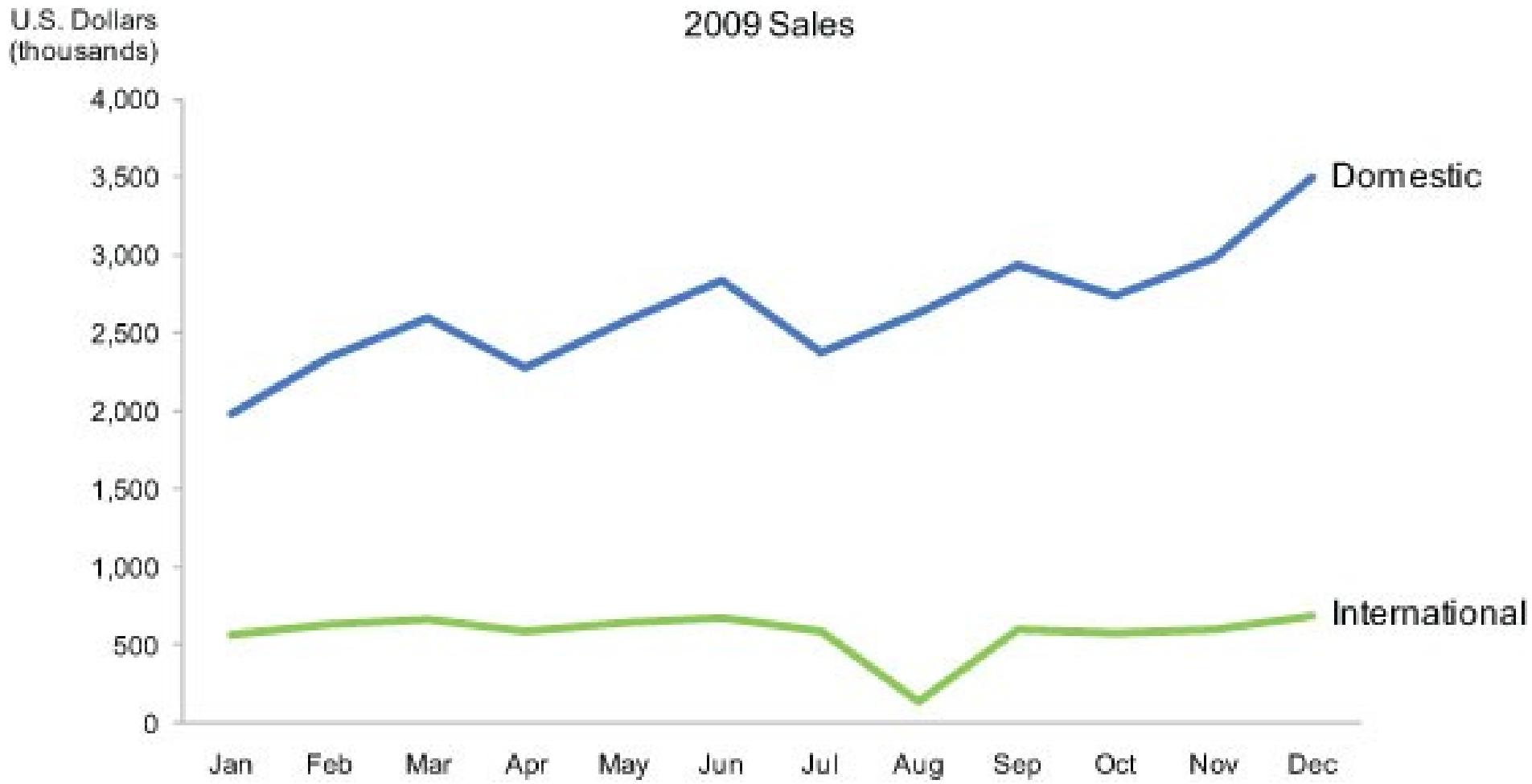
Data visualization

2009 Sales (thousands of U.S. $)

Region	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Total
Domestic	1,983	2,343	2,593	2,283	2,574	2,838	2,382	2,634	2,938	2,739	2,983	3,493	31,783
International	574	636	673	593	644	679	593	139	599	583	602	690	7,005
Total	2,557	2,979	3,266	2,876	3,218	3,517	2,975	2,773	3,537	3,322	3,585	4,183	38,788

2009 Sales (thousands of U.S. $)

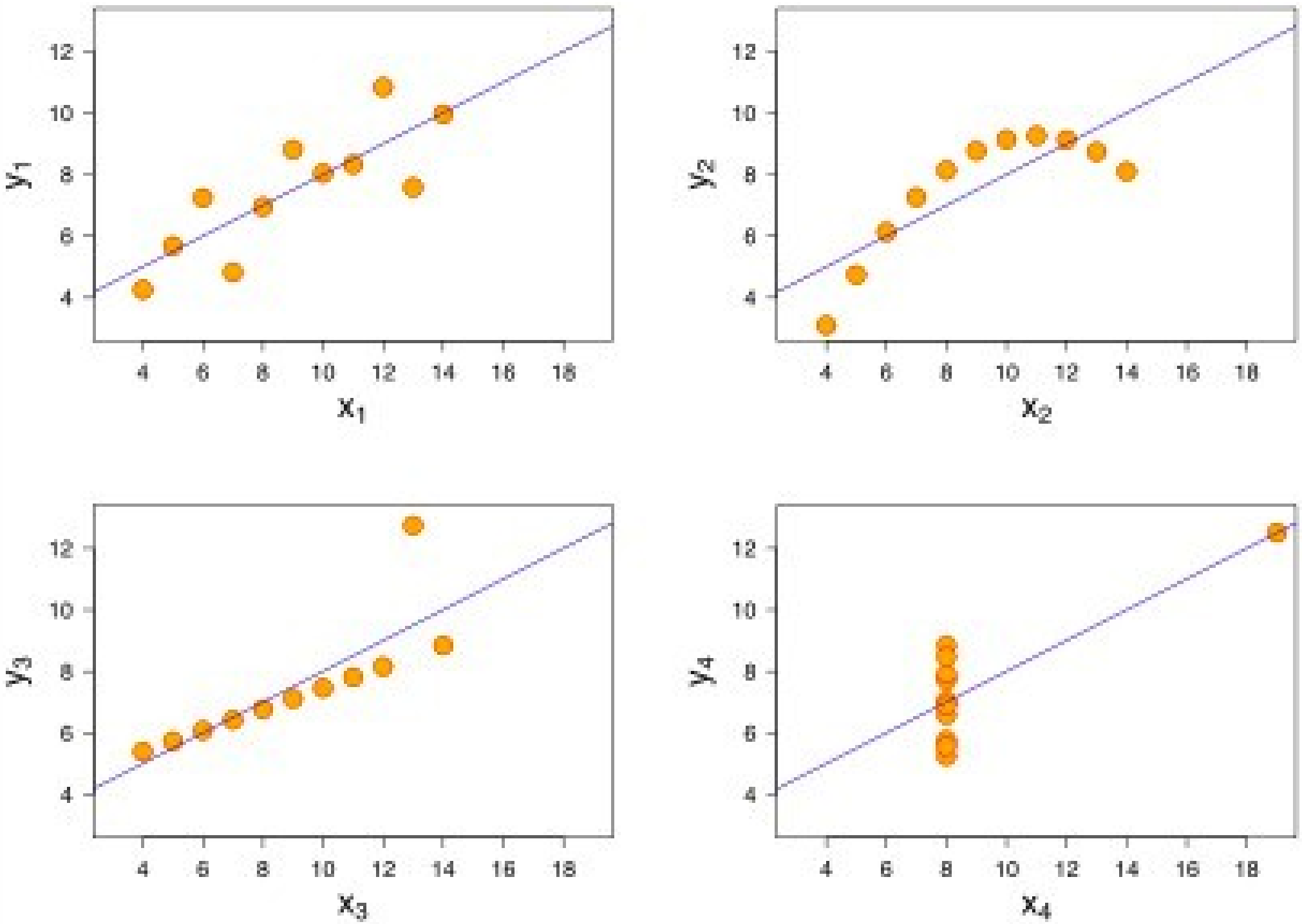
Region	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Total
Domestic	1,983	2,343	2,593	2,283	2,574	2,838	2,382	2,634	2,938	2,739	2,983	3,493	31,783
International	574	636	673	593	644	679	593	139	599	583	602	690	7,005
Total	2,557	2,979	3,266	2,876	3,218	3,517	2,975	2,773	3,537	3,322	3,585	4,183	38,788



Anscombe's quartet

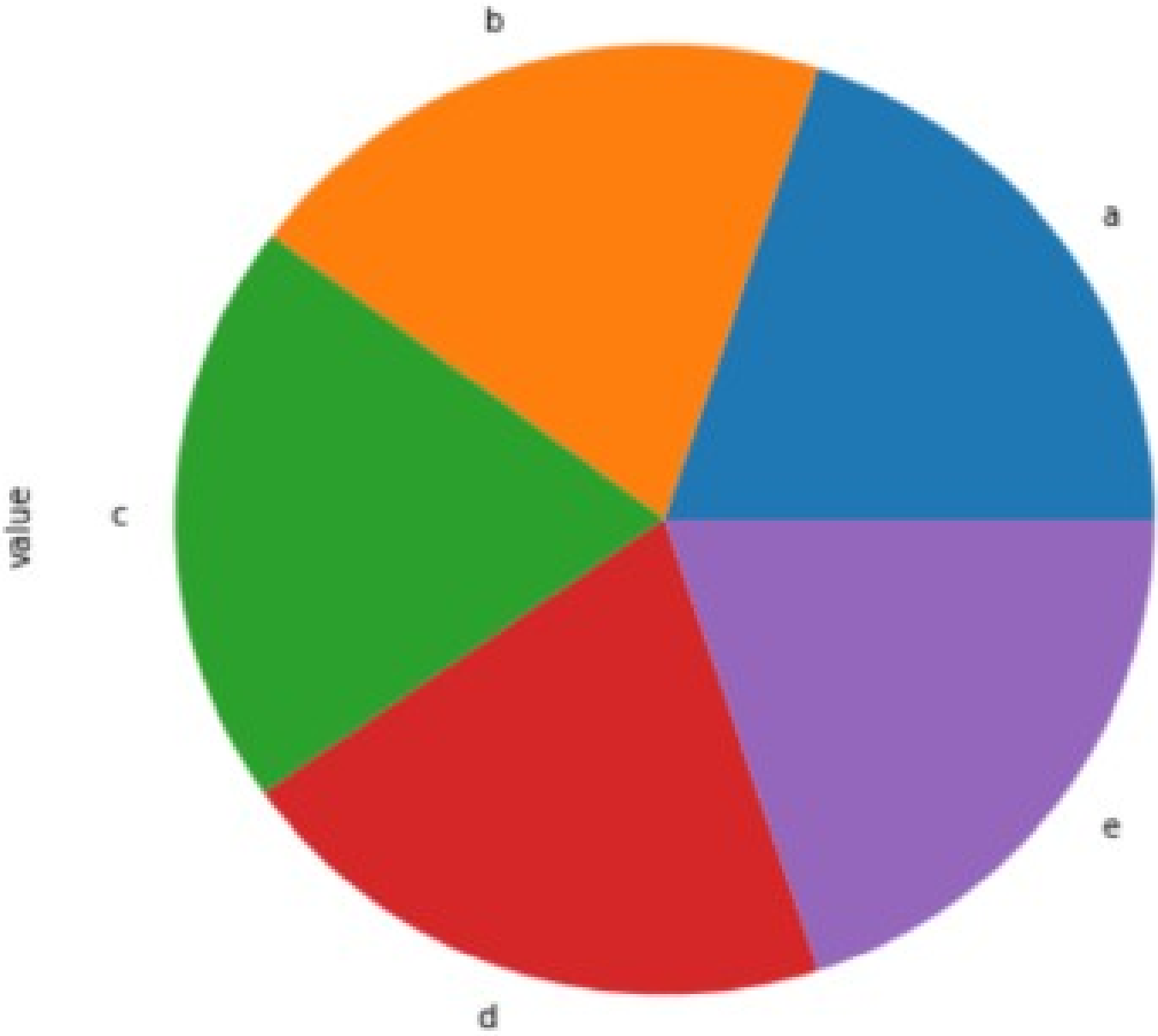
I		II		III		IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

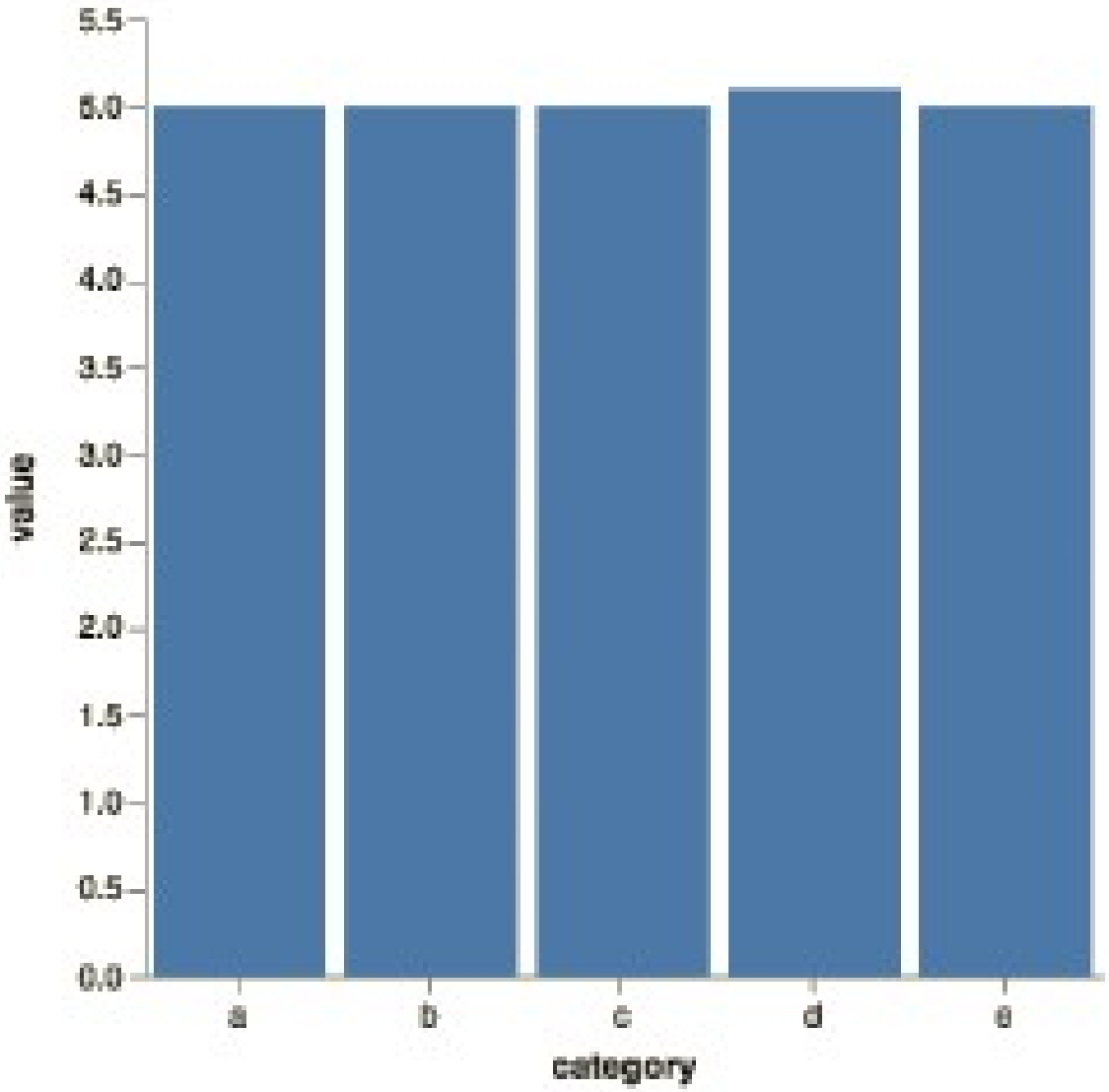
Property	Value
Mean of x in each case	9 (exact)
Variance of x in each case	11 (exact)
Mean of y in each case	7.50 (to 2 decimal places)
Variance of y in each case	4.122 or 4.127 (to 3 decimal places)
Correlation between x and y in each case	0.816 (to 3 decimal places)
Linear regression line in each case	$y = 3.00 + 0.500x$ (to 2 and 3 decimal places, respectively)


y1
x1
y2
x2
y3
x3
y4
x4
4
6
8
10
12
14
16
18

Why visualization

- Show comparisons, contrasts and differences
- Show causality, mechanism, explanation and systematic structure
- Show multivariate data: more than one or two variables


b
a
c
value
e
d


5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
value
a
b
c
d
e
category

AMERICA'S ECONOMY
2010 GROSS DOMESTIC PRODUCT
United States
$14.6 TRILLION
China $5.7 TRILLION
Japan $5.3 TRILLION
Germany $3.3 TRILLION
France $2.5 TRILLION

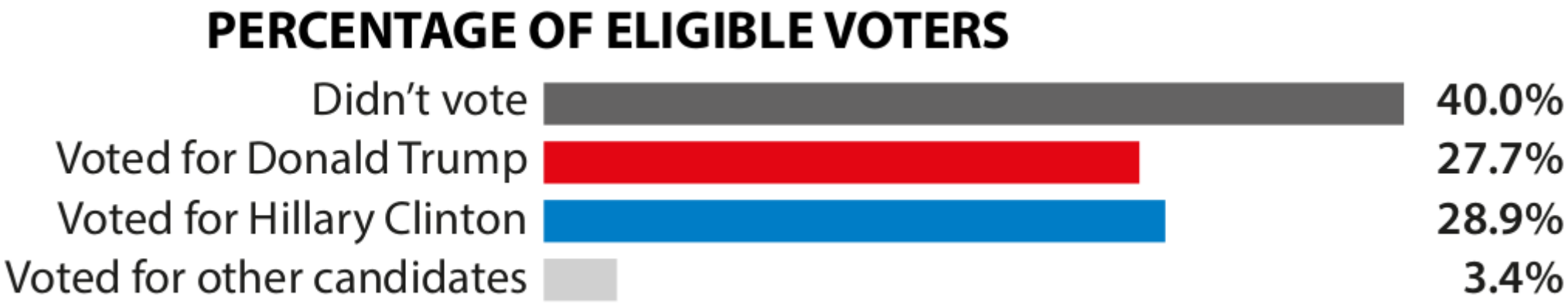
AMERICA'S ECONOMY

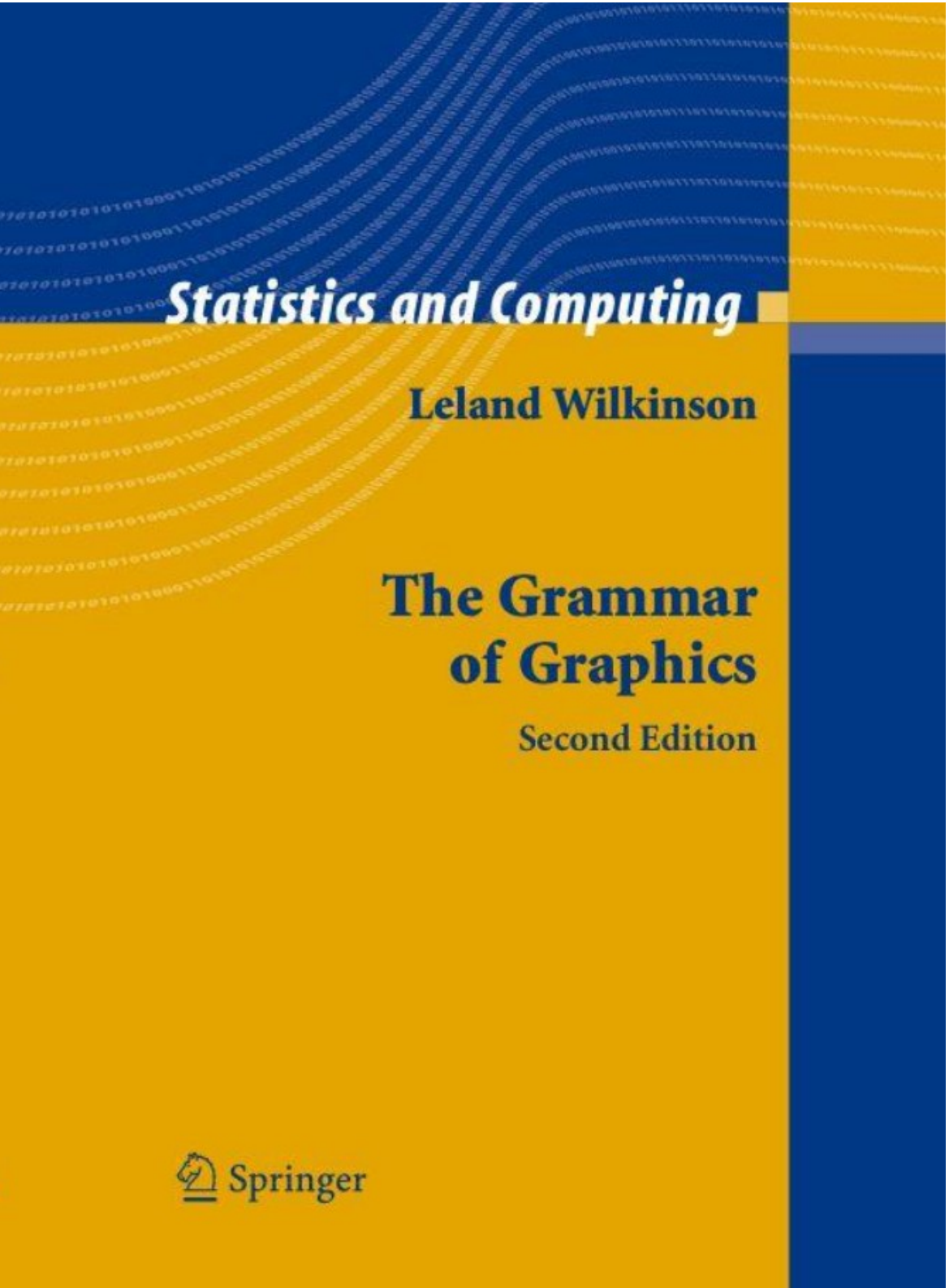
2010 GROSS DOMESTIC PRODUCT

- United States 14.6 Trillion
- China 5.7 Trillion
- Japan 5.3 Trillion
- Germany 3.3 Trillion
- France 2.5 Trillion

Surface on the
county-level map:
Red: 80%
Blue: 20%
VICTORY BY A LANDSLIDE!!!!!!

SHARE OF THE POPULAR VOTE IN THE 2016 PRESIDENTIAL ELECTION


PERCENTAGE OF ELIGIBLE VOTERS
Didn't vote
40.0%
Voted for Donald Trump
27.7%
Voted for Hillary Clinton
28.9%
Voted for other candidates
3.4%

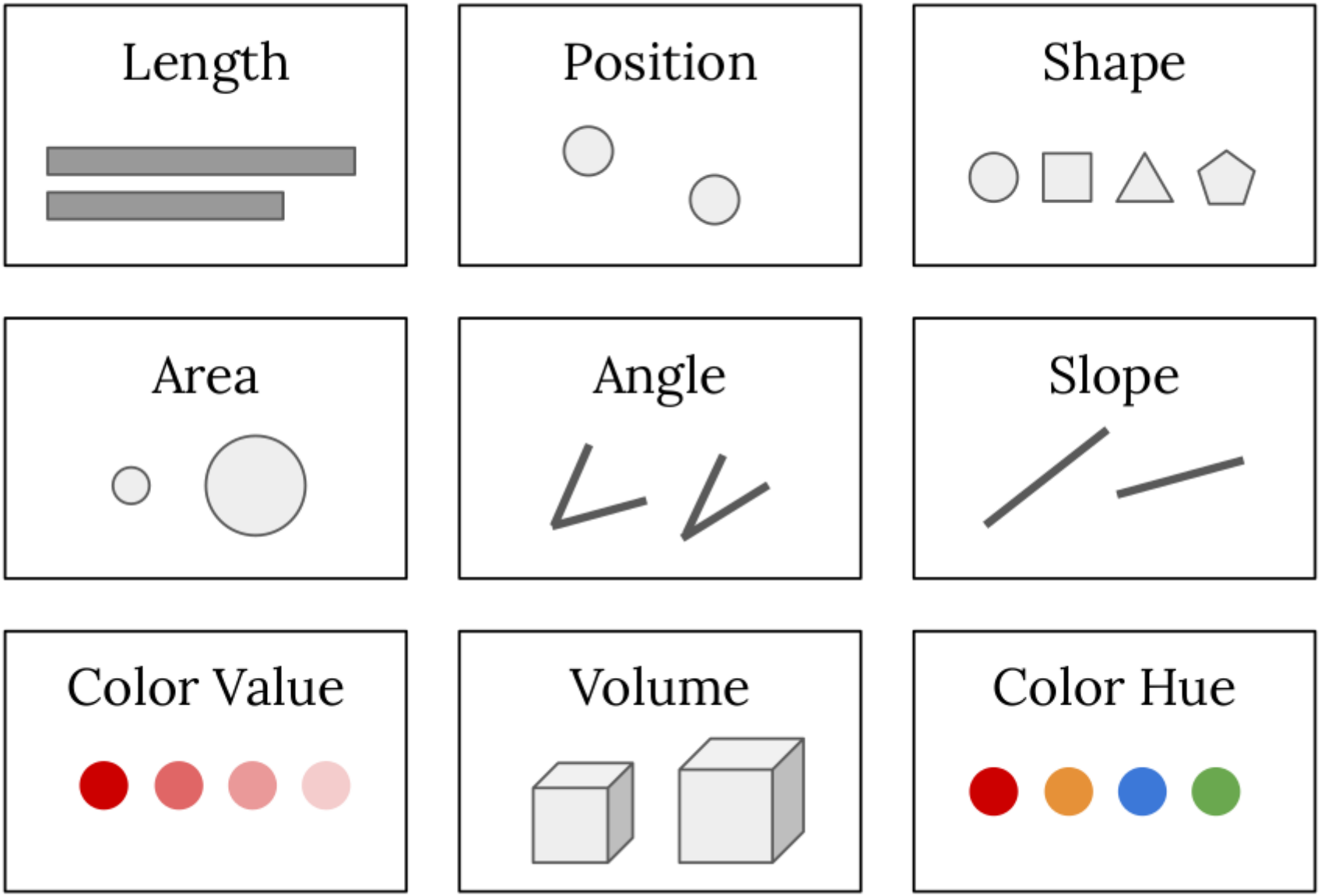


Data: describe observations using variables
Mappings: map data variables to visual qualities
Scales: map values in data space to values in aesthetic space (create axes and legends)
Geometric objects (geoms) constitute the objects seen on a plot
Guides - visualizations of scales (axes, legends, etc.)

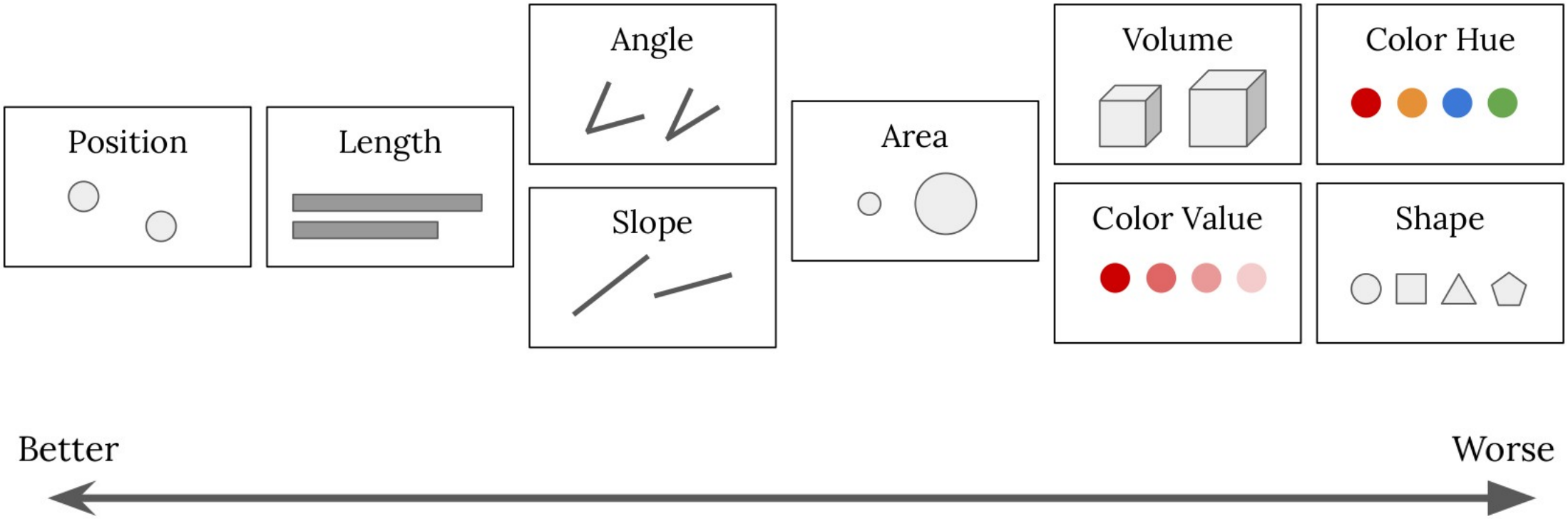
Grammar - data

Make	Model	Year	Engine Displacement	Cylinders	Transmission	Drivetrain	Vehicle Class	Fuel Type	Fuel Barrels/Year
AM General	DJ Po Vehicle 2WD	1984	2.5	4.0	Automatic 3-spd	2-Wheel Drive	Special Purpose Vehicle 2WD	Regular	19.388824
AM General	FJ8c Post Office	1984	4.2	6.0	Automatic 3-spd	2-Wheel Drive	Special Purpose Vehicle 2WD	Regular	25.354615
AM General	Post Office DJ5 2WD	1985	2.5	4.0	Automatic 3-spd	Rear-Wheel Drive	Special Purpose Vehicle 2WD	Regular	20.600625
AM General	Post Office DJ8 2WD	1985	4.2	6.0	Automatic 3-spd	Rear-Wheel Drive	Special Purpose Vehicle 2WD	Regular	25.354615

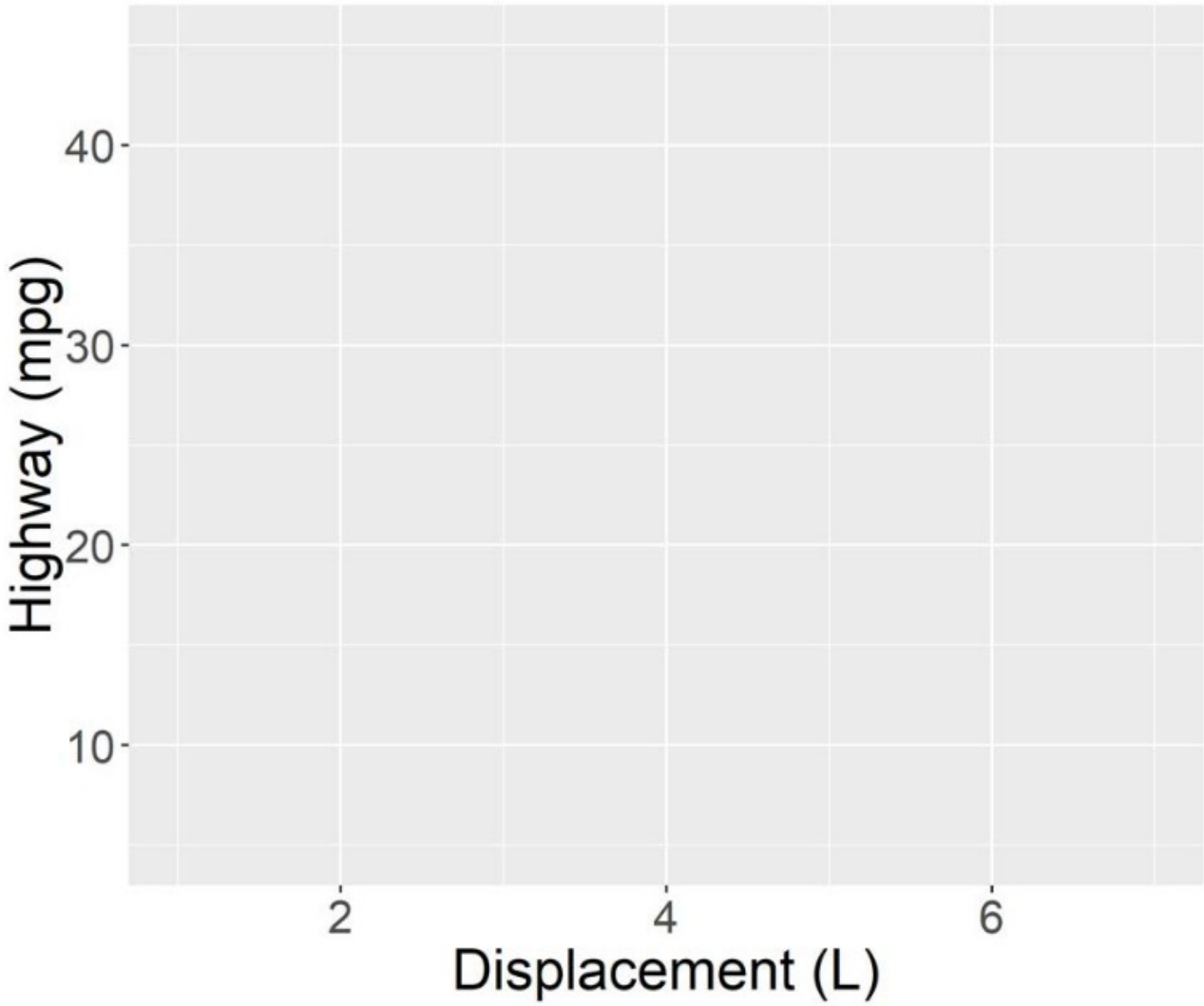
Grammar - mappings



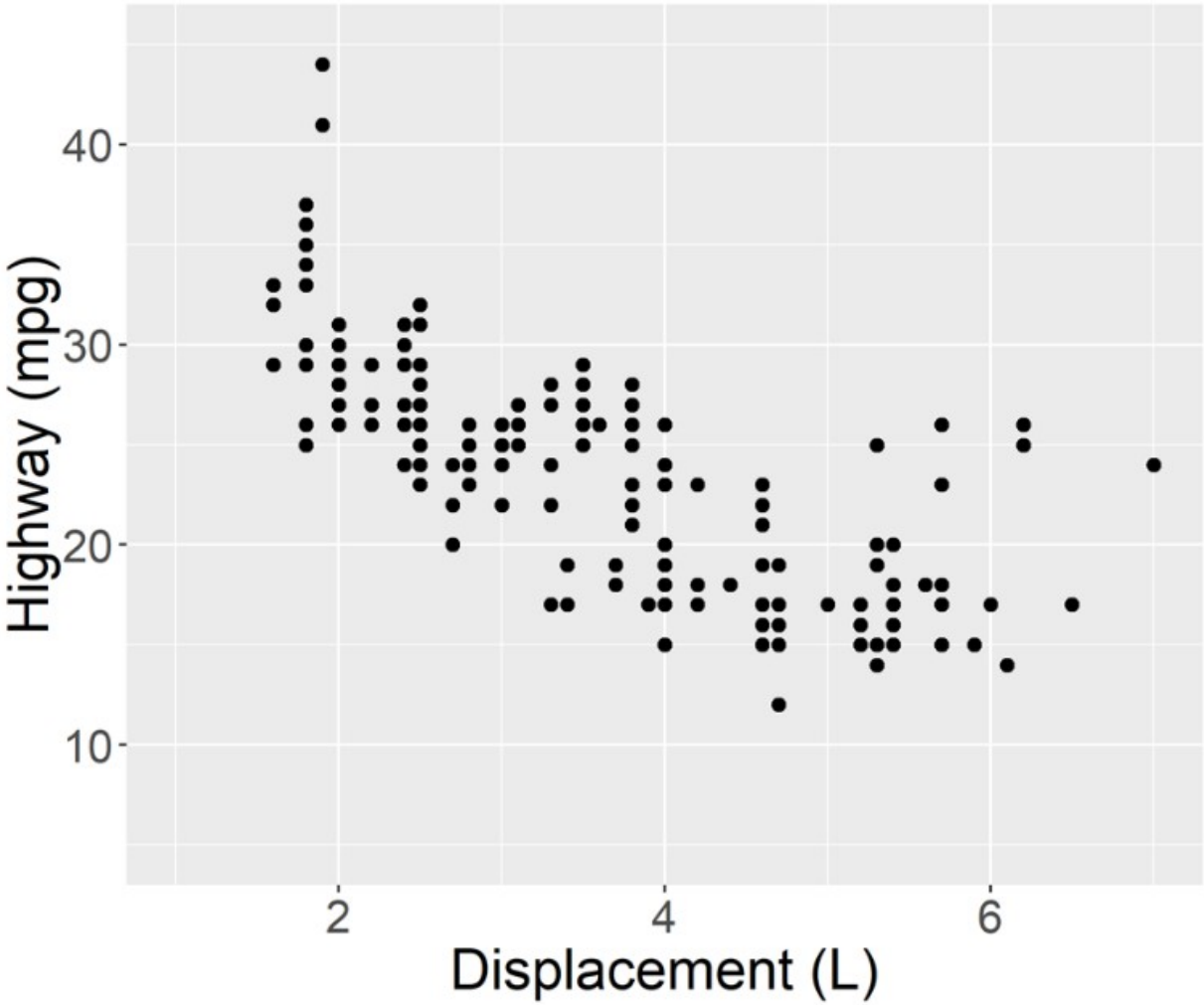
Grammar - mappings

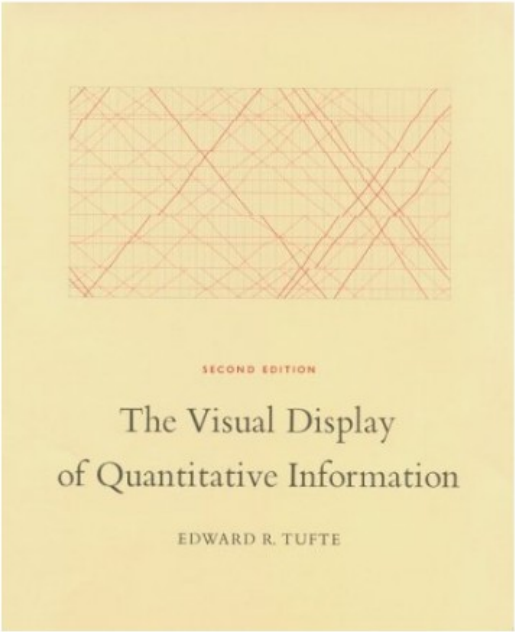


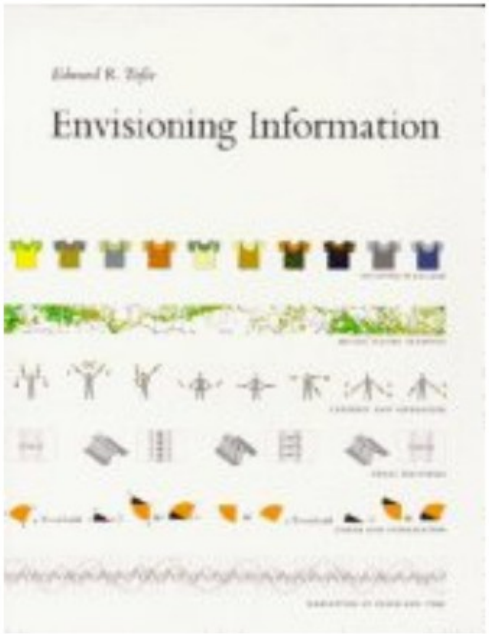
Grammar - scales

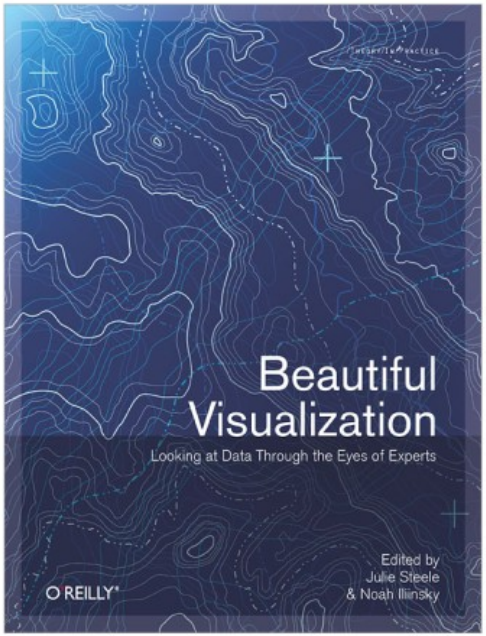


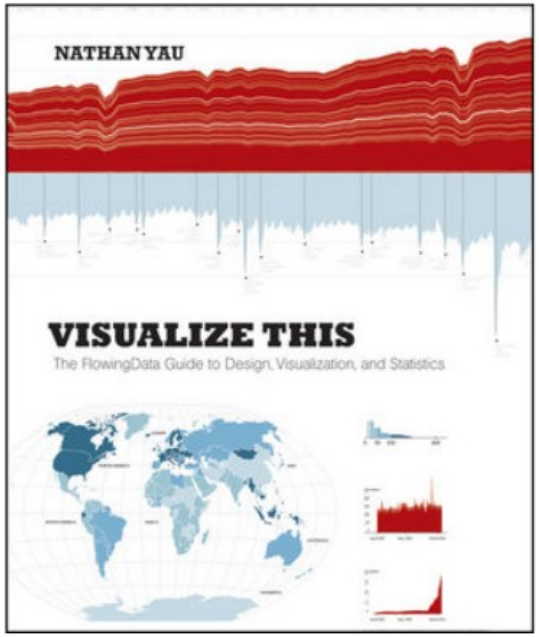
Grammar - marks

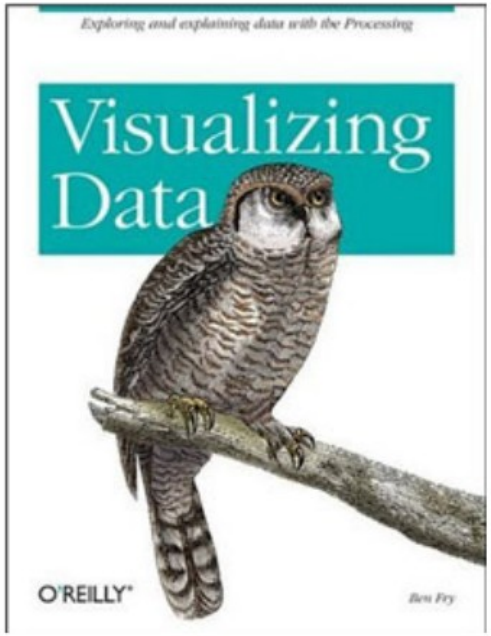


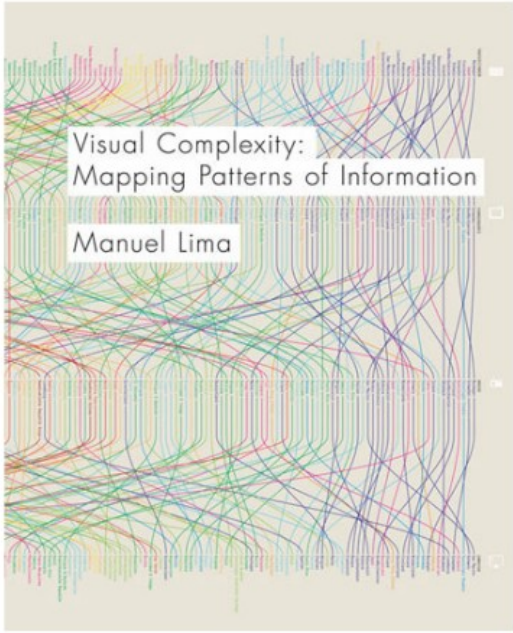

SECOND EDITION
The Visual Display of Quantitative Information
EDWARD R. TUFTE

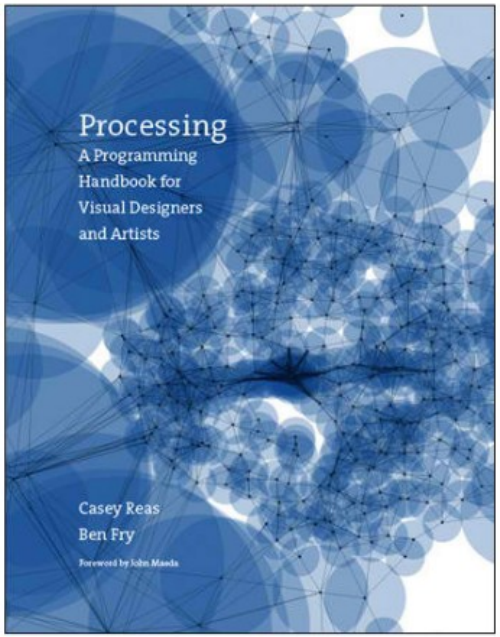

Edward R. Tufte
Envisioning Information

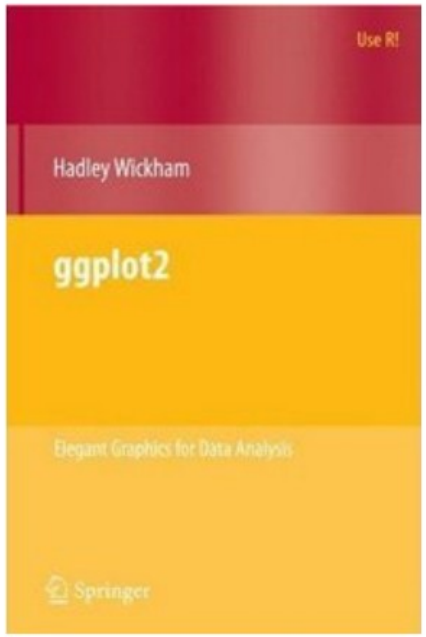

Beautiful Visualization
Looking at Data Through the Eyes of Experts
Edited by Julie Steele & Noah Iliinsky
O'REILLY


NATHAN YAU
VISUALIZE THIS
The FlowingData Guide to Design, Visualization, and Statistics


Exploring and explaining data with the Processing
Visualizing Data
O'REILLY
Ben Fry


Visual Complexity: Mapping Patterns of Information
Manuel Lima


Processing
A Programming Handbook for Visual Designers and Artists
Casey Reas
Ben Fry


Use R!
Hadley Wickham
ggplot2
Elegant Graphics for Data Analysis
Springer

matplotlib

Matplotlib

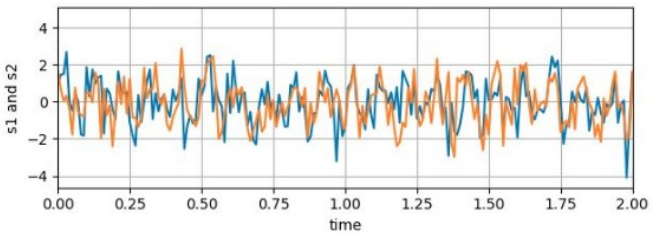
- Matplotlib was created by John D. Hunter in 2003 to visualize electrocorticography data.

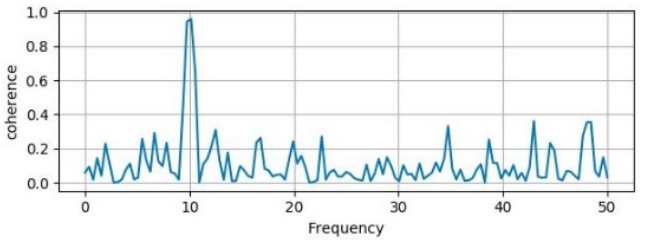
  My goal is to make high quality, publication quality plotting easy in python, with a a syntax familiar to matlab users.

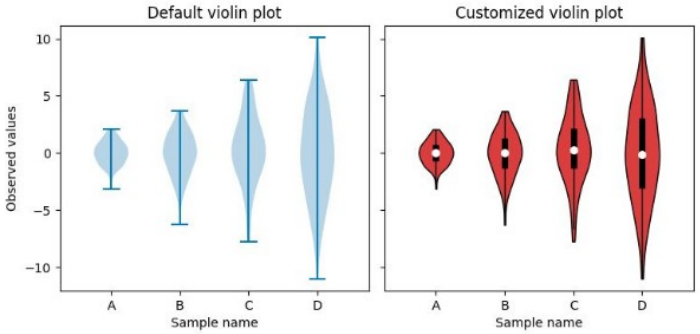
  - *make easy things easy*
  - *make hard things possible*

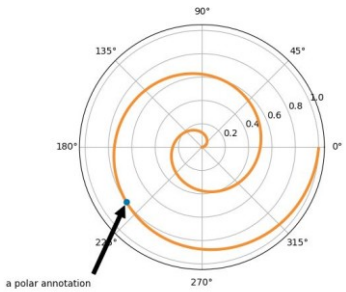
Matplotlib

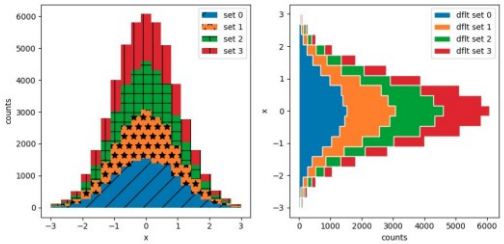
- Can reproduce just about any plot (with a bit of effort)

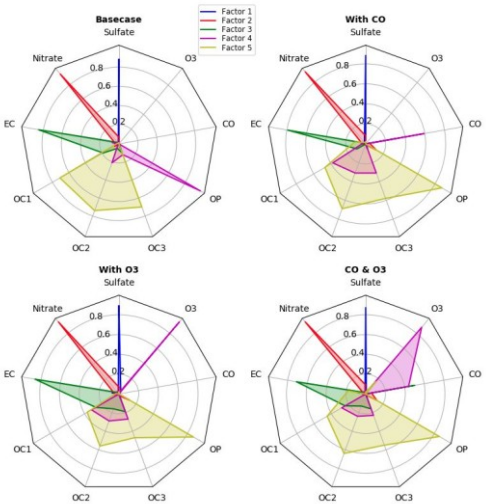


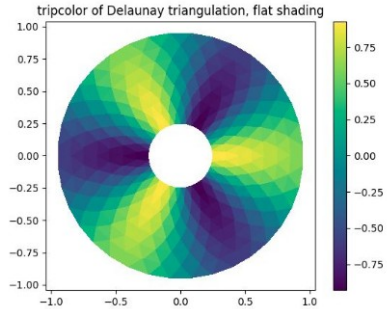


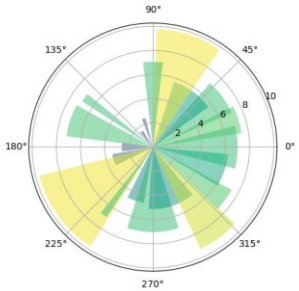


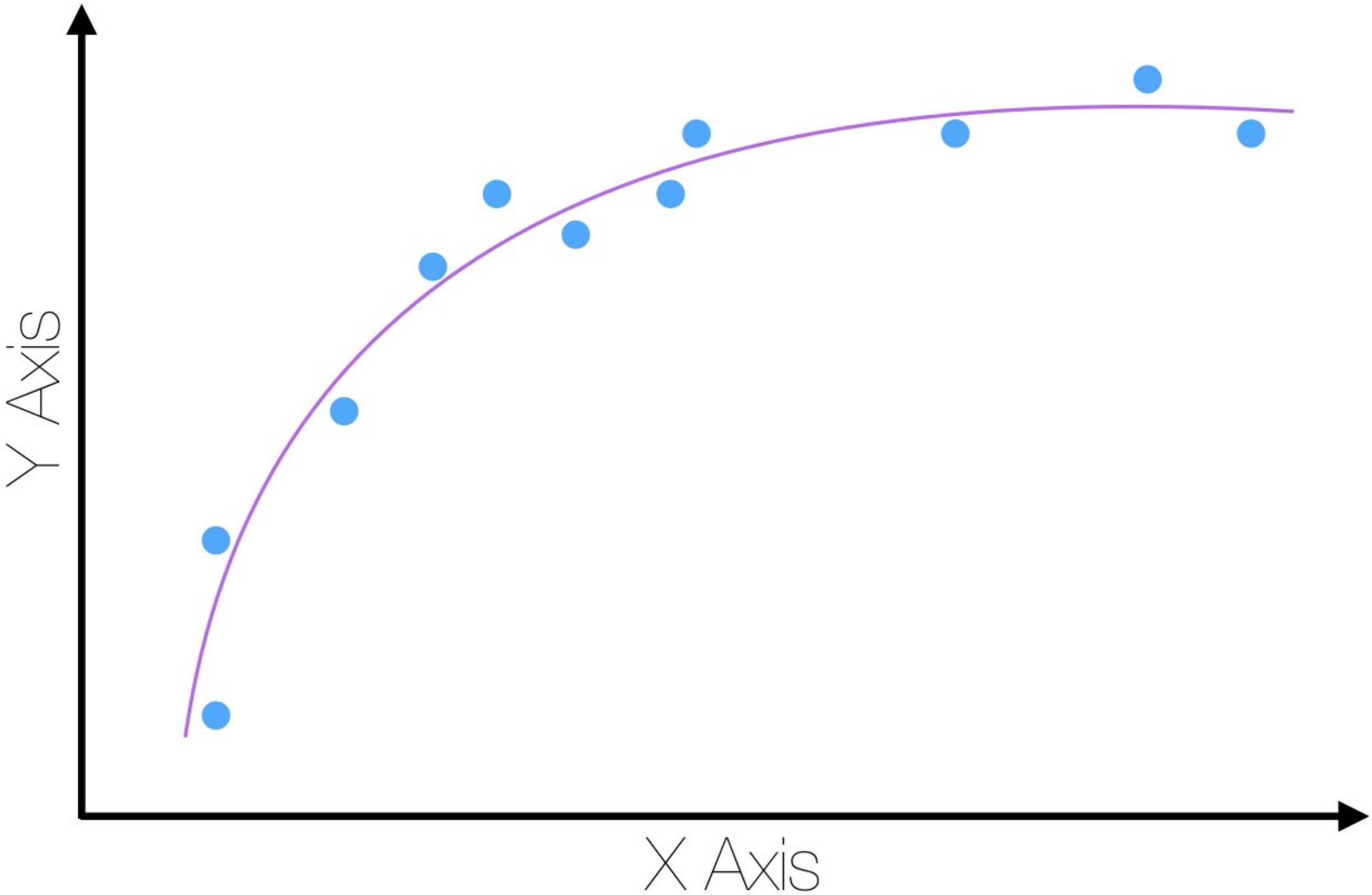


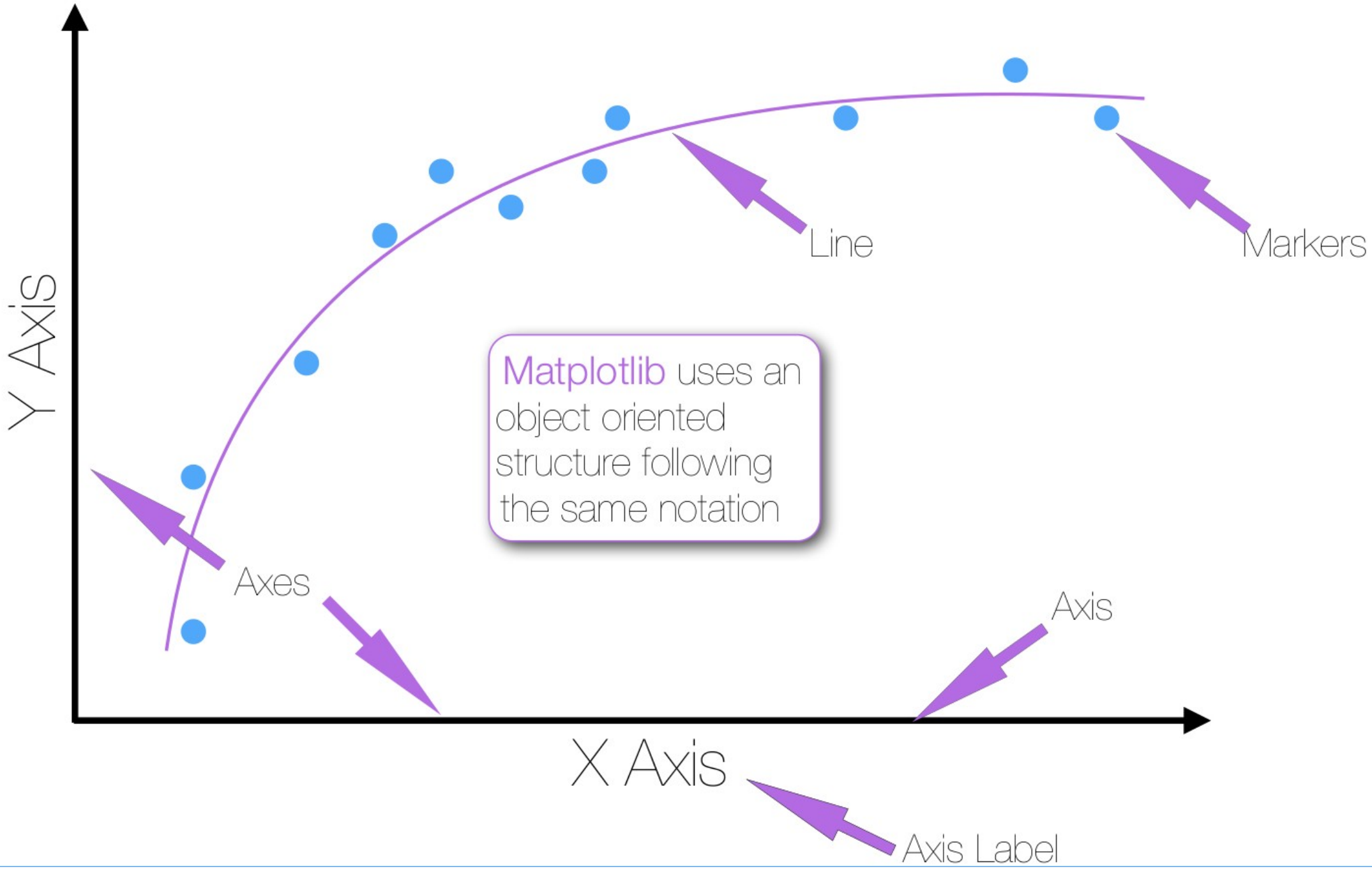









Y Axis
X Axis


Y Axis
Line
Markers
Matplotlib uses an object oriented structure following the same notation
Axes
Axis
X Axis
Axis Label

Matplotlib objects

- Matplotlib uses an object oriented structure following an intuitive notation
- A Figure is a set of one or more Axes.
- An Axes object is a subplot of a Figure.
- Each Axes object contains one or more Axis objects.
- Each Axes is associated with exactly one Figure and each set of Markers is associated with exactly one Axes.
- Markers/Lines represent a dataset that is plotted against one or more Axis.

Figure

While the Figure object controls the way in which the figure is displayed.

- .gca() - Get the current Axes, creating one if necessary
- .show() - Show the final figure
- .savefig("filename.ext", dpi=300) - Save the figure to "filename.ext" where ".ext" defines the format the saved image ()

```
import matplotlib.pyplot as plt
fig = plt.figure()
```

Axes

The convention is to import pyplot as plt

To create subplots (Axes) you use .subplots(nrows, ncols, sharex=False, sharey=False) instead of .figure(). set sharex and/or sharey to True to keep the same scale in both cases.

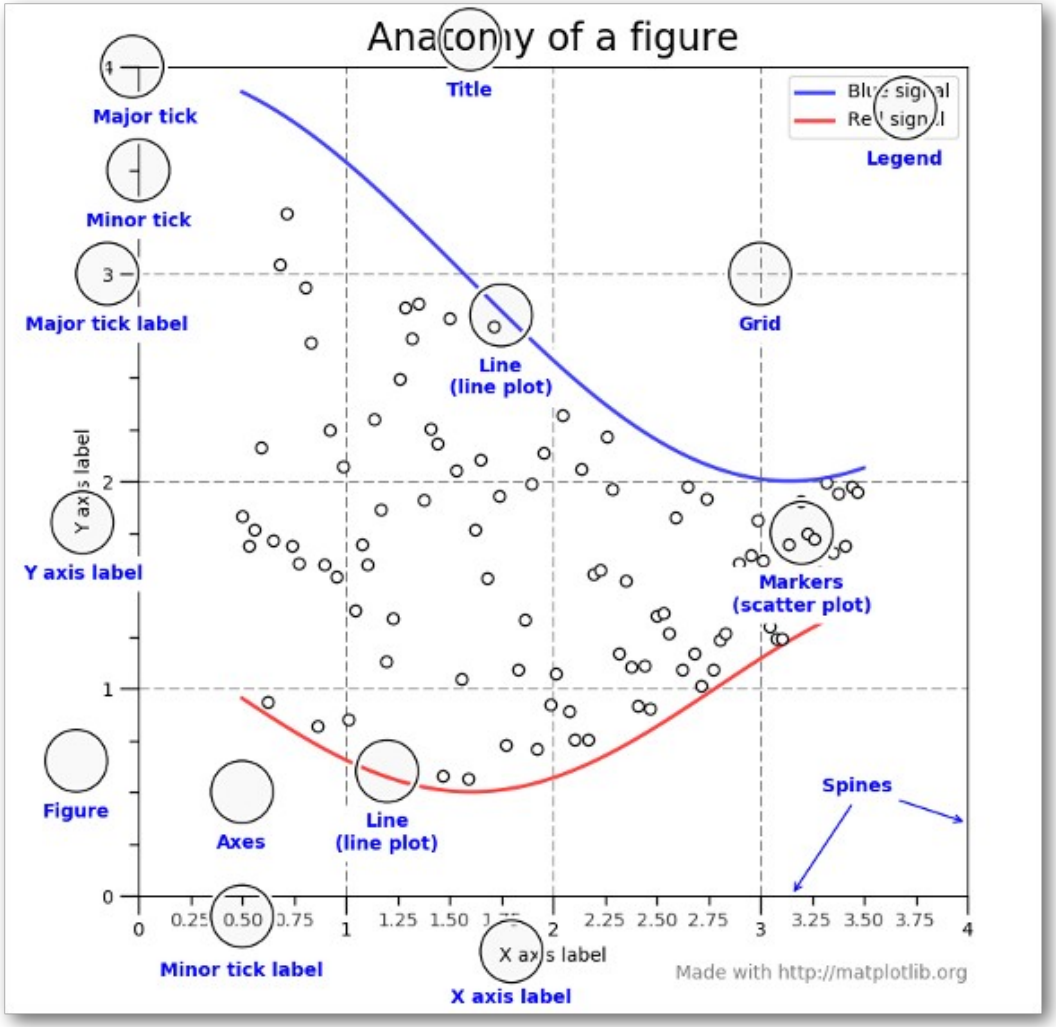
.subplots - returns a (fig, ax_lst) tuple where ax_lst is a list of Axes and fig is the Figure.

Axes

Axes have several methods of interest:

- .plot(x, y) - Make a scatter or line plot from a list of x, y coordinates.
- .imshow(mat) - Plot a matrix as if it were an image. Element 0,0 is plotted in the top right corner.
- .bar(x, y) - Make a bar plot where x is a list of the lower left coordinates of each bar and y is the respective height.
- .pie(values, labels=labels) - Produce a pie plot out of a list of values list and labeled with labels
- .savefig(filename) - Write the current figure as an static image

Guides



The respective functions are named in an intuitive way, Every Axes object has as methods:

- .set_xlabel(label)
- .set_ylabel(label)
- .set_title(title)

And axis limits can be set using:

- .set_xlim(xmin, xmax)
- .set_ylim(ymin, ymax)

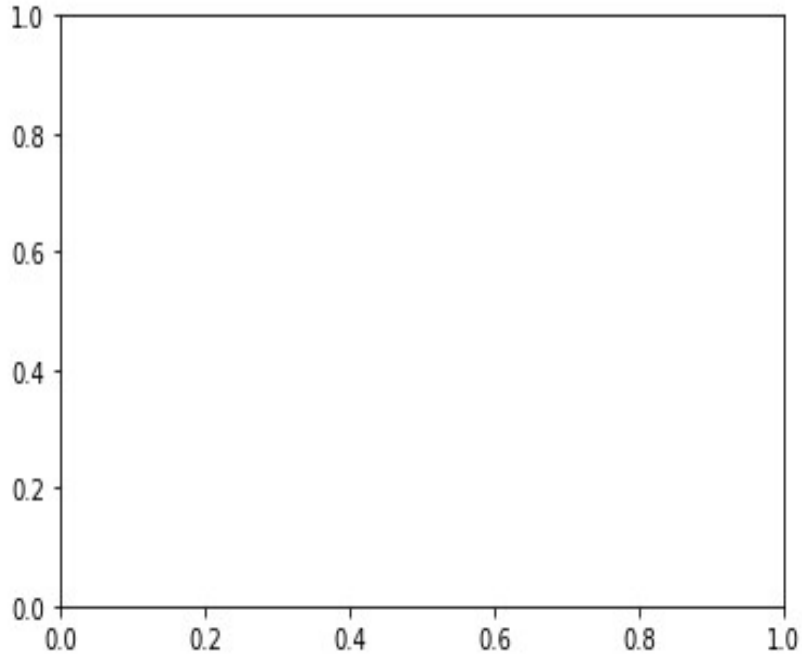
Tick marks and labels are set using:

- .set_xticks(ticks)/.set_yticks(ticks)
- .set_xticklabels(labels)/.set_yticklabels(labels)

pyplot interface

In [1]:
```
import matplotlib.pyplot as plt
%matplotlib inline

fig, ax = plt.subplots()
plt.show()
```

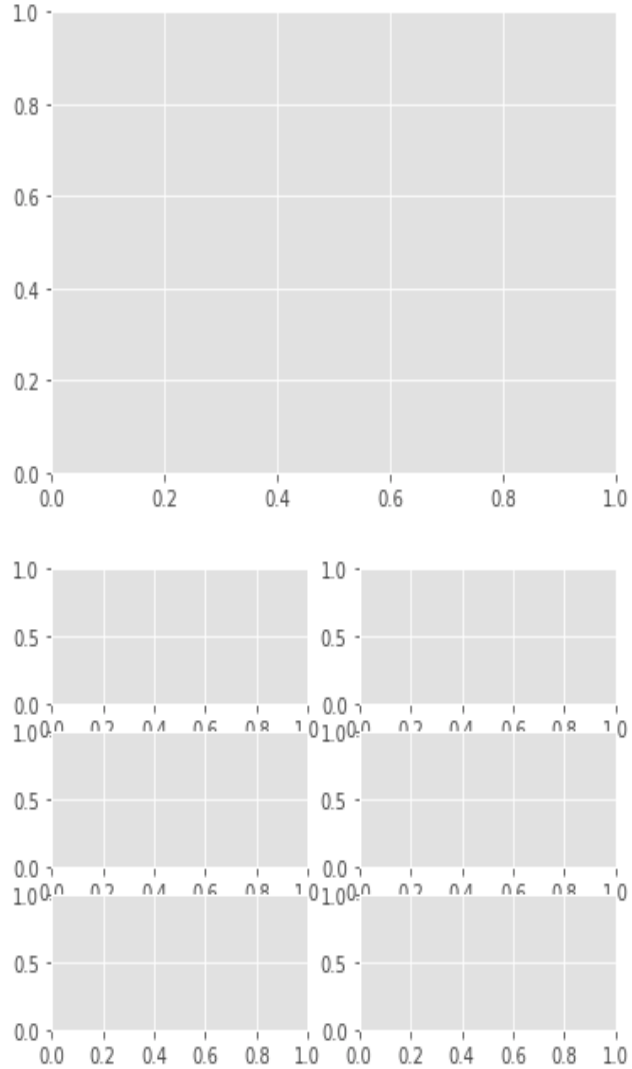
executed in 612ms, finished 10:36:18 2019-07-25



Subplots

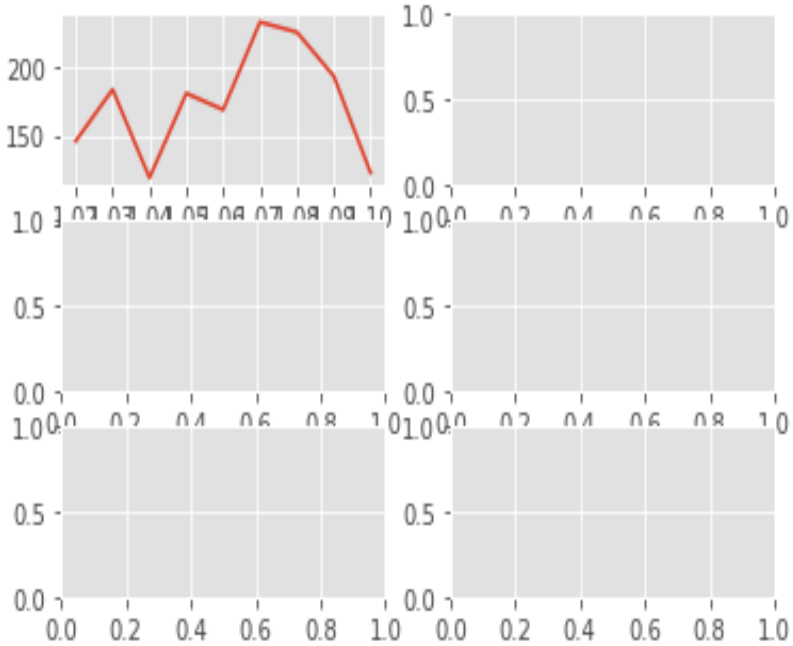
```
In [31]: fig, ax = plt.subplots()
         fig, ax = plt.subplots(3, 2)
         plt.show()
```

executed in 586ms, finished 11:28:41 2019-07-25



```
In [36]: fig, ax = plt.subplots(3, 2)

         ax.shape

         ax[0,0].plot(shampoo['Month'],shampoo['Sales'])
         plt.show()
```

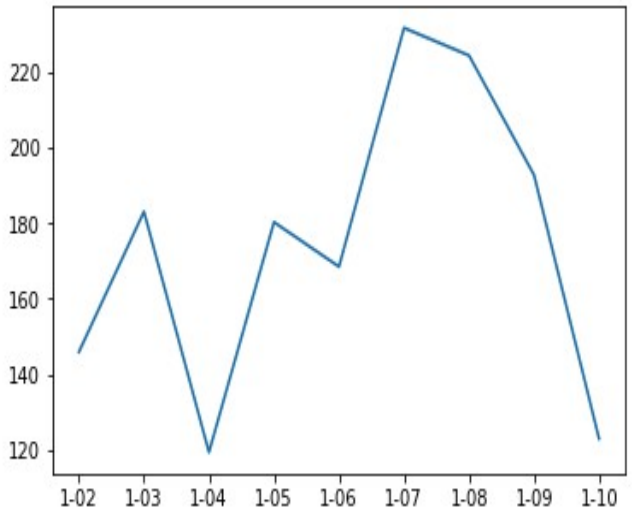
executed in 476ms, finished 11:30:38 2019-07-25



Adding data

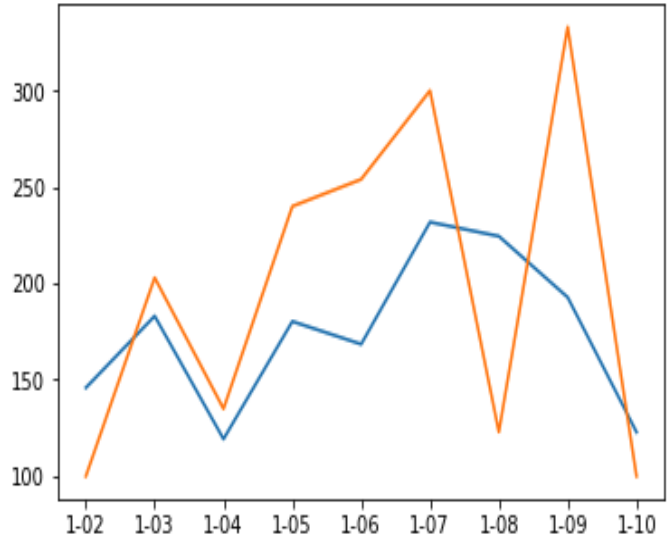
```
In [33]: %matplotlib inline

         fig, ax = plt.subplots()
         ax.plot(shampoo['Month'],shampoo['Sales'])
         plt.show()
```

executed in 119ms, finished 10:11:19 2019-07-25




```
In [36]: fig, ax = plt.subplots()
         ax.plot(shampoo['Month'],shampoo['Sales'])
         ax.plot(shampoo['Month'],shampoo['Sales2'])
         plt.show()
```

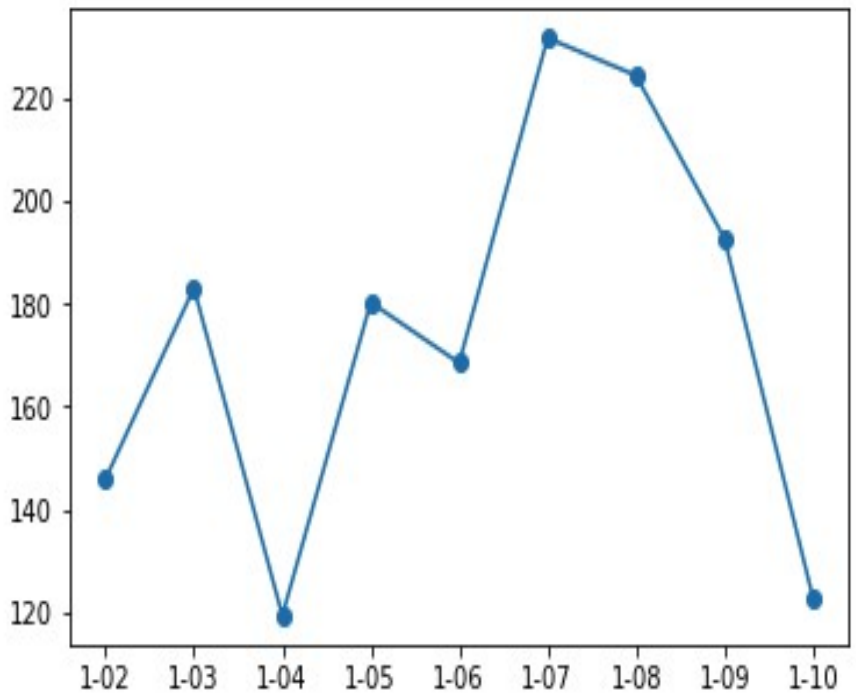
executed in 122ms, finished 10:26:40 2019-07-25



Customizing plots - markers

```
In [9]: fig, ax = plt.subplots()
        ax.plot(shampoo['Month'],shampoo['Sales'], marker = 'o')
        plt.show()
```

executed in 115ms, finished 10:41:41 2019-07-25



matplotlib.markers

This module contains functions to handle markers. Used by both the marker functionality of `plot`

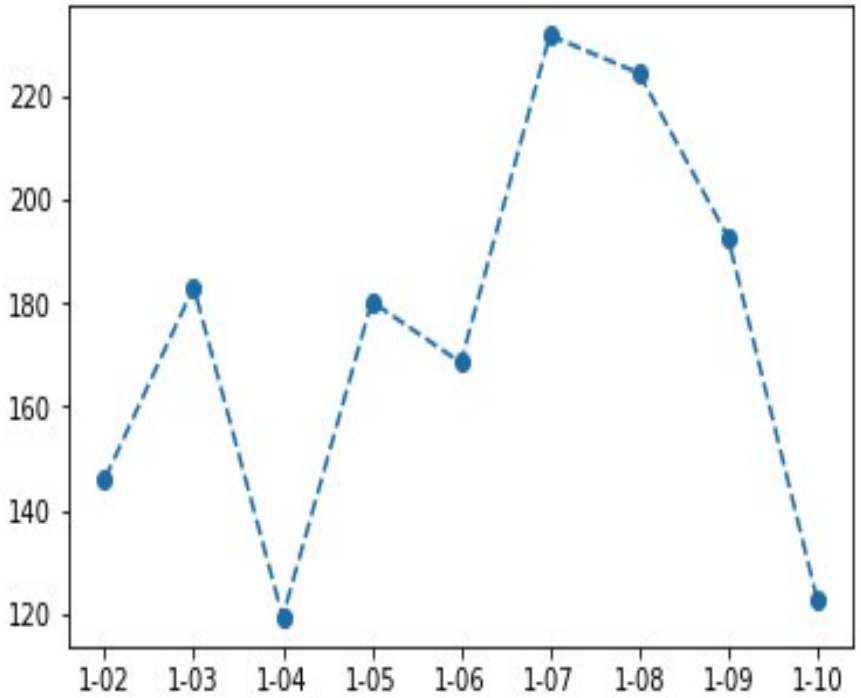
All possible markers are defined here:

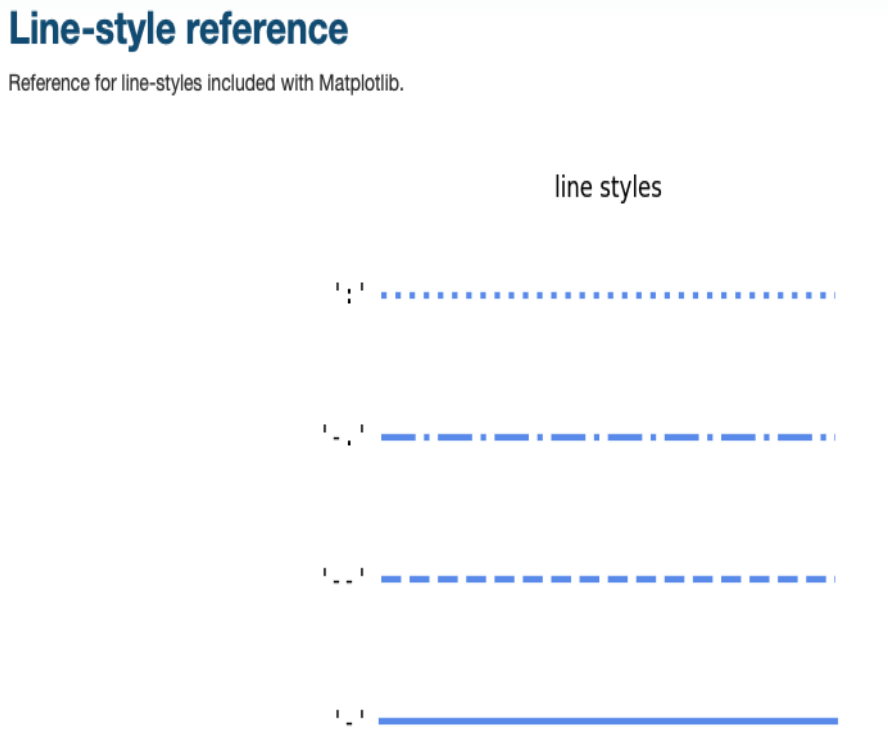
marker	symbol	description
`"."`	•	point
`","`	·	pixel
`"o"`	●	circle
`"v"`	▼	triangle_down
`"^"`	▲	triangle_up
`"<"`	◀	triangle_left
`">"`	▶	triangle_right
`"1"`	⊤	tri_down
`"2"`	⊥	tri_up
`"3"`	≺	tri_left
`"4"`	≻	tri_right
`"8"`	●	octagon
`"s"`	■	square
`"p"`	⬟	pentagon
`"P"`	✚	plus (filled)

Customizing plots - linestyle

In [10]:
```
fig, ax = plt.subplots()
ax.plot(shampoo['Month'],shampoo['Sales'], marker = 'o', linestyle = '--')
plt.show()
```

executed in 121ms, finished 10:42:56 2019-07-25

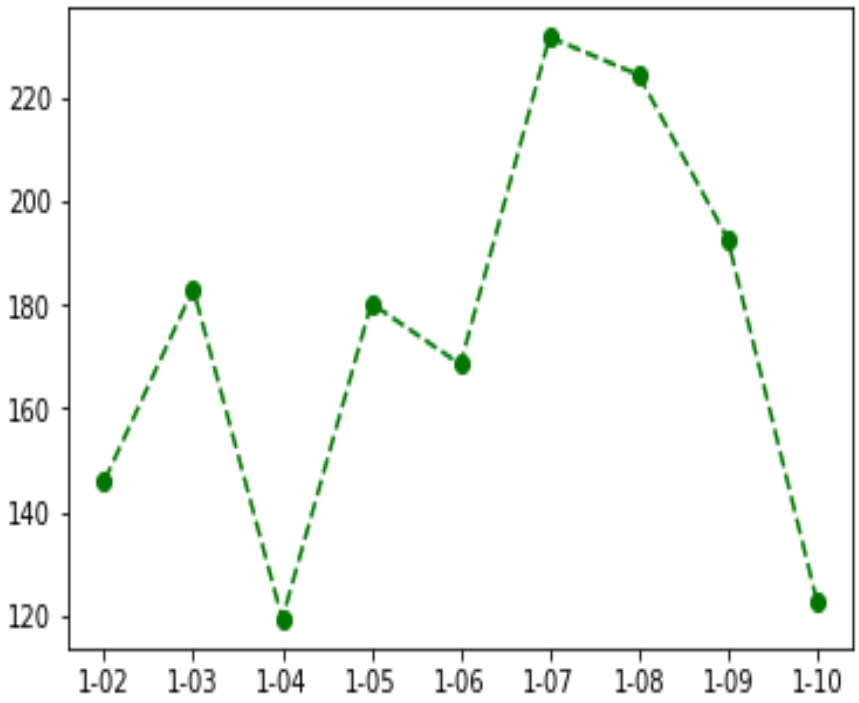




Customizing plots - color

In [11]:
```
fig, ax = plt.subplots()
ax.plot(shampoo['Month'],shampoo['Sales'],
        marker = 'o', linestyle = '--',color = 'g')
plt.show()
```

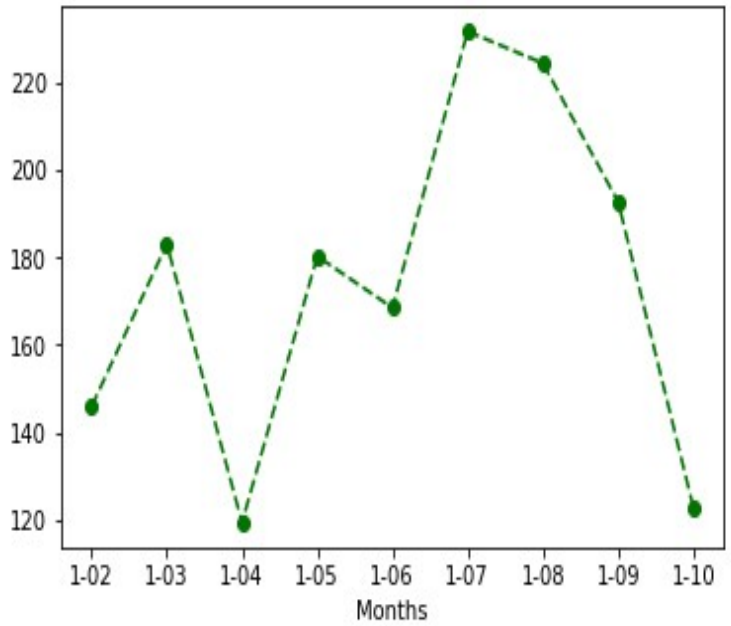
executed in 195ms, finished 10:44:16 2019-07-25



Customizing plots - axes labels

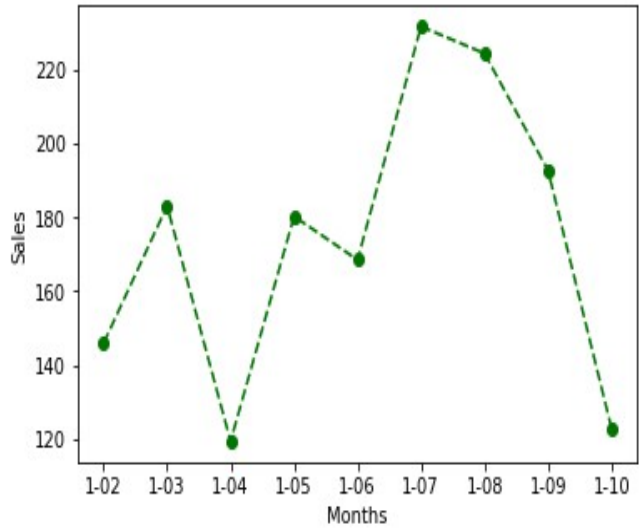
In [12]:
```
fig, ax = plt.subplots()
ax.plot(shampoo['Month'],shampoo['Sales'],
        marker = 'o', linestyle = '--',color = 'g')
ax.set_xlabel("Months")
plt.show()
```

executed in 121ms, finished 10:45:16 2019-07-25



In [13]:
```
fig, ax = plt.subplots()
ax.plot(shampoo['Month'],shampoo['Sales'],
        marker = 'o', linestyle = '--',color = 'g')
ax.set_xlabel("Months")
ax.set_ylabel("Sales")
plt.show()
```

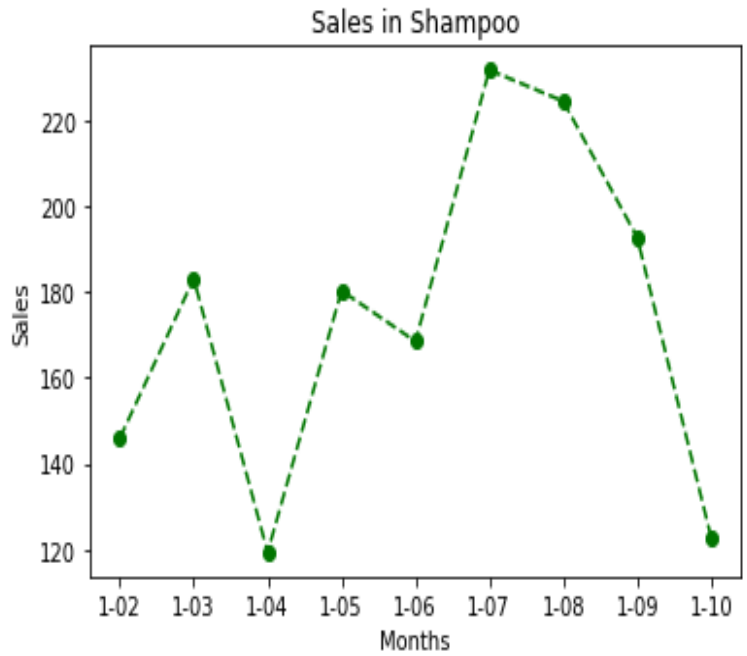
executed in 123ms, finished 10:48:52 2019-07-25



Customizing plots - title

```
In [14]: fig, ax = plt.subplots()
         ax.plot(shampoo['Month'],shampoo['Sales'],
                 marker = 'o', linestyle = '--',color = 'g')
         ax.set_xlabel("Months")
         ax.set_ylabel("Sales")
         ax.set_title('Sales in Shampoo')
         plt.show()
```

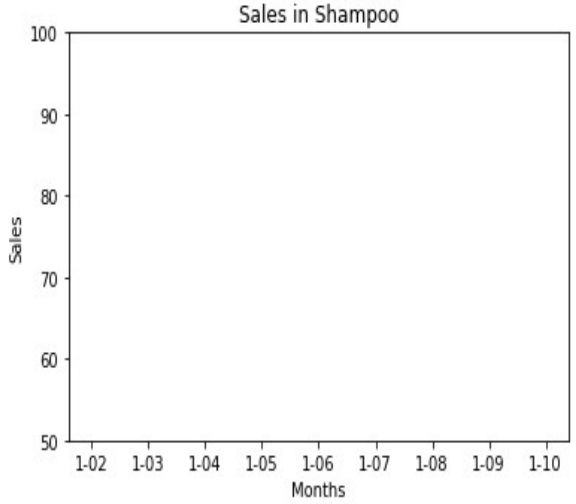
executed in 129ms, finished 10:49:46 2019-07-25



Customizing plots - axis extents

- axis([xmin, xmax, ymin, ymax]) sets axis extents
- Control over individual axis extents
  - xlim([xmin, xmax])
  - ylim([ymin, ymax])
- Can use tuples, lists for extents
  - e.g., xlim((-2, 3)) works
  - e.g., xlim([-2, 3]) works also

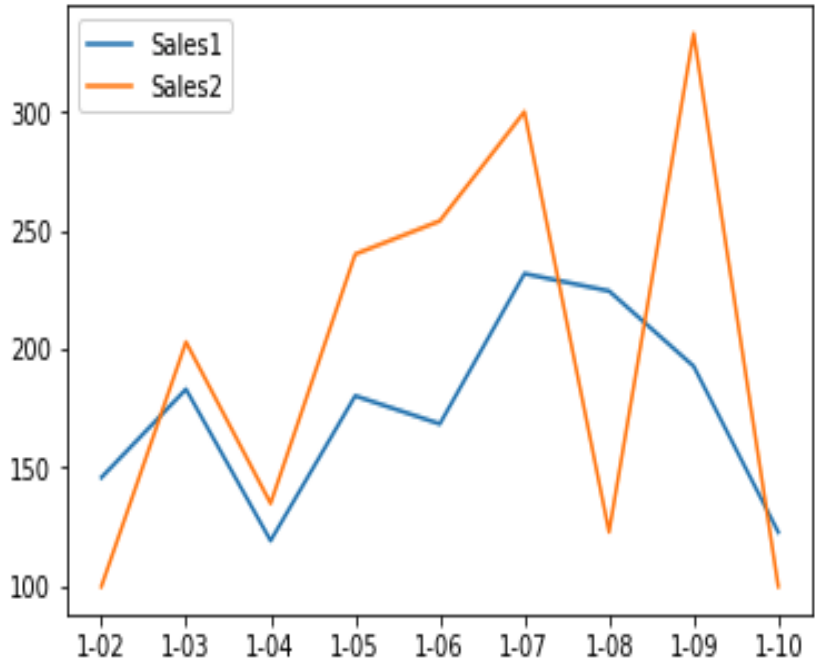
```
In [16]: fig, ax = plt.subplots()
         ax.plot(shampoo['Month'],shampoo['Sales'],
                 marker = 'o', linestyle = '--',color = 'g')
         ax.set_xlabel("Months")
         ax.set_ylabel("Sales")
         plt.ylim((50,100))
         ax.set_title('Sales in Shampoo')

         executed in 138ms, finished 11:05:16 2019-07-25

Out[16]: Text(0.5, 1.0, 'Sales in Shampoo')
```



Customizing plots - legend

In [18]:
```
fig, ax = plt.subplots()
ax.plot(shampoo['Month'],shampoo['Sales'],label = 'Sales1')
ax.plot(shampoo['Month'],shampoo['Sales2'],label = 'Sales2'
plt.legend()
plt.show()
```

executed in 171ms, finished 11:06:36 2019-07-25



Legend locations

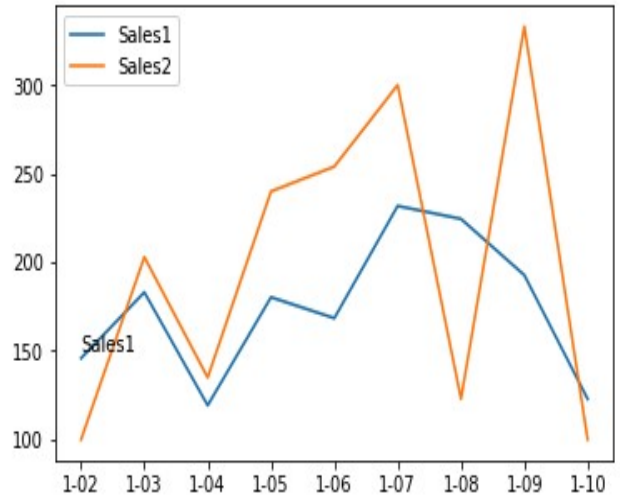
string	code	string	code	string	code
'upper left'	2	'upper center'	9	'upper right'	1
'center left'	6	'center'	10	'center right'	7
'lower left'	3	'lower center'	8	'lower right'	4
'best'	0			'right'	5

Customizing plots - annotations

- Text labels and arrows using annotate() method
- Flexible specification of coordinates
- Keyword arrowprops: dict of arrow properties
  - width
  - color
  - etc.

```
In [21]: fig, ax = plt.subplots()
         ax.plot(shampoo['Month'],shampoo['Sales'],label = 'Sales1')
         ax.plot(shampoo['Month'],shampoo['Sales2'],label = 'Sales2')
         plt.legend()
         plt.annotate('Sales1',xy=('1-02',150))
         plt.show()
```

executed in 142ms, finished 11:10:09 2019-07-25



Customizing plots - style

```
In [22]: plt.style.use("ggplot")

         fig, ax = plt.subplots()

         ax.plot(shampoo['Month'],shampoo['Sales'],label = 'Sales1')
         ax.plot(shampoo['Month'],shampoo['Sales2'],label = 'Sales2')
         plt.legend()
         plt.annotate('Sales1',xy=('1-02',150))
         plt.show()
```

executed in 426ms, finished 11:17:47 2019-07-25

- stylesheets allow us to define our own figure defaults that will apply to every figure from then on.
- matplotlib comes with several pre-defined styles
- plt.style.available returns a list of all available styles
- plt.style.use('*style*')

Customizing plots - size

In [29]:
```
fig, ax = plt.subplots()
fig.set_size_inches([30, 14])

ax.plot(shampoo['Month'],shampoo['Sales'],label = 'Sales1')
ax.plot(shampoo['Month'],shampoo['Sales2'],label = 'Sales2')
plt.legend()
plt.annotate('Sales1',xy=('1-02',150))
plt.show()
```

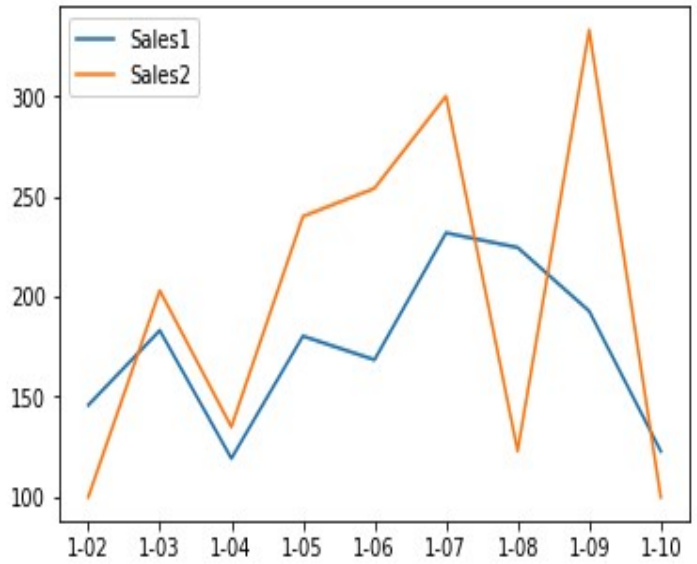
executed in 340ms, finished 11:23:46 2019-07-25

Lines

Use: Compare the relationship between numerical fields

```
In [18]: fig, ax = plt.subplots()
         ax.plot(shampoo['Month'],shampoo['Sales'],label = 'Sales1')
         ax.plot(shampoo['Month'],shampoo['Sales2'],label = 'Sales2')
         plt.legend()
         plt.show()
```

executed in 171ms, finished 11:06:36 2019-07-25

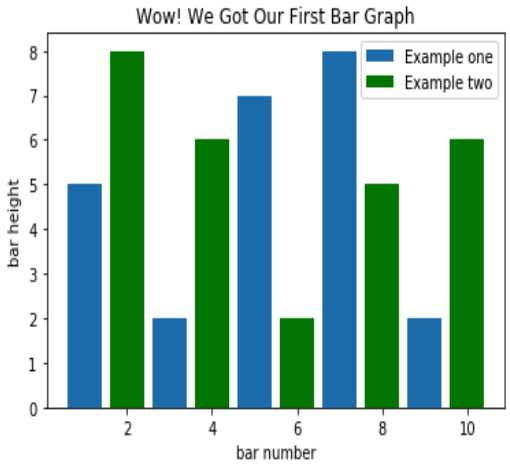


Bars

Use: When plotting categorical data, there is value to plotting two or more groups side by side and being able to compare them

```
In [5]: plt.bar([1,3,5,7,9],[5,2,7,8,2], label='Example one')
        plt.bar([2,4,6,8,10],[8,6,2,5,6], label='Example two', color='g')

        plt.legend()
        plt.xlabel('bar number')
        plt.ylabel('bar height')
        plt.title('Wow! We Got Our First Bar Graph')
        plt.show()
```

executed in 171ms, finished 10:01:40 2019-07-25




```
In [58]: plt.barh([1,3,5,7,9],[5,2,7,8,2], label='Example one')
         plt.barh([2,4,6,8,10],[8,6,2,5,6], label='Example two', color='g')

         plt.legend()
         plt.xlabel('bar number')
         plt.ylabel('bar height')
         plt.title('Wow! We Got Our First Bar Graph')
         plt.show()
```

executed in 154ms, finished 12:38:44 2019-07-25

Saving figure

```
In [ ]: plt.style.use("ggplot")

        fig, ax = plt.subplots()

        ax.plot(shampoo['Month'],shampoo['Sales'],label = 'Sales1')
        ax.plot(shampoo['Month'],shampoo['Sales2'],label = 'Sales2')
        plt.legend()
        plt.annotate('Sales1',xy=('1-02',150))
        plt.show()

        fig.savefig("sales.png")
```

```
fig.savefig("sales.png")
fig.savefig("sales.png",quality= 20)
fig.savefig("sales.svg")
```

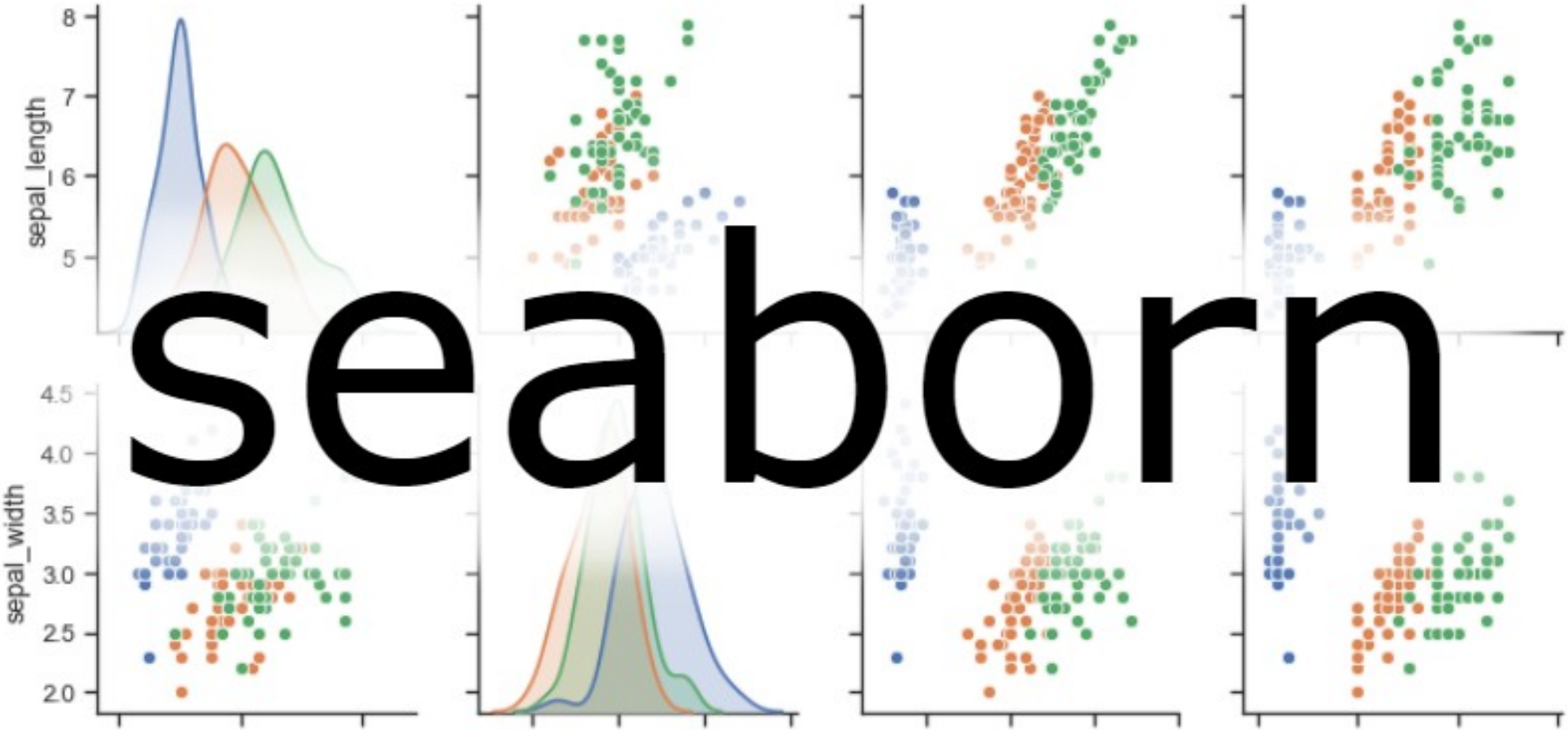
Loading images

- plt.imshow(img) - Display an image on a set of axes.
- plt.imshow(fig, extent=(xllcorner, xurcorner, yllcorner, yurcorner), zorder=-1)
- img can be any matrix of numbers

Matplotlib

- Weaknesses
  - API is imperative & often overly verbose
  - Sometimes poor stylistic defaults
  - Poor support for web/interactive graphs
  - Often slow for large & complicated data

seaborn



Seaborn

- Seaborn is a data visualization library that was created to complement matplotlib and it is closely integrated with pandas data structures. Some of the advantages of this library are:
  - Streamlines the amount of code needed to write to create matplotlib visualizations
  - Enables us to work with pandas dataframes which is a big advantage over matplotlib
  - Provides specialized support for using categorical variables
  - Has functions to examine relationships between multiple variables
  - Provides convenient views onto the overall structure of complex datasets

Seaborn

```
sns.get_dataset_names()
sns.load_dataset(name)
```

- anscombe.csv
- attention.csv
- brain_networks.csv
- car_crashes.csv
- diamonds.csv
- dots.csv
- exercise.csv
- flights.csv
- fmri.csv
- gammas.csv
- iris.csv
- mpg.csv
- planets.csv
- tips.csv
- titanic.csv

Seaborn

- sns.scatterplot()
- sns.lineplot()

Relationship plots

- sns.stripplot()
- sns.swarmplot()
- sns.boxplot()
- sns.violinplot()
- sns.boxenplot()
- sns.pointplot()
- sns.barplot()
- sns.countplot()

Categorical plots

Seaborn

Axes level

- sns.scatterplot()
- sns.lineplot()

- sns.stripplot()
- sns.swarmplot()
- sns.boxplot()
- sns.violinplot()
- sns.boxenplot()
- sns.pointplot()
- sns.barplot()
- sns.countplot()

Figure level

Relationship plots

sns.relplot()

- kind="scatter"
- kind="line"

Figure level functions are equivalent to the axes level ones, (with the right kind setting) but more adapted for ease of exploration

Categorical plots

sns.catplot()

- kind="strip"
- kind="swarm"
- kind="box"
- kind="violin"
- kind="boxen"
- kind="point"
- kind="bar"
- kind="count"

For convenience, we will focus on figure level functions

Seaborn

The basic plotting functions are designed to work directly with pandas data frames

Every plotting function takes a data parameter that is used to pass the correct data frame to the function

There is no limitation on the number of columns that the dataframe can contain.

The specific columns to be plotted are passed by setting other parameters to the respective column names. In particular:

- x - column to plot along the x-axis
- y - column to plot along the y axis

Plot properties such as hue, size, style, etc can also be set using column names

In the case of figure level functions, the kind parameter determines what specific type of plot is produced.

Seaborn

- sns.scatterplot()
- sns.lineplot()

- sns.stripplot()
- sns.swarmplot()
- sns.boxplot()
- sns.violinplot()
- sns.boxenplot()
- sns.pointplot()
- sns.barplot()
- sns.countplot()

Relationship plots

x, y - column names to plot in each axes

hue - column name specifying how to color each data element

size - column name to use to determine the size of each element

style - column name to use to determine the style of each element

data - dataframe containing the data to plot

Categorical plots

orient - specifies the orientation of the plot ('v' or 'h' for vertical or horizontal)

order - specifies the order in which the categorical variable (x or y) is plotted

*_order - family of parameters that determiner the order in which the respective categorical value (hue_order, style_order, size_order, etc…) is plotted

Subplots

One of the main advantages of figure level functions is their ability to easily generate subplots by just passing a couple of extra parameters:

- row, col - specifies the column names to be used to split the dataset and plot along rows and columns
- col_wrap - width at which to wrap the column. Incompatible with a row setting.
- row_order, col_order - work similarly to the other *_order parameters

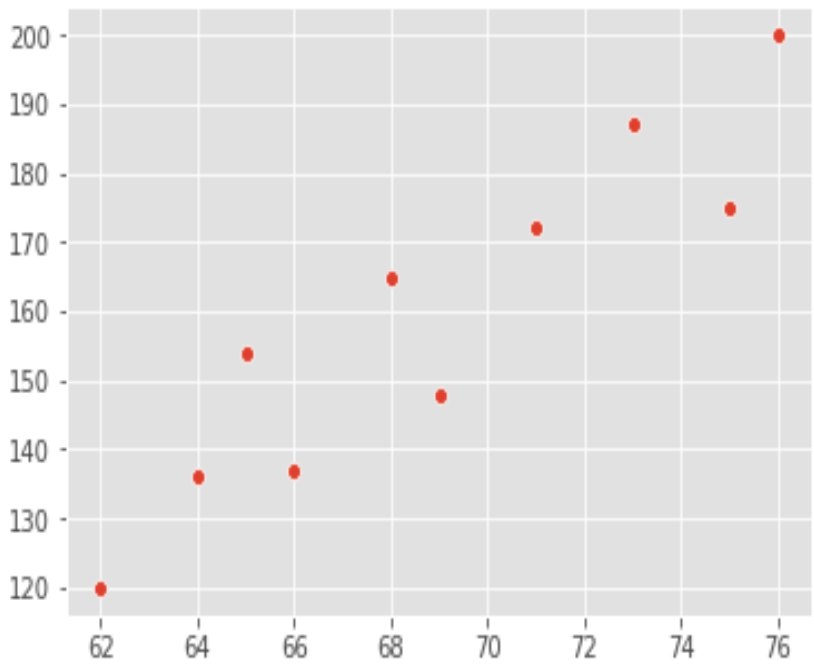
axes level functions return a matplotlib Axis object

figure level functions return a seaborn FacetGrid object

Basic charts

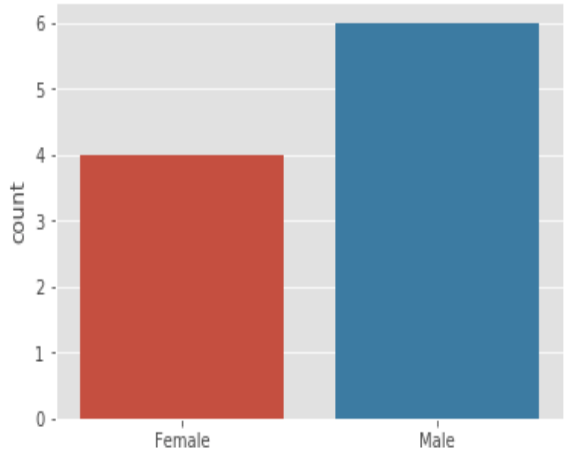
```
In [40]: height = [62, 64, 69, 75, 66, 68, 65, 71, 76, 73]
         weight = [120, 136, 148, 175, 137, 165, 154, 172, 200, 187]

         sns.scatterplot(x=height, y=weight)
         plt.show()
```

executed in 150ms, finished 11:43:41 2019-07-25




```
In [42]: gender = ["Female", "Female", "Female", "Female",
         "Male", "Male", "Male", "Male", "Male", "Male"]
         sns.countplot(x=gender)
         plt.show()
```

executed in 103ms, finished 11:43:55 2019-07-25



Basic charts with pandas

```
In [39]: tips = sns.load_dataset("tips")
         tips.head()
```

executed in 756ms, finished 11:37:14 2019-07-25

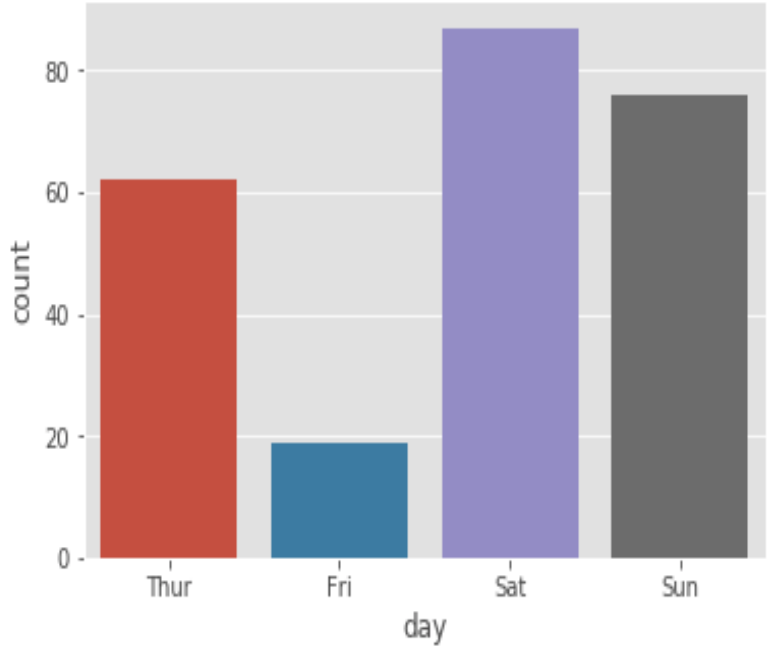
Out[39]:

	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

click to scroll output; double click to hide

```
In [44]: sns.countplot(x="day",data=tips)
         plt.show()
```

executed in 110ms, finished 11:45:06 2019-07-25

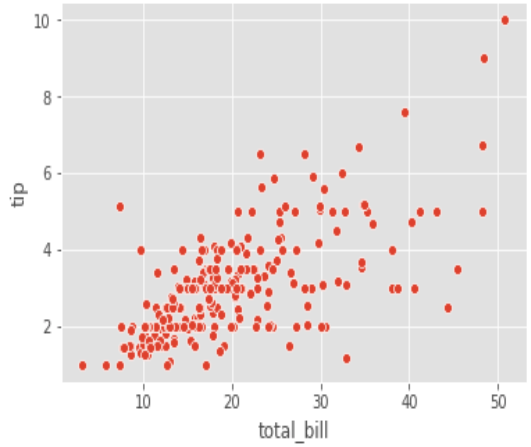


Basic charts with pandas and hue

```
In [46]: sns.scatterplot(x="total_bill", y="tip",data=tips)
```

executed in 154ms, finished 11:47:31 2019-07-25

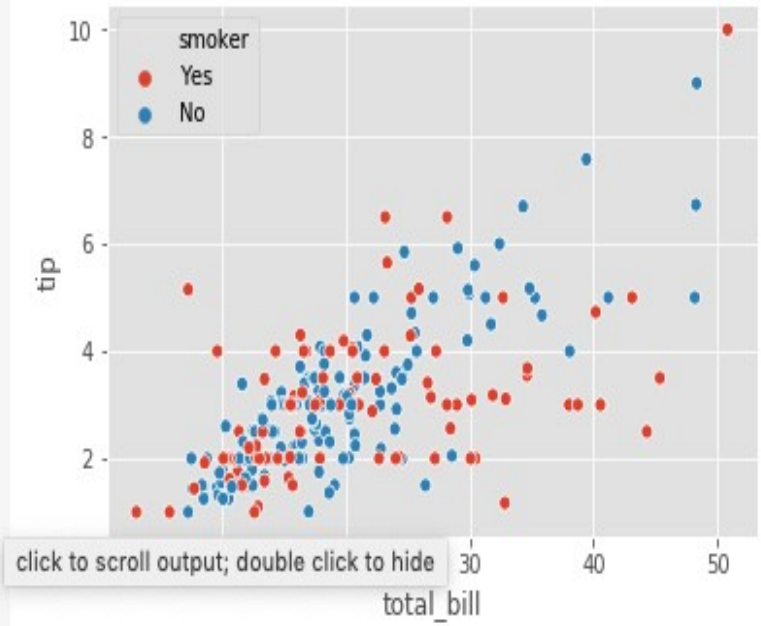
```
Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x1a23b52748>
```




```
In [47]: sns.scatterplot(x="total_bill", y="tip",data=tips, hue='smoker')
```

executed in 194ms, finished 11:47:56 2019-07-25

```
Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x1a238b16d8>
```



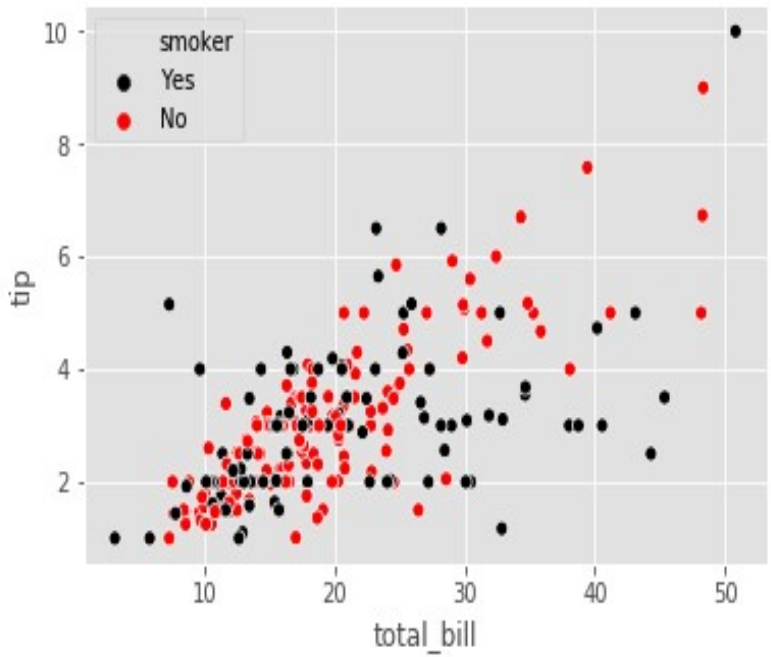
click to scroll output; double click to hide

Basic charts with pandas and hue

```
In [48]: hue_colors = {"Yes": "black", "No": "red"}

         sns.scatterplot(x="total_bill", y="tip",data=tips, hue='smoker', palette=hue_colors)
```

executed in 198ms, finished 11:49:44 2019-07-25

```
Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x1a23905cf8>
```

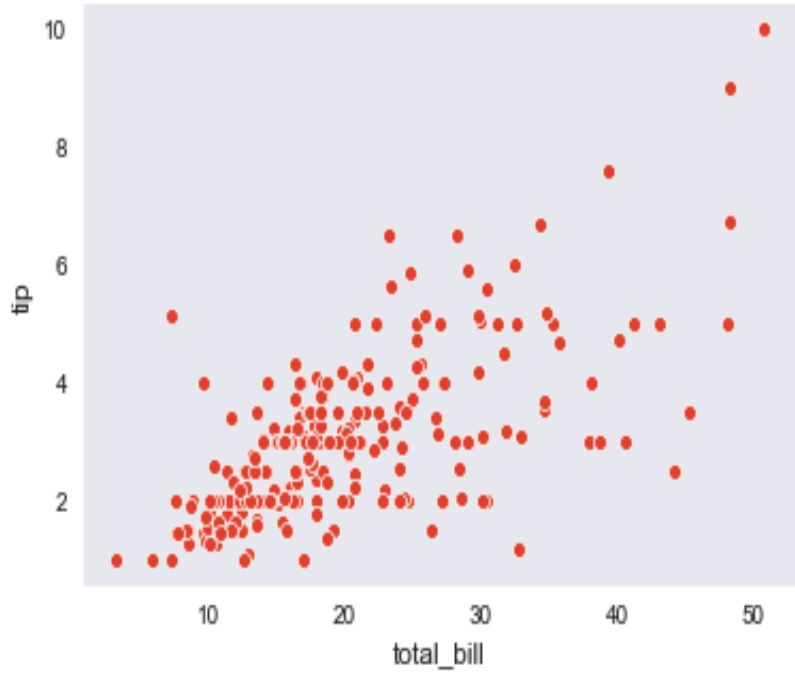


Customizing plots - style

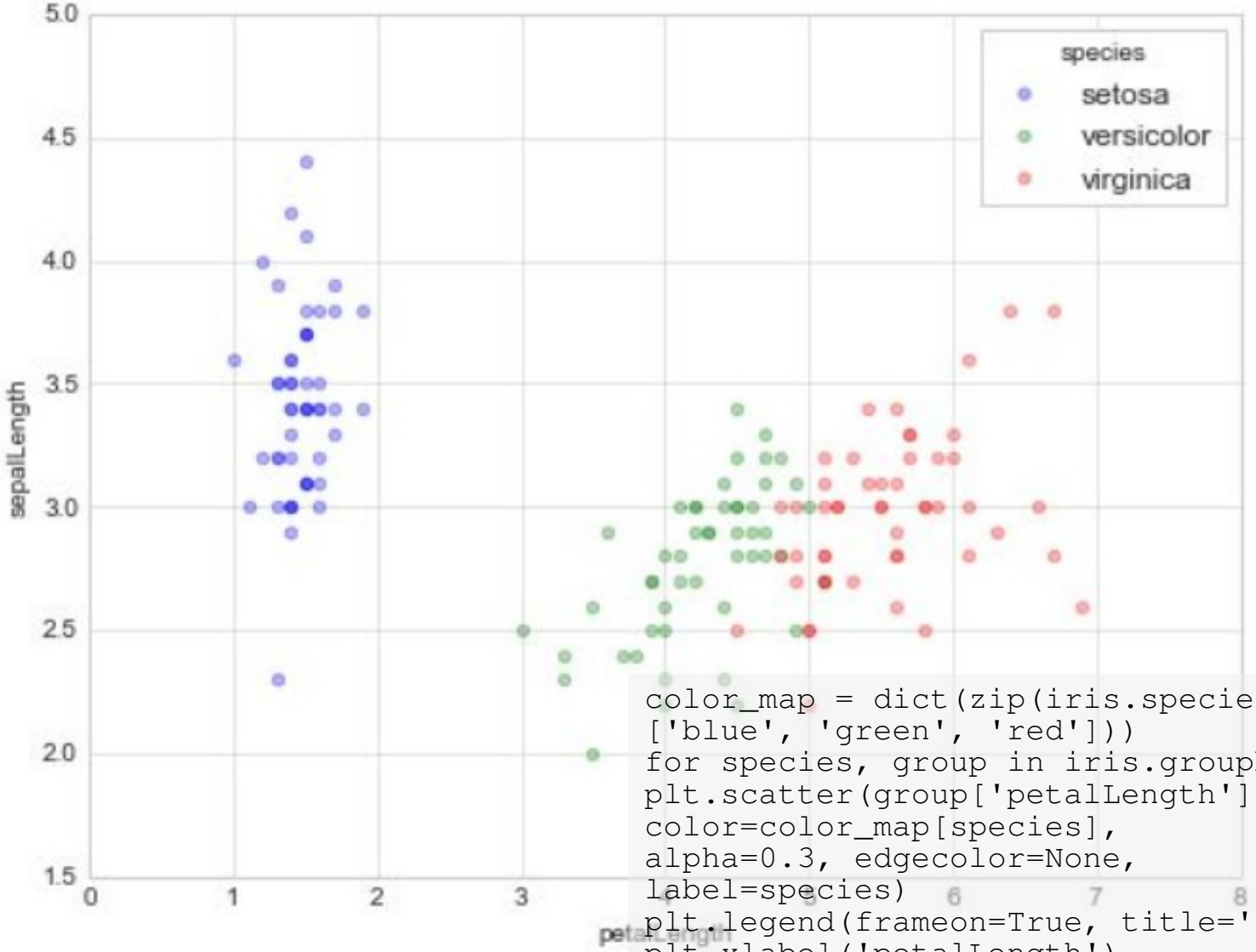
```
In [49]: sns.set_style("dark")

         sns.scatterplot(x="total_bill", y="tip",data=tips)
```

executed in 143ms, finished 11:51:49 2019-07-25

```
Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x1a232d5fd0>
```



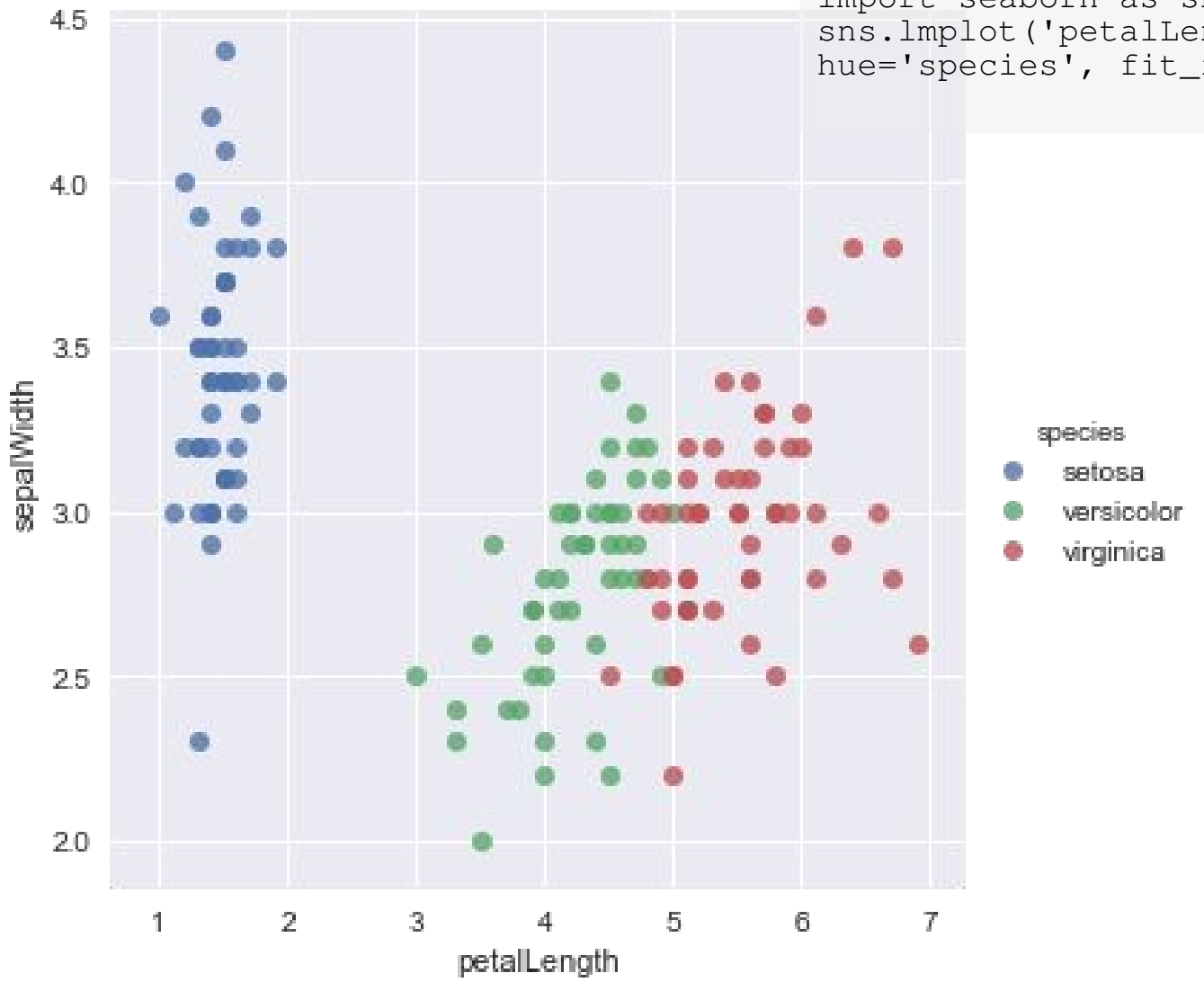
Matplotlib




```
color_map = dict(zip(iris.species.unique(),
['blue', 'green', 'red']))
for species, group in iris.groupby('species'):
plt.scatter(group['petalLength'], group['sepalLength'],
color=color_map[species],
alpha=0.3, edgecolor=None,
label=species)
plt.legend(frameon=True, title='species')
plt.xlabel('petalLength')
plt.ylabel('sepalLength')
```

Seaborn

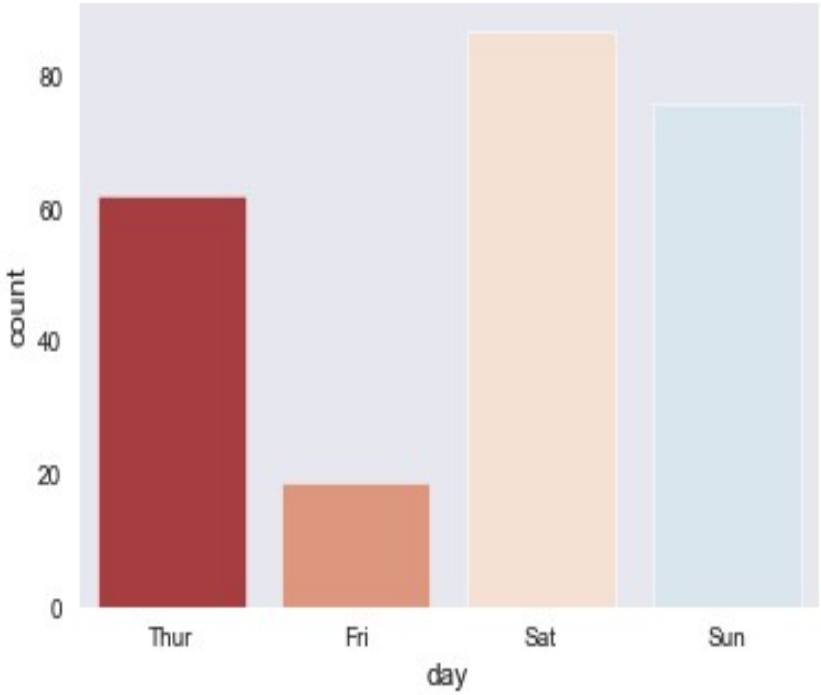
```
import seaborn as sns
sns.lmplot('petalLength', 'sepalWidth', iris,
hue='species', fit_reg=False)
```



Customizing plots - palette

```
In [50]: sns.set_palette("RdBu")

         sns.countplot(x="day",data=tips)
         plt.show()
```

executed in 103ms, finished 11:52:51 2019-07-25

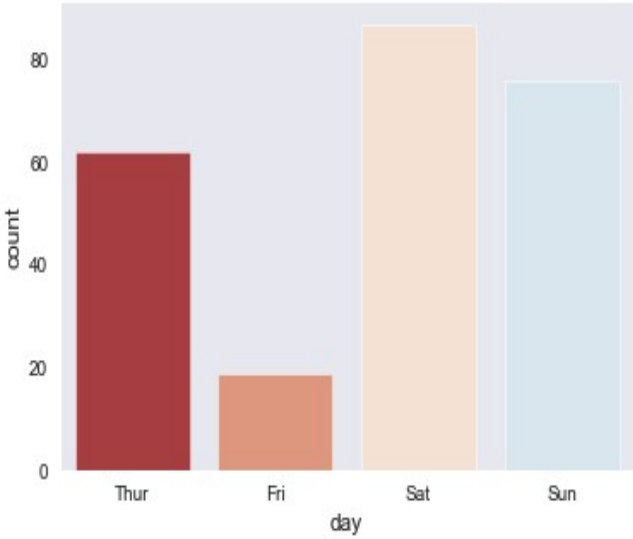


https://matplotlib.org/tutorials/colors/colormaps.html
https://seaborn.pydata.org/tutorial/color_palettes.html

Customizing plots - scale

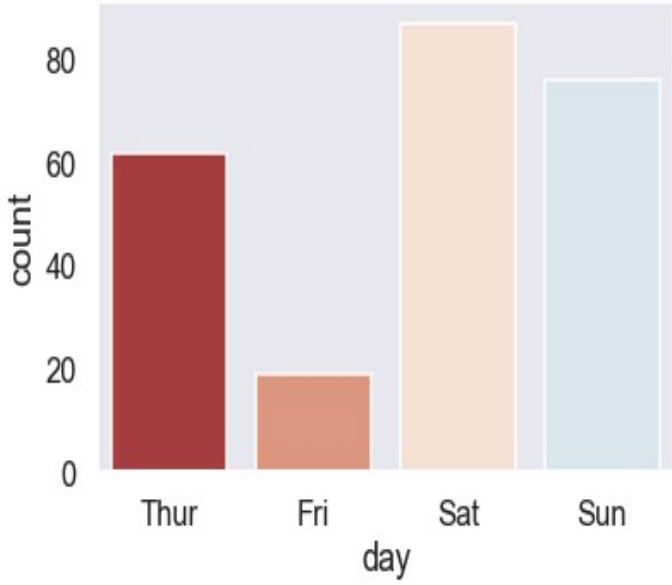
```
In [50]: sns.set_palette("RdBu")

         sns.countplot(x="day",data=tips)
         plt.show()
```

executed in 103ms, finished 11:52:51 2019-07-25




```
In [51]: sns.set_context("talk")

         sns.set_palette("RdBu")

         sns.countplot(x="day",data=tips)
         plt.show()
```

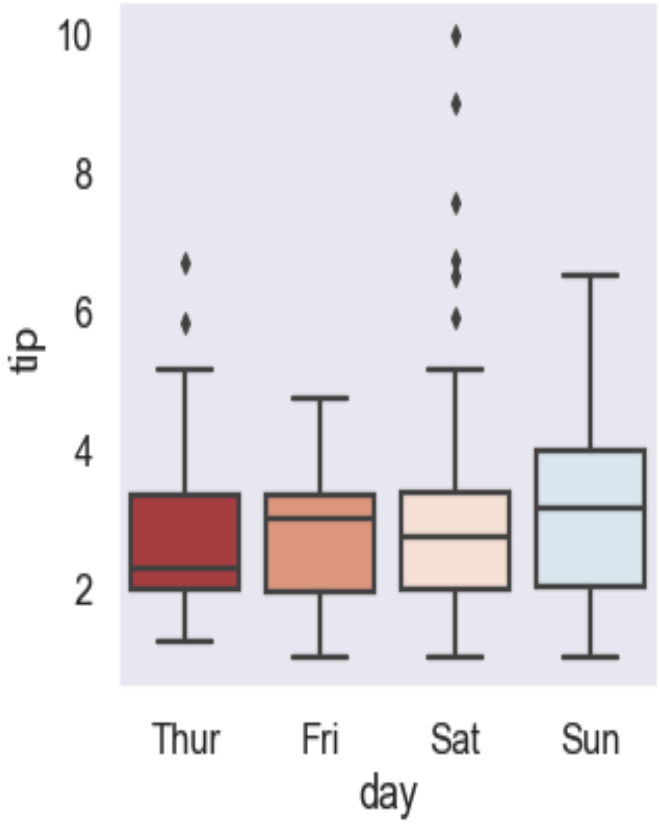
executed in 109ms, finished 11:54:36 2019-07-25



Customizing plots - axis labels

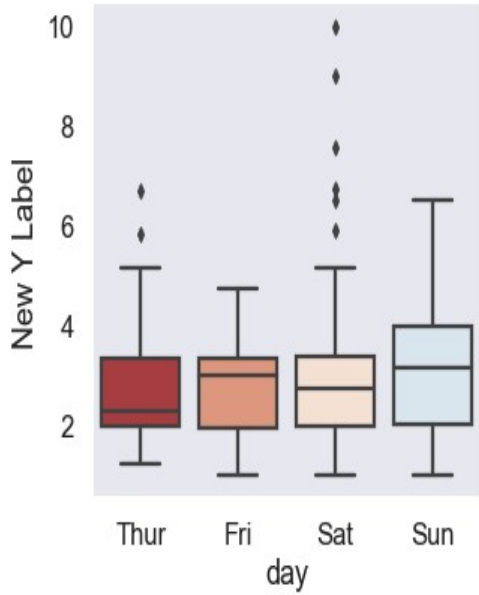
```
In [54]: sns.catplot(x="day",y='tip',data=tips,kind='box
         plt.show()
```

executed in 181ms, finished 12:04:26 2019-07-25




```
In [56]: g = sns.catplot(x="day",y='tip',data=tips,kind='box')

         g.set(ylabel="New Y Label")

         plt.show()
```

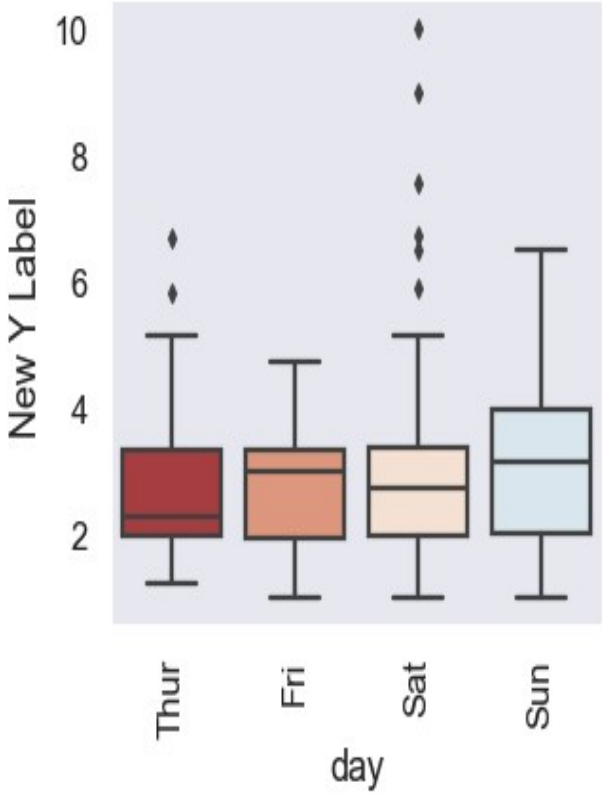
executed in 183ms, finished 12:05:53 2019-07-25



Customizing plots - ticks

```
In [57]: g = sns.catplot(x="day",y='tip',data=tips,kind='box')

         g.set(ylabel="New Y Label")

         plt.xticks(rotation=90)
         plt.show()
```

executed in 191ms, finished 12:06:59 2019-07-25



Customizing plots - titles

Adding a title to AxesSubplot

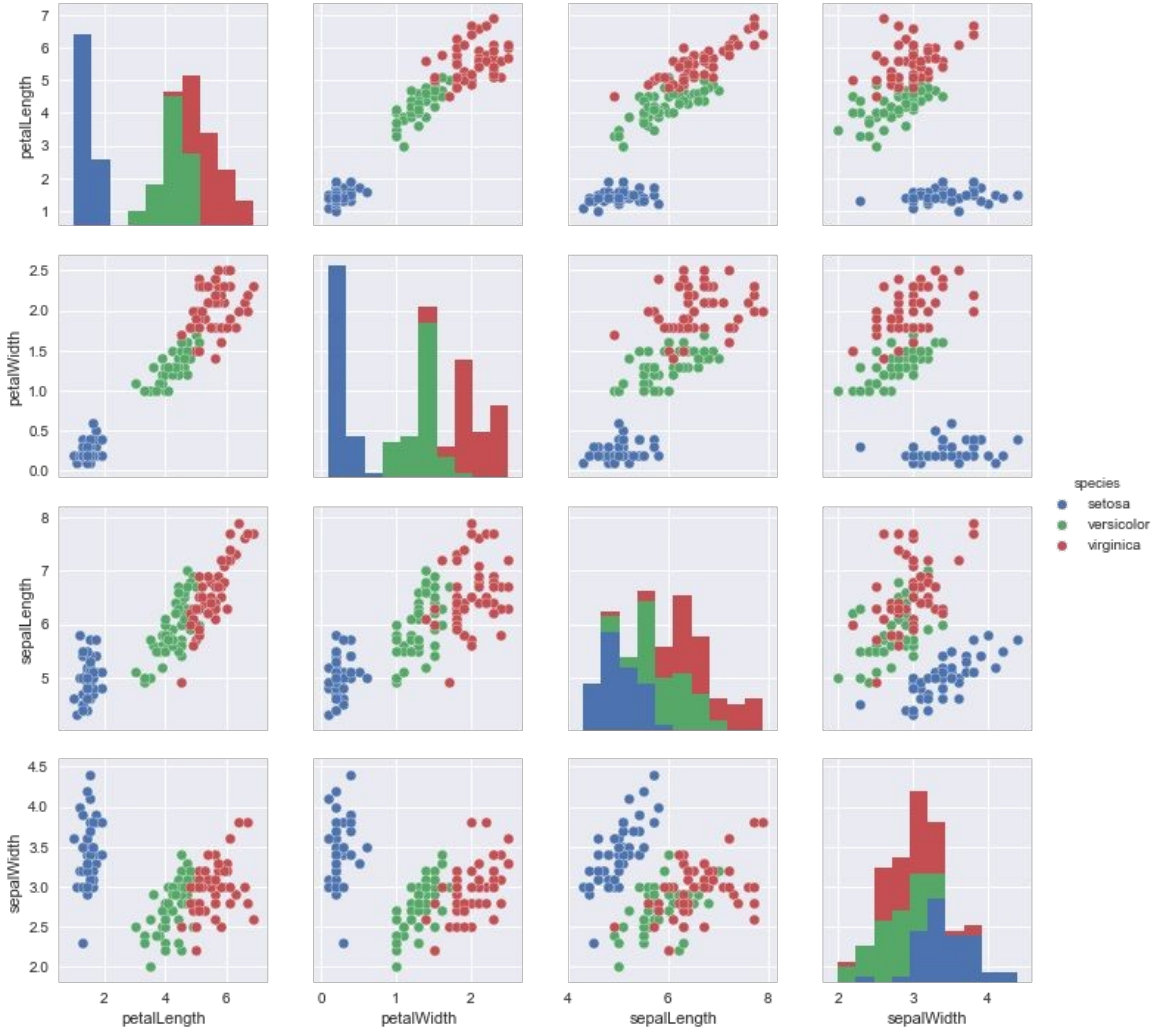
FacetGrid

```
g = sns.catplot(x="Region",
                y="Birthrate",
                data=gdp_data,
                kind="box")


g.fig.suptitle("New Title",
               y=1.03)
```

AxesSubplot

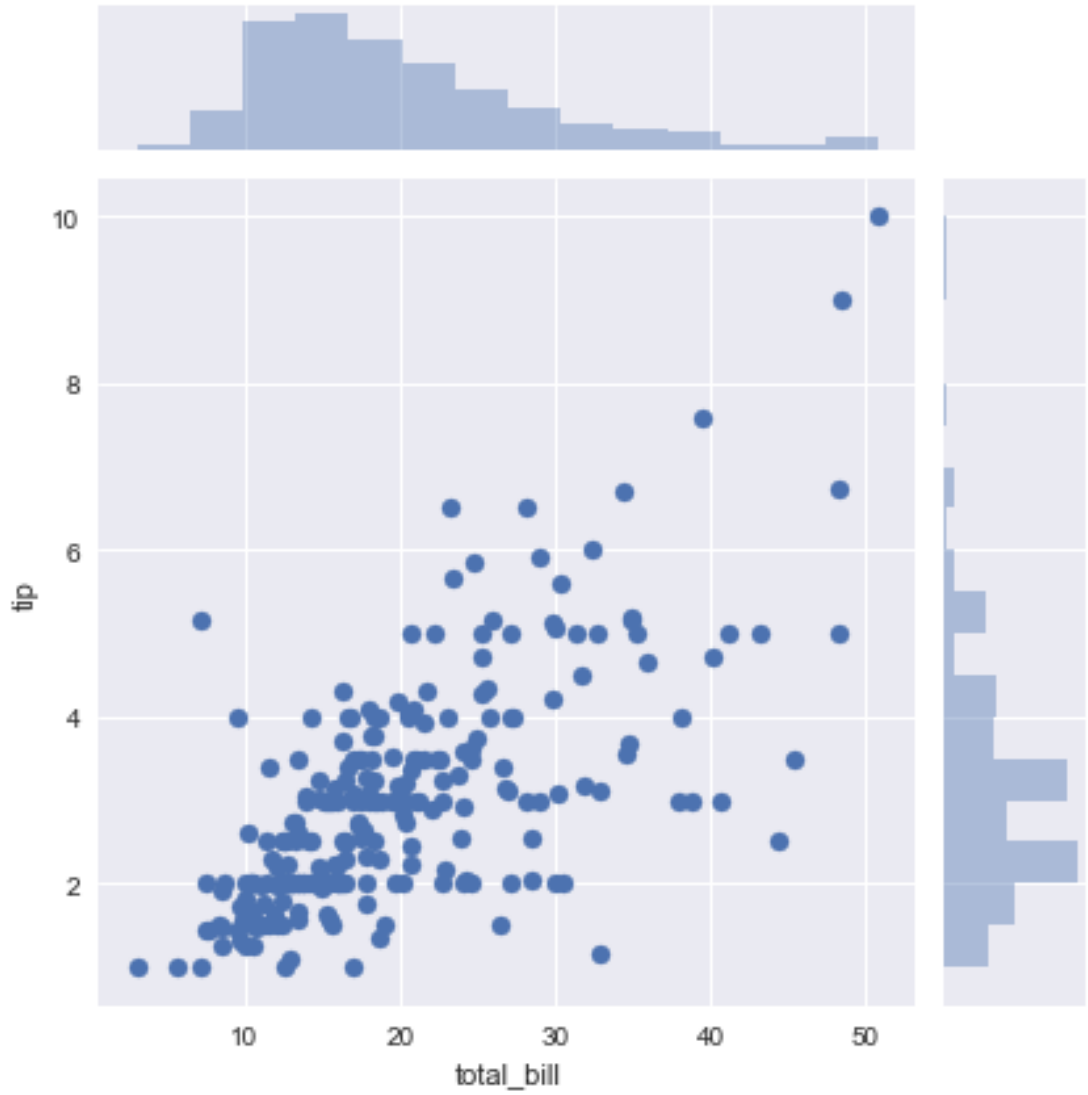
```
g = sns.boxplot(x="Region",
                y="Birthrate",
                data=gdp_data)



g.set_title("New Title",
            y=1.03)
```

Advanced charts




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

Join plot



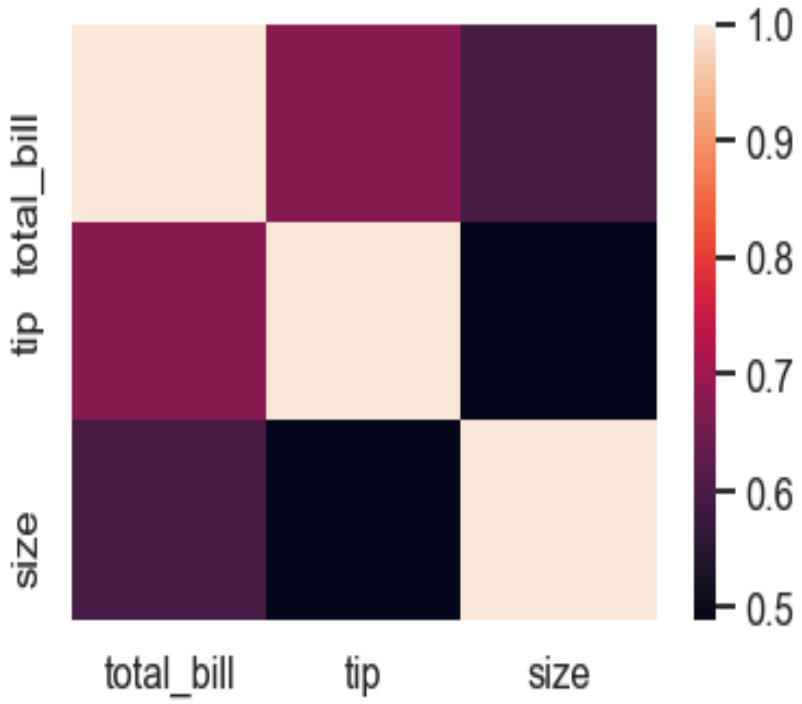

```
sns.jointplot("x", "y", data)
```

Heatmap

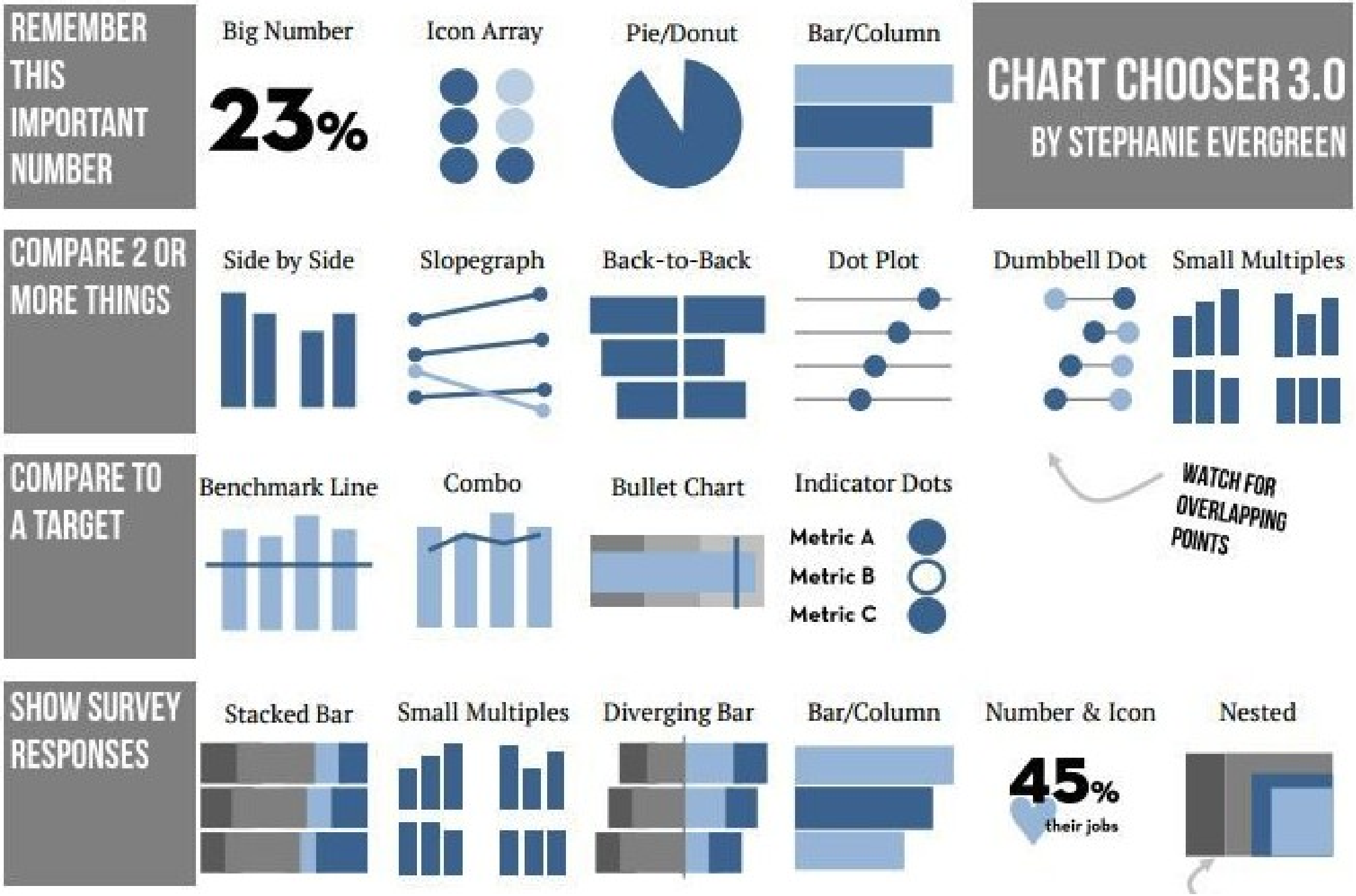
```
In [61]: sns.heatmap(tips.corr(),
                     xticklabels=tips.corr().columns,
                     yticklabels=tips.corr().columns)
```

executed in 161ms, finished 12:43:04 2019-07-25

```
Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0x11fdac7f0>
```



https://github.com/ft-interactive/chart-doctor/blob/master/visual-vocabulary/Visual-vocabulary.pdf

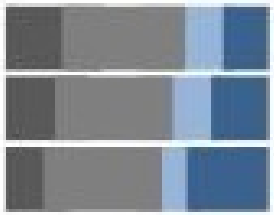


https://stephanieevergreen.com/qualitative-chart-chooser-3/

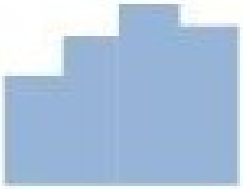
THESE ARE THE PARTS OF THIS WHOLE

Pie/Donut

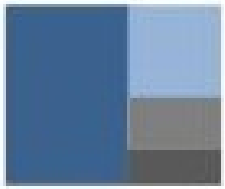
Stacked Bar



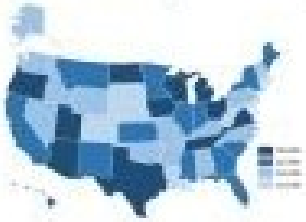
Histogram



Tree Map



Map



FOR BRANCHING QUESTIONS!

VISUALIZE OPEN-ENDED COMMENTS

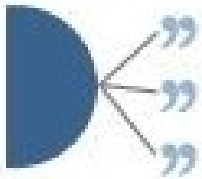
Quote & Pic

Word cloud

Stock photo Rep

Heat Map

Prezi

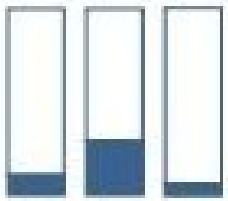


ALSO GOOD FOR SHOWING FLOW

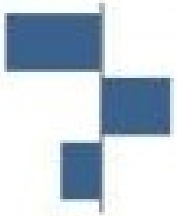
HEY, THINGS CHANGED OVER TIME

Line

Stacked Column

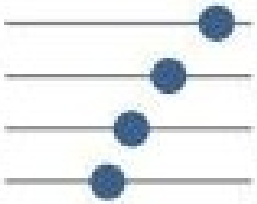


Deviation Bar



Slopegraph

Dot Plot

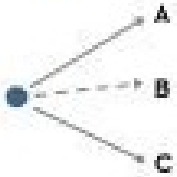


Sankey

THIS THING CHANGES WHEN THAT THING DOES

Scatterplot

Draw It

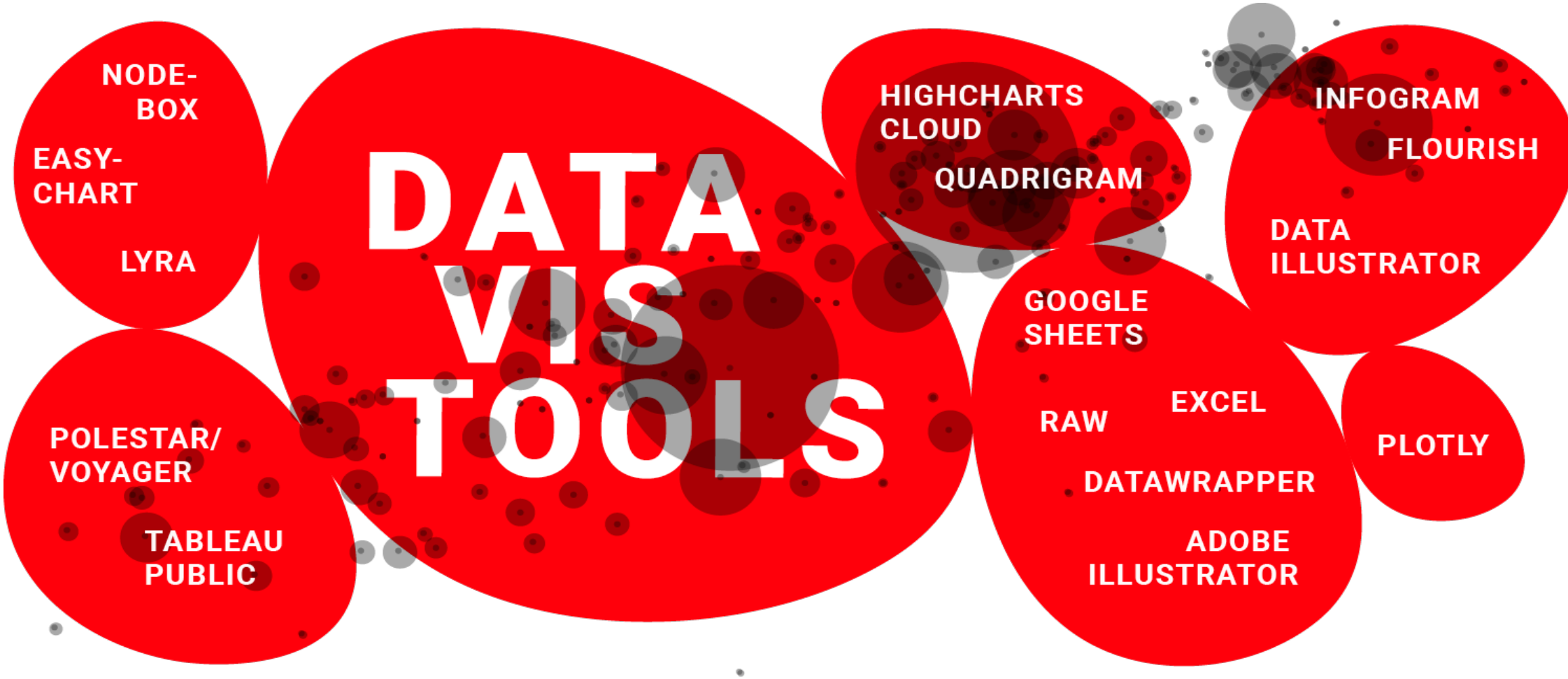


FOR MORE SEE

STEPHANIEEVERGREEN.COM/TAG/STEP-BY-STEP

STEPHANIEEVERGREEN.COM/BLOG

PRESENTING DATA EFFECTIVELY



https://lisacharlotterost.de/datavistools-revisited