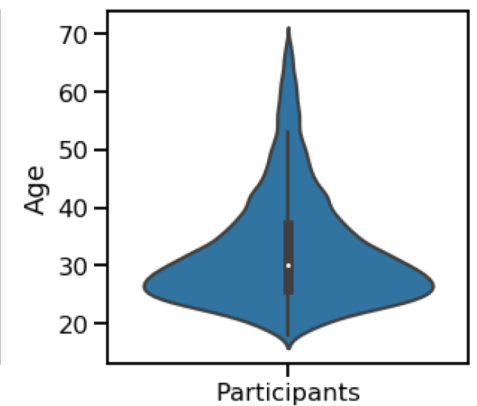
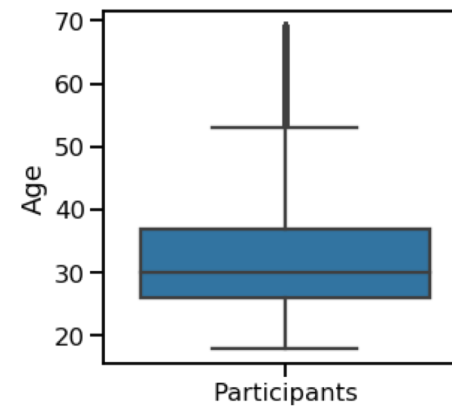
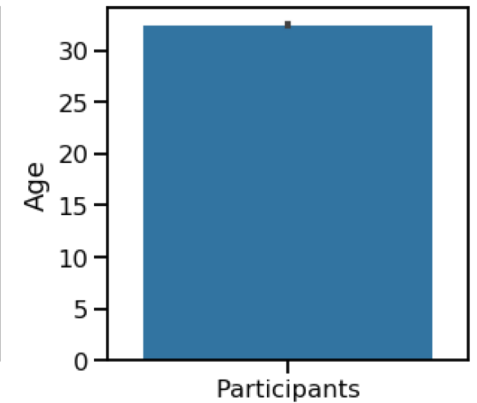
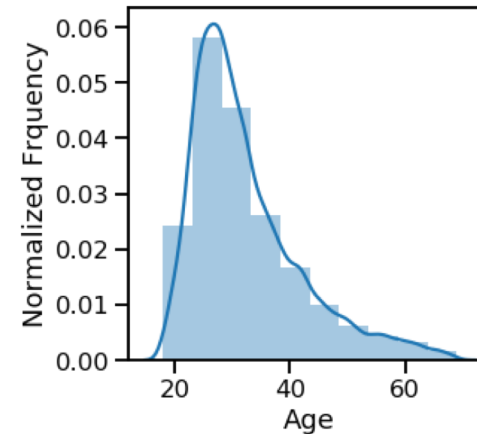


Graphical Data Analysis (Exercise 4): Background information session



Learning Objectives

1. Essential statistics for data analysis
2. Plotting types for data analysis
3. Addressing a hypothetical problem

1. Essential statistics

Essential statistics definitions:

- **Mean (arithmetic, average):** the sum of a collection of numbers divided by the count of numbers in the collection

[1, 2, 2, 2, 3, 3]

$\text{mean} = (1+2+2+2+3+3)/6 = 2.17$

- **Median:** the value separating the higher half from the lower half of a dataset

[1, 2, 2, 2, 3, 3]

median = 2

- **Mode:** the value that appears more often in a dataset

[1, 2, 2, 2, 3, 3]

mode = 2

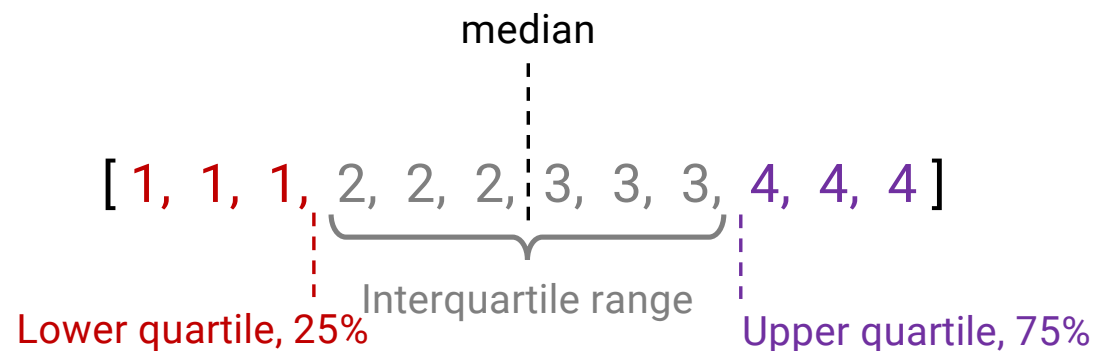
```
import pandas as pd
df = pd.read_excel('filename.ext')

avg = df['column1'].mean()
median = df['column1'].median()
mode = df['column1'].mode()
```

1. Essential statistics

Essential statistics definitions:

- **Quartile:** a division of the data into four subgroups
- **Upper quartile:** splits the highest 25% of the data and lowest 75% of the data
- **Lower quartile:** the lowest 25% of the data
- **Interquartile range:** the middle 50% of the data



1. Essential statistics

Essential statistics definitions:

- **Standard deviation (σ):** gives information on the variability of the samples w.r.t. the mean; how much do the sample values differ from the mean

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}}$$

\bar{x} = mean

x_i = each value in the dataset

n = size of the population

- **Standard error ($\sigma_{\bar{x}}$):** gives information regarding sampling; how far away is the sample mean from the population mean

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$$

```
import pandas as pd
df = pd.read_excel('filename.ext')

std = df['column1'].std()
sem = df['column1'].sem()
```

When plotting values with error, it is important to indicate which value you plot.

1. Essential statistics

Dataset: [1, 2, 2, 2, 3, 3]

- **Standard deviation (σ)**

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}}$$

$$\bar{x} = 2.17$$

$$\sum_{i=1}^n (x_i - \bar{x})^2 = ((2.17 - 1)^2 + 3 \cdot (2.17 - 2)^2 + 2 \cdot (2.17 - 3)^2) = 2.84$$

$$n = 6$$

$$\sigma = 0.75$$

- **Standard error ($\sigma_{\bar{x}}$)**

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$$

$$\sigma_{\bar{x}} = \frac{0.75}{\sqrt{6}} = 0.31$$

When plotting values with error, it is important to indicate which value you plot.

Plotting standard error without indication is deceptive!

2. Plotting types: Data wrangling

```
# Let's analyze the age of people that  
are on a popular dating website.
```

```
# Import the necessary modules
```

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

```
# Import the data using pandas
```


```
df = pd.read_csv('fake-age-data.csv')
```

```
df.head()
```

```
# Let's calculate how many full decades  
each person has lived.
```

```
df_decade = df/10  
df_decade = df_decade.astype(int)
```

```
df_decade.head()
```



	age
0	22
1	38
2	23
3	29
4	29



	age
0	2
1	3
2	2
3	2
4	2

This dataframe has only one column ('age') and an index.

Note: converting to an integer does not round! It merely drops off the decimal values.

dataframe.head() prints the first 5 rows.
dataframe.head(#) prints the first # rows.
dataframe.tail(#) prints the last # rows.

Note: this dataset is very large, so we are using a csv file.

2. Plotting types: Examining data with pandas

```
# Import the necessary modules
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Import the data using pandas
df = pd.read_csv('fake-age-data.csv')

df.head()
```

	age
0	22
1	38
2	23
3	29
4	29

```
age_len = len(df['age'])
age_avg = round(df['age'].mean(), 2)

print("The length of the dataset is  
      "+str(age_len)+".")

print("The mean of the dataset is  
      "+str(age_avg)+".")
```

The length of the dataset is 45952.
The mean of the dataset is 32.51.

Is there an easier way to collect the statistical information?

2. Plotting types: Examining data with pandas

```
# Import the necessary modules
```

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

```
# Import the data using pandas
```

```
df = pd.read_csv('fake-age-data.csv')
```

```
df.head()
```

```
df.describe()
```

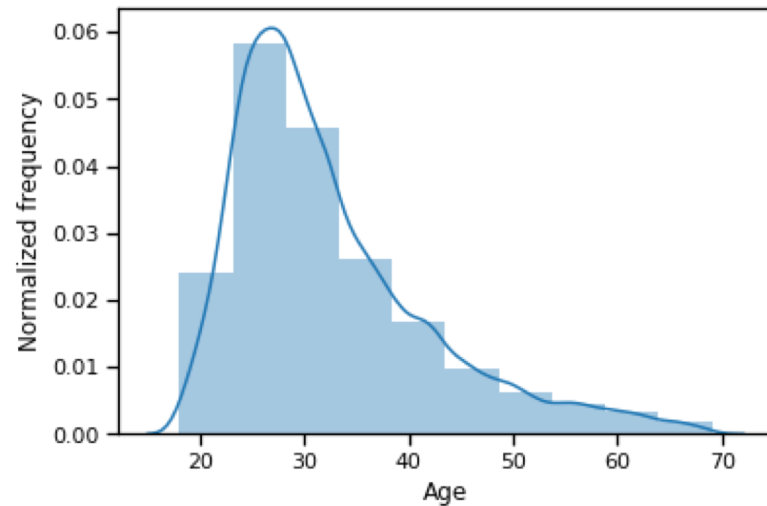
	age
0	22
1	38
2	23
3	29
4	29

	age
count	45952.000000
mean	32.512121
std	9.569849
min	18.000000
25%	26.000000
50%	30.000000
75%	37.000000
max	69.000000

How can we use to visualize the data?

`dataframe.describe()` prints some statistical information about the columns.

2. Plotting types: Histogram



```
sns.distplot(df, bins = 10)

plt.xlabel('Age')
plt.ylabel('Normalized frequency')

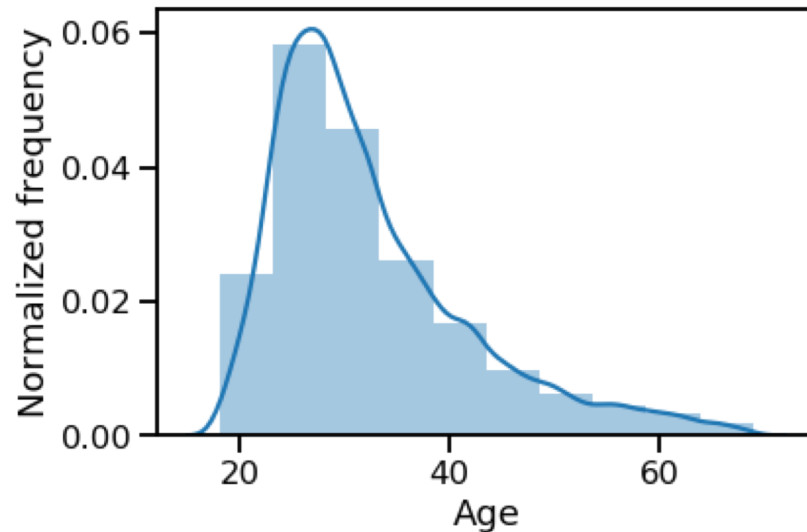
plt.show()
plt.clf()
```

*Note: seaborn uses distplot
whereas matplotlib uses hist*

Which statistical metric is easy to see with this plot?

Mean, median, mode, interquartile range, standard deviation, or standard error

2. Plotting types: Histogram



Which statistical metric is easy to see with this plot?

Mean, median, mode, interquartile range, standard deviation, or standard error

```
sns.set_context('talk')

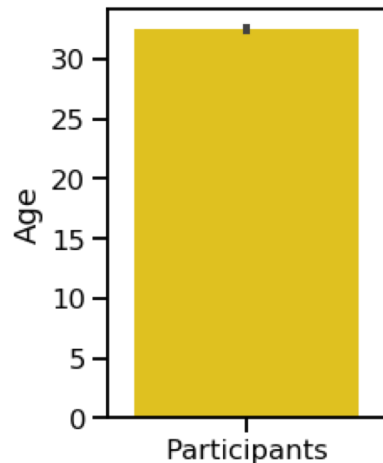
sns.distplot(df, bins = 10)

plt.xlabel('Age')
plt.ylabel('Normalized frequency')

plt.show()
plt.clf()
```

*Seaborn context can be
paper, notebook, talk, or
poster*

2. Plotting types: Bar plot



A bar plot shows the mean. For seaborn, the error bar is by default the standard deviation.

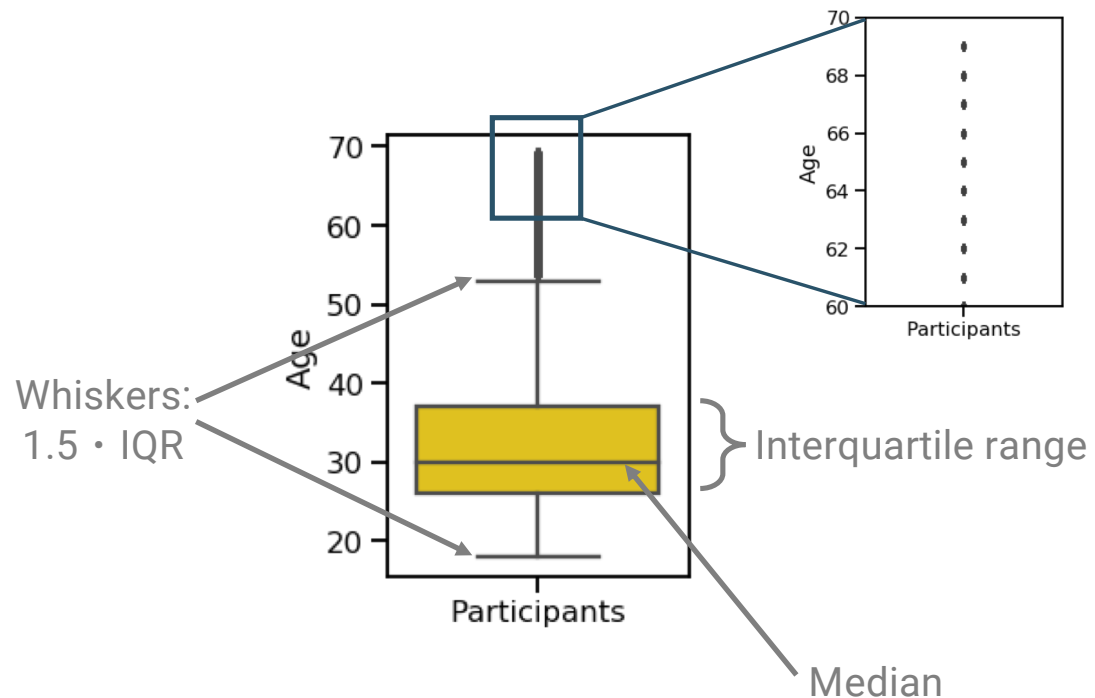
Which statistical metrics are easy to see with this plot?

Mean, median, mode, interquartile range, standard deviation, or standard error

```
fig, ax = plt.subplots(figsize = (3, 4.5))  
  
sns.barplot(data = df)  
  
ax.set_xticklabels(labels=['Participants'])  
plt.ylabel('Age')  
  
plt.show()  
plt.close()
```

For small datasets, you could add the individual points with a swarm plot. This data set has over 40,000 points, so it is not possible here.

2. Plotting types: Box plot



Which statistical metrics are easy to see with this plot?

Mean, median, mode, interquartile range, standard deviation, or standard error

```
fig, ax = plt.subplots(figsize = (3, 4.5))

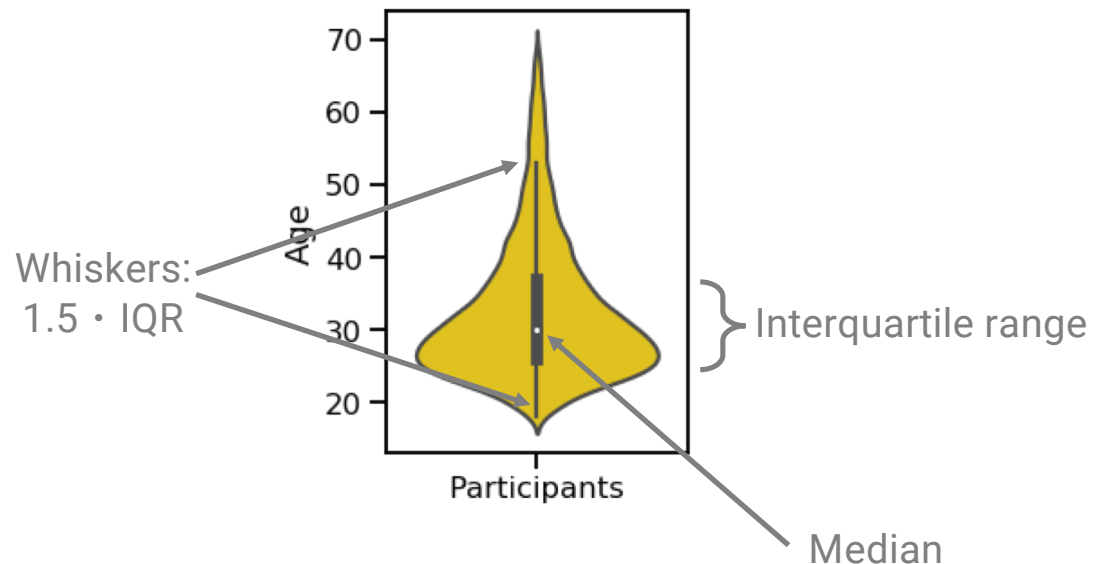
sns.boxplot(data = df)

ax.set_xticklabels(labels=['Participants'])
plt.ylabel('Age')

plt.show()
plt.close()
```

Outliers drawn as points

2. Plotting types: Violin plot



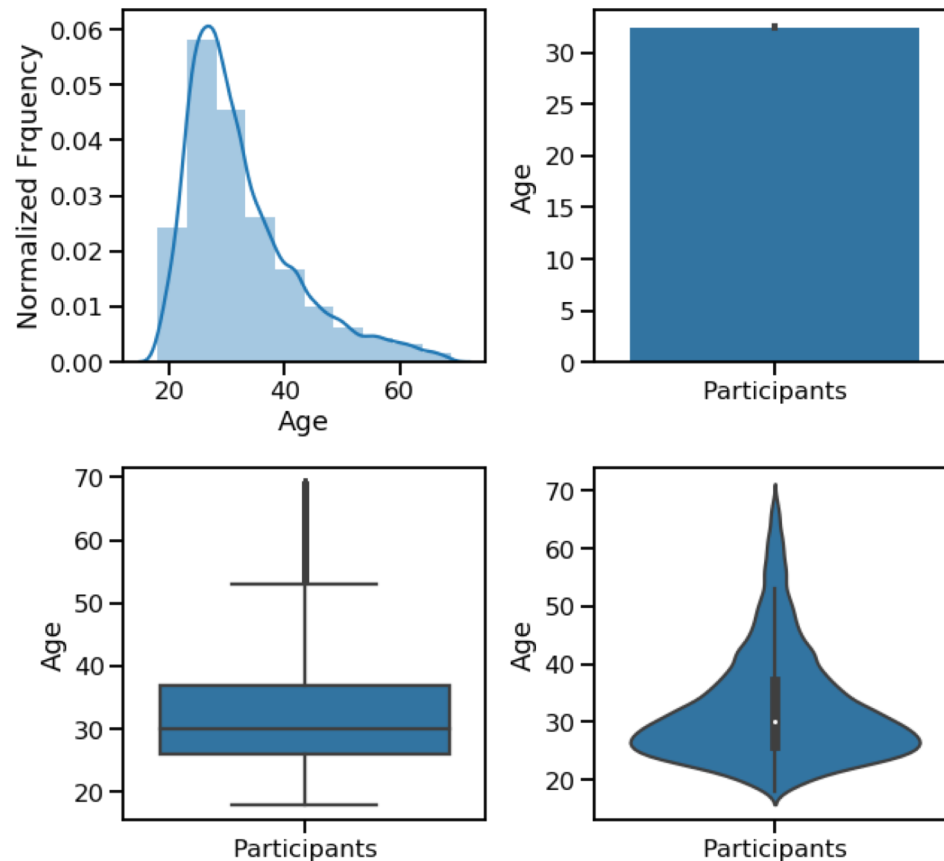
Which statistical metrics are easy to see with this plot?

Mean, median, mode, interquartile range, standard deviation, or standard error

```
fig, ax = plt.subplots(figsize = (3, 4.5))  
  
sns.violinplot(data = df)  
  
ax.set_xticklabels(labels=['Participants'])  
plt.ylabel('Age')  
  
plt.show()  
plt.close()
```

A violin plot combines features of the box plot and the histogram.

2. Plotting types: Combining the plots



```
fig, axs = plt.subplots(2, 2, figsize=(10, 10))

ax1 = sns.distplot(df, bins = 10, ax=axs[0,0])
ax1.set_xlabel('Age')
ax1.set_ylabel('Normalized Frequency')

ax2 = sns.barplot(data = df, ax=axs[0, 1])
ax2.set_ylabel('Age')
ax2.set_xticklabels(labels=['Participants'])

ax3 = sns.boxplot(data = df, ax=axs[1, 0])
ax3.set_ylabel('Age')
ax3.set_xticklabels(labels=['Participants'])

ax4 = sns.violinplot(data = df, ax=axs[1, 1])
ax4.set_ylabel('Age')
ax4.set_xticklabels(labels=['Participants'])

plt.subplots_adjust(wspace= 0.3, hspace = 0.3)

plt.show()
plt.clf()
```

3. Hypothetical problem: Set up

Data analysis problem solving:

- **Platform:** Jupyter notebook
- **Plotting tool:** seaborn
- **Problem:** We want to examine the total turnover number (TTN) of a new catalyst that we have made. We made a stock solution of the catalyst, and we tested the activity of this stock solution. In order to get proper error assessment, we conducted three replicates on at least 5 days. Now that we have this data, we want to analyze the data to understand more about the catalyst.

Would people prefer to keep using powerpoint or look at the jupyter notebook format?

3. Hypothetical problem: Importing and exploring data

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

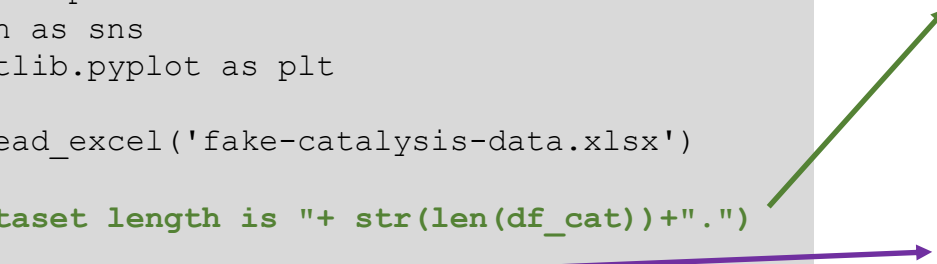
df_cat = pd.read_excel('fake-catalysis-data.xlsx')

print("The dataset length is " + str(len(df_cat)) + ".")


df_cat.head()

df_cat.describe()
```

The dataset length is 18.



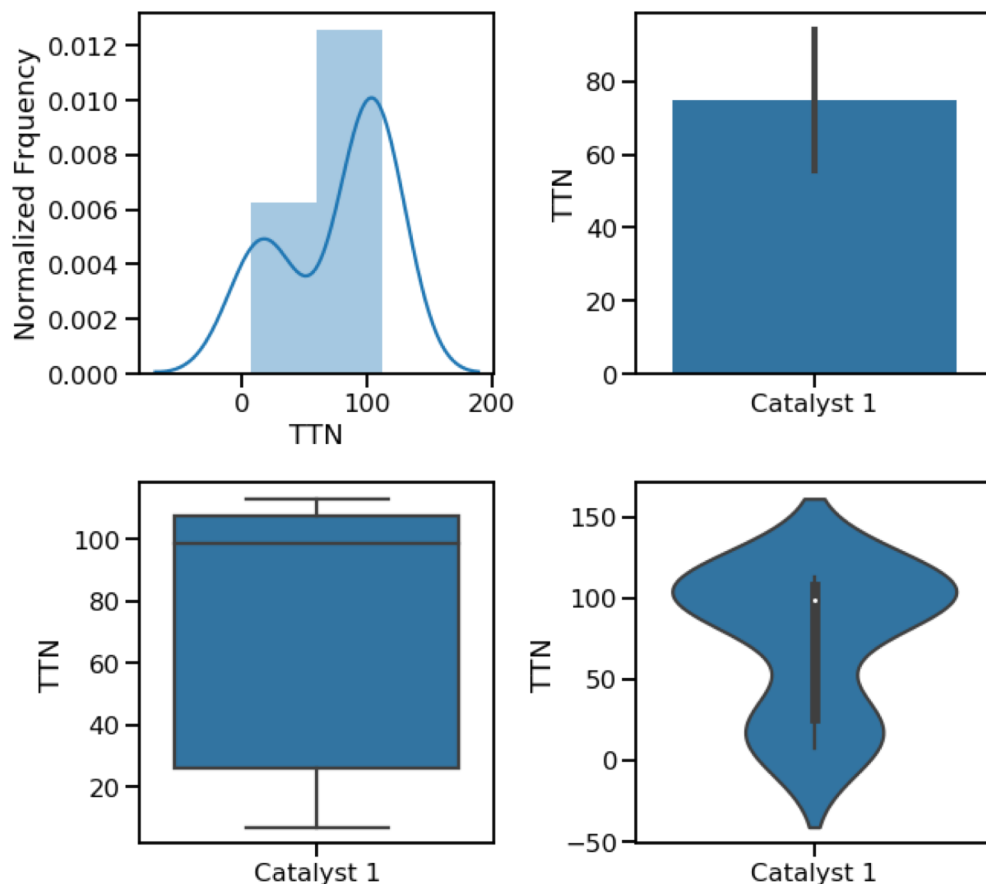
	Day	TTN
0	1	100
1	1	110
2	1	105
3	2	95
4	2	108



	Day	TTN
count	18.000000	18.000000
mean	3.500000	75.055556
std	1.757338	42.880782
min	1.000000	7.000000
25%	2.000000	26.250000
50%	3.500000	98.500000
75%	5.000000	107.250000
max	6.000000	113.000000

Note: this dataset is very small and I made it up, so we are using an excel file.

3. Hypothetical problem: Plotting the compiled data



It looks like there is something strange about the data...

```
fig, axs = plt.subplots(2, 2, figsize=(10, 10))

ax1 = sns.distplot(df_cat['TTN'], ax=axs[0,0])
ax1.set_xlabel('TTN')
ax1.set_ylabel('Normalized Frquency')

ax2 = sns.barplot(data = df_cat['TTN'],
                  ax=axs[0, 1])
ax2.set_ylabel('TTN')
ax2.set_xticklabels(labels=['Catalyst 1'])

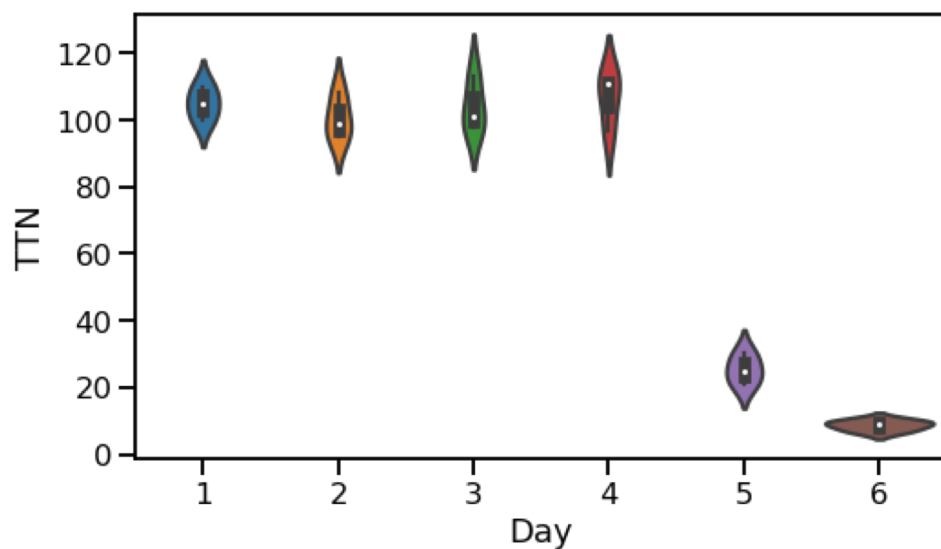
ax3 = sns.boxplot(data = df_cat['TTN'],
                  ax=axs[1, 0])
ax3.set_ylabel('TTN')
ax3.set_xticklabels(labels=['Catalyst 1'])

ax4 = sns.violinplot(data = df_cat['TTN'],
                    ax=axs[1, 1])
ax4.set_ylabel('TTN')
ax4.set_xticklabels(labels=['Catalyst 1'])

plt.subplots_adjust(wspace= 0.4, hspace = 0.3)

plt.show()
plt.clf()
```

3. Hypothetical problem: Looking at the data by day



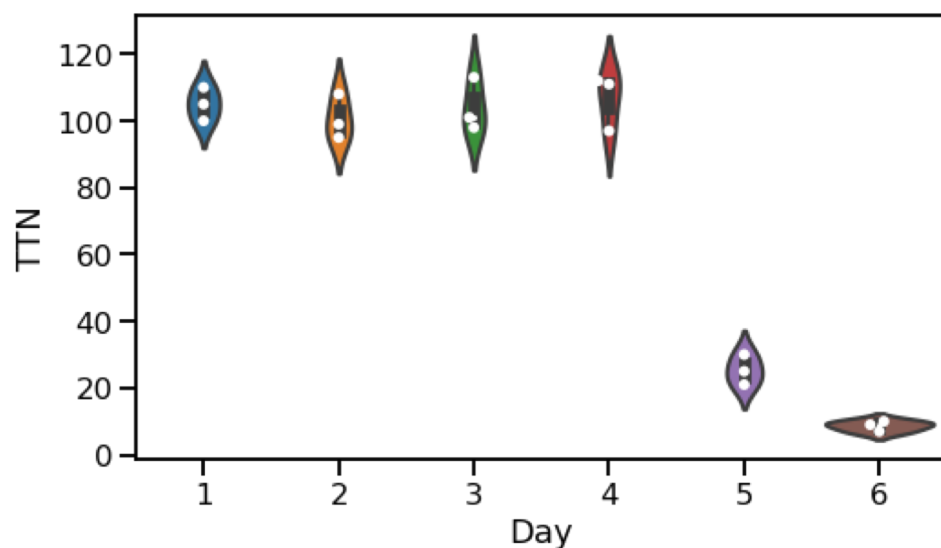
```
fig, ax = plt.subplots(figsize = (8, 4.5))

sns.violinplot(data = df_cat,
               y = df_cat['TTN'],
               x=df_cat['Day'])

plt.show()
plt.close()
```

Can we learn something about the catalyst based on this assessment?

3. Hypothetical problem: Looking at the data by day



Can we learn something about the catalyst based on this assessment?

```
fig, ax = plt.subplots(figsize = (8, 4.5))

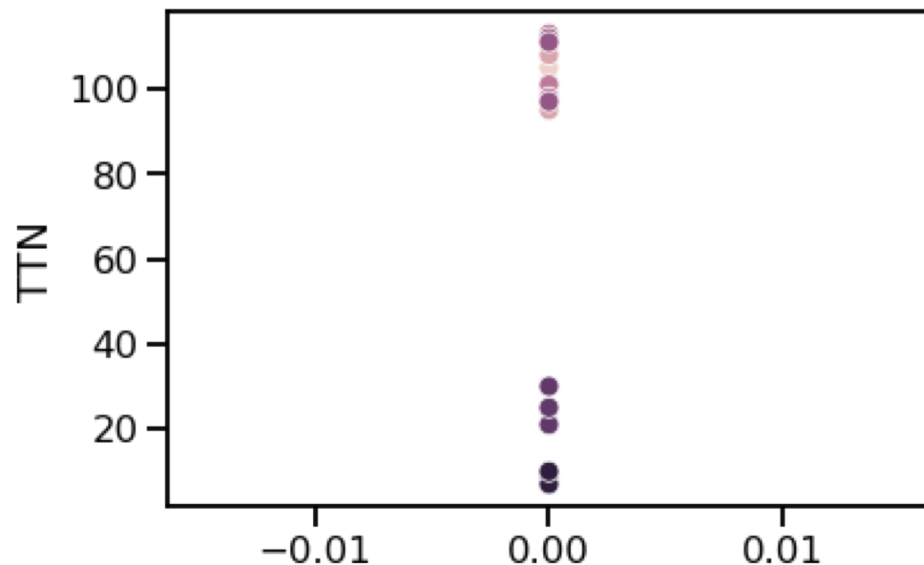
sns.violinplot(data = df_cat,
               y = df_cat['TTN'],
               x=df_cat['Day'])

sns.swarmplot(data = df_cat,
              y = df_cat['TTN'],
              x=df_cat['Day'],
              color='white', size = 6)

plt.show()
plt.close()
```

If your data set is small you can also create a swarm plot or overlay a swarm plot onto another type of plot (here violin + swarm).

3. Hypothetical problem: Looking at the data by day



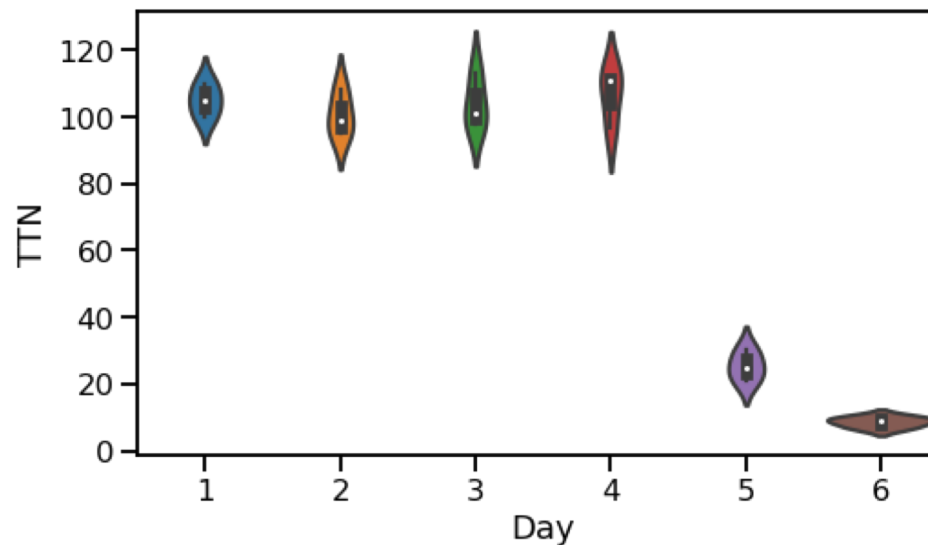
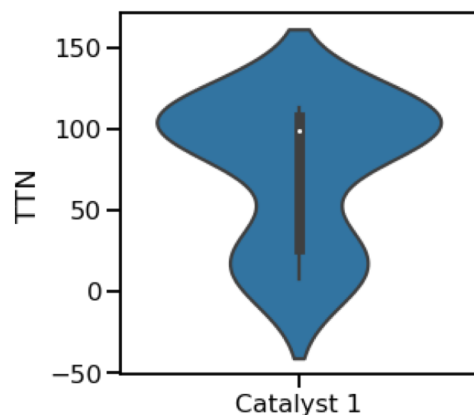
Can we learn something about the catalyst based on this assessment?

```
sns.scatterplot(data = df_cat,  
                x = [0]*len(df_cat),  
                y = df_cat['TTN'],  
                hue = df_cat['Day'],  
                legend= False)  
  
plt.show()  
plt.close()
```

Here 'x' is artificially set because we are only looking at one catalyst (column), normally 'x' would be the multiple columns.

Finally, you could also color code the data points according to day.

3. Hypothetical problem: Conclusions regarding data



Based on the plots above, it seems clear that the catalyst has stopped working well around day 5.

Three possible explanations:

- i) the catalyst stock solution has decomposed,
- ii) the substrate stock solution has decomposed, or
- iii) we changed something on day 5 and 6.

What could we do to test this hypothesis?

4. Questions

- i. Are violin plots always superior?

No. Why? Let's imagine that we have > 20 days to compare?

- ii. How can you make all of these graphs even more meaningful?

Add p-values – *outside of the bounds of this lecture, but food for thought*

- iii. Questions from you?