



# Python - Data Analysis Essentials

David Pinezich

[david.pinezich@gmail.com](mailto:david.pinezich@gmail.com)



## Timeline

Part 1: Introduction, Course objectives, Python basics, Setting up Pycharm, Jupyter, Getting started with numpy theory(array creation, slicing, utility functions) and exercises(puzzles)

Part 2: Continue Numpy theory(concatenating, splitting, universal functions, aggregations, boolean masking, reading and writing data) and exercises(puzzles)

Part 3: Pandas theory(series and dataframe creation, basic dataframe and series methods, data selection, universal functions) and exercises(puzzles)

Part 4: Continue Pandas theory(Reading and writing data, aggregations, filters, groupby) and exercises(finish puzzles, 3 case studies), visualizations using Seaborn, small visualization example of covid



**Universität  
Zürich** <sup>UZH</sup>

IT Training and Continuing Education

# Using Seaborn to display our data



## Learning Objectives

- You know:
  - What Seaborn is and how it helps to display data
  - How to combine your **Series** and **DataFrame from Pandas** with Seaborn
  - Some basic methods in Seaborn and where **to find them** in the documentation
  - How to use the **default datasets** of Seaborn
  - How to construct more **advanced examples** of **statistical data**



## Seaborn

- Seaborn is a Python data visualization library based on matplotlib
- It provides a high-level interface for drawing attractive and informative statistical graphics
- And it works very well with Pandas
- Seaborn documentation: <https://seaborn.pydata.org/>

```
In [1]: import matplotlib.pyplot as plt  
import seaborn as sns  
sns.__version__
```

JUPYTER NB

```
Out [1]: '0.11.2'
```



## A First Example

- A Seaborn **visualization** works very well with our Pandas data
  - For this example we define **two basic objects**

```
In [1]: df_obj1 = pd.DataFrame({"x": np.random.randn(500),  
                                "y": np.random.randn(500)})  
  
        df_obj2 = pd.DataFrame({"x": np.random.randn(500),  
                                "y": np.random.randint(0, 100, 500)})
```

JUPYTER NB

- And a **jointplot** of the two random DataFrames

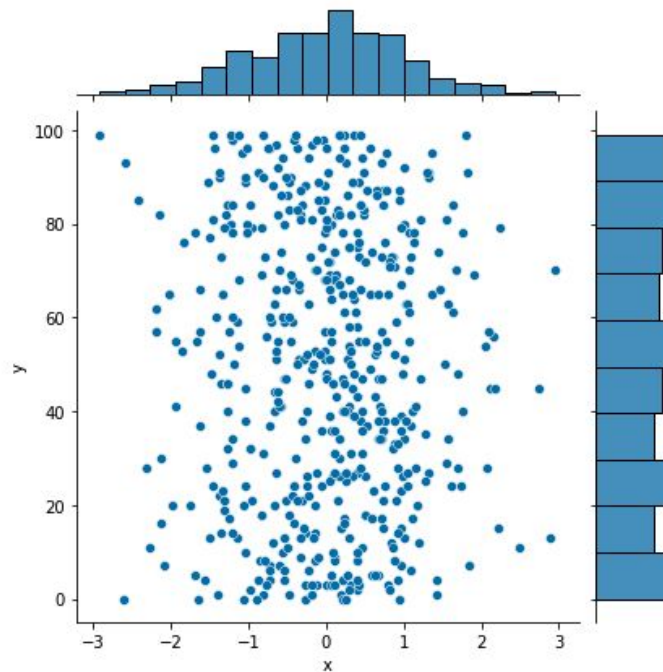
```
In [2]: sns.jointplot(x="x", y="y", data=df_obj2)
```

JUPYTER NB

- Note: the plot will not use both data objects, but they are used in different examples / kinds

## A First Example

- If everything works, you should get a nice plot which **combines (joins) bars** and **points**:





## The kind of Plot

The easiest manipulation of our first example is to change the **kind** of plot we see

- The following kinds are available: { “scatter” | “kde” | “hist” | “hex” | “reg” | “resid” }
- <https://seaborn.pydata.org/generated/seaborn.jointplot.html>

```
In [1]: sns.jointplot(x="x", y="y", data=df_obj2, kind="hex")
```

JUPYTER NB

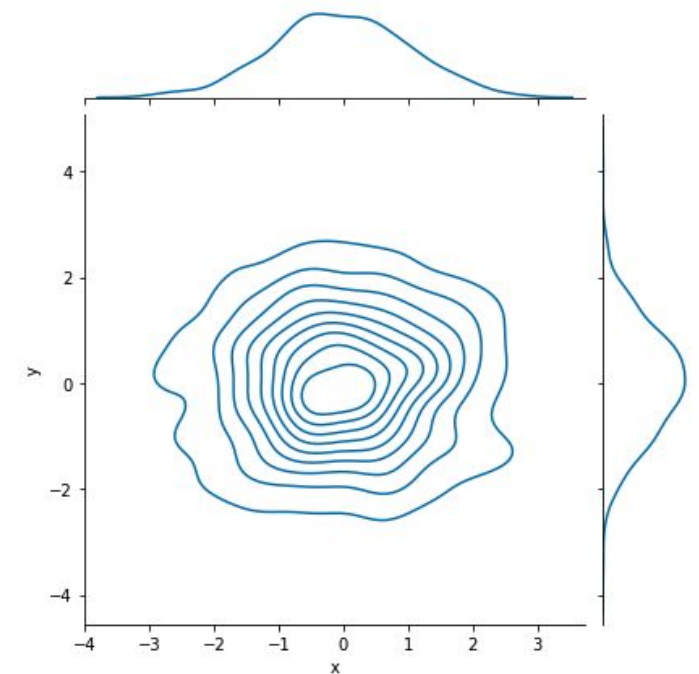
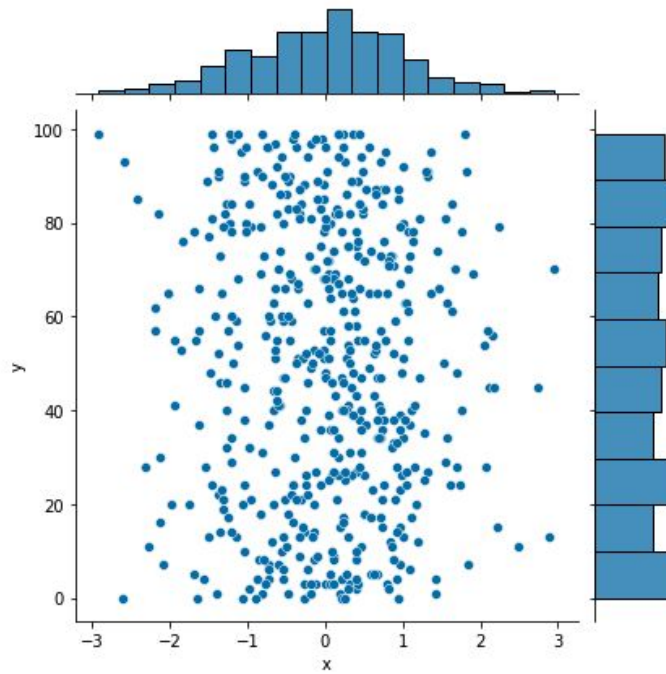
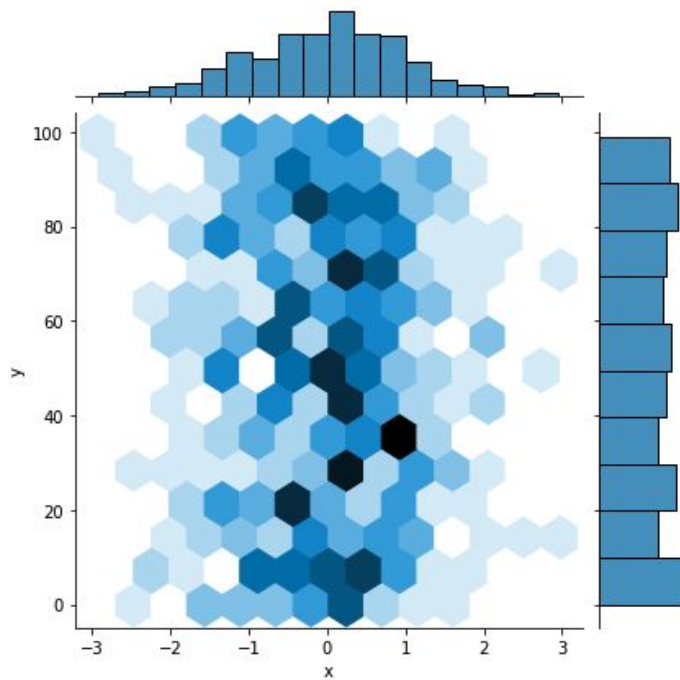
```
In [2]: sns.jointplot(x="x", y="y", data=df_obj1, kind="kde")
```

JUPYTER NB



## The kind of Plot

- Your plots should look like the images underneath, can you describe which one is which?

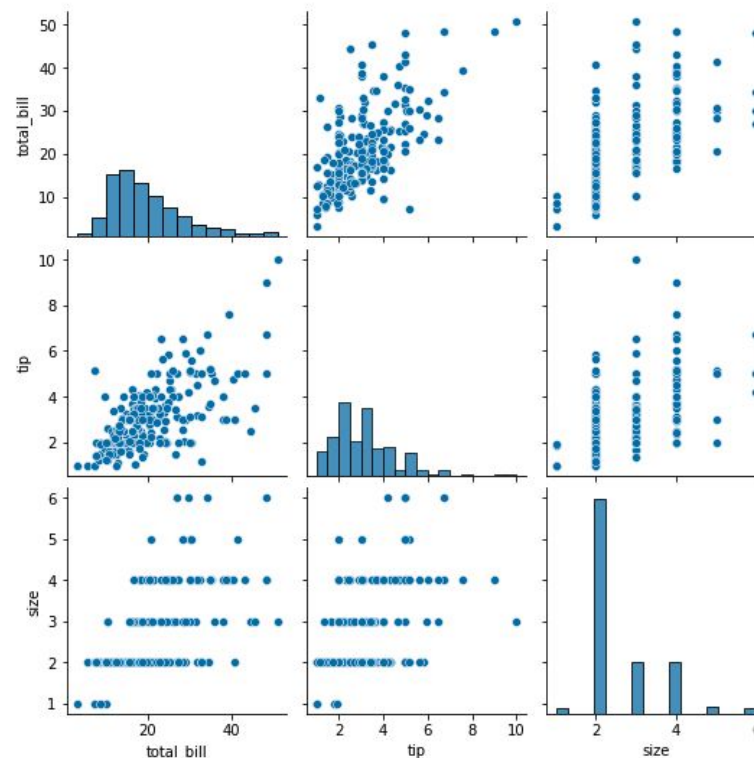


## A Pairplot of Default Data

- As described at the beginning, importing the **seaborn module** brings also some **default data sets**
- One of this datasets is the **“tips” dataset**
- A **pairplot** plots **pairwise relationships** in a dataset

```
In [1]: dataset = sns.load_dataset("tips")  
        sns.pairplot(dataset)
```

- We can see, that this dataset consists of size, tip, total\_bill
- Is it true that **high bills** lead to **high tips**?
- <https://seaborn.pydata.org/generated/seaborn.pairplot.html>





## Default datasets built in to Seaborn

- For the time being, Seaborn brings **17 default datasets**
- These datasets are great to **learn Seaborn** and eventually try out a **fitting plot** before cleaning your data

```
In [1]: sns.get_dataset_names()
```

JUPYTER NB



## Default datasets built in to Seaborn

- Loading a dataset is very easy with `load_dataset('dataset')`

```
In [1]: exercise = sns.load_dataset('exercise')
```

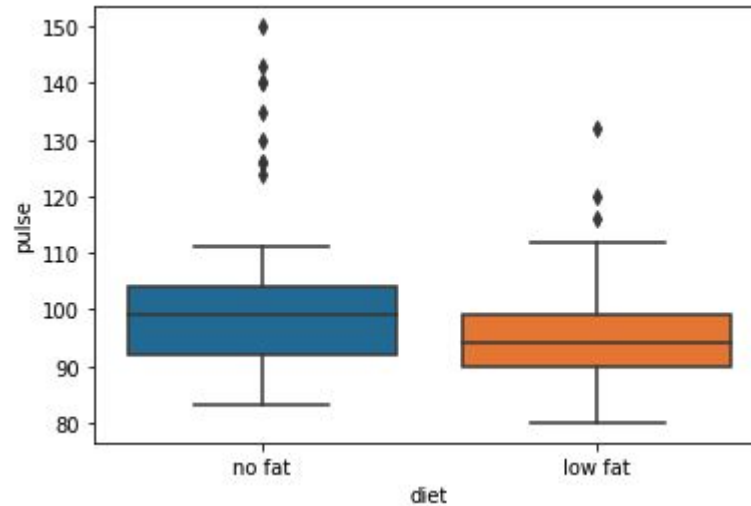
JUPYTER NB

```
sns.stripplot(x="diet", y="pulse", data=exercise)
#sns.swarmplot(x="diet", y="pulse", data=exercise, hue='kind')
#sns.boxplot(x="diet", y="pulse", data=exercise)
#sns.boxplot(x="diet", y="pulse", data=exercise, hue='kind')
#sns.violinplot(x="diet", y="pulse", data=exercise, hue='kind')
#sns.barplot(x="diet", y="pulse", data=exercise, hue='kind')
#sns.pointplot(x="diet", y="pulse", data=exercise, hue='kind')

plt.show()
```

## Default datasets built in to Seaborn

- This is a representation of `sns.stripplot(x="diet", y="pulse", data=exercise)`





## A Deeper Look into **built-in datasets** of Seaborn

- With `print`, you can easily get a first overview of a dataset

```
In [1]: titanic = sns.load_dataset('titanic')  
        print(titanic)
```

JUPYTER NB



## A Deeper Look into `built-in` datasets of Seaborn

```
survived  pclass    sex  age  sibsp  parch    fare embarked  class \
0         0        3  male  22.0    1     0   7.2500      S   Third
1         1        1 female  38.0    1     0  71.2833      C   First
2         1        3 female  26.0    0     0   7.9250      S   Third
3         1        1 female  35.0    1     0  53.1000      S   First
4         0        3  male  35.0    0     0   8.0500      S   Third
..      ...      ...      ...      ...      ...      ...      ...
886        0        2  male  27.0    0     0  13.0000      S  Second
887        1        1 female  19.0    0     0  30.0000      S   First
888        0        3 female   NaN    1     2  23.4500      S   Third
889        1        1  male  26.0    0     0  30.0000      C   First
890        0        3  male  32.0    0     0   7.7500      Q   Third
```

```
who  adult_male  deck  embark_town  alive  alone
0    man         True  NaN  Southampton    no  False
1  woman        False   C   Cherbourg    yes  False
2  woman        False  NaN  Southampton    yes  True
3  woman        False   C   Southampton    yes  False
4    man         True  NaN  Southampton    no  True
..      ...      ...      ...      ...      ...
886  man         True  NaN  Southampton    no  True
887 woman        False   B   Southampton    yes  True
888 woman        False  NaN  Southampton    no  False
889  man         True   C   Cherbourg    yes  True
890  man         True  NaN  Queenstown    no  True
```

```
[891 rows x 15 columns]
```

## A Deeper Look into `built-in datasets` of Seaborn

- If `print(dataset)` is too detailed for your needs, you can also use the already known `df.head()`

```
In [1]: df = sns.load_dataset('car_crashes')  
        print(df.head())
```

JUPYTER NB

	total	speeding	alcohol	not_distracted	no_previous	ins_premium \
0	18.8	7.332	5.640	18.048	15.040	784.55
1	18.1	7.421	4.525	16.290	17.014	1053.48
2	18.6	6.510	5.208	15.624	17.856	899.47
3	22.4	4.032	5.824	21.056	21.280	827.34
4	12.0	4.200	3.360	10.920	10.680	878.41

	ins_losses	abbrev
0	145.08	AL
1	133.93	AK
2	110.35	AZ
3	142.39	AR
4	165.63	CA





**Universität  
Zürich** <sup>UZH</sup>

IT Training and Continuing Education

# Styling and Themes



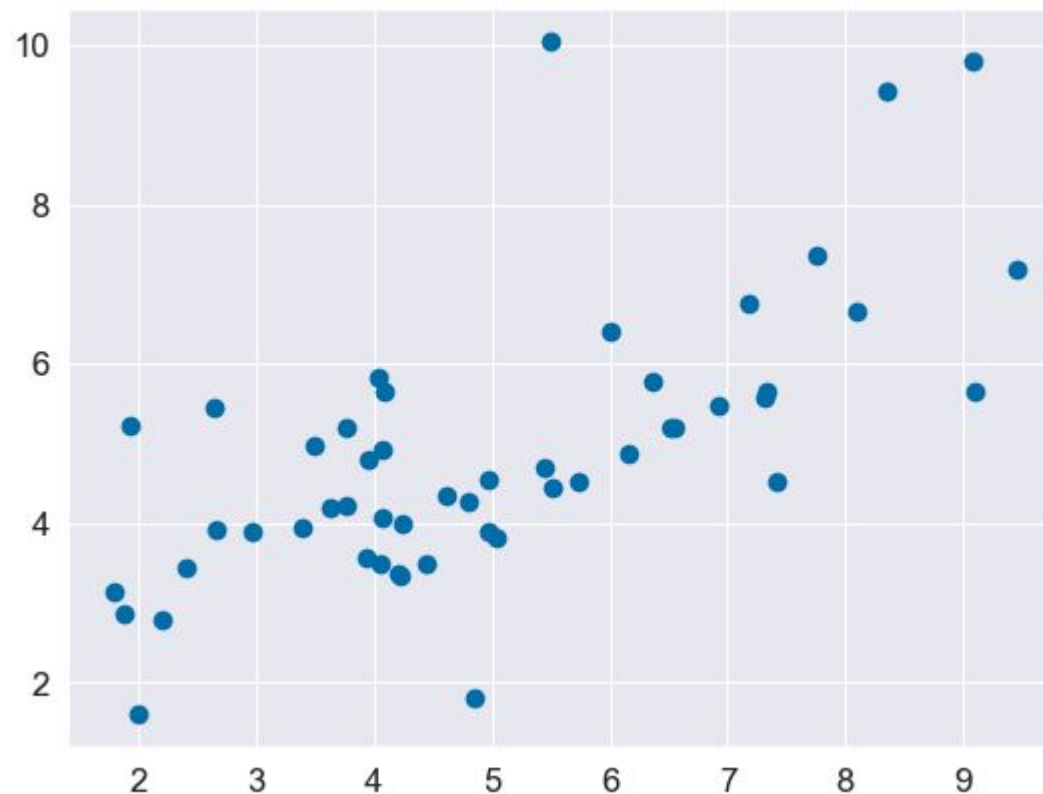
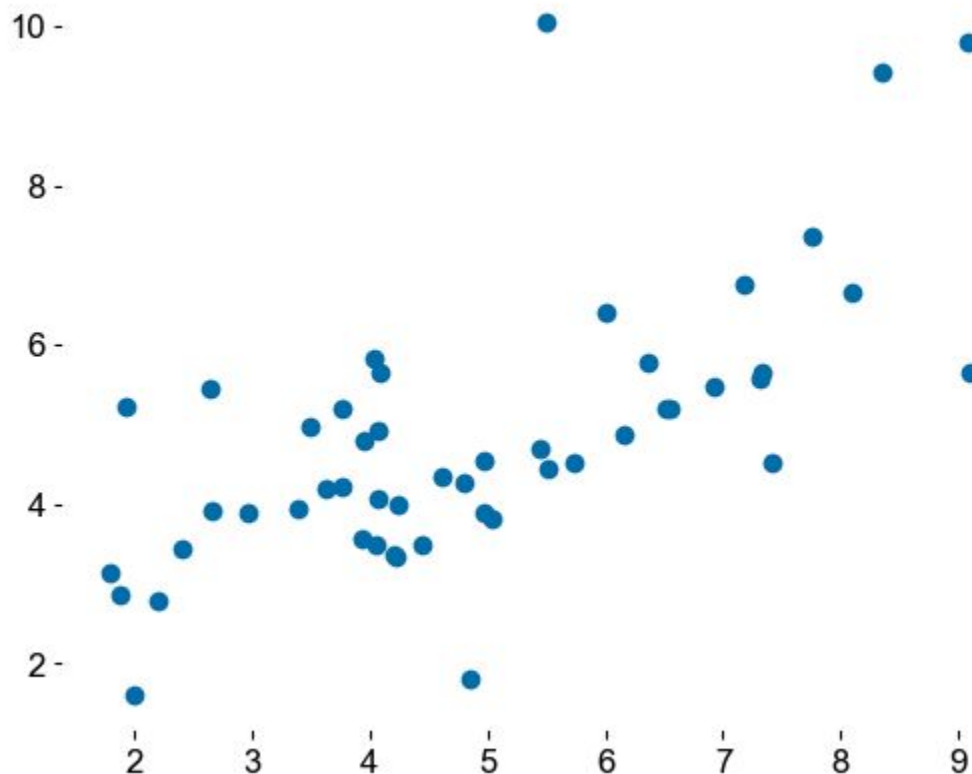
## The `set_style` Option

- The Matplotlib library is already **highly customizable** but a bit dull when it comes to styling options
- It can be very **hard to style** a matplotlib-visualization to your needs
- Here comes Seaborn very handy with its **customized themes and a high-level interface** for customizing and controlling the look of Matplotlib figures

```
In [1]: from matplotlib import pyplot as plt
import seaborn as sns
plt.scatter(df.speeding, df.alcohol)
sns.set_style("darkgrid")
plt.show()
```

JUPYTER NB

## The `set_style` Option





## The `set_style` Option

Seaborn supports various themes that can make styling the plots really easy and save a lot of time. Using the `set_style()` function of Seaborn we can set any of the themes available on Seaborn library. Here are a few of the popular themes:

- Darkgrid
- Whitegrid
- Dark
- White
- Ticks



## The `set_context` Option

Seaborn also allows us to **control individual elements of our graphs** and thus we can control the scale of these elements or the plot by using the `set_context()` function. We have four preset templates for contexts, based on relative size, the contexts are named as follows:

- Paper
- Notebook
- Talk
- Poster
- ...



## Reset Please!

Sometimes you need to **reset your plot-defaults**, because your chart has already been styled but that styling should be different for the next chart (themes are set somewhat universal).

```
In [1]: import matplotlib as mpl
import importlib
importlib.reload(mpl)
importlib.reload(plt)
importlib.reload(sns)
```

JUPYTER NB



**Universität  
Zürich** <sup>UZH</sup>

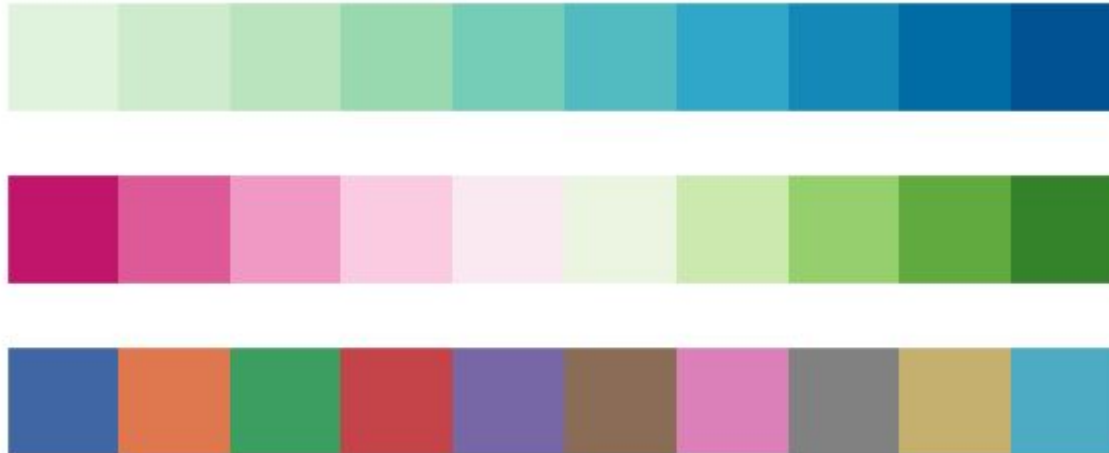
**IT Training and Continuing Education**

# Colors

## The Color Palette of Seaborn

Let us also have a look at the various color palettes seaborn offers (only a small extract):

- `sns.palplot(sns.color_palette("GnBu", 10))`
- `sns.palplot(sns.color_palette("PiYG", 10))`
- `sns.palplot(sns.color_palette("deep", 10))`



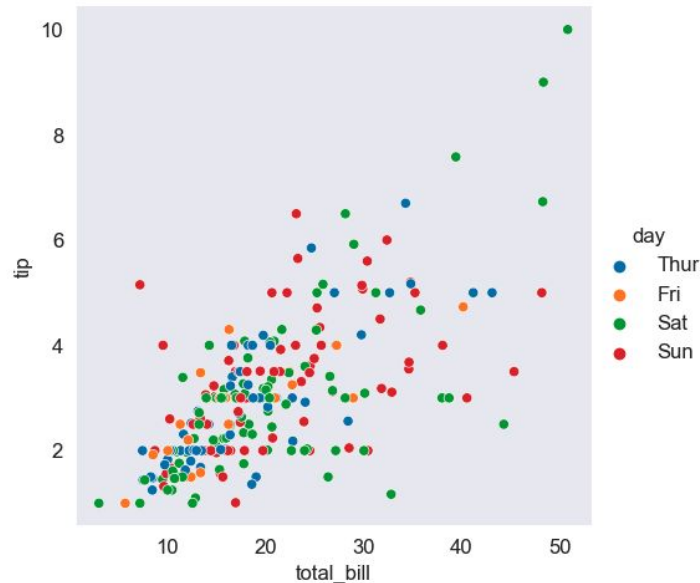


## The Color Palette of Seaborn and plots (hue option)

You can imagine, coloring a plot is as straightforward as configuring the plot itself:

```
In [1]: tips = sns.load_dataset("tips")  
        sns.relplot(data=tips, x="total_bill", y="tip", hue="day")
```

JUPYTER NB





**Universität  
Zürich** <sup>UZH</sup>

**IT Training and Continuing Education**

# Several Examples



## Histogram (repetition)

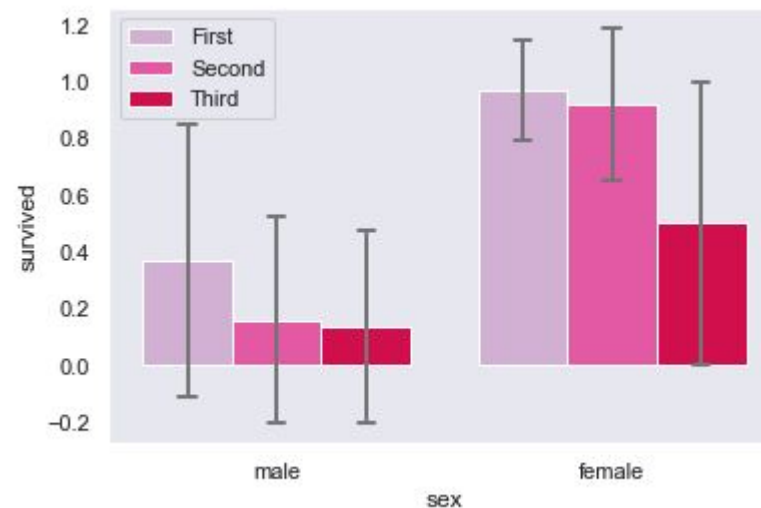
Please take some time and try this plot and see what happens, can you describe all steps

```
In [1]: import seaborn as sns
        from matplotlib import pyplot as plt
        df = sns.load_dataset('iris')
        sns.distplot(df['petal_length'], kde = False)
```

JUPYTER NB

## Barplot+

```
import matplotlib.pyplot as plt
import seaborn as sns
titanic = sns.load_dataset('titanic')
# create plot
sns.barplot(x = 'sex', y = 'survived', hue = 'class', data = titanic,
            palette = 'PuRd',
            order = ['male', 'female'],
            capsize = 0.05,
            saturation = 8,
            errcolor = 'gray', errwidth = 2,
            ci = 'sd'
            )
plt.legend()
plt.show()
```



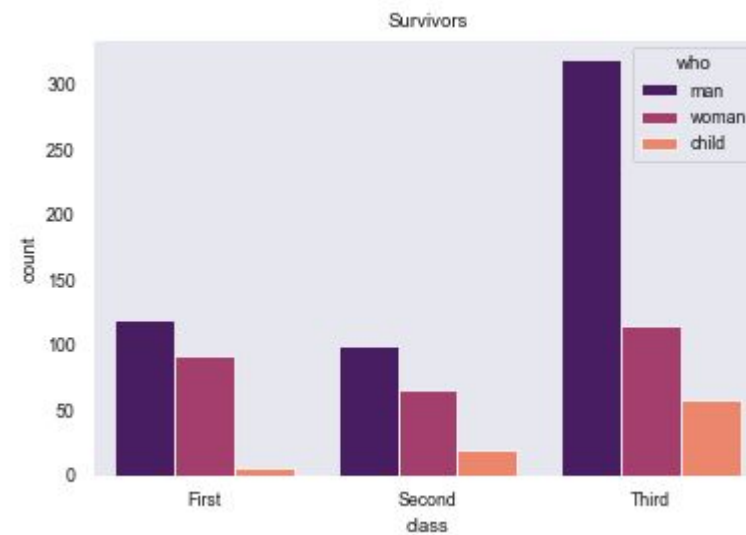


## Countplot

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_context('paper')

# load dataset
titanic = sns.load_dataset('titanic')
# create plot
sns.countplot(x = 'class', hue = 'who', data = titanic, palette = 'magma')
plt.title('Survivors')
plt.show()
```

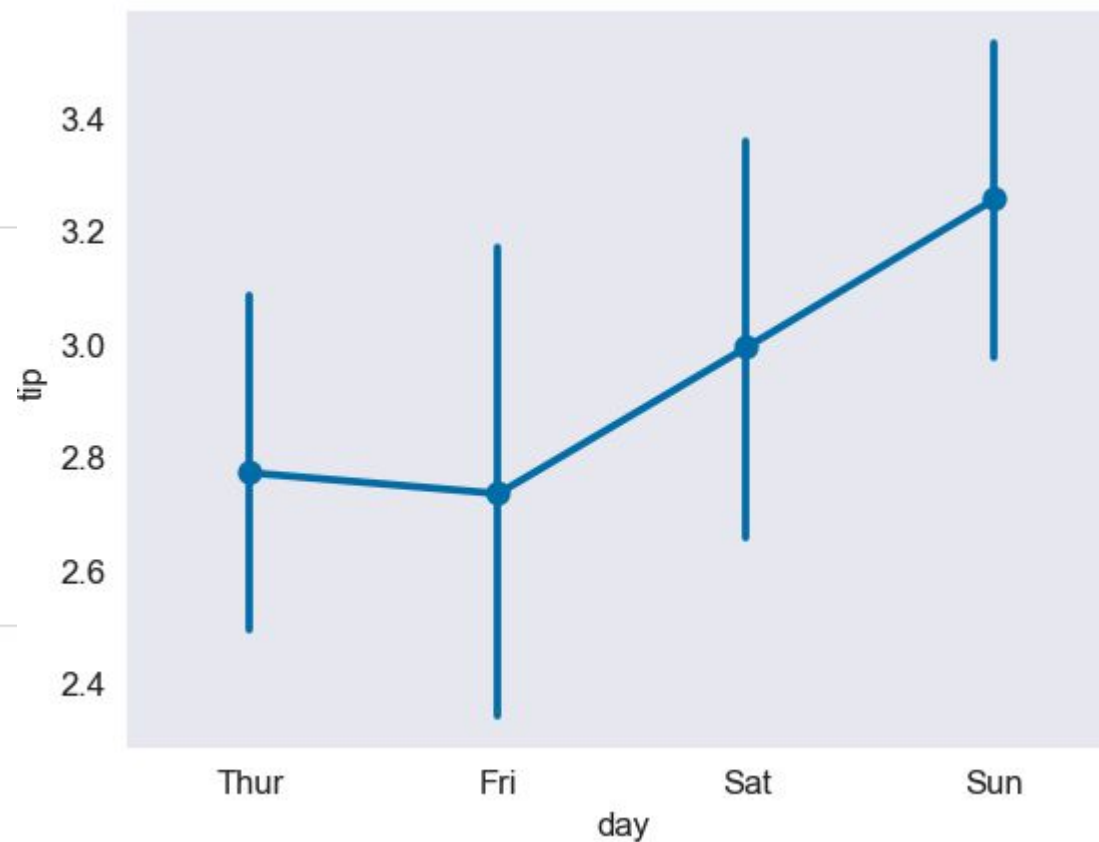




## Pointplot

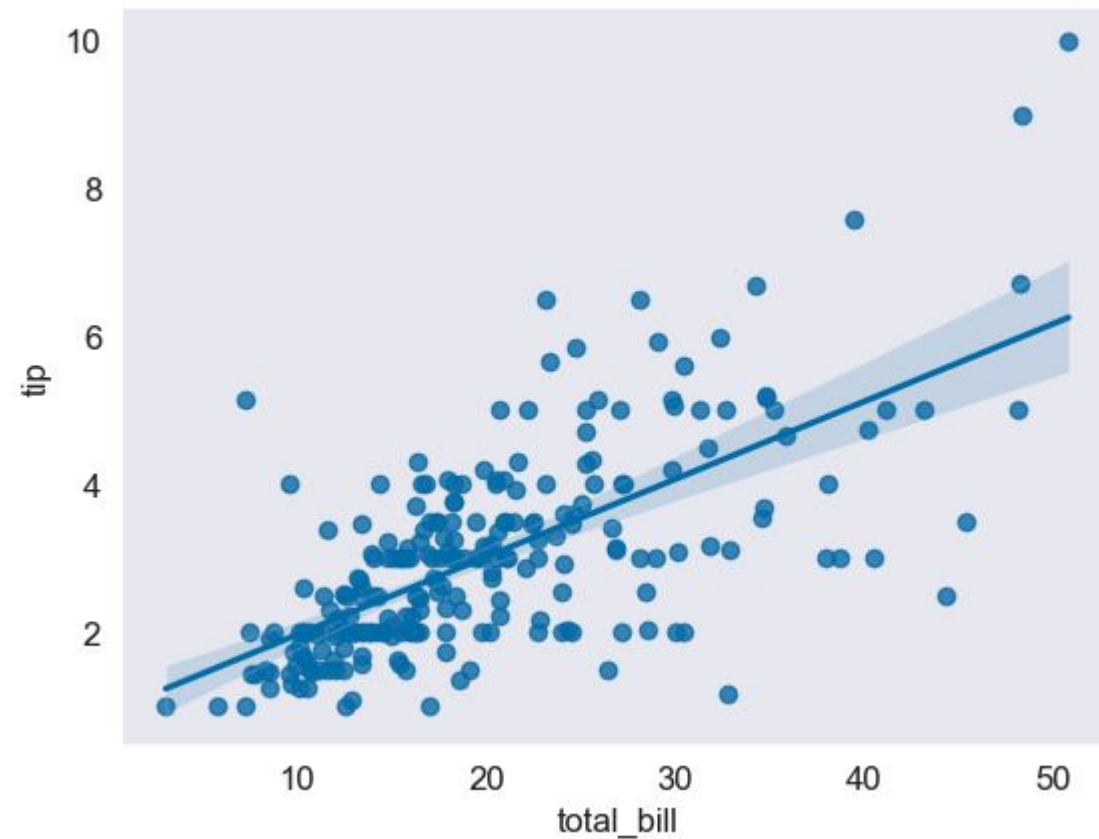
```
# importing required packages
import seaborn as sns
import matplotlib.pyplot as plt

# loading dataset
data = sns.load_dataset("tips")
sns.pointplot(x="day", y="tip", data=data)
plt.show()
```



## Regplot

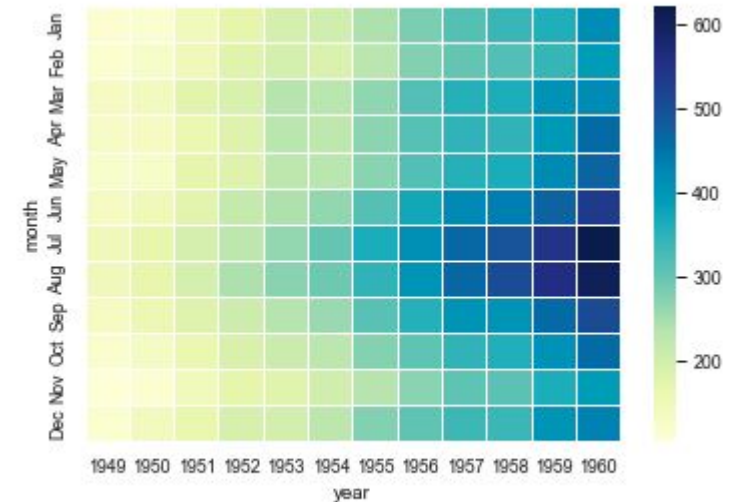
```
import seaborn as sns
tips = sns.load_dataset("tips")
ax = sns.regplot(x="total_bill", y="tip", data=tips)
```



## Final example

```
flights=sns.load_dataset("flights")
flights = flights.pivot("month", "year", "passengers")
print(flights)
sns.heatmap(flights,linewidths=.5,cmap="YlGnBu")
```

year	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
month												
Jan	112	115	145	171	196	204	242	284	315	340	360	417
Feb	118	126	150	180	196	188	233	277	301	318	342	391
Mar	132	141	178	193	236	235	267	317	356	362	406	419
Apr	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
Jun	135	149	178	218	243	264	315	374	422	435	472	535
Jul	148	170	199	230	264	302	364	413	465	491	548	622
Aug	148	170	199	242	272	293	347	405	467	505	559	606
Sep	136	158	184	209	237	259	312	355	404	404	463	508
Oct	119	133	162	191	211	229	274	306	347	359	407	461
Nov	104	114	146	172	180	203	237	271	305	310	362	390
Dec	118	140	166	194	201	229	278	306	336	337	405	432





## Final example cont

```
import seaborn as sns
car_crashes = sns.load_dataset("car_crashes")
corr=car_crashes.corr()
print(corr)
sns.heatmap(corr,annot=True,linewidths=.5,cmap="YlGnBu")
```

	total	speeding	alcohol	not_distracted	no_previous	\
total	1.000000	0.611548	0.852613	0.827560	0.956179	
speeding	0.611548	1.000000	0.669719	0.588010	0.571976	
alcohol	0.852613	0.669719	1.000000	0.732816	0.783520	
not_distracted	0.827560	0.588010	0.732816	1.000000	0.747307	
no_previous	0.956179	0.571976	0.783520	0.747307	1.000000	
ins_premium	-0.199702	-0.077675	-0.170612	-0.174856	-0.156895	
ins_losses	-0.036011	-0.065928	-0.112547	-0.075970	-0.006359	

	ins_premium	ins_losses
total	-0.199702	-0.036011
speeding	-0.077675	-0.065928
alcohol	-0.170612	-0.112547
not_distracted	-0.174856	-0.075970
no_previous	-0.156895	-0.006359
ins_premium	1.000000	0.623116
ins_losses	0.623116	1.000000

