In [297…
```
%reset
```

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

Make sure that you have sected y or n to proceed.

Tutorial 1: Step 1. Read in the "MedicineCap_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 [368…
```
# 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 MedicineCap_NeedsCleaning, 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, here MC (we are naming the dataframe or df objet as MC). You need to have the dataset uploaded to the coding area for this to work.

Read in the dataset and name it, such as MC = pd.read_csv('MedicineCap_NeedsCleaning.csv')

In [304…
```

```

Then you can see it or work with it as a named df.

See the dataset by calling the df, here MC

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 MachineA and MachineB 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

Convert to the datatype that you are interested in using.

Here, you want both MachineA and MachineB values to be float64 type data.

Pandas dtype	Python type	NumPy type	Usage
object	str or mixed	string_, unicode_, mixed types	Text or mixed numeric and non-numeric values
int64	int	int_, int8, int16, int32, int64, uint8, uint16, uint32, uint64	Integer numbers
float64	float	float_, float16, float32, float64	Floating point numbers
bool	bool	bool_	True/False values
datetime64	NA	datetime64[ns]	Date and time values
timedelta[ns]	NA	NA	Differences between two datetimes
category	NA	NA	Finite list of text values

Check for your dataframes datatypes

In [ ]:

```

```

Change the MC['MachineA'] variable to a float64

You can specify the column variable by using the df["column_name"]

Can coerce the conversion of the whole df into being a float number using MC= MC.apply(pd.to_numeric, errors='coerce')

In [315…

```
# If you try to convert Process A from object to float using DF.astype you get an error.
# MC=MC.astype({"MachineA":float},errors='raise')
```

Creating or overwriting dataframe names

It is often good practice to create a new df name for steps to keep track of updates to your df, or you can overwrite assign your df name keeping it as is, or even overwrite the called df command using (inplace=True) to update the df (without creating a new variable name or keeping the old varaible name with df =)

Here we will create a new df with a new name at each step.

Create a new df name and then chang your df to numbers; Use the df.apply(pd.to_numeric, errors='coerce') method to coerce your dataframe to be updated to numbers.

So here, MC_A = MC.apply(pd.to_numeric, errors='coerce') would create an updated df called MC_A

Coercing it changes any problems with converting to numbers into NaNs.

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.apply.html

https://pandas.pydata.org/docs/reference/api/pandas.to_numeric.html

```
In [ ]:
```

Can check to see if the object to number conversion worked.

```
In [ ]:
```

Working with NAN: Not a Number

Working with NaN in datasets is easy. So changing problem data to NaNs is often a useful step.

So, could convert the string 'A" at row 15 to a NaN and then convert:
MC = MC.replace('A', np.nan,inplace=True)

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

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

Since we coerced the df and replaced problems with NaNs, we can drop these using the df.dropna() command, but want to specify that you wnat to drop NaNs by row (index, or 0), and that you only wnat to drop the row if all of the values are NaNs.

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html

Drop unwanted rows

MC_A.dropna(axis = 'index', how = 'all')

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

Reset the index values

Since you have filtered by rows, you can reset the index values so that the df has ordered rows.

such as: use MC_B.reset_index(drop=True)

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.reset_index.html

```
In [ ]:
```

Tutorial 1: Step 3. Select one variable to show and then look for data validity and correct.

What do you think the limits of plausible data would be for MachineA and MachineB? Go ahead and clean the data based on deleting any nonsense values (say that a force 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 exprafult). The query expressions can be used with multiple conditions.

Show only Machine A data and then show Machine A data's mean

The column variable as a Series can be selected using ['variable_name'] or ["variable_name"] from the Dataframe, so here MC_C["MachineA"]

Statistical summary information can be found using numpys (for arrays or series) or pandas (for dataframes):

- np.mean(Array)
- np.mean(Array,ddof=1)
- np.var(Array)
- np.median(Array)
- DF.describe()
- DF.mean()

```
In [ ]:
```

```
In [ ]:
```

Correct data validity

You can use the DF.query function for the dataframe to filter the dataframe based on your specifications, here greater or equal to 0 and less than 50.

The query string will accept & and | operators for the Bollean and and or.

Such as:

MC_D = MC_C.query('0<= MachineA < 50 | MachineA.isnull()')

MC_D = MC_D.query('0<= MachineB < 50 | MachineB.isnull()')

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.query.html

```
In [ ]:
```

Reset the index values and then you will have a prepared dataset for MedicineCap.

```
In [22]: # rest index
```

```
In [ ]:
```

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 of array or df Series (column)

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

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

Show the summary data for the df

```
In [ ]:
```

Show the boxplots

```
In [ ]:
```

Bonus Information

Take your Tutorial 1 Quiz

You can print from flie---> print to pdf if export is not working

If you are using an old version of python or jupyter you may want to update your version.

Also update if your plots are not working with seaborn, or use matplotlib.

Try ChatGPT and Ananconda Assistant for AI assistance with code

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)

an example with CompStrength:

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 [34]:
```
fig,axes =plt.subplots(2,2,sharex=True)
sns.boxplot(x=MC_clean["MachineA"], ax=axes[0,0])
sns.boxplot(x=MC_clean["MachineB"], ax=axes[0,1])
sns.histplot(data=MC_clean["MachineA"],ax=axes[1,0])
sns.histplot(data=MC_clean["MachineB"],ax=axes[1,1])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[34], line 1
----> 1 fig,axes =plt.subplots(2,2,sharex=True)
      2 sns.boxplot(x=MC_clean["MachineA"], ax=axes[0,0])
      3 sns.boxplot(x=MC_clean["MachineB"], ax=axes[0,1])

NameError: name 'plt' is not defined
```

In [ ]:

In [ ]: