Data Processing, Analysis and Visualization in Python

Seaborn

1

Outline

- Introduction
- Setup and import data
- Histograms, KDEs, and densities – histograms
- Pair plots
- Faceted histograms
- Factor Plots (Box)
- Joint distributions
- Bar Plots
- Pie Charts
- Histograms

Seaborn

2

What is Matplotlib?

- Seaborn is a multi-platform data visualization library built on NumPy arrays and MatPlotLib, and designed to work with the broader SciPy stack.

NumPy matplotlib IP[y]: IPython SymPy pandas

https://seaborn.pydata.org/

3

What is Seaborn?

- Seaborn provides an Application Programming Interface (API) on top of Matplotlib that defines simple high-level functions for common statistical plot types and integrates with the functionality provided by pandas DataFrames.

Seaborn

https://seaborn.pydata.org/

4

Seaborn vs Matplotlib

```
>>> import seaborn as sns
>>> sns.__version__
'0.11.2'
```

- You'll also like to do:

```
>>> import numpy as np
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
>>> plt.style.use('default')
```

- If you're working in Jupyter notebook or other IPython

```
%matplotlib inline
%reload_ext autoreload
%autoreload 2
```

5

Seaborn vs Matplotlib

```
>>> import matplotlib.pyplot as plt
>>> plt.style.use('default')
>>> import numpy as np
>>> import pandas as pd
```

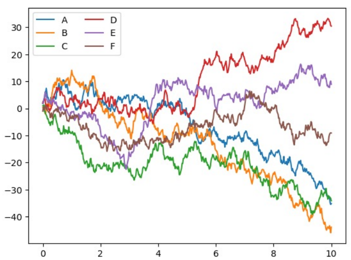
- Example create random walk process

```
>>> rng = np.random.RandomState(0)
>>> x = np.linspace(0, 10, 500)
>>> y = np.cumsum(rng.randn(500, 6), 0)
```

6

Seaborn vs Matplotlib

```
>>> # Plot the data with Matplotlib defaults
>>> plt.plot(x, y); plt.legend('ABCDEF', ncol=2,
                               loc='upper left')
```



7

Seaborn vs Matplotlib

```
>>> import seaborn as sns
>>> sns.set() # set style
>>> # Plot the data with seaborn defaults (same code)
>>> plt.plot(x, y); plt.legend('ABCDEF', ncol=2,
                               loc='upper left')
```

8

Seaborn plots

- Seaborn provides high-level commands to create a variety of plot types useful for statistical data exploration, and even some statistical model fitting.
- Note that all of the following examples could be done using raw Matplotlib commands (this is, in fact, what Seaborn does under the hood) but the Seaborn API is much more convenient.

9

Seaborn plotting steps

- The basic steps to creating plots with seaborn are:

1. Prepare data
2. Control figure aesthetics
3. Plot
4. Customize plot
5. Save plot
6. Show plot

10

Seaborn datasets

- Seaborn has a set of built-in datasets for practice:

```
>>> print(sns.get_dataset_names())
['anagrams', 'anscombe', 'attention',
'brain_networks', 'car_crashes',
'diamonds', 'dots', 'exercise',
'flights', 'fmri', 'gammas', 'geyser',
'iris', 'mpg', 'penguins', 'planets',
'taxis', 'tips', 'titanic']
```

11

Seaborn datasets

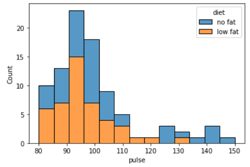
- Seaborn has a set of built-in datasets for practice:

```
>>> exercise = sns.load_dataset('exercise')
>>> exercise.head()
Unnamed:     0  id     diet  pulse    time  kind
0            0   1  low fat     85   1 min  rest
1            1   1  low fat     85  15 min  rest
2            2   1  low fat     88  30 min  rest
3            3   2  low fat     90   1 min  rest
4            4   2  low fat     92  15 min  rest
```

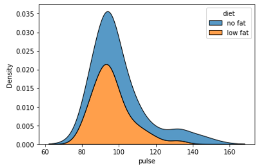
12

Seaborn Figure-level plots

```
>>> sns.histplot(data=exercise, x='pulse', hue='diet',
                 multiple='stack')
```



```
>>> sns.kdeplot(data=exercise, x='pulse', hue='diet',
                multiple='stack')
```

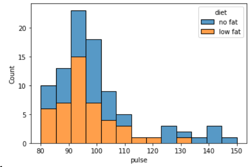


Which column in the data frame, you want to use for color encoding

13

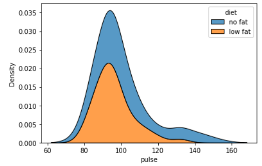
Seaborn Axes-level plots

```
>>> sns.displot(data=exercise, x='pulse', hue='diet',
                multiple='stack', kind='hist')
```



Specify the # of bins


```
>>> sns.displot(data=exercise, x='pulse', hue='diet',
                multiple='stack', kind='kde')
```



14

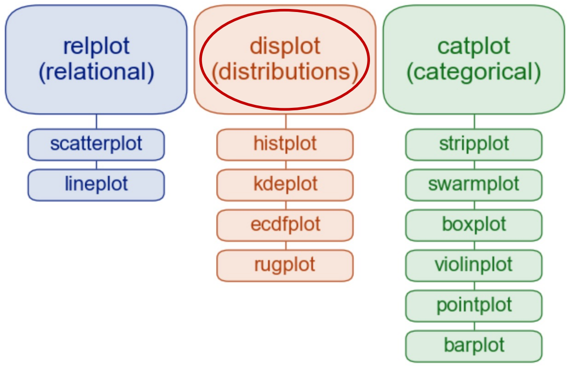
Play with Seaborn datasets

- Load additional Seaborn datasets and display the data

```
['anagrams', 'anscombe',
'attention', 'brain_networks',
'car_crashes', ...ds', 'dots',
'dowjones', ' ', 'flights',
'fmri', 'glue',
'healthexp', 'iris', 'mpg',
'penguins', 'planets', 'seaice',
'taxis', 'tips', 'titanic']
```

15

Seaborn structure



https://seaborn.pydata.org/tutorial/function_overview.html

16

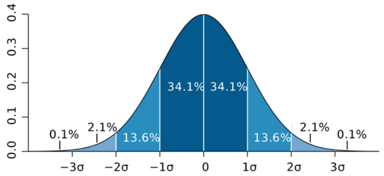
Histograms, KDEs, and densities – histograms

- Normal distribution

$$y = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Mean Variance
(σ - standard deviation)

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$



*d*D

$$y = \frac{1}{\sqrt{|\Sigma|(2\pi)^d}} e^{-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)^T}$$

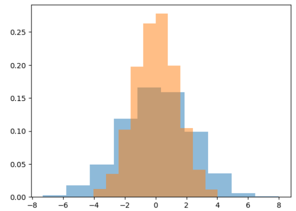
Mean (dimension d) Variance (dimension d²)

17

Histograms, KDEs, and densities – histograms

- Often in statistical data visualization, one wants to plot histograms and joint distributions of variables. This is relatively straightforward in Matplotlib (1D example):

```
>>> mean = [0, 0]
>>> cov = [[5, 2],[2, 2]]
>>> data=np.random.multivariate_normal(mean, cov, size=100)
>>> data = pd.DataFrame(data, columns=['x', 'y'])
>>> for col in 'xy':
        plt.hist(data[col], density=True, stacked=True, alpha=0.5)
```



$$y = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

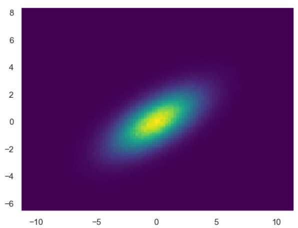
Mean Variance

18

Histograms, KDEs, and densities – histograms

- Often in statistical data visualization, one wants to plot histograms and joint distributions of variables. This is relatively straightforward in Matplotlib (2D example):

```
>>> mean = [0, 0]
>>> cov = [[5, 2],[2, 2]]
>>> data=np.random.multivariate_normal(mean, cov, size=100)
>>> data = pd.DataFrame(data, columns=['x', 'y'])
>>> h = plt.hist2d(x=data['x'], y=data['y'], bins=100, cmap='viridis')
```



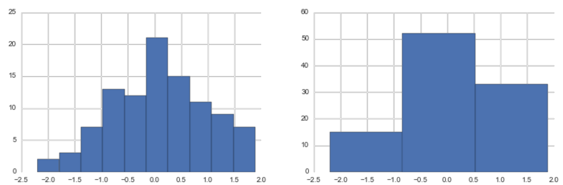
$$y = \frac{1}{\sqrt{|\Sigma|(2\pi)^d}} e^{-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)^T}$$

Mean (dimension d) Variance (dimension d²)

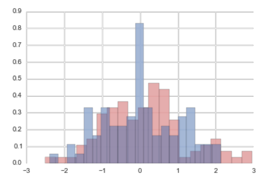
19

Histograms, KDEs, and densities – kdeplots

- Problems with histograms:
  - What bin size to use?



  - What's the correct x-axis alignment?
  - What should the width of the bin be?



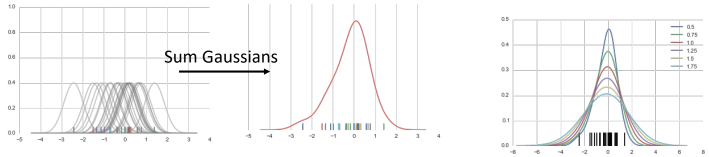
20

Histograms, KDEs, and densities

- Rather than a histogram, we can get a smooth estimate of the distribution using Kernel Density Approximation (KDE)

$$f^{KDE}(x) = \sum_{i=1}^{N_C} c_i \left\{ \frac{1}{hN_{K_i}} \cdot \sum_{j=1}^{N_{K_i}} K_{ij} \left(\frac{x - x_{ij}}{h} \right) \right\}$$

- KDE work by
  - Passing a strongly sharped peak (kernel function, like a Gaussian) over each data point on the x-axis
  - To get the KDE we simply sum the value of all the gaussians
  - The bandwidth of the Gaussian changes the plot. As a rule of thumb, a wider bandwidth is used for smooth data sets, but a narrow bandwidth is good for data sets with lots of wiggle.



21

Histograms, KDEs, and densities

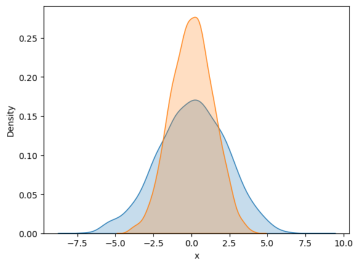
- Advantages of KDE plots over histograms:
  - Information isn't lost by "binning" as is in histograms, this means KDEs are unique for a given bandwidth and kernel.
  - They are smoother, which is easier for feeding back into a computer for further calculations.

22

Histograms, KDEs, and densities – kdeplots

- Rather than a histogram, we can get a smooth estimate of the distribution using Seaborn's kernel density estimation (KDE) plot `sns.kdeplot()`:

```
>>> for col in 'xy':
        sns.kdeplot(data[col], shade=True)
```

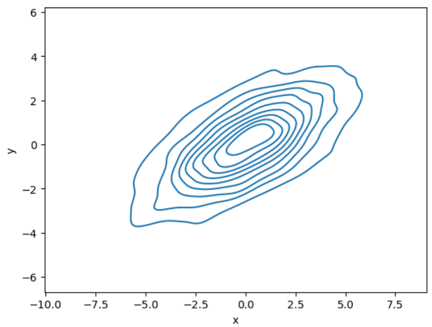



```
bw_method=0.5
bw_method=0.05
bw_method='scott'
bw_method='silverman'
```

23

Histograms, KDEs, and densities – kdeplots

```
>>> sns.kdeplot(x=data.x, y=data.y)
```

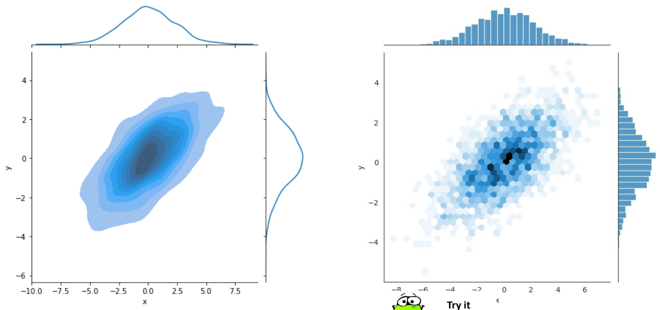


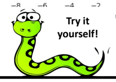

```
shade=True, cmap="Blues", thresh=0.05
```

24

Histograms, KDEs, and densities – jointplots

```
>>> with sns.axes_style('white'):
        sns.jointplot(x=data.x, y=data.y, kind='kde', shade=True)
        sns.jointplot(x=data.x, y=data.y, data, kind='hex')
```



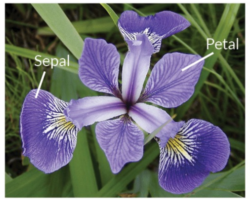


Try with regression

25

Pair plots

- Generalize joint plots to datasets of larger dimensions
- Useful for exploring correlations between multidimensional data
- https://github.com/mwaskom/seaborn-data
- Famous iris dataset with 150 iris flowers with
  - 3 species: “setosa”, “virginica”, and “versicolor”
    4 features: “sepal_length”, “sepal_width”, “petal_length”, “petal_width”



Iris Versicolor

Iris Setosa

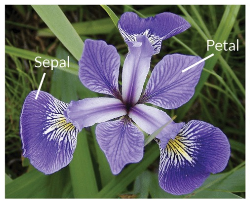
Iris Virginica

26

Pair plots

```
>>> iris = sns.load_dataset("iris"); iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa



Iris Versicolor

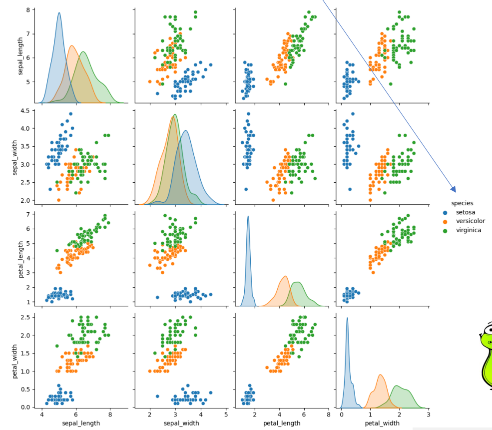
Iris Setosa

Iris Virginica

27

Pair plots

```
>>> sns.pairplot(iris, hue='species', height=2.5)
```



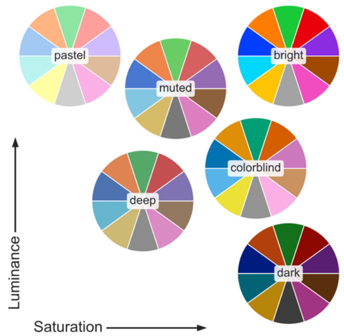
https://seaborn.pydata.org/tutorial/color_palettes.html

Try changing the palette

28

Palettes

- Seaborn in fact has six variations of matplotlib's palette:



29

Faceted histograms

- Histograms of subsets

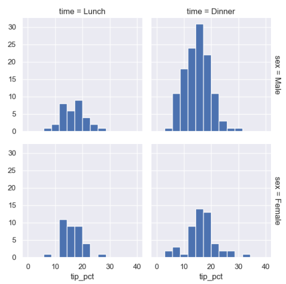
```
>>> tips = sns.load_dataset('tips'); tips.head()
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
```

30

Faceted histograms

- Histograms of subsets

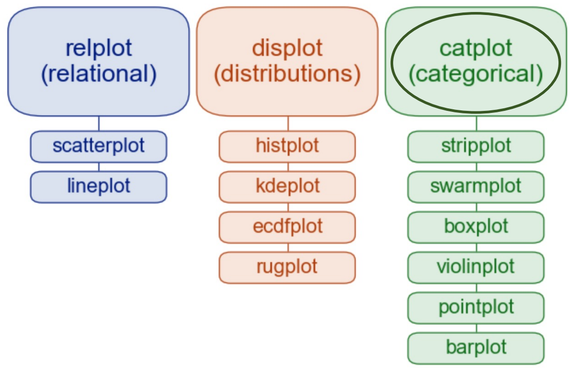
```
>>> tips['tip_pct'] = 100 * tips['tip'] /
                          tips['total_bill']
>>> grid = sns.FacetGrid(tips, row='sex', col='time',
           margin_titles=True)
>>> grid.map(plt.hist, 'tip_pct', bins=np.linspace(0,
             40, 15))
```



Compare using
`plt.style.use('default')` and
`sns.set()`

31

Seaborn structure



https://seaborn.pydata.org/tutorial/function_overview.html

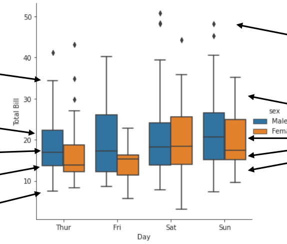
34

Box Plots

- View the distribution of a parameter (total_bill) within bins defined by any other parameter (day)

darkgrid, whitegrid, dark, white, and ticks

```
>>> with sns.axes_style(style='ticks'):
        g=sns.catplot(x='day', y='total_bill', hue='sex',
                         data=tips, kind='box')
        g.set_axis_labels('Day', 'Total Bill')
```



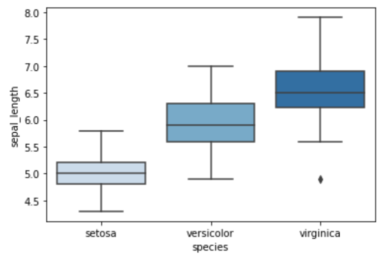
max
Q$_3$
median
Q$_1$
min
outliers
25% of the data in each region

35

Box Plots

- Returning to our flowers …

```
>>> sns.boxplot(x='species', y='sepal_length',
                palette='Blues', data=iris)
```



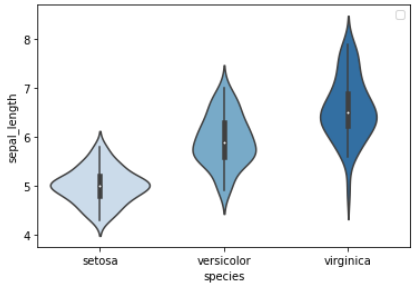
Reverse the order of the box plots along the x-axis
Reverse horizontally

36

Violin Plots

- Violin plot is a combination of box and kde plots

```
>>> sns.violinplot(x='species', y='sepal_length',
palette='Blues', data=iris)
```



Try the following built-in styles:

```
sns.set_context('talk') # 'paper', 'talk', 'poster'
```

37

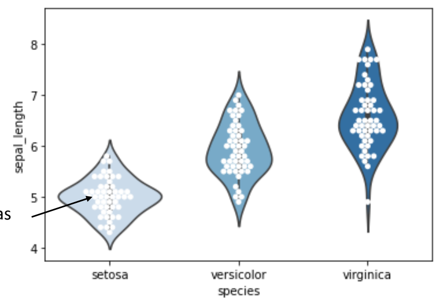
Violin & Swarm Plots

- Swarm plots may complement box or violin plots

```
>>> sns.violinplot(x='species', y='sepal_length',
     palette='Blues', data=iris)
>>> sns.swarmplot(x='species', y='sepal_length',
     color='White', data=iris)
```

- Change shape of figure
- Try palette='PuBuGn '

Only position along y-axis has meaning

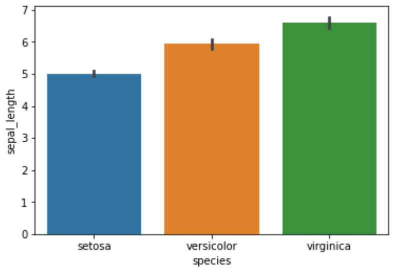


38

Bar Plots

- Time series can be plotted using `sns.factorplot()`

```
>>> sns.barplot(x='species', y='sepal_length', data=iris,
                estimator=np.mean)
median
std
var
covar
min
max
```

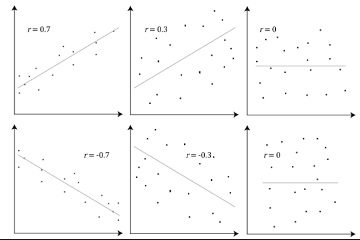


41

Matrix plots – Heat maps

- Correlations in data may be calculated using correlation coefficients.
- Pearson correlation can identify linear correlation in data:

$$C_{xy} = \frac{\sum_{i=1}^{N}(x_i - x)\sum_{i=1}^{N}(y_i - y)}{\sqrt{\sum_{i=1}^{N}(x_i - x)^2}\sqrt{\sum_{i=1}^{N}(y_i - y)^2}} = \frac{\mathrm{cov}(X, Y)}{\sigma_X \sigma_Y}$$



43

Matrix plots – Heat maps

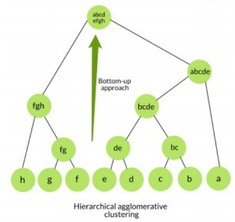
```
>>> iris_mat = iris.corr(method='pearson') # kendall, spearman
>>> print(iris_mat)
>>> plt.figure(figsize=(8,6))
>>> sns.set_style('white')
>>> sns.set_context('paper', font_scale=1.4);
>>> sns.heatmap(iris_mat, annot=True, cmap='Blues',
                linecolor='grey', linewidth=3)
```

To move the colorbar, use sns.heatmap with, e.g.,
`cbar_kws = dict(use_gridspec=False,location='top')`

44

Matrix plots – Cluster maps

- Cluster maps allow us to discover structure in heatmap data using an agglomerative (bottom-up) hierarchical clustering.
- In agglomerative clustering, we start with considering each data point as a cluster and then repeatedly combine two nearest clusters into larger clusters until we are left with a single cluster.
- The graph we plot after performing agglomerative clustering on data is called Dendrogram.



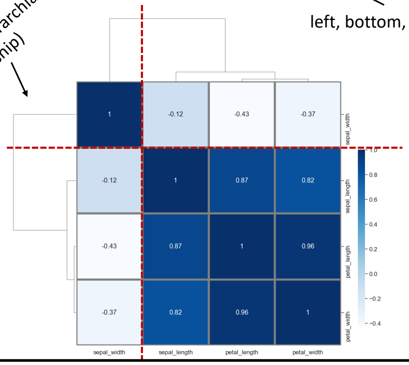
45

Matrix plots – Cluster maps

```
>>> g = sns.clustermap(iris_mat, annot=True, cmap='Blues',
                       linecolor='grey', linewidth=3)
>>> g.ax_cbar.set_position((1.0, 0.1, 0.02, 0.5))
```

left, bottom, width, height

Dendogram (hierarchial relationship)

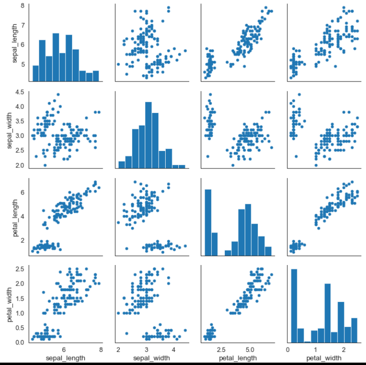


46

PairGrid

- We can use PairGrid to control the many figures on a grid

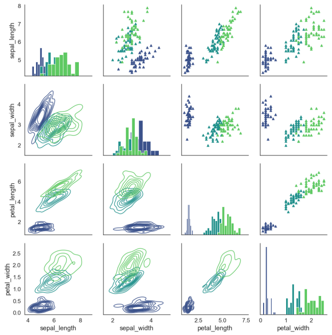
```
>>> iris_g = sns.PairGrid(iris)
>>> iris_g.map_offdiag(plt.scatter)
>>> iris_g.map_diag(plt.hist)
```



47

PairGrid

```
>>> iris_g = sns.PairGrid(iris, hue='species', palette='viridis')
>>> iris_g.map_diag(plt.hist)
>>> iris_g.map_upper(plt.scatter, marker='^')
>>> iris_g.map_lower(sns.kdeplot)
```

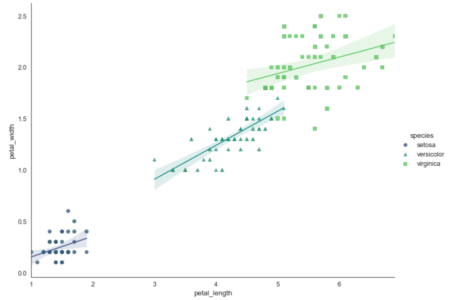


48

Simple Regression plots

- Regression plots can be plotted using `sns.lmplot()`

```
>>> sns.lmplot(data=iris, hue='species', palette='viridis',
      x='petal_length', y='petal_width', markers=['o', '^',
      's'], scatter_kws={'s': 50, 'linewidth':[0.1, 0.3,
      0.5], 'edgecolor':'g'})
```

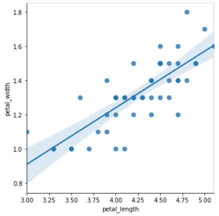


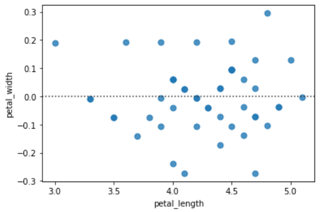
51

Simple Regression plots

- Separating the data series `sns.lmplot()`

```
>>> iris_st = iris[iris['species']=='setosa']
>>> iris_vr = iris[iris['species']=='virginica']
>>> iris_vc = iris[iris['species']=='versicolor']
>>> sns.lmplot(x='petal_length', y='petal_width',
data=iris_vc, markers=['o'], scatter_kws={'s': 50})
>>> sns.residplot(x='petal_length', y='petal_width',
data=iris_vc, scatter_kws={'s': 50})
```





52

56