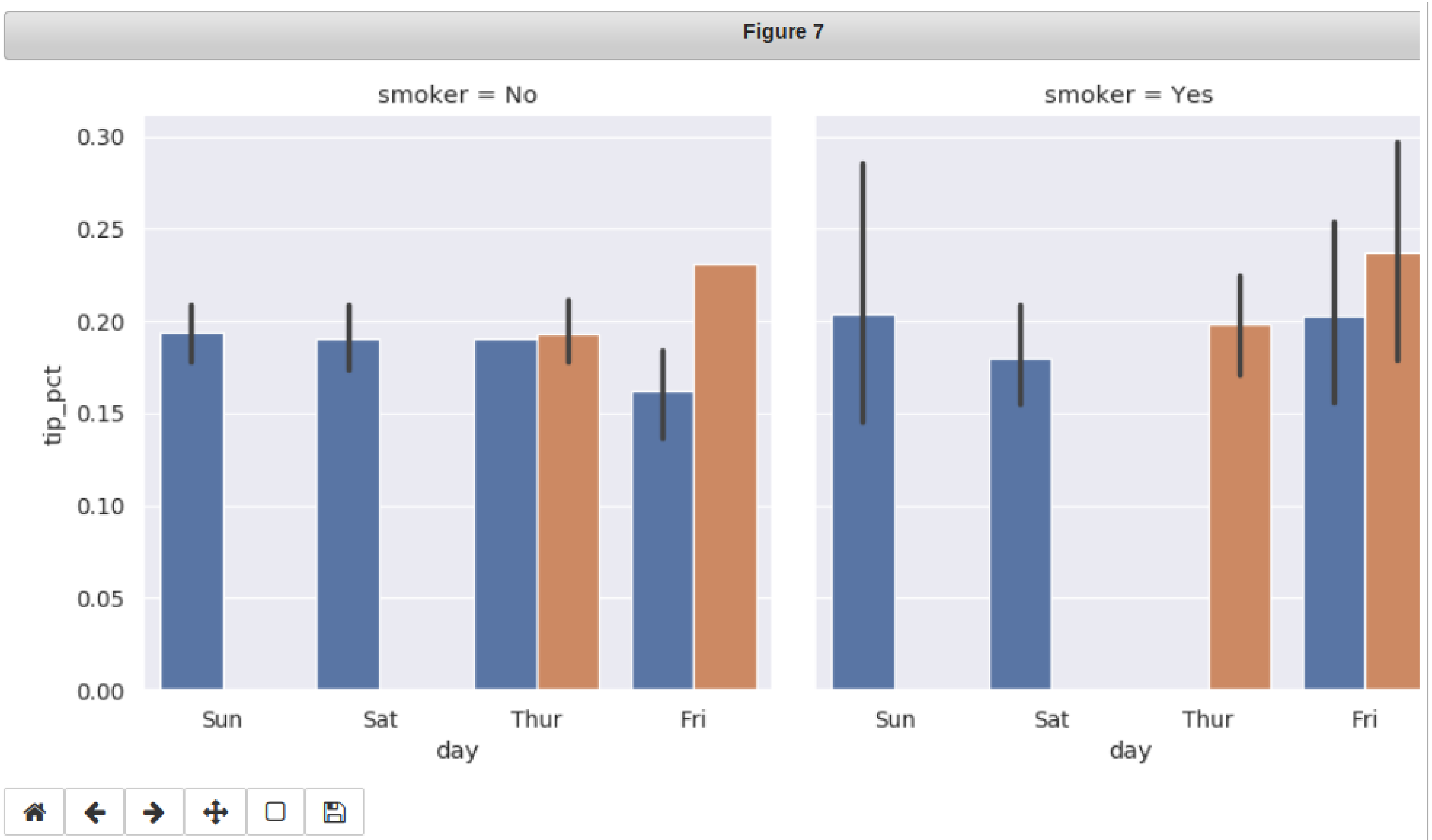


Figure 7
smoker = No
smoker = Yes
tip_pct
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Sun
Sat
Thur
Fri
day
Sun
Sat
Thur
Fri
day

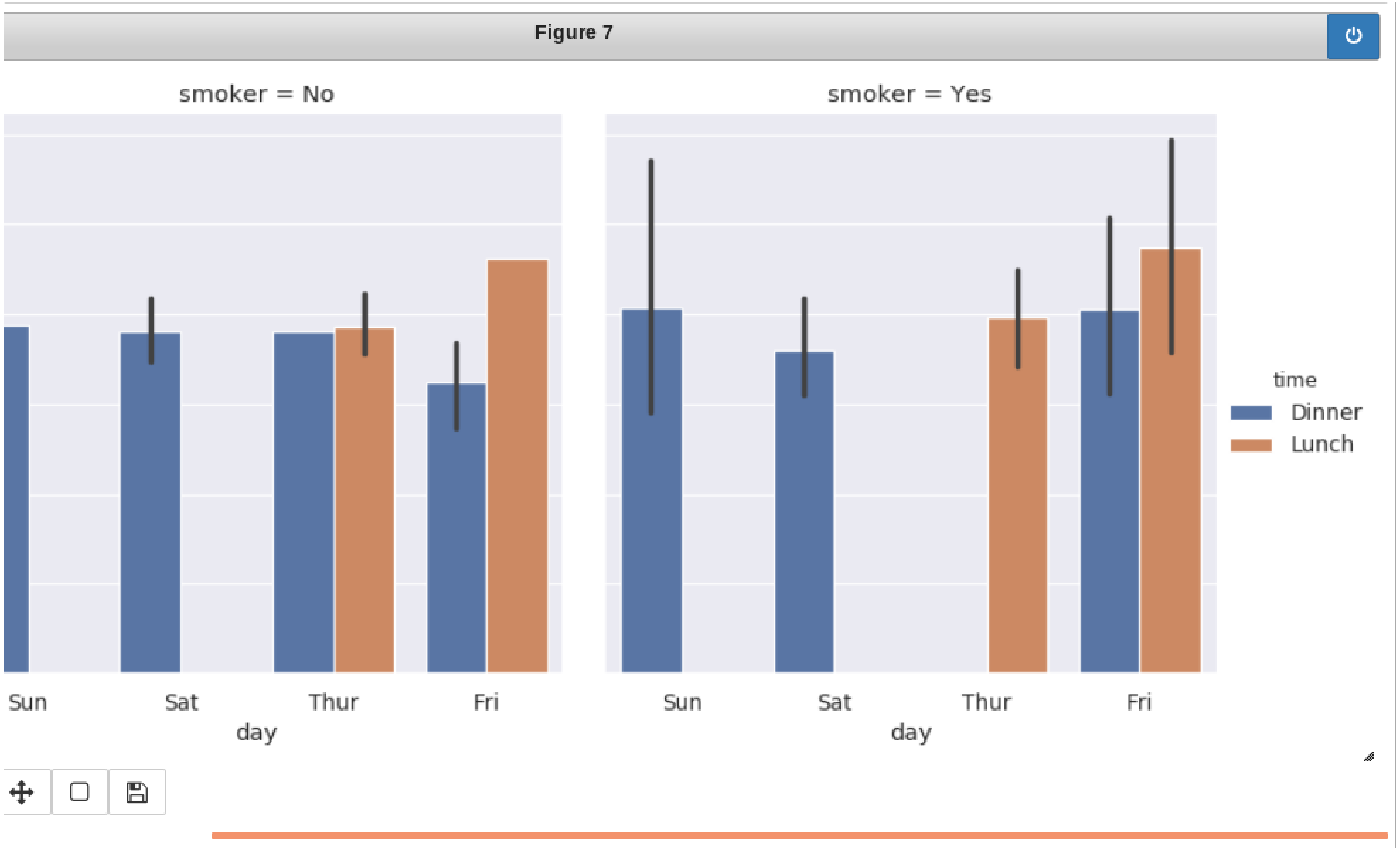
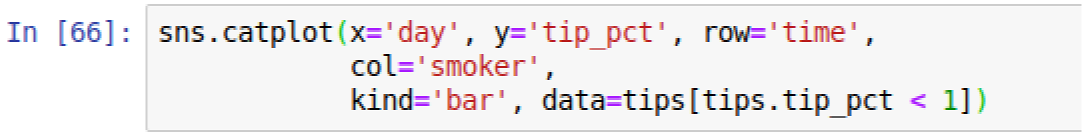


Figure 7
smoker = No
smoker = Yes
Sun
Sat
Thur
Fri
day
Sun
Sat
Thur
Fri
day
time
Dinner
Lunch

- Instead of grouping by `'time'` by different bar colors within a facet, we can also expand the facet grid by adding one row per `time` value:



```
In [66]: sns.catplot(x='day', y='tip_pct', row='time',
                     col='smoker',
                     kind='bar', data=tips[tips.tip_pct < 1])
```

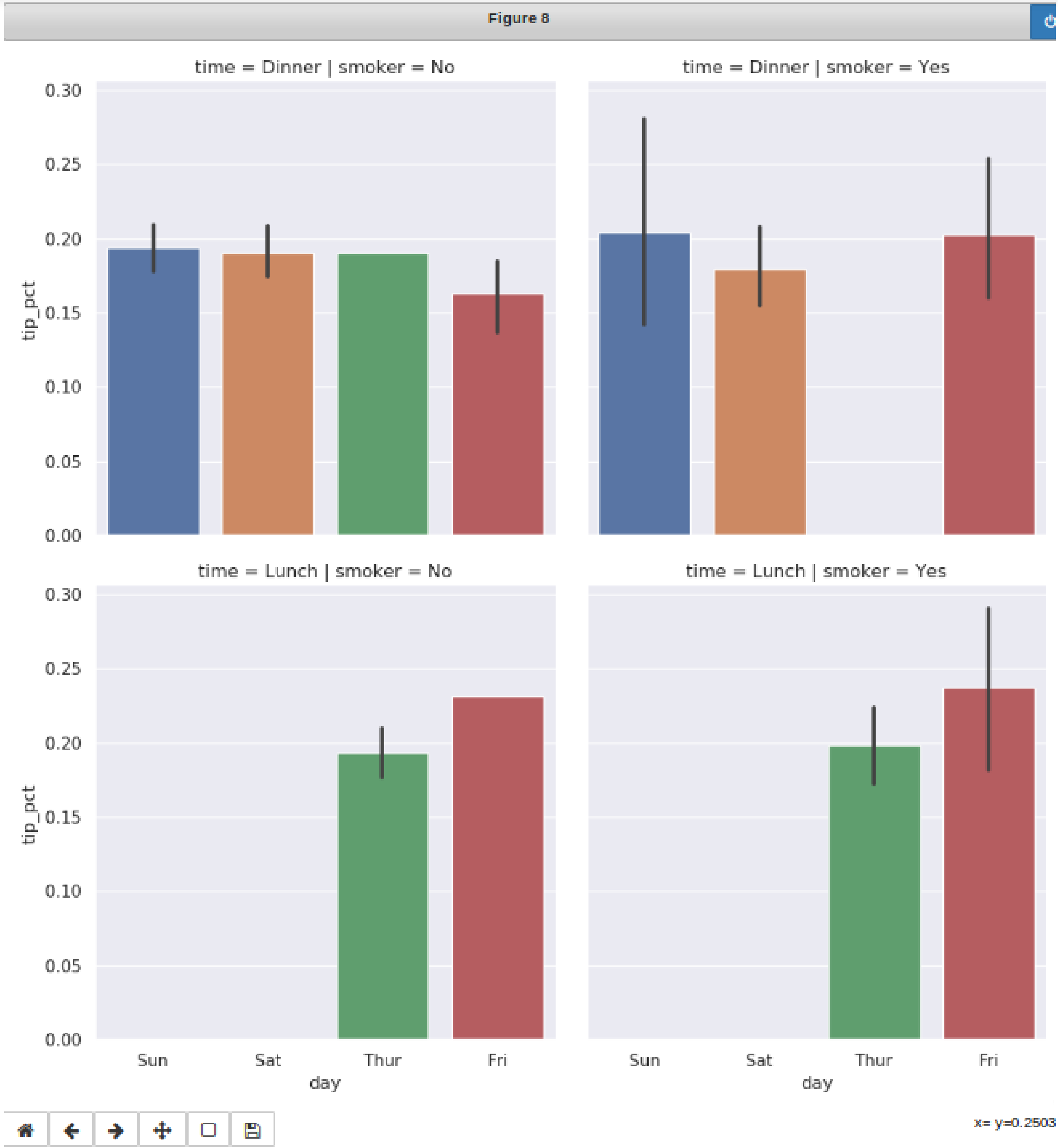
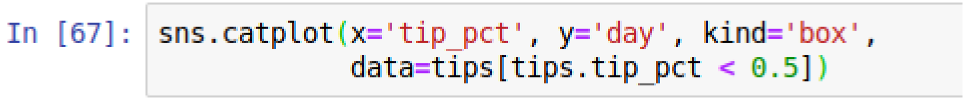


Figure 8
time = Dinner | smoker = No
time = Dinner | smoker = Yes
time = Lunch | smoker = No
time = Lunch | smoker = Yes
tip_pct
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Sun
Sat
Thur
Fri
day
x= y=0.2503

- `catplot` supports other plot types that may be useful depending on what you are trying to display.
- For example, box plots (which show the median, quartiles, and outliers) can be an effective visualization type:



In [67]:

```
sns.catplot(x='tip_pct', y='day', kind='box',
            data=tips[tips.tip_pct < 0.5])
```

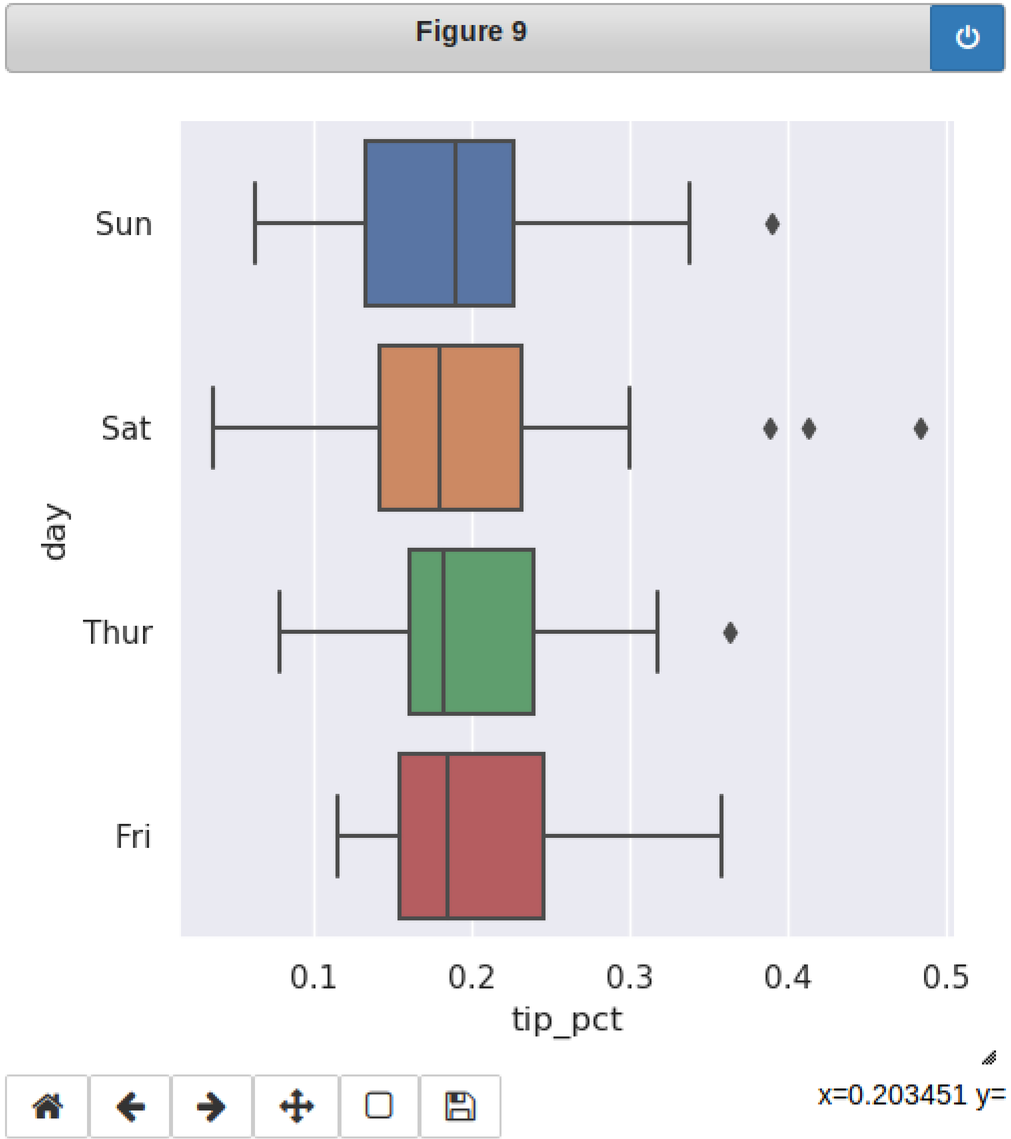


Figure 9
Sun
Sat
Thur
Fri
day
0.1
0.2
0.3
0.4
0.5
tip_pct
x=0.203451 y=