|  |  |
| --- | --- |
|  | Visualizing Regressions |
|  | In [1]: import pandas as pd  In [2]: import matplotlib.pyplot as plt  In [3]: import seaborn as sns  In [4]: tips =sns.load\_dataset('tips') |
| lmplot() | In [5]: sns.lmplot(x= 'total\_bill', y='tip', data=tips)  In [6]: plt.show() |
| hue=…  (Grouping Factors) | In [7]: sns.lmplot(x='total\_bill', y='tip', data=tips, hue='sex',  ...: palette='Set1') |
| col=…  (subplots) | In [9]: sns.lmplot(x='total\_bill', y='tip', data=tips, col='sex') |
| residplot() | In [11]: sns.residplot(x='age',y='fare',data=tips,color='indianred') |
|  | Visualizing univariate distributions |
| stripplot() | In [1]: sns.stripplot(y= 'tip', data=tips) |
| In [4]: sns.stripplot(x='day', y='tip', data=tip). ---🡪 (grouping with srtipplot()) |
| In [7]: sns.stripplot(x='day', y='tip', data=tip, size=4, jitter=True) --🡪 (spreading out) |
| swarmplot() | In [10]: sns.swarmplot(x='day', y='tip', data=tips) |
| In [13]: sns.swarmplot(x='day', y='tip', data=tips, hue='sex') --🡪 grouping |
| In [16]: sns.swarmplot(x='tip', y='day', data=tips, hue='sex', orient='h') |
| boxplot() | In [20]: sns.boxplot(x='day', y='tip', data=tips) |
| violinplot() | In [23]: sns.violinplot(x='day', y='tip', data=tips) |
| Combining plots | In [27]: sns.violinplot(x='day', y='tip', data=tips, inner=None, color='lightgray')  In [28]: sns.stripplot(x='day', y='tip', data=tips, size=4, jitter=True) |
|  | Visualizing Multivariate Distributions |
| jointplot() | In [1]: sns.jointplot(x= 'total\_bill', y= 'tip', data=tips) |
| kde=True | In [3]: sns.jointplot(x='total\_bill', y= 'tip', data=tips, kind='kde') |
| pairplot() | In [5]: sns.pairplot(tips) |
| In [7]: sns.pairplot(tips, kind=’reg’, hue='sex') --🡪 grouping |
| heatmap() | In [10]: sns.heatmap(covariance) |
|  | Exercises |
| plt.scatter() | plt.scatter(auto['weight'], auto['mpg'], label='data', color='red', marker='o') |
| sns.regplot() | sns.regplot(x='weight', y='mpg', data=auto, scatter=None, color='blue', label='order 1') |
|  | Visualizing time series |
|  | Datetime index: specialized slicing  ● e.g., weather[‘2010-07-04’]  ● e.g., weather[‘2010-03’:’2010-04’]  ● e.g., weather[‘2010-05’] |
| Slicing time series | In [2]: march\_apr = temperature['2010-03':'2010-04'] # data of March & April 2010 only  In [4]: march\_apr.iloc[-4:] |
| Ploting time series slices | In [1]: plt.plot(temperature['2010-01'], color='red', label='Temperature')  In [2] dew point = weather['DewPoint']  In [3]: plt.plot(dewpoint['2010-01'], color='blue', label='Dewpoint')  In [4]: plt.legend(loc='upper right')  In [5]: plt.xticks(rotation=60)  In [6]: plt.show() |
| Selecting & formating dates | In [1]: jan = temperature['2010-01']  In [2]: dates = jan.index[::96] # Pick every 4th day ## DatetimeIndex([‘2010-01-01’, ‘2010-01-05’,….  In [4]: labels = dates.strftime('%b %d') # Make formatted labels ## [‘Jan 01’ ‘Jan 05’…. |
|  | Time series with moving windows |
| Viewing 24 hour averages | In [1]: plt.plot(smoothed['1d']) # moving average over 24 hours |
| Viewing all moving averages | In [1]: plt.plot(smoothed['2010-01']) # plot DataFrame for January |
| Moving standard deviations | In [1]: plt.plot(variances['2010-01'])  In [2]: plt.legend(variances.columns) |
|  | Histogram equalization in images |
| Image histograms | In [1]: orig = plt.imread('low-contrast-moon.jpg')  In [2]: pixels = orig.flatten()  In [3]: plt.hist(pixels, bins=256, range=(0,256), normed=True, color='blue', alpha=0.3)  In [4]: plt.show()  In [5]: minval, maxval = orig.min(), orig.max()  In [6]: print(minval, maxval). ###. 125 244 |
| Rescaling the image | In [3]: rescaled = (255/(maxval-minval)) \* (pixels - minval)  In [4]: print(rescaled.min(), rescaled.max()). #### 0.0 255.0 |
| Original & rescaled histograms | In [1]: plt.hist(orig.flatten(), bins=256, range=(0,255), normed=True, color='blue', alpha=0.2))  In [2]: plt.hist(rescaled.flatten(), bins=256, range=(0,255), normed=True, color='green', alpha=0.2))  In [3]: plt.legend(['original', 'rescaled']) |
| Image histogram & CDF | In [1]: plt.hist(pixels, bins=256, range=(0,256), normed=True, color='blue', alpha=0.3)  In [2]: plt.twinx()  In [3]: orig\_cdf, bins, patches = plt.hist(pixels, cumulative=True, bins=256, range=(0,256), normed=True, color='red', alpha=0.3) |
| Equalizing intensity values | In [1]: new\_pixels = np.interp(pixels, bins[:-1], orig\_cdf\*255)  In [2]: new = new\_pixels.reshape(orig.shape) |
| Equalized histogram & CDF | In [1]: plt.hist(new\_pixels, bins=256, range=(0,256), normed=True, color='blue', alpha=0.3)  In [2]: plt.twinx()  In [3]: plt.hist(new\_pixels, cumulative=True, bins=256, range=(0,256), normed=True, color='red', alpha=0.1) |