Pandas plotting capabilities

Pandas built-in capabilities for data visualization it's built-off of matplotlib, but it's baked into pandas for easier usage. It provides the basic statistic plot types.
Let's take again the mortgage example in the file train.csv (http://bit.do/train-csv)

```
import pandas as pd
import numpy as np
loan_data = pd.read_csv("train.csv", index_col="Loan_ID")
loan_data.head()
```

Now say that you want to make a histogram of the column 'LoanAmount'. You can find all the possible matplotlib output on the gallery page
http://matplotlib.org/gallery.html
Call the appropriate method with:

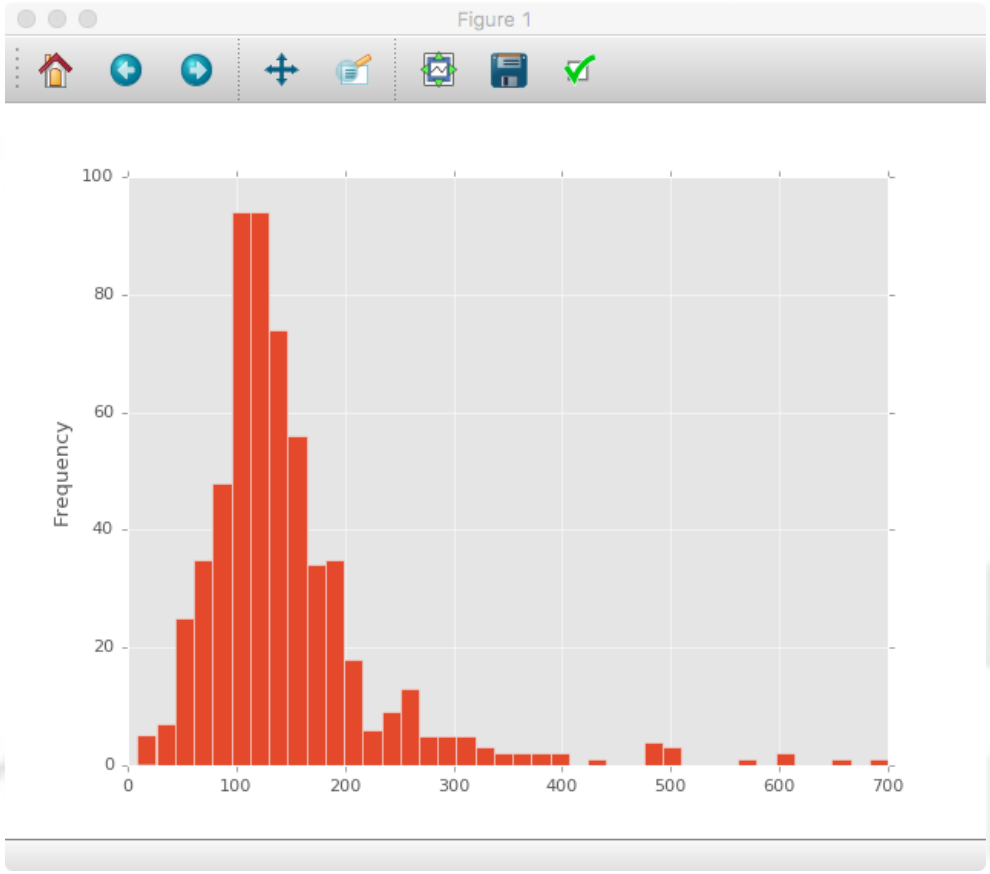
```
loan_data['LoanAmount'].plot.hist()
```

As you can see the default binning it's not very descriptive in our case, so let's change some parameter

```
loan_data['LoanAmount'].plot.hist(bins=40)
```

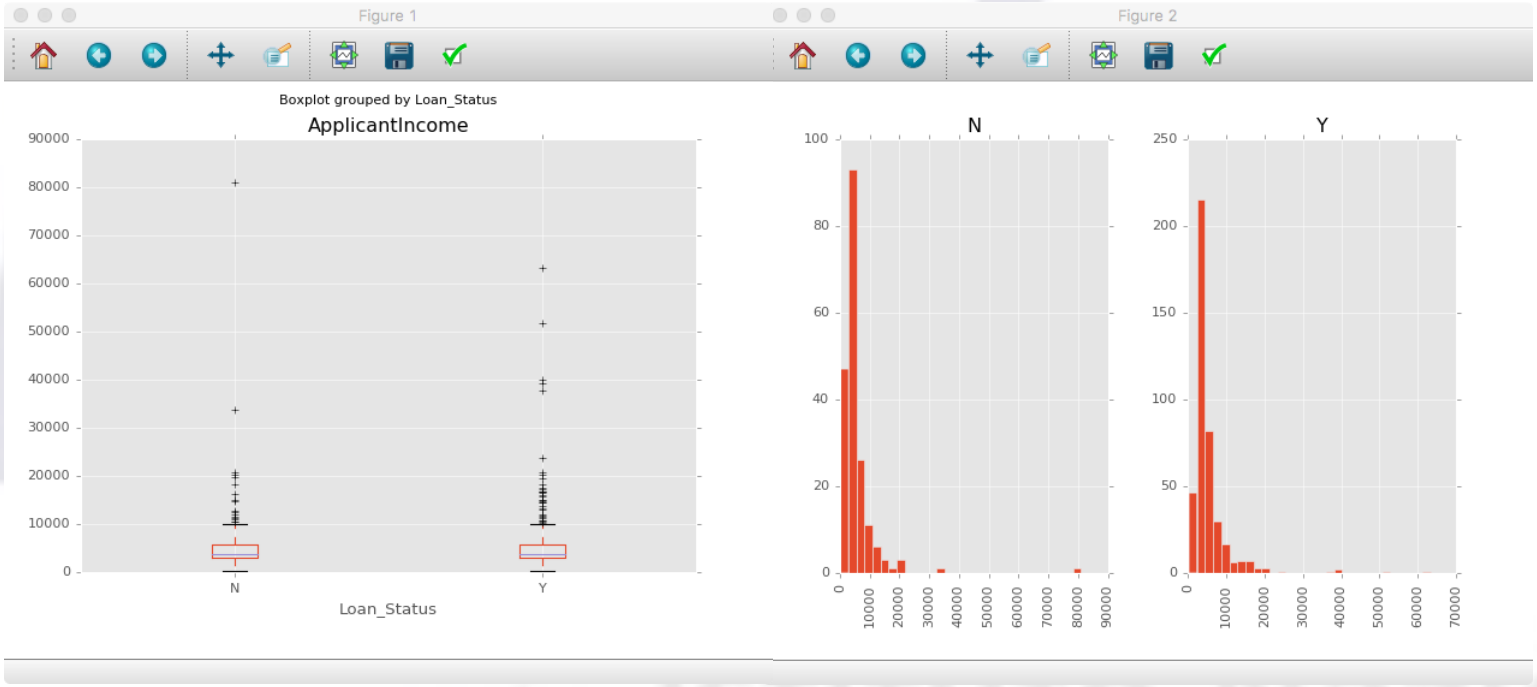
If the default appearance seems to you quite dull, you can change it using a stylesheet

```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
loan_data['LoanAmount'].plot.hist(bins=40)
```



Now we want to compare the distribution of ApplicantIncome by Loan_Status:

```
loan_data.boxplot(column="ApplicantIncome",by="Loan_Status")
loan_data.hist(column="ApplicantIncome",by="Loan_Status",bins=30)
```



It seems that the Income is not a big deciding factor on its own, as there is no appreciable difference between the people who received and were denied the loan

Seaborn

- If you want to use advanced plotting features you can import seaborn in your code. Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics.

- Here:

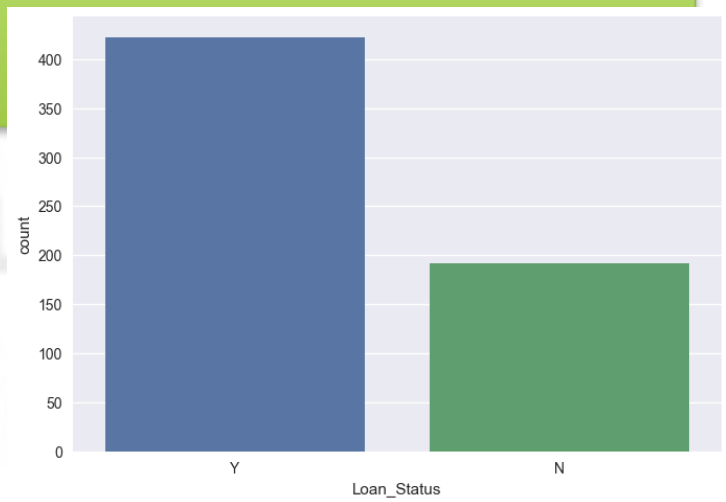
  http://seaborn.pydata.org/examples/index.html

  you can find an extensive examples gallery about what you can obtain from it.

- Generally speaking:

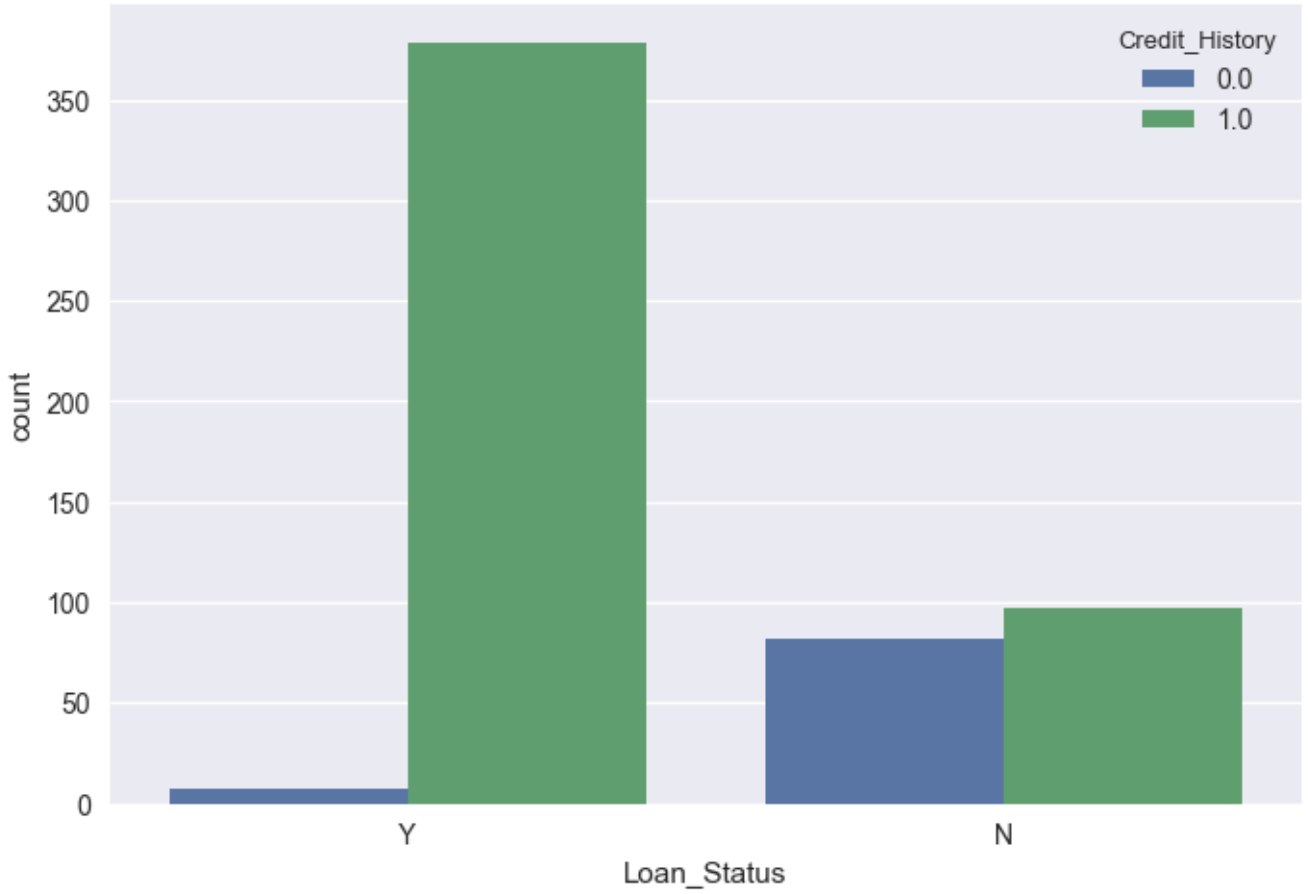
1. scatter plots help you to understand relation between two continuous variables.
2. box-plot helps you to find outliers in your data.
3. Histogram helps you find spread of the data when continuous.
4. Correlation plot helps you learn correlation with variables.
5. Barplot helps you to understand relation between your continous and categorical variable.

```
import seaborn as sns
import pandas as pd
import numpy as np
loan_data = pd.read_csv("train.csv", index_col="Loan_ID")

impute_grps = loan_data.pivot_table(values=["LoanAmount"],
index=["Gender","Married","Self_Employed"], aggfunc=np.mean)

loan_data['LoanAmount'].fillna(value=(impute_grps.loc[tuple([loan_data['Gende
r'][0], loan_data['Married'][0], loan_data['Self_Employed'][0]])][0]),
inplace=True)

loan_data['Gender'].fillna(value=(loan_data['Gender'].mode())[0],
inplace=True)
loan_data['Married'].fillna(value=(loan_data['Married'].mode())[0],
inplace=True)
loan_data['Self_Employed'].fillna(value=(loan_data['Self_Employed'].mode())[0
], inplace=True)

sns.countplot(x='Loan_Status',data=loan_data)
```



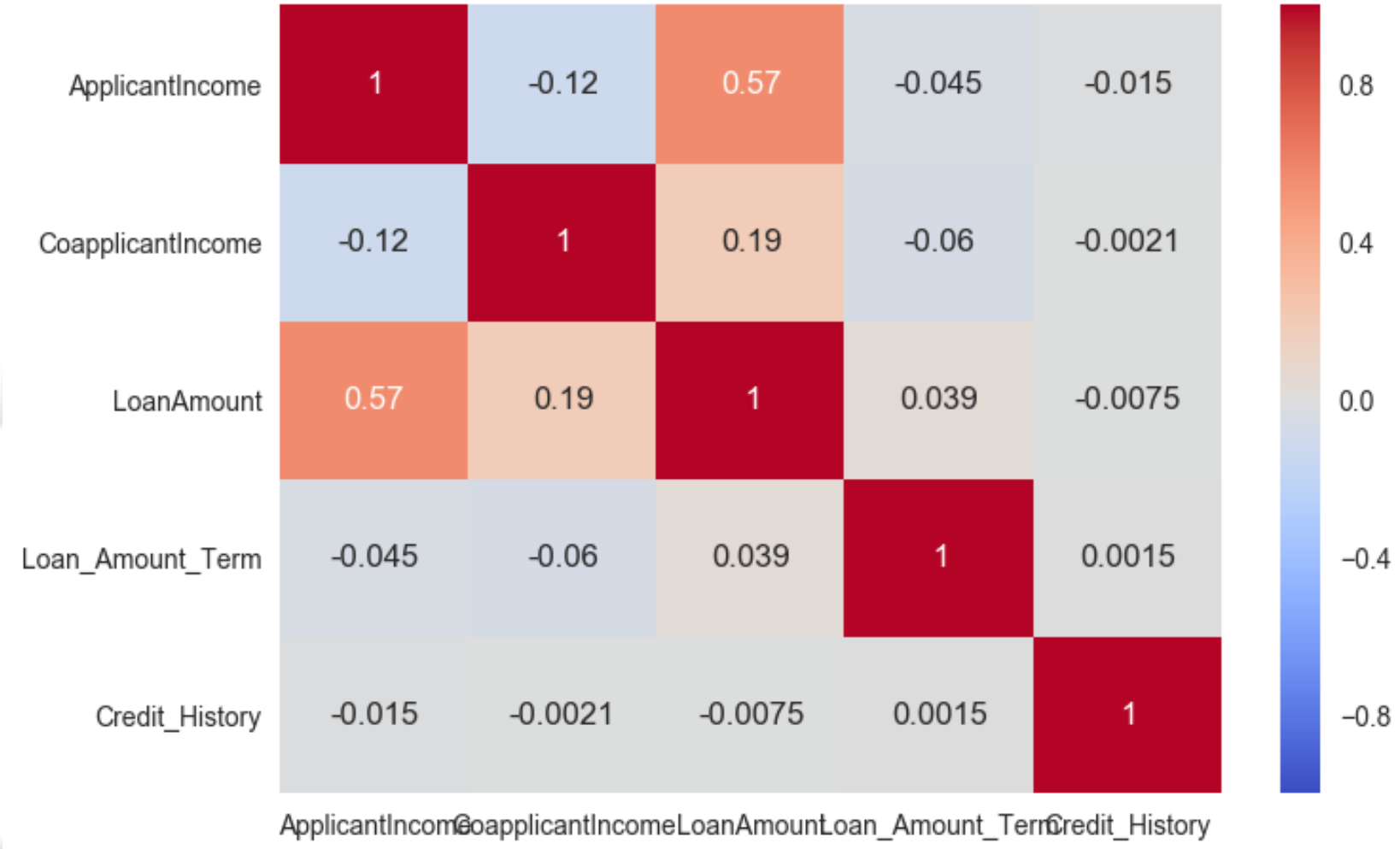
This is a simple count of the categorical column 'Loan_Status'. But let's insert a distinction between people with or without a credit history: the plot will tell us that it's high advisable to have a credit history if you want a loan…

```
sns.countplot(x='Loan_Status',hue='Credit_History',data=loan_data)
```



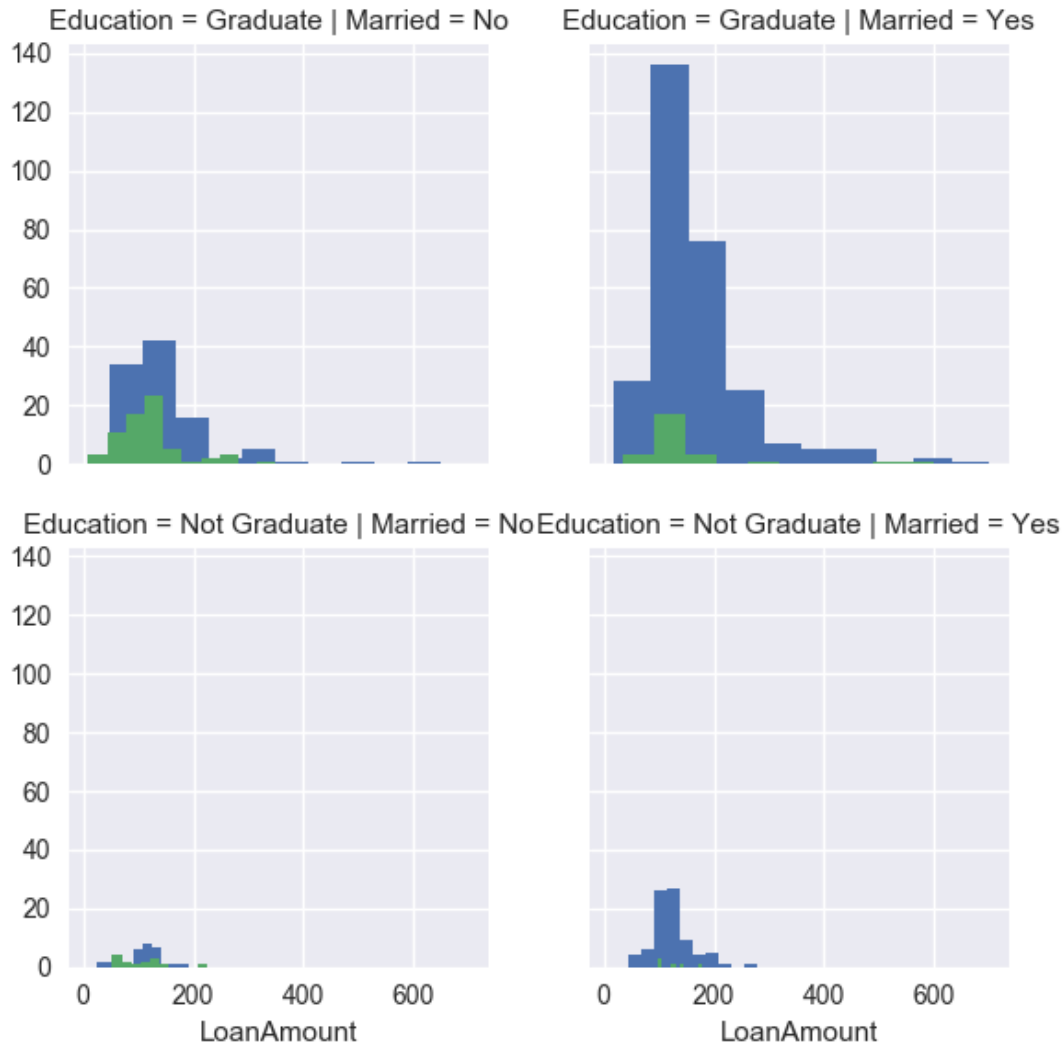
With the `corr()` function you can obtain a matrix formed by a pairwise correlation of numerical columns, excluding nan values, that can give a useful "first look" about the data:

```
sns.heatmap(loan_data.corr(),cmap='coolwarm',annot=True)
```



FacetGrid is the general way to create grids of plots based off of a feature:

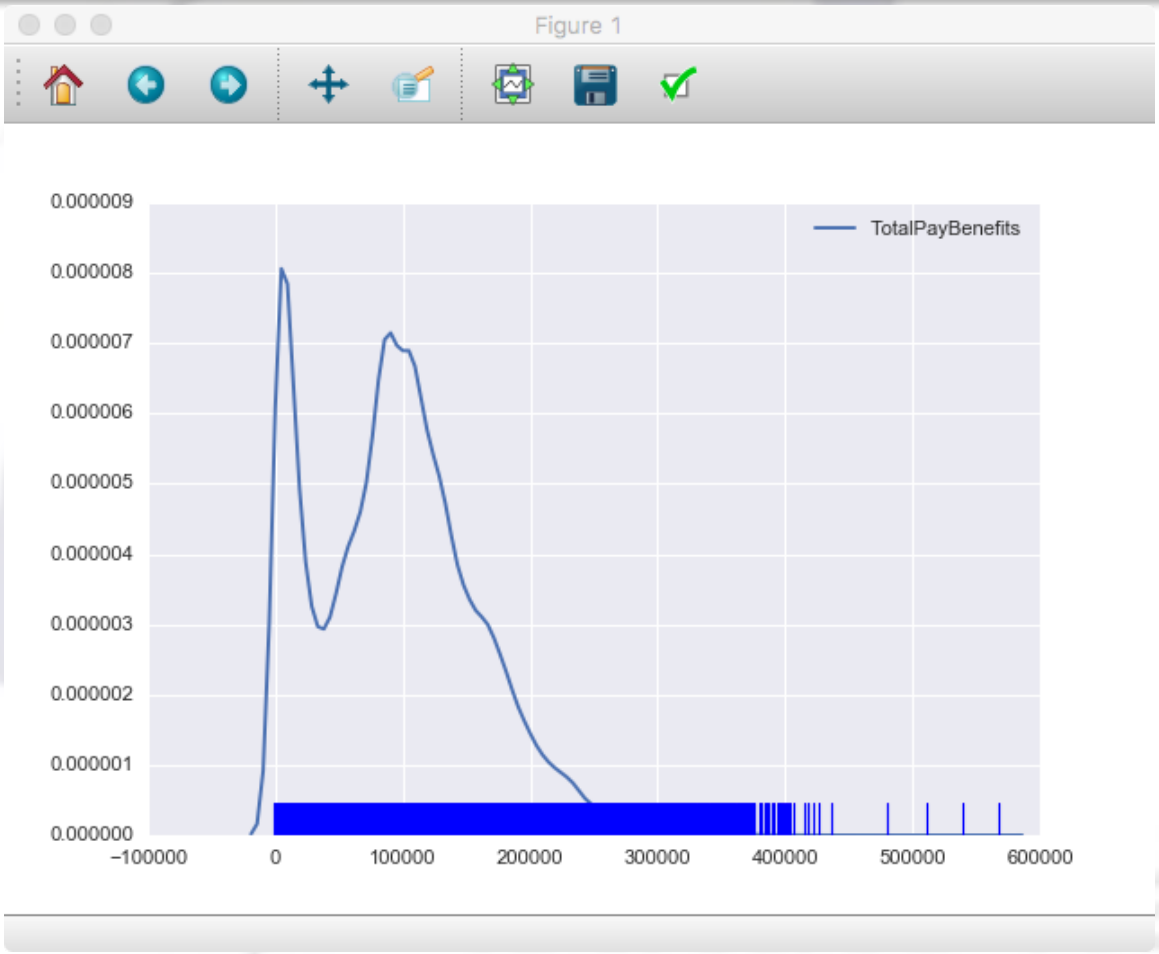
```
import matplotlib.pyplot as plt
g = sns.FacetGrid(loan_data, col="Married", row="Education",hue="Gender")
g = g.map(plt.hist, "LoanAmount")
```



Now let's reload the kaggle dataset salaries.csv: we will see other types of seaborn data visualization (http://bit.do/salaries-csv)

```
salaries=pd.read_csv("Salaries.csv", index_col='Id')
salaries.head()
sns.jointplot(x='BasePay',y='Benefits',data=salaries,kind='scatter')
#try reg,hex, kde
```

```
sns.rugplot(salaries["TotalPayBenefits"])
sns.kdeplot(salaries["TotalPayBenefits"])
```

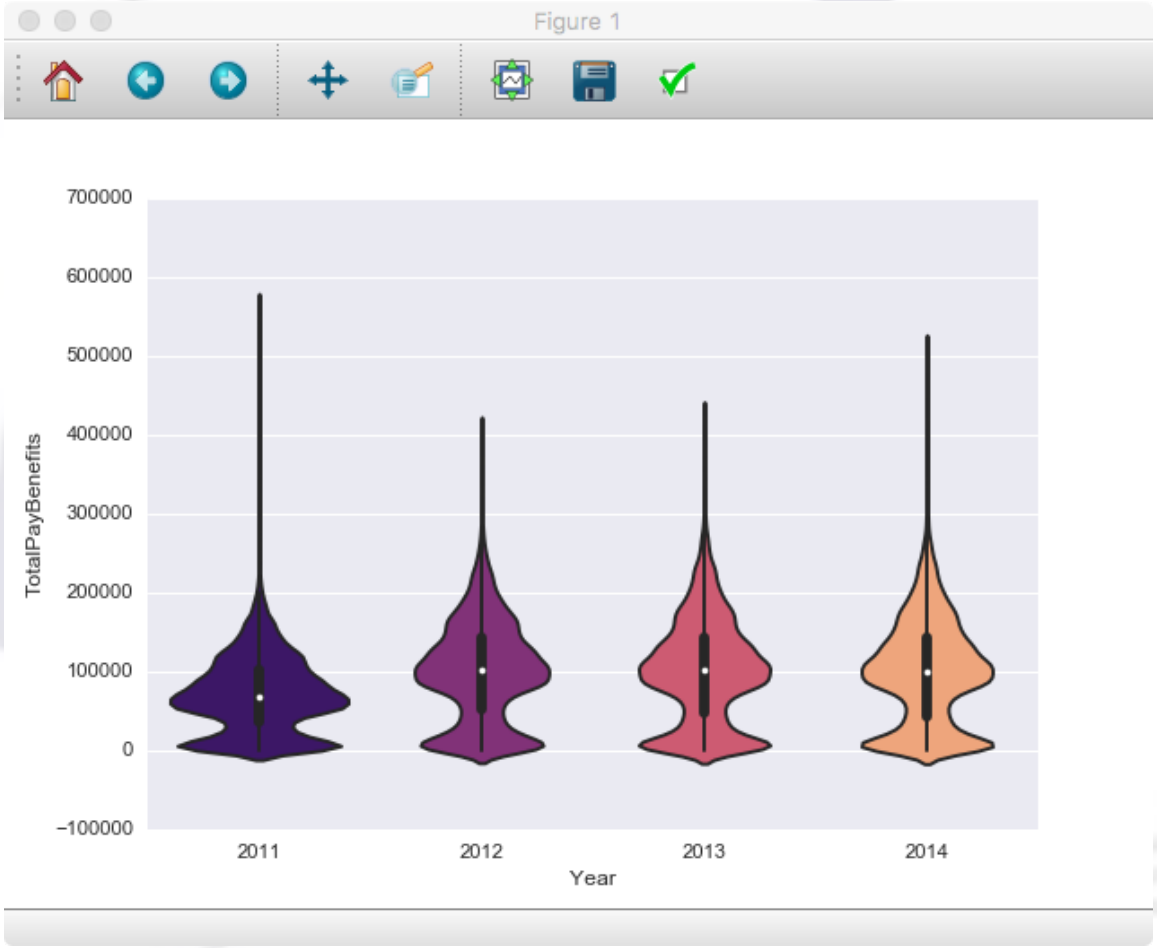


rugplots just draw a dash mark for every point on a univariate distribution. They are the building block of a KDE (Kernel Density Estimator) plot. These KDE plots replace every single observation with a Gaussian (Normal) distribution centered around that value.

```
sns.boxplot(x="Year", y="TotalPayBenefits",data=salaries,palette='rainbow')
```

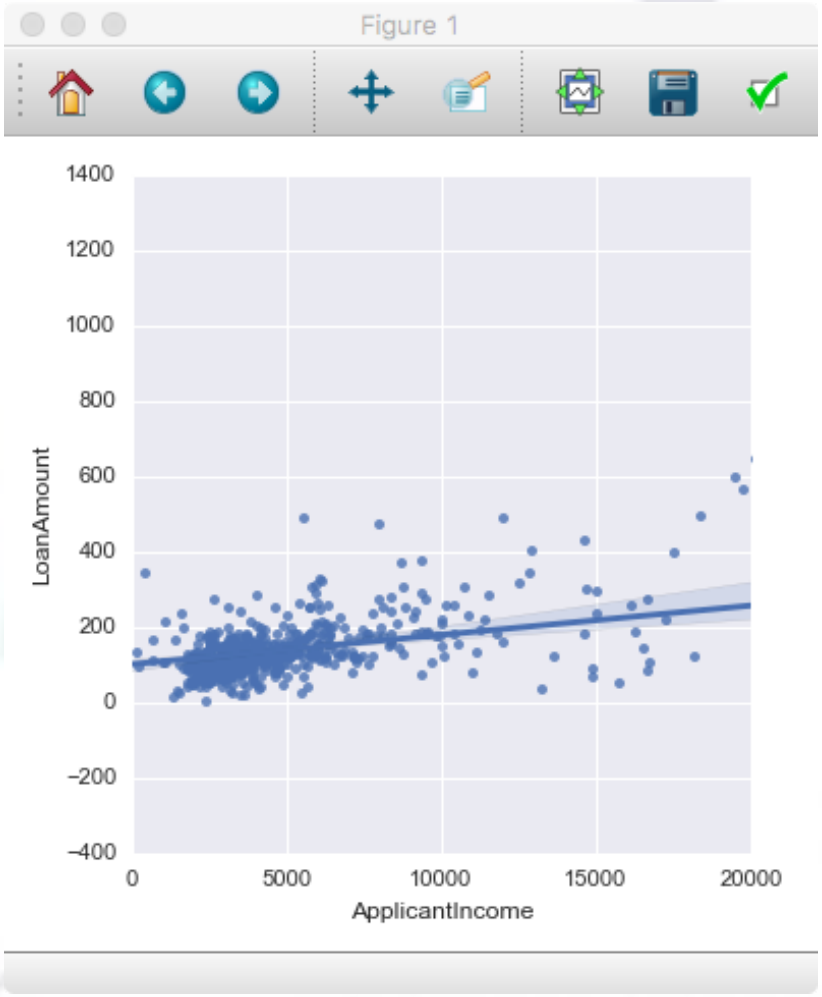
The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be “outliers” using a method that is a function of the inter-quartile range.

```
sns.violinplot(x="Year", y="TotalPayBenefits",data=salaries,palette='magma')
```



A violin plot shows the distribution of quantitative data across several levels of one (or more) categorical variables such that those distributions can be compared. Unlike a box plot, in which all of the plot components correspond to actual datapoints, the violin plot features a kernel density estimation of the underlying distribution.

```
lm =
sns.lmplot(x='ApplicantIncome',y='LoanAmount',data=loan_data,palette='coolwarm')
axes=lm.axes
axes[0,0].set_xlim(0,20000)
```



lmplot allows you to display linear models, but it also conveniently allows you to split up those plots based off of features, as well as coloring the hue based off of features.

Plotly and Cufflinks

- Plotly is a library that allows you to create interactive plots that you can use in dashboards or websites (you can save them as html files or static images).
- Cufflinks is a library that allows you to call plots directly off of a pandas dataframe.
- To see directly the generated plots we will use *Jupyter notebook,* that Anaconda has installed for you

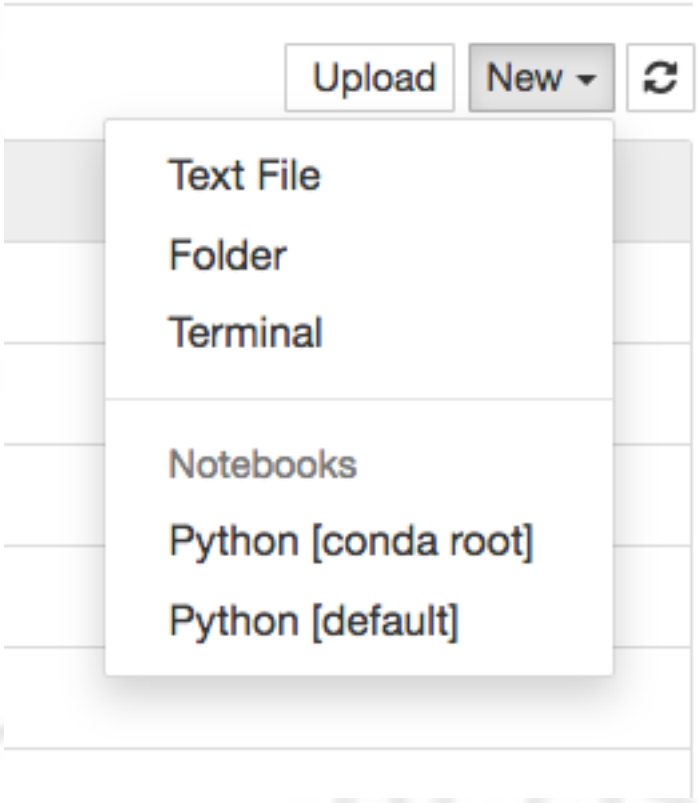
- These libraries are not currently available through conda but are available through pip. So you have to install pip in Anaconda, and use it to install them:

```
conda install pip
pip install plotly
pip install cufflinks
```

Use a terminal CLI to write the following command:

```
jupyter notebook
```

The default browser will be opened with a file browser of the directory from which you have issued the command.
With this browser you can open a previously saved Jupiter notebook file, or you can create a new notebook with the "New" button on the right:



Select the Python [conda root] and a new tab will be opened with the familiar IPython prompt

In [1]:
```
import pandas as pd
import numpy as np
%matplotlib inline
```

In [2]:
```
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
```

In [3]:
```
import cufflinks as cf
```

In [4]:
```
# For Notebooks
init_notebook_mode(connected=True)
```

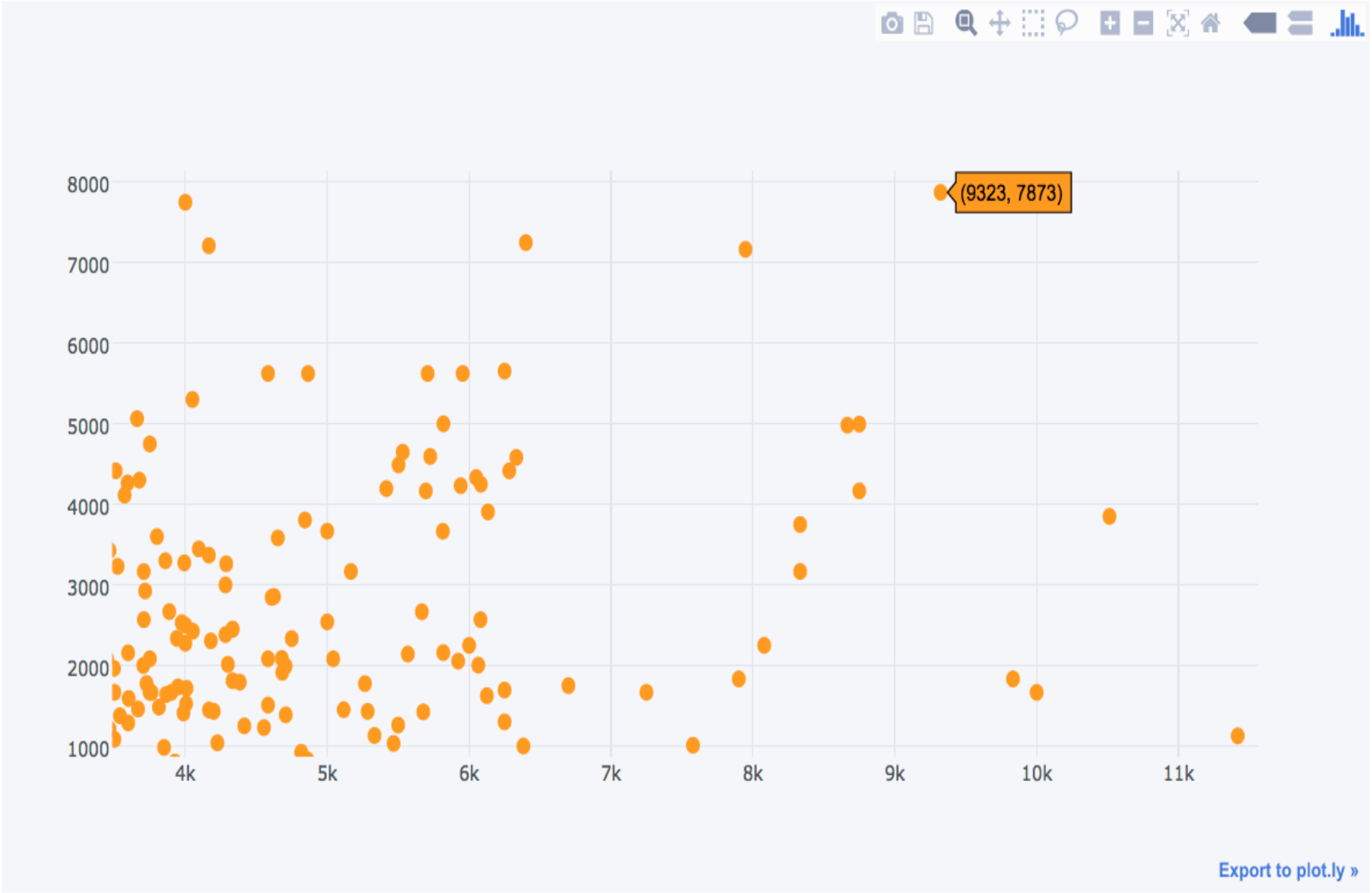
In [5]:
```
# For offline use
cf.go_offline()
```

If you type a command and press Enter the code cell will enter a new line: to execute the code you must press Shift+Enter

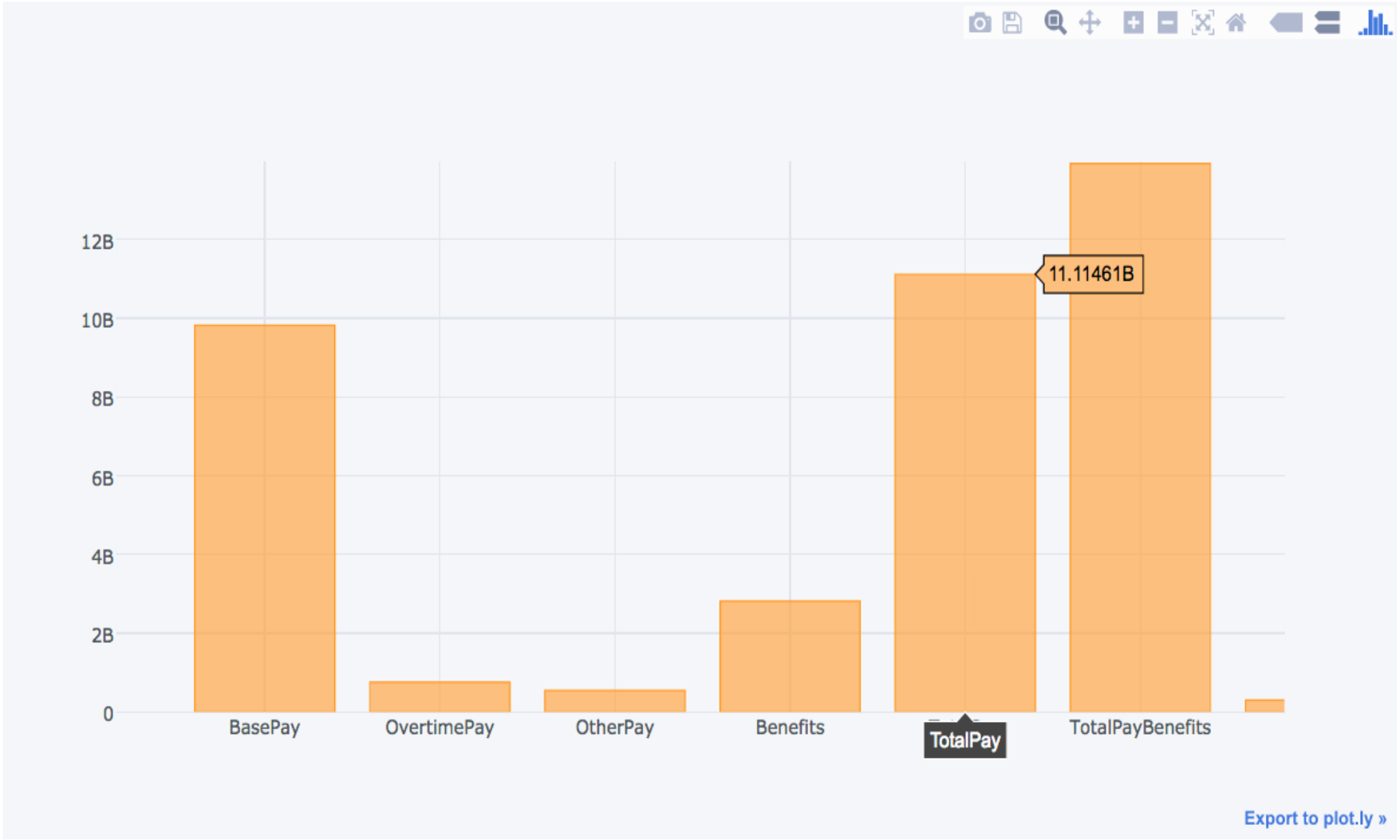
Plotly is open-source, but you need to tell it that you're working offline. As you can see on the site http://plot.ly, there is a corresponding company that makes money hosting your visualizations and dashboards. The site hosts a great gallery of examples.

In [8]:
```
loan_data = pd.read_csv("train.csv", index_col="Loan_ID")
salaries=pd.read_csv("Salaries.csv", index_col='Id')
```

In [12]:
```
loan_data.iplot(kind='scatter',x='ApplicantIncome',y='CoapplicantIncome',mode='markers', size=10)
```

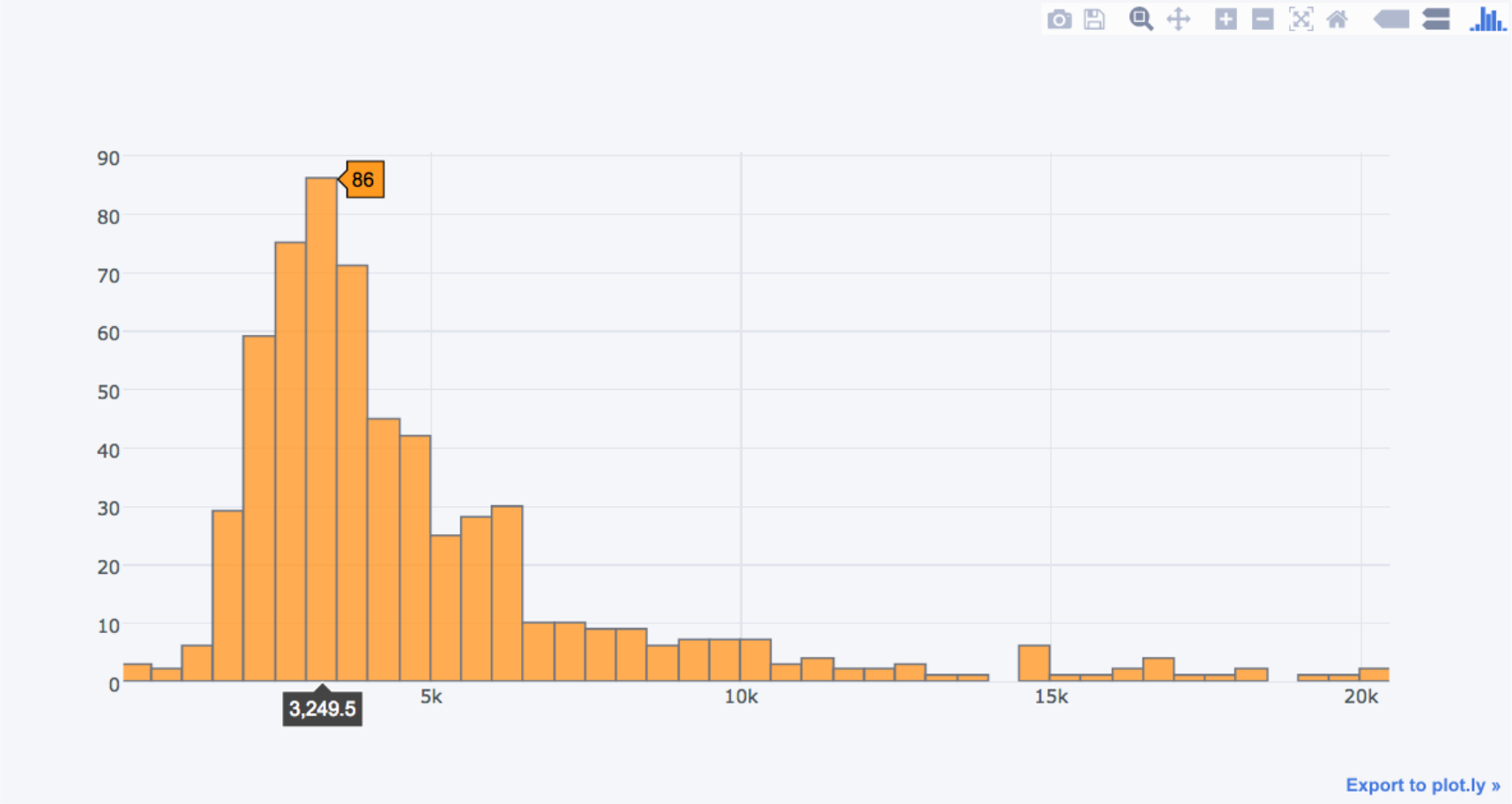


In [6]:
```
salaries.sum().iplot(kind='bar')
```

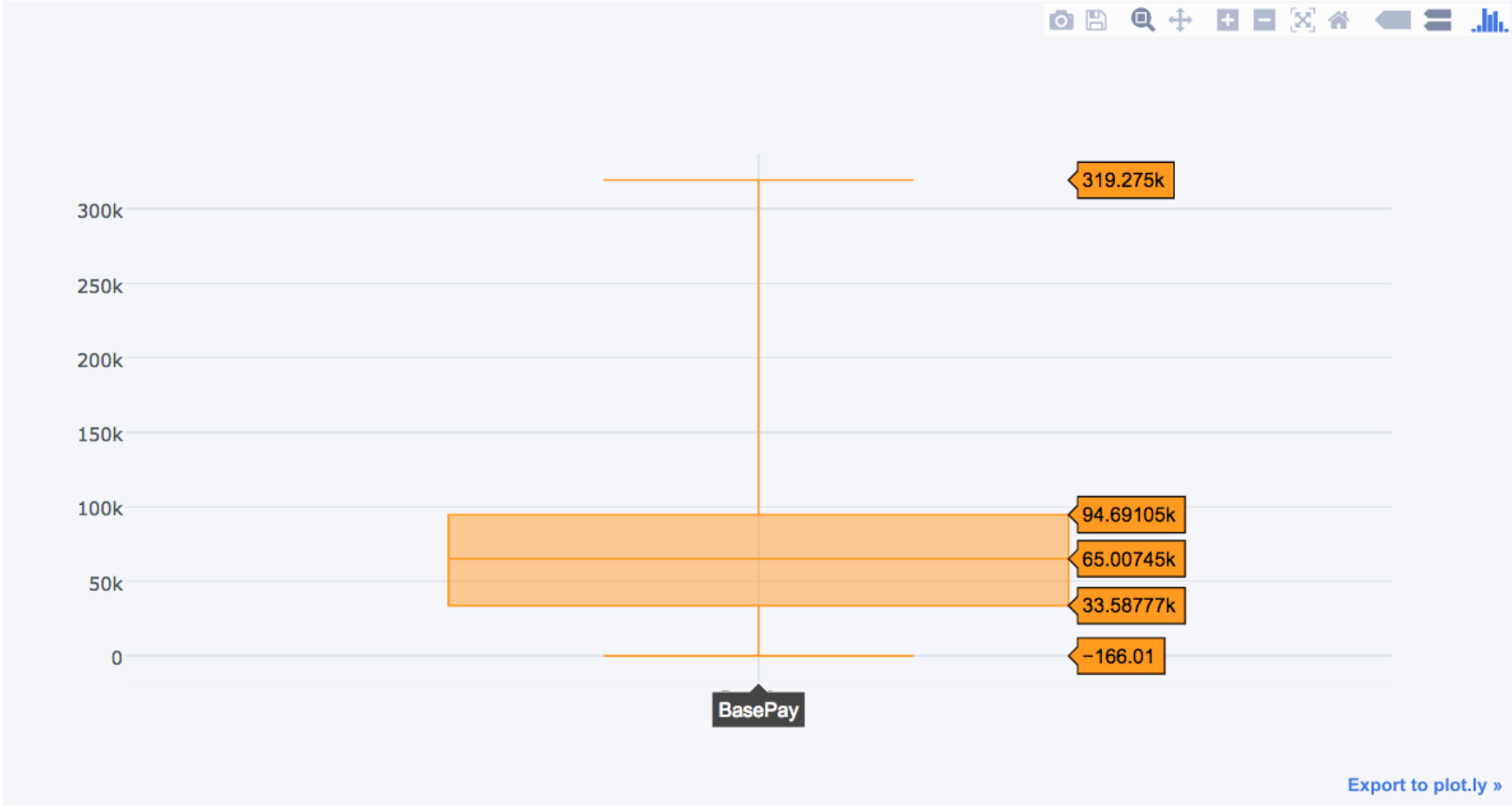


In [15]:
```
loan_data.corr().iplot(kind='heatmap',colorscale='spectral')
```

In [41]:
```
loan_data['ApplicantIncome'].iplot(kind='hist',bins=250)
```



```
In [52]: salaries['BasePay'].iplot(kind='box')
```



If you want to use this library in IPython you can save the interactive plots on a file and then use `plot()`

```
fig = dataframe.iplot(kind='bar',asFigure=True)
plot(fig)
```

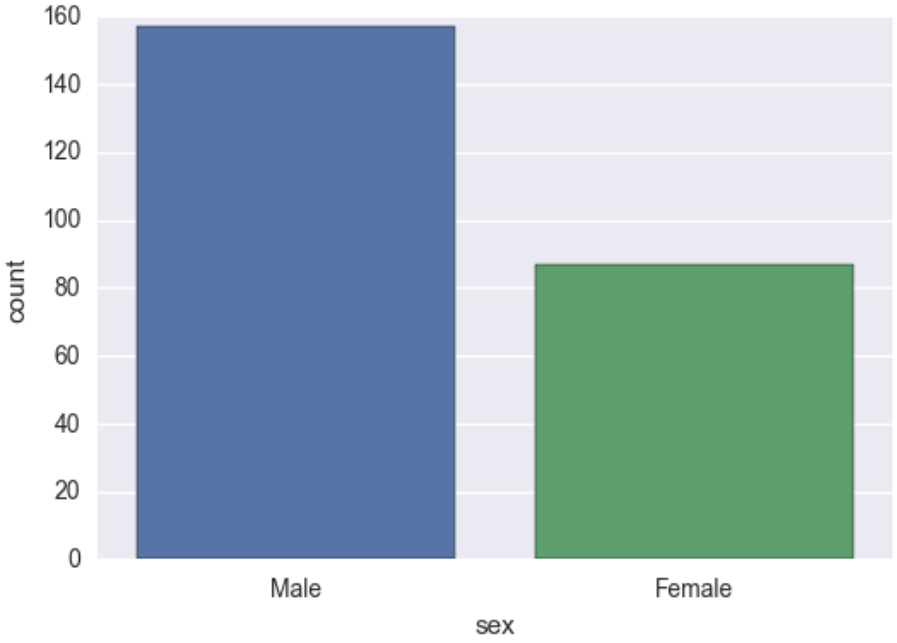
Exercises!

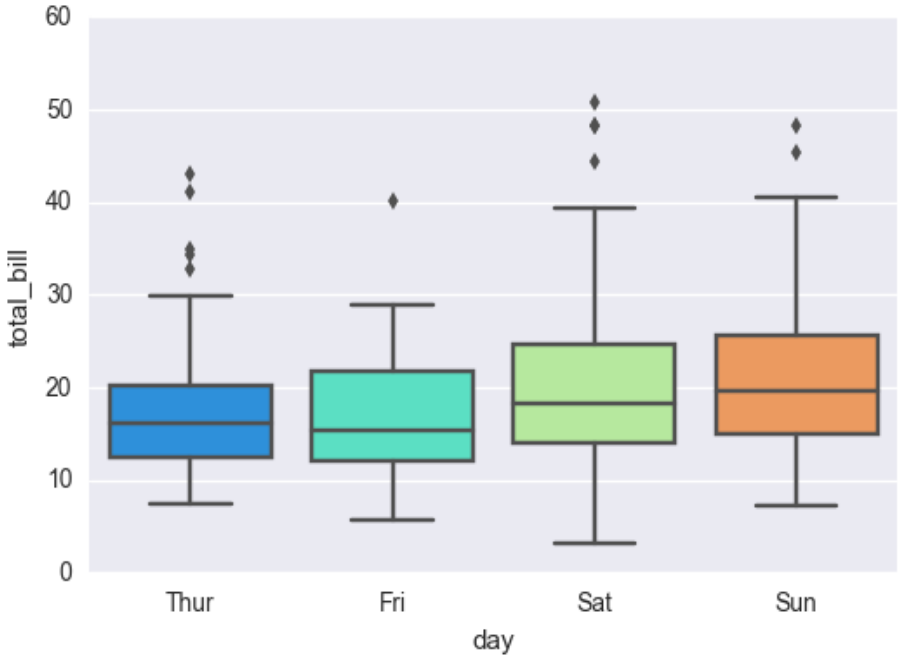
Seaborn comes with the possibility to load a dataset from an online repository (requires internet), with the function `load_dataset(string)`.
You can obtain the list of available datasets with `get_dataset_names()`
You can find in the references the page on github that hosts them.

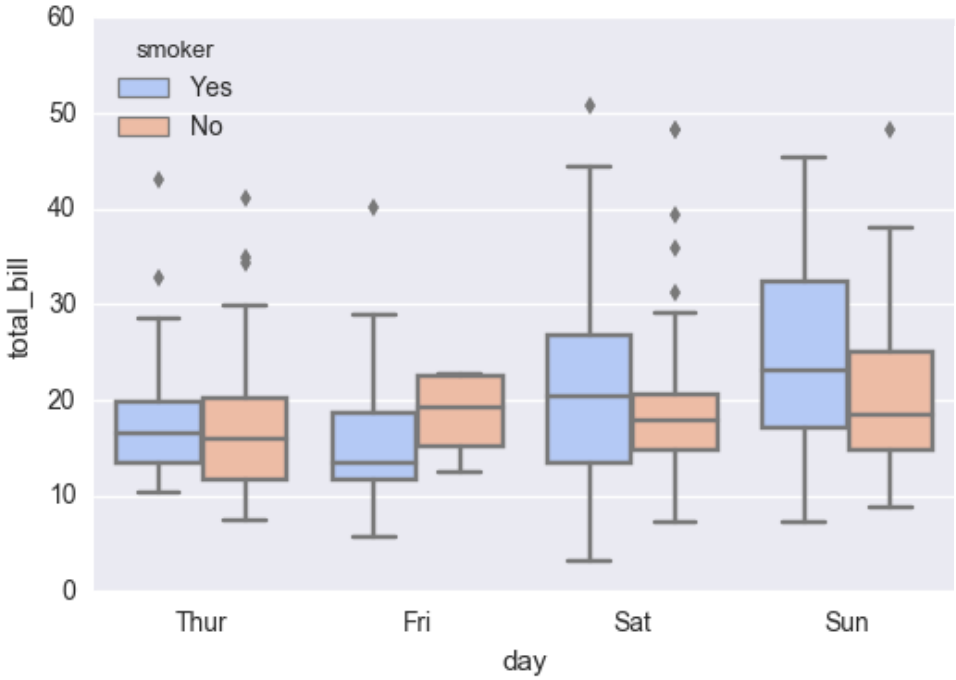
```
import seaborn as sns
%matplotlib inline
sns.get_dataset_names()
tips = sns.load_dataset('tips')
tips.head()
```

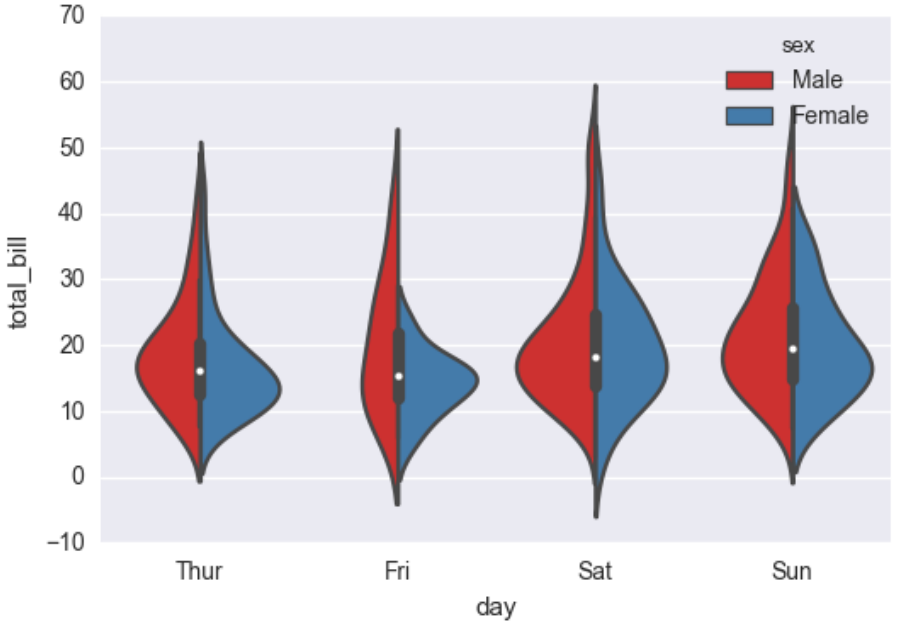
Feel free to experiment with all the available datasets.
For our exercises you must use the tips dataset, that gives 244 record about the customers of a restaurant.

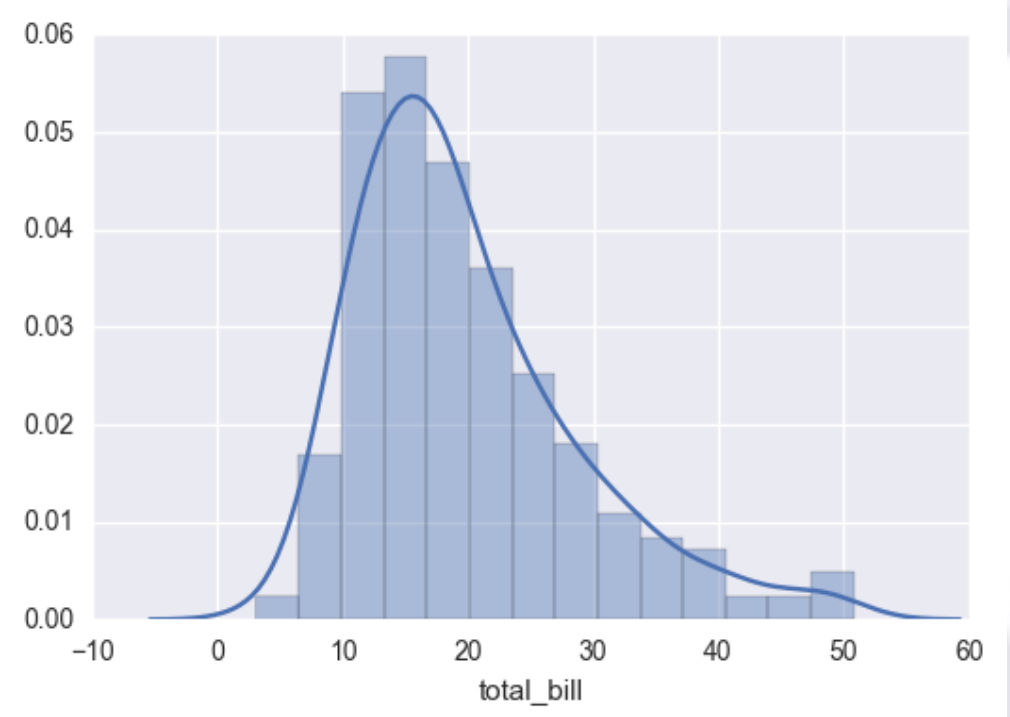
Your job will be to reproduce the plot shown in the following pages:

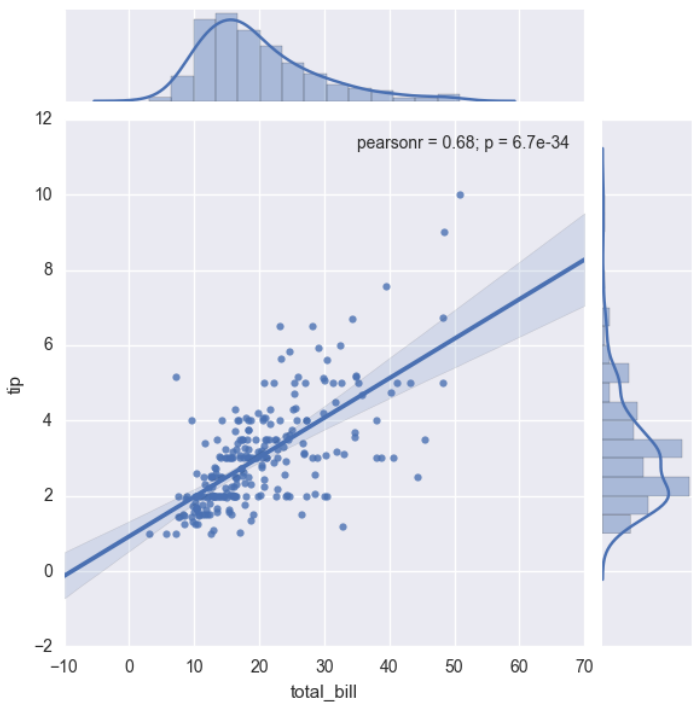

160
140
120
100
80
60
40
20
0
count
Male
Female
sex

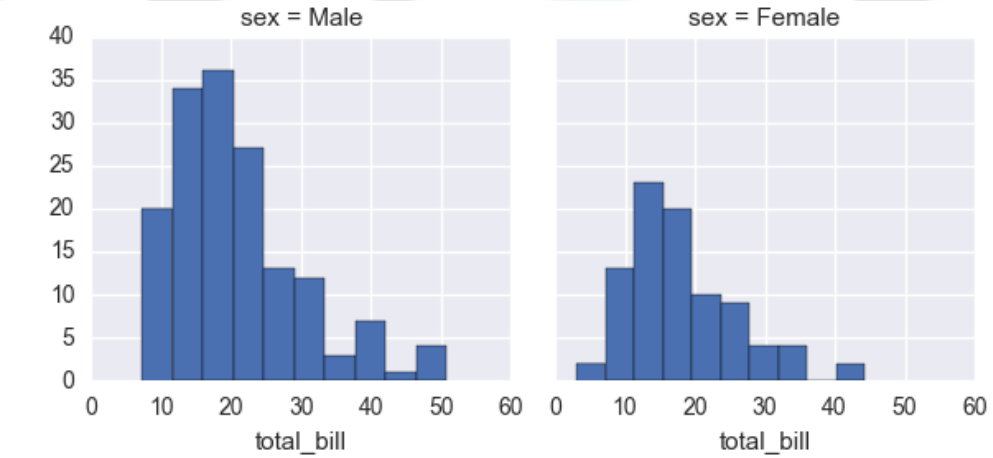

60
50
40
30
20
10
0
total_bill
Thur
Fri
Sat
Sun
day

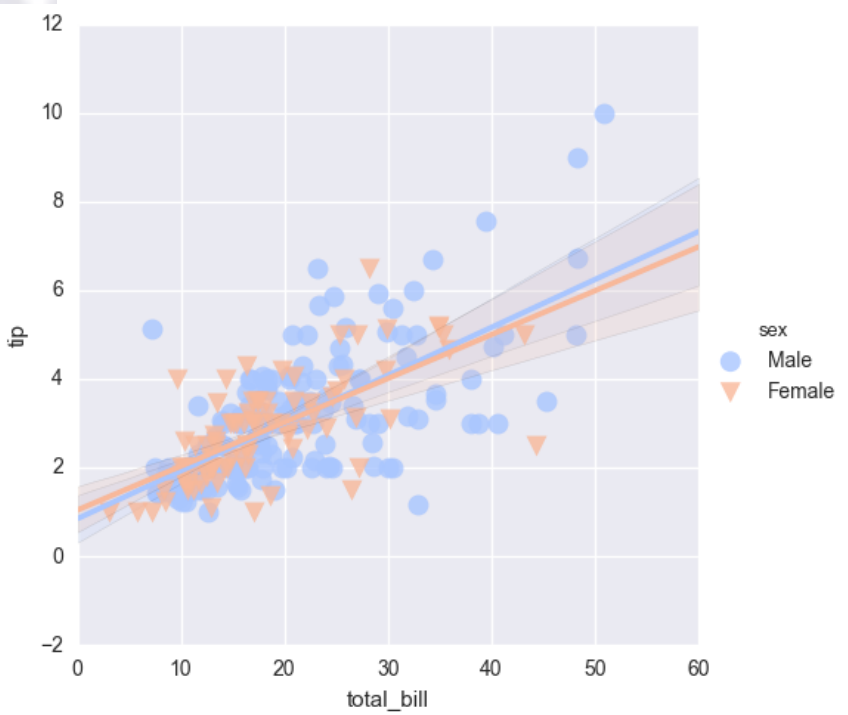

smoker
Yes
No
60
50
40
30
20
10
0
total_bill
Thur
Fri
Sat
Sun
day

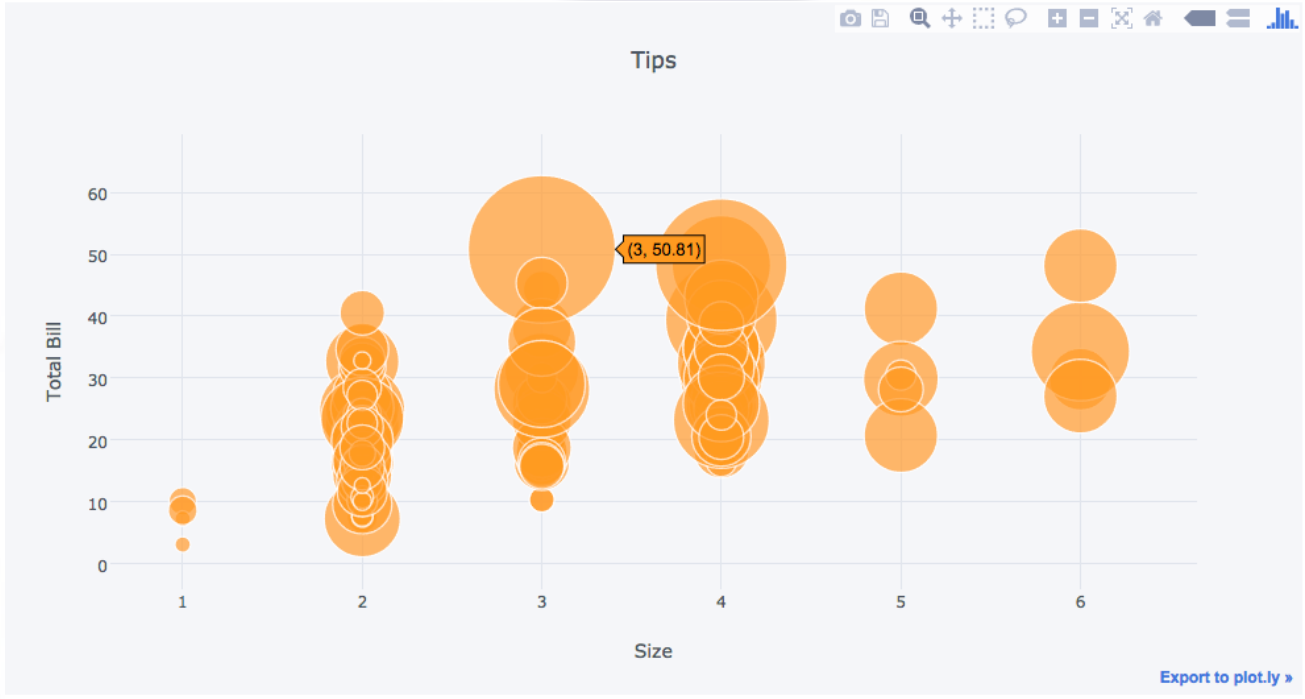

sex
Male
Female
70
60
50
40
30
20
10
0
−10
total_bill
Thur
Fri
Sat
Sun
day

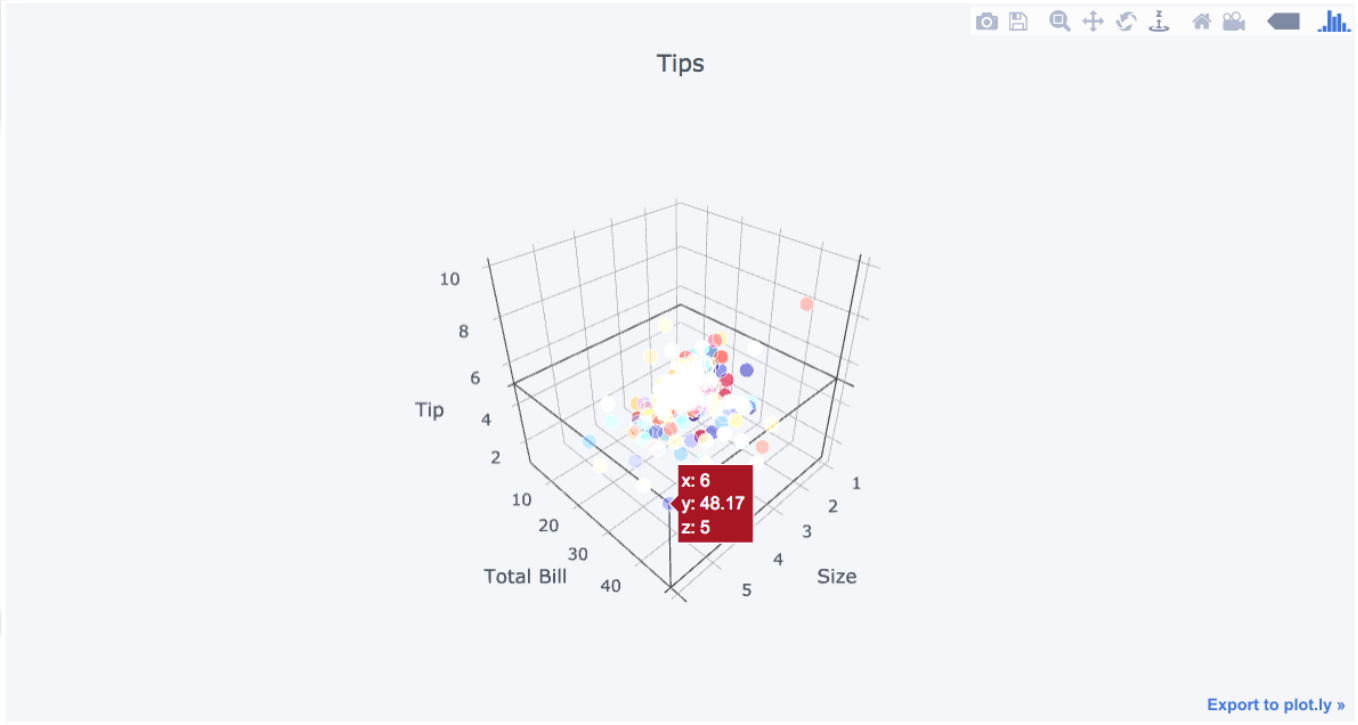

0.06
0.05
0.04
0.03
0.02
0.01
0.00
−10
0
10
20
30
40
50
60
total_bill


pearsonr = 0.68; p = 6.7e-34
tip
total_bill
12
10
8
6
4
2
0
−2
−10
0
10
20
30
40
50
60
70


sex = Male
sex = Female
40
35
30
25
20
15
10
5
0
0
10
20
30
40
50
60
total_bill


tip
total_bill
sex
Male
Female
12
10
8
6
4
2
0
−2
0
10
20
30
40
50
60


Tips
Total Bill
Size
(3, 50.81)
0
10
20
30
40
50
60
1
2
3
4
5
6
Export to plot.ly »


Tips
Tip
Total Bill
Size
x: 6
y: 48.17
z: 5
Export to plot.ly »

Exercise

On Kaggle you can find a database that contains the 911 calls of the Montgomery County, PA. The file is downloaded for you and you can find it here:

http://bit.do/911-csv

Explore the data, and build a heatmap of the number of calls per Hour and Day of Week, showing clearly that between 16:00 and 17:00 there is a maximum.

Montgomery County, locally also referred to as Montco, is a county located in the Commonwealth of Pennsylvania. As of the 2010 census, the population was 799,874, making it the third-most populous county in Pennsylvania, after Philadelphia and Allegheny Counties. The county seat is Norristown. Montgomery County is very diverse, ranging from farms and open land in Upper Hanover to densely populated rowhouse streets in Cheltenham.

- http://matplotlib.org/users/pyplot_tutorial.html
- http://matplotlib.org/gallery.html
- http://www.labri.fr/perso/nrougier/teaching/matplotlib/
- http://matplotlib.org/api/pyplot_api.html
- http://scipy-lectures.github.io/matplotlib/matplotlib.html
- http://seaborn.pydata.org/
- http://seaborn.pydata.org/examples/index.html
- https://plot.ly/python/reference/
- http://web.quant-platform.com/trial/yves/Plotly_Cufflinks.html
- http://images.plot.ly/plotly-documentation/images/python_cheat_sheet.pdf
- https://plot.ly/ipython-notebooks/cufflinks/
- https://github.com/mwaskom/seaborn-data