Statistics validations

April 16, 2025

1 Statistic checks of the collected data

[ ]:
```
import numpy as np
from scipy import stats
from matplotlib import pyplot as plt

import pandas as pd
from scipy.signal import medfilt

%matplotlib inline
alpha = 0.05
```

1.0.1 Verifying if the data is normally distribuated and has similar variance

Loading the test dataset of 3 Aspect Ratios. Each dataset contain 10 experiments. We will verify the normallity and the variance on the 10 values x3 datasets.

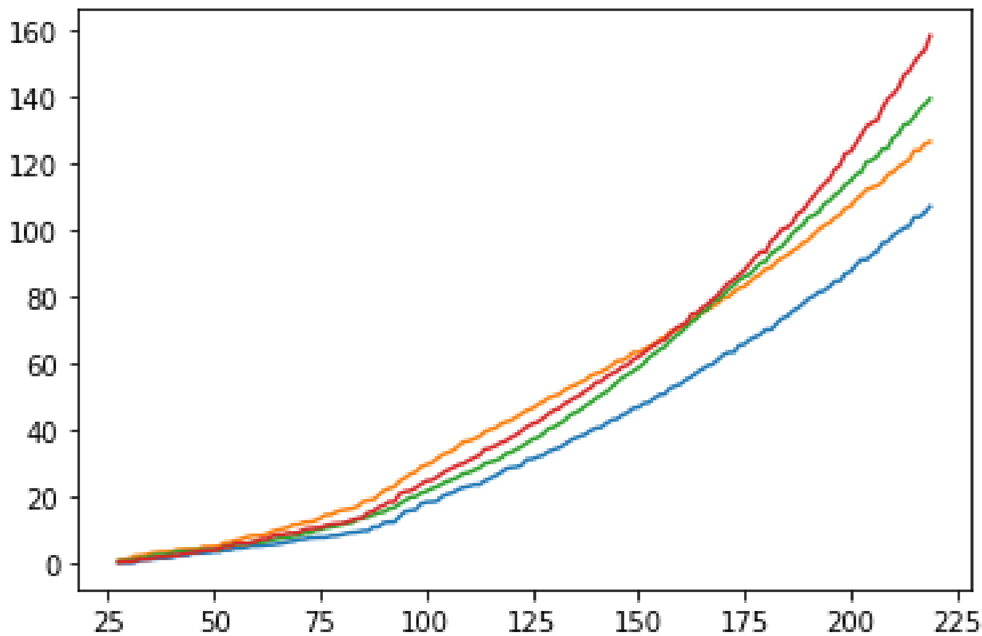
[ ]:
```
def import_CSV(fileslist):

    expNbr = len(fileslist)
    expData = []
    for i in range(expNbr):
        expData.append(pd.read_csv(fileslist[i]))

    expDataSize = list(len(df['force'].values) for df in expData)
    dataCrop = min(expDataSize)

    #Clean data
    return_table = dict()
    medKernel = 5
    for i,data in enumerate(expData):
        data['force'] = medfilt(data['force'], kernel_size=medKernel)  # Apply␣
↪median filter
        data['force'] = data['force'][0:-medKernel]

    dataCrop -= medKernel

    #Extract indicators
```

```python
    return_table["depth"] = expData[0]['steps'].values[:dataCrop]
    for i in range(expNbr):
        return_table["exp_" + str(i)] = expData[i]['force'].values[:dataCrop]
        # print(np.max(expData[i]['steps'].iloc[-1]))

    return pd.DataFrame(return_table)
```

[ ]:
```python
# Aspect ratio 1
group_1_files =["./expData/dataTest/AR1/FR2_0.csv",
                "./expData/dataTest/AR1/FR2_1.csv",
                "./expData/dataTest/AR1/FR2_2.csv",
                "./expData/dataTest/AR1/FR2_3.csv",
                "./expData/dataTest/AR1/FR2_4.csv",
                "./expData/dataTest/AR1/FR2_5.csv",
                "./expData/dataTest/AR1/FR2_6.csv",
                "./expData/dataTest/AR1/FR2_7.csv",
                "./expData/dataTest/AR1/FR2_8.csv",
                "./expData/dataTest/AR1/FR2_9.csv"]

# Aspect ratio 2
group_2_files =["./expData/dataTest/AR25/FR5_0.csv",
                "./expData/dataTest/AR25/FR5_1.csv",
                "./expData/dataTest/AR25/FR5_2.csv",
                "./expData/dataTest/AR25/FR5_3.csv",
                "./expData/dataTest/AR25/FR5_4.csv",
                "./expData/dataTest/AR25/FR5_5.csv",
                "./expData/dataTest/AR25/FR5_6.csv",
                "./expData/dataTest/AR25/FR5_7.csv",
                "./expData/dataTest/AR25/FR5_8.csv",
                "./expData/dataTest/AR25/FR5_9.csv"]

# Aspect ratio 3
group_3_files =["./expData/dataTest/AR4/FR8_0.csv",
                "./expData/dataTest/AR4/FR8_1.csv",
                "./expData/dataTest/AR4/FR8_2.csv",
                "./expData/dataTest/AR4/FR8_3.csv",
                "./expData/dataTest/AR4/FR8_4.csv",
                "./expData/dataTest/AR4/FR8_5.csv",
                "./expData/dataTest/AR4/FR8_6.csv",
                "./expData/dataTest/AR4/FR8_7.csv",
                "./expData/dataTest/AR4/FR8_8.csv",
                "./expData/dataTest/AR4/FR8_9.csv"]

group_1 = import_CSV(group_1_files)
group_2 = import_CSV(group_2_files)
group_3 = import_CSV(group_3_files)
```

```python
plt.plot(group_2["depth"],group_2["exp_1"])
plt.plot(group_2["depth"],group_2["exp_2"])
plt.plot(group_2["depth"],group_2["exp_3"])
plt.plot(group_2["depth"],group_2["exp_4"])
```

[ ]: [<matplotlib.lines.Line2D at 0x237931f1f60>]

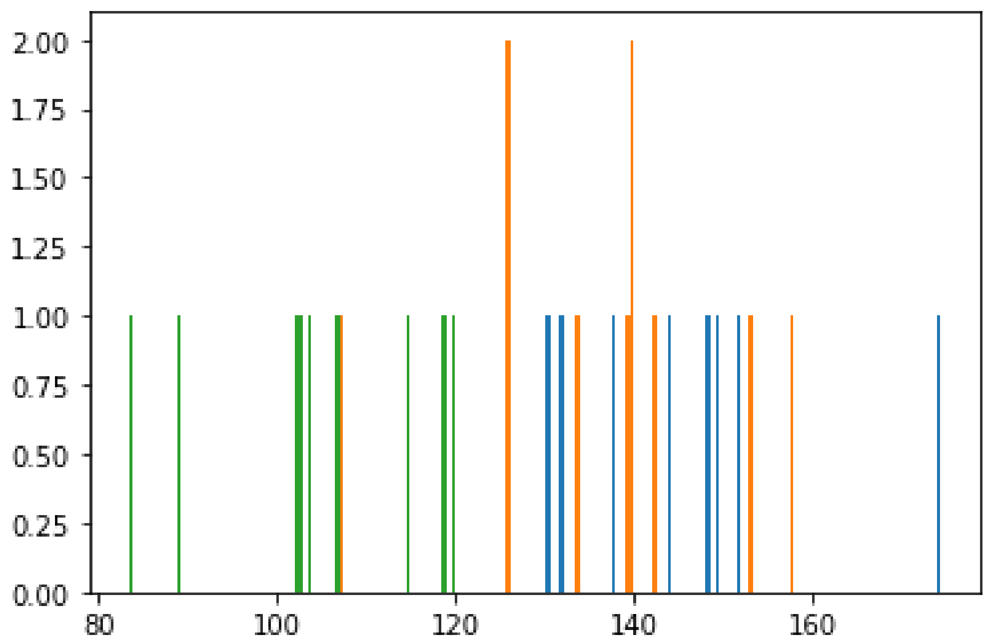


Extract the last depth values

```python
group_1_sub, group_2_sub, group_3_sub = [], [], []
for i in range(10):
    group_1_sub.append(group_1["exp_"+str(i)].iloc[-1])
    group_2_sub.append(group_2["exp_"+str(i)].iloc[-1])
    group_3_sub.append(group_3["exp_"+str(i)].iloc[-1])
```

```python
print(group_1_sub)
plt.hist(group_1_sub,bins=100);
plt.hist(group_2_sub,bins=100);
plt.hist(group_3_sub,bins=100);
```

```
[125.81, 125.9, 174.4, 137.9, 152.03, 130.57, 143.83, 148.58, 149.23, 132.09]
```



[ ]:
```
alpha = 0.05

for group_x in [group_1_sub,group_2_sub, group_3_sub]:
    #Check the normality
    shapiro_test = stats.shapiro(group_x)
    if(shapiro_test.pvalue > alpha):
        print("  Normal distribution accepted" + f" ({shapiro_test.pvalue})")
    else:
        print("  Normal distribution rejected")

#Check the variance
levene = stats.levene(group_1_sub, group_2_sub,group_3_sub, center='mean')
if(levene.pvalue > alpha):
    print("  Homegeinity of variance accepted" + f" ({levene.pvalue})")
else:
    print("  Homegeinity of variance rejected")
```

```
 Normal distribution accepted (0.2532499204145696)
 Normal distribution accepted (0.6629100671090086)
 Normal distribution accepted (0.4633562895813254)
 Homegeinity of variance accepted (0.7233546183413981)
```

1.0.2 Appliying T-Test on the data

Once the conditions are verifyed, we can process a t-test on the vertical and horizontal force data. Comparing the data to AR1, which is the reference.

1.0.3 Figure 1: Vertical penetration AR1 AR2.5 AR4 AR1asym

[ ]:
```
group_AR1 = [7.0503125, 9.3308125, 7.895068750000001]
group_AR1_asym = [10.5254625, 10.33055, 9.971512500000003]
group_AR25 = [4.971775000000001, 4.30201875, 4.914106250000001]
group_AR4 = [3.1286, 2.7672000000000008, 2.5889250000000006]

print("AR1 vs AR1 asym")
result = stats.ttest_ind(group_AR1, group_AR1_asym, alternative='two-sided')
if result.pvalue > alpha:
    print("H0: The two independent samples have identical average (expected)␣
↪values."+ f" ({result.pvalue})")
else:
    print("H1: The means of the distributions underlying the samples are␣
↪unequal."+ f" ({result.pvalue})")

print("AR1 vs AR25")
result = stats.ttest_ind(group_AR1, group_AR25, alternative='two-sided')
if result.pvalue > alpha:
    print("H0: The two independent samples have identical average (expected)␣
↪values."+ f" ({result.pvalue})")
else:
    print("H1: The means of the distributions underlying the samples are␣
↪unequal."+ f" ({result.pvalue})")

print("AR1 vs AR4")
result = stats.ttest_ind(group_AR1, group_AR4, alternative='two-sided')
if result.pvalue > alpha:
    print("H0: The two independent samples have identical average (expected)␣
↪values."+ f" ({result.pvalue})")
else:
    print("H1: The means of the distributions underlying the samples are␣
↪unequal."+ f" ({result.pvalue})")
```

```
AR1 vs AR1 asym
H1: The means of the distributions underlying the samples are unequal.
(0.03330004605113471)
AR1 vs AR25
H1: The means of the distributions underlying the samples are unequal.
(0.008592803303155155)
AR1 vs AR4
H1: The means of the distributions underlying the samples are unequal.
(0.0015366260089722042)
```

1.0.4 Figure 2: Horizontal penetration

[ ]:
```
depths = [120,220,320]


group_AR1 = [[4.656375000000001, 4.0143625, 4.341843750000001, 3.902825, 2.
↪1696125, 2.61651875],
             [6.069131250000001, 3.4680687500000005, 3.41640625, 3.
↪8237250000000005],
              [15.288543749999999, 5.549837500000001, 14.258000000000001, 8.
↪31511875, 9.288775000000001, 8.0668375]]
group_AR25 = [[3.9430625, 3.5901812500000005, 4.16218125, 3.05588125, 2.
↪34490625, 2.3912750000000003, 2.8868062500000002],
             [5.346606250000001, 3.1081812500000003, 2.5341312499999997, 3.
↪17195],
              [14.23845625, 6.3012625, 4.61883125, 5.2333, 6.114218750000001]]
group_AR4 = [[3.9067937500000003, 4.723625, 5.014100000000001, 2.
↪9722375000000003, 2.1194, 2.9676125000000004, 2.2592125000000003],
            [6.874543750000001, 4.5127, 4.758018750000001, 4.5131625, 7.
↪062275000000001, 3.2807875000000006, 3.105975, 3.7428687500000004, 6.
↪0931625, 3.23635, 3.3171625000000002, 3.6472562500000003],
            [11.8154875, 3.03289375, 4.61306875, 3.9226687500000006]]

for i,d in enumerate(depths):
    print("AR1 vs AR25 "+ str(d))
    result = stats.ttest_ind(group_AR1[i], group_AR25[i],␣
↪alternative='two-sided',equal_var=False)
    if result.pvalue > alpha:
        print("\tH0: The two independent samples have identical average␣
↪(expected) values.")
    else:
        print("\tH1: The means of the distributions underlying the samples are␣
↪unequal.")
    print("\t",result.pvalue)
    print("AR1 vs AR4 "+ str(d))
    result = stats.ttest_ind(group_AR1[i], group_AR4[i],␣
↪alternative='two-sided',equal_var=False)
    if result.pvalue > alpha:
        print("\tH0: The two independent samples have identical average␣
↪(expected) values.")
    else:
        print("\tH1: The means of the distributions underlying the samples are␣
↪unequal.")
    print("\t",result.pvalue)
    print("----------")
```

```
AR1 vs AR25 120
```

```
        H0: The two independent samples have identical average (expected)
values.
         0.4122115792185961
AR1 vs AR4 120
        H0: The two independent samples have identical average (expected)
values.
         0.7507136357571126
----------
AR1 vs AR25 220
        H0: The two independent samples have identical average (expected)
values.
         0.48738207953915624
AR1 vs AR4 220
        H0: The two independent samples have identical average (expected)
values.
         0.6888427921287368
----------
AR1 vs AR25 320
        H0: The two independent samples have identical average (expected)
values.
         0.261647353605683
AR1 vs AR4 320
        H0: The two independent samples have identical average (expected)
values.
         0.14148466394979073
----------
```

Compare asymetric orientation

[ ]:
```
group_AR1 = [4.656375000000001, 4.0143625, 4.341843750000001, 3.902825, 2.
 ↪1696125, 2.61651875]
group_AR1sym_UP = [4.76405625, 4.827481250000001, 4.6944125, 3.
 ↪0783750000000003, 3.890225, 2.8605, 2.69065625, 2.9896625000000006, 2.
 ↪9456125, 2.0453437500000002]
group_AR1sym_DOWN = [2.8477625, 1.930225, 1.8477125]

print("AR1 vs DOWN")
result = stats.ttest_ind(group_AR1, group_AR1sym_DOWN,␣
 ↪alternative='two-sided',equal_var=False)
if result.pvalue > alpha:
    print("\tH0: The two independent samples have identical average (expected)␣
 ↪values. " + str(result.pvalue))
else:
    print("\tH1: The means of the distributions underlying the samples are␣
 ↪unequal. " + str(result.pvalue))
print("AR1 vs UP")
```

```
result = stats.ttest_ind(group_AR1, group_AR1sym_UP,␣
 ↪alternative='two-sided',equal_var=False)
if result.pvalue > alpha:
    print("\tH0: The two indepependent samples have identical average (expected)␣
 ↪values. " + str(result.pvalue))
else:
    print("\tH1: The means of the distributions underlying the samples are␣
 ↪unequal. "  + str(result.pvalue))


print("UP vs DOWN")
result = stats.ttest_ind(group_AR1sym_UP,group_AR1sym_DOWN,␣
 ↪alternative='two-sided',equal_var=False)
if result.pvalue > alpha:
    print("\tH0: The two indepependent samples have identical average (expected)␣
 ↪values. " + str(result.pvalue))
else:
    print("\tH1: The means of the distributions underlying the samples are␣
 ↪unequal. " + str(result.pvalue))
```

```
AR1 vs DOWN
        H1: The means of the distributions underlying the samples are unequal.
0.030981644294252115
AR1 vs UP
        H0: The two indepependent samples have identical average (expected)
values. 0.7926338984539102
UP vs DOWN
        H1: The means of the distributions underlying the samples are unequal.
0.02800861714769658
```

[ ]:
```
group_AR1 = [7.293575000000001, 9.727518750000002, 8.198150000000002]
group_AR1_AIR = [9.451316367599926, 6.929253867599925, 7.792341367599924, 7.
 ↪682916367599926]

print("AR1 vs AIR")
result = stats.ttest_ind(group_AR1, group_AR1_AIR,␣
 ↪alternative='two-sided',equal_var=False)
if result.pvalue > alpha:
    print("\tH0: The two independent samples have identical average (expected)␣
 ↪values. " + str(result.pvalue))
else:
    print("\tH1: The means of the distributions underlying the samples are␣
 ↪unequal. " + str(result.pvalue))
```

```
AR1 vs AIR
        H0: The two independent samples have identical average (expected)
values. 0.6440001440168176
```

[ ]:
```
group_AR1 = [16.491016950000002, 18.913584050000004, 9.55505235, 11.
↪905008650000001]
group_AR1_AIR = [6.0289506, 4.17028810000000005]

print("AR1 vs AIR")
result = stats.ttest_ind(group_AR1, group_AR1_AIR,␣
↪alternative='two-sided',equal_var=False)
if result.pvalue > alpha:
    print("\tH0: The two independent samples have identical average (expected)␣
↪values. " + str(result.pvalue))
else:
    print("\tH1: The means of the distributions underlying the samples are␣
↪unequal. " + str(result.pvalue))
```

```
AR1 vs AIR
        H1: The means of the distributions underlying the samples are unequal.
0.018583642210370174
```