```
In [90]: %reset
```

Often, as a standard practice, we will reset any user defined variables to create a clean run using the %reset call.

## Tutorial 1: Step 1. Read in the "Viscosity_NeedsCleaning.csv" dataset and show it.

You will need to import some packages to use to do this.

In general you will always want to have available Numpy for array data, Pandas for series and dataframe data, and the plotting packages of Matplotlib and Seaborn. Anaconda distribution comes available with these installed by default. If you need to install, then use an install call on your promt or command line.

You then import the required libraries By convention, you import Numpy with a np alias, Pandas with a pd alias, etc.

```
In [97]: # Importing Necessary Libraries

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

With Pandas you can use the read_csv(path_to_file) to automatically read and parse your csv. The only required arguement is the path to the csv file as a string or path object. The file can be hosted locally on your computer or online. Documentation for this method can be found here at https://pandas.pydata.org/pandas-docs/dev/reference/api/pandas.read_csv.html.

In this example we'll use the Assignment 1 dataset on Viscosities, but then we will need to do a bit of data preparation using it,

There are many ways to read the dataset. Use the .read_csv() to read in the dataset and store it as a Dataframe object in the variable that we choose to name it.

Such as ViscosityNC= ViscosityNC = pd.read_csv('Viscosity_NeedsCleaning.csv')

Then show the data by calling it.

```
In [ ]:
```

```
In [ ]:
```

## Tutorial 1: Step 2. Look for data-type validity and Correct.

Let's check that the data type matches what you would want the data to be used for. What are the data types for ProcessA and Process B in the dataset? If the data type does not match what you want it to be, discuss what the data type is that you want and why, and the reasons that you find in the dataset for any datatype mismatch, and correct this.

You can use dataframe_name.info() to see the the datatypes.

Other attributes you can look at for the dataframe (df) include:

- type(df)
- len(df)
- df.shape
- dfs.ndim
- df.size
- df.columns
- df.index
- dfs.head()
- df.tail()
- df.describe()
- df.dtypes rds.dtypes

Convert to the datatype that you are interested in using.

Here, you want both Process A and Process B values to be float64 type data.

## Working with NAN: Not a Number

Working with NaN in datasets is easy. So, could convert the string 'A" at row 15 to a NaN and then convert:
ViscosityNC = ViscosityNC.replace('A', np.nan,inplace=True)

You could drop by row for NaNs: ViscosityNC= ViscosityNC.dropna()

Can drop by column for NaNs: ViscosityNC.dropna(subset=['ProcessA'],inplace=True)

Can coerce the conversion using ViscosityNC= ViscosityNC.apply(pd.to_numeric, errors='coerce')

In [20]:
```
# If you try to convert Process A from oject to float you get an error.
# ViscosityNC=ViscosityNC.astype({"ProcessA":float},errors='raise')
```

Check the datatypes in the dataframe.

In [ ]:
```
```

Can drop unwanted rows.

In [ ]:
```
```

In [ ]:
```
```

Check that the datatype is corrected.

```
In [ ]:
```

```
In [ ]:
```

Can reset the index using df.reset_index(drop=True, inplace=True)

```
In [ ]:
```

```
In [ ]:
```

## Tutorial 1: Step 3. Look for data validity and correct.

What do you think the limits of plausible data would be for Process A and Process B? Go ahead and clean the data based on deleting any nonsense values (say that a viscosity measurement of >=50 is not possible, and negative values are not possible).

You can use iloc, square brackets, .query, or other methods to filter or slide data.

Columns can be access by using their name in square brackers [] while rows can be access using their row (index) number and the .iloc property.Can u

use .loc for label inde; loc gets rows (and/or columns) with particular labels. xing

It is also possible to filter the dataframe based on the value in certain columns or rows. Filtering the values returns a new dataframe with just the values that meet the condition.

We will also learn about the .query method. A pandas Dataframe query is a way to query the columns of a Dataframe with a boolean expression. The .query( ,inplace=TRUE) returns no new dataframe and .query( ,inplace=FALSE) returns a new dataframe resulting from the provided query expression (FALSE is the deafult). The query expressions can be used with multiple con

So, df = df.query('0<= ProcessA < 50', inplace=False) would create or overwrite df as the filtered df.ditions.

```
In [ ]:
```

Check the df to verify that it is prepared for analysis.

## Tutorial 1: Step 4. Summarize and Visualize the data.

Use appropriate data numerical summaries and plots to describe and visualize the data.

Use df.describe() to get a numerical summary of the dataset.

Can use np.mean(df['columnname']) for mean

np.var; np.std(ddof=1 for sample standard deviation)

Can plot easily with seaborn, such as calling the seaborn boxplot for the dataframe.

```
In [ ]:
```

Show the data summary for Process A and Process B

df.describe()

Show a plot comparing Process A and Process B

sns.boxplot(df)

```
In [ ]:
```

```
In [ ]:
```

## You can set the figure and the axes to specify plots

Matplotlib graphs your data on Figures each of which can contain one or more Axes, an area where points can be specified in terms of x-y coordinates (or theta-r in a polar plot, x-y-z in a 3D plot, etc.).

Here, we are asking for a figure to be created (fig) and creating 2 Axes, and asking to plot subplots (here, 2 of them with a shared x axis)

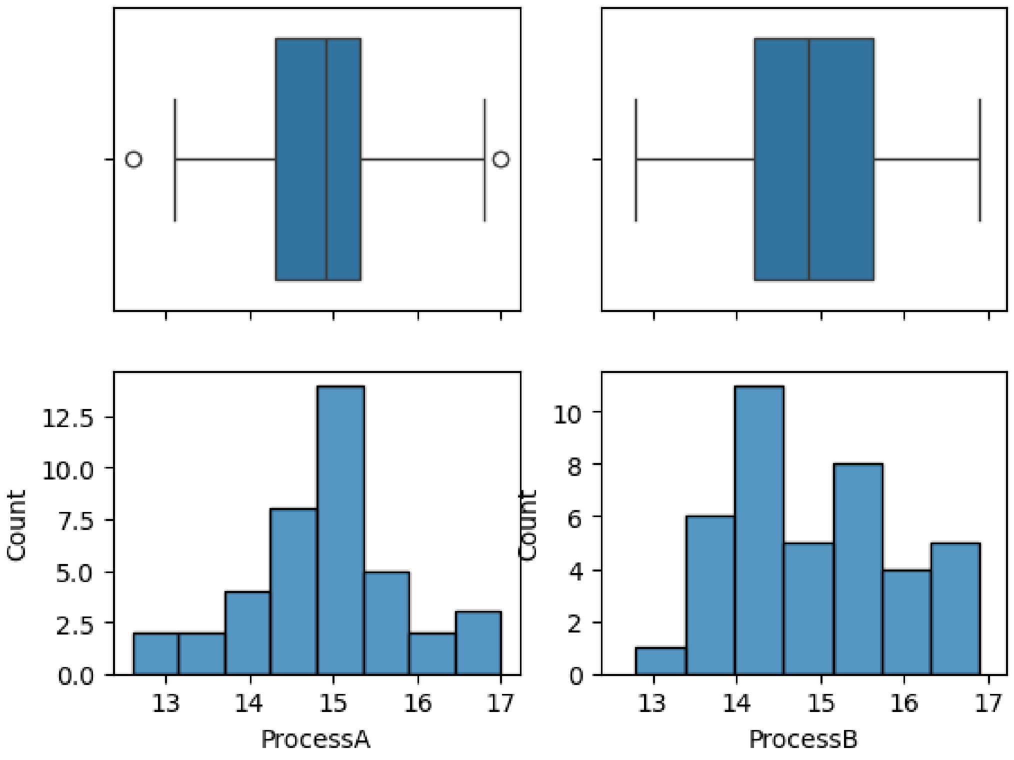
fig,(ax1,ax2)=plt.subplots(2,sharex=True,gridspec_kw={"height_ratios":(0.15,0.85)}) sns.boxplot(x=CompStrength["CompStrength"], ax=ax1) sns.histplot(data=CompStrength,ax=ax2)

Here we are asking for a 2 by 2 subplots

fig, axes = plt.subplots(3, 4, sharex=True, figsize=(16,8)) fig.suptitle('3 rows x 4 columns axes withdataa' )

```
In [86]: fig,axes =plt.subplots(2,2,sharex=True)
         sns.boxplot(x=ViscosityNC["ProcessA"], ax=axes[0,0])
         sns.boxplot(x=ViscosityNC["ProcessB"], ax=axes[0,1])
         sns.histplot(data=ViscosityNC["ProcessA"],ax=axes[1,0])
         sns.histplot(data=ViscosityNC["ProcessB"],ax=axes[1,1])
```

```
Out[86]: <Axes: xlabel='ProcessB', ylabel='Count'>
```


Count
12.5
10.0
7.5
5.0
2.5
0.0
13
14
15
16
17
ProcessA
Count
10
8
6
4
2
0
13
14
15
16
17
ProcessB

```
In [ ]:
```