Visualization principles


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib as mpl
```

1. Show the data - Jitter

Google it: Jitter in python

Documentation contents such a good example we'll just follow it.

```
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

```
sns.boxplot(x="smoker", y="tips", data=tips)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a0a0a0a90>
```


```
sns.stripplot(x="day", y="total_bill", data=tips)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a0a0a0a90>
```

Use a smaller amount of jitter

```
sns.stripplot(x="day", y="total_bill", data=tips, jitter=0.05)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a0a0a0a90>
```

Jitter plus a boxplot

```
ax = sns.boxplot(x="day", y="tips", data=tips)
ax = sns.stripplot(x="day", y="tips", data=tips, jitter=.3)
```

(menu)

2. Consider the scale - logs

```
diamonds = sns.load_dataset("diamonds")
diamonds.head()
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

scaling the y axis

```
diamonds_2 = diamonds.loc[(diamonds.cut == 'Ideal')&(diamonds.color == 'E')]
ax = sns.boxplot(x="clarity", y = diamonds_2.price, data=diamonds_2)
ax.set_yscale("log")
plt.show()
```

scaling the x axis

```
big_sample = pd.sample(frac=0.2, n=, replace=)
ax = sns.boxplot(x="price", y = big_sample.clarity, data=big_sample)
ax.boxplot(vert=False,positions=ax.get_yticks(), [big_price > 5]ax, ax = axes[1])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a0a0a0a90>
```

(menu)

3. Baselines

Let look at a case were we have 2 binary variables. 'sex' and 'survived'

```
titanic = sns.load_dataset("titanic")
titanic.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	True

Plotting one categorical variable using countplot:

Exercise: plot the sex, plot the survived

```
sns.countplot(x = "sex", data = titanic)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a0a0a0a90>
```


```
sns.countplot(x = "class", data = titanic)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a0a0a0a90>
```

advanced: plot them side by side

```
big_sample = pd.sample(frac=0.2, n=, replace=)
ax = sns.boxplot(x="price", y = big_sample.clarity, data=big_sample)
ax.boxplot(vert=False,positions=ax.get_yticks(), [big_price > 5]ax, ax = axes[1])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a0a0a0a90>
```

Add 'hue' to plot a countplot that compares two variables

```
sns.countplot(x="sex", y=[survived], hue="class", ax = None, data=titanic)
plt.title("Titanic survivors by gender and class")
plt.xlabel("Percentage of survivors")
plt.ylabel("")
plt.show()
```

Use barplot to plot percentages for two variables

```
sns.barplot(x="sex", y=[survived], hue="class", ax = None, data=titanic)
plt.title("Titanic survivors by gender and class")
plt.xlabel("Percentage of survivors")
plt.ylabel("")
plt.show()
```

That was easy since SURVIVAL is binary. How do we plot percentages for non-binary data?

For example let's plot the fare percentage

> Exercise: what is the fare's sum?
>
> How do you suggest to compute percentage?

```
titanic["fare"].sum()
```

```
14454.9683
```

```
sns.barplot(x="class", y="survived", hue="sex", data=titanic)
plt.show()
```

(menu)

4. Ease comparisons

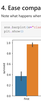
Note what happens when we swap the hue and the x axis. You can also remove the hue altogether (check what happens).

```
sns.barplot(x="class", y="survived", hue="sex", data=titanic)
plt.show()
```

Change the order of the x axis

```
sns.barplot(x="class", y="survived", hue="sex", data=titanic, order=["Third", "Second", "First"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a0a0a0a90>
```



Add labels to the plot

Google it: add text to plot seaborn

```
penguins = sns.load_dataset("penguins")
penguins.head()
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female

> Exercise: plot the relationship between: bill_length_mm and bill_depth_mm
>
> advanced: color the different species:

```
ax = sns.scatterplot(data=penguins, x="bill_length_mm", y="bill_depth_mm", hue = "species", palette = "colblind")
```

Without a legend but with text

```
ax = sns.scatterplot(data=penguins, x="bill_length_mm", y="bill_depth_mm", hue = "species", palette = "colblind",
legend=False)
texts = dict(zip(zip(*ax.collections[0].get_offsets.T), ax.get_text()))
ax.text(50.75, 20, "Chinstrap", **texts)
ax.text(45.25, 17.5, "Adelie", **texts)
plt.show()
```

(menu)

5. Set the scale

Google it: set scale seaborn

One option:

```
ax = sns.stripplot(x="day", y="total_bill", data=tips, jitter=0.05)
ax.set_ylim(0, 100)
```

```
(0.0, 100.0)
```

Another option:

```
plt.ylim(0, 100)
ax = sns.stripplot(x="day", y="total_bill", data=tips, jitter=0.05)
```