Statistics validations

August 3, 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.

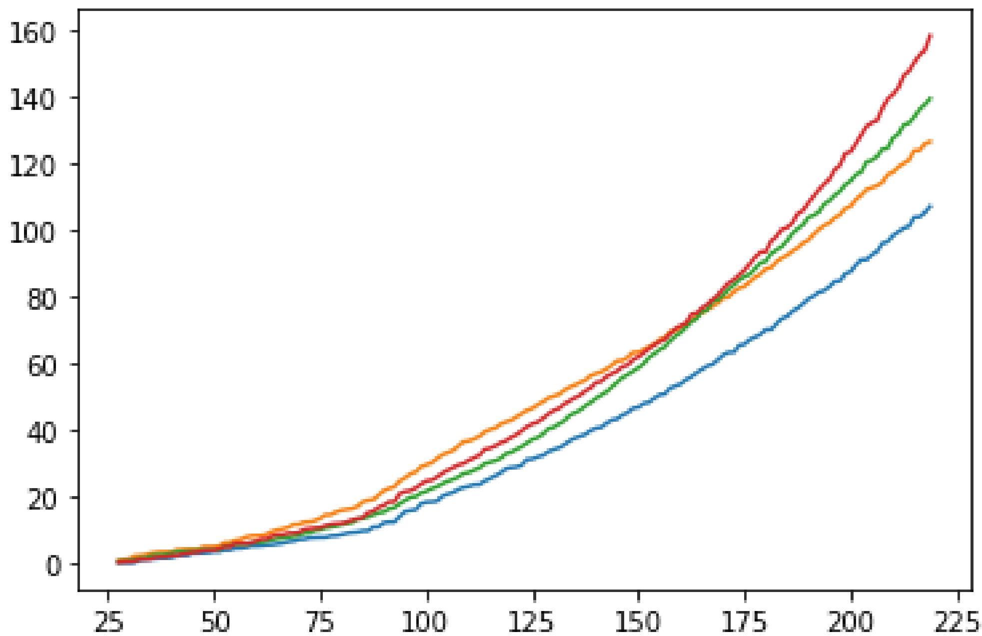
[ ]:
```
def import_CSV(fileslist):

    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
```

```
    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)
```

[ ]:
```
# 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 0x29ed9a88a30>]

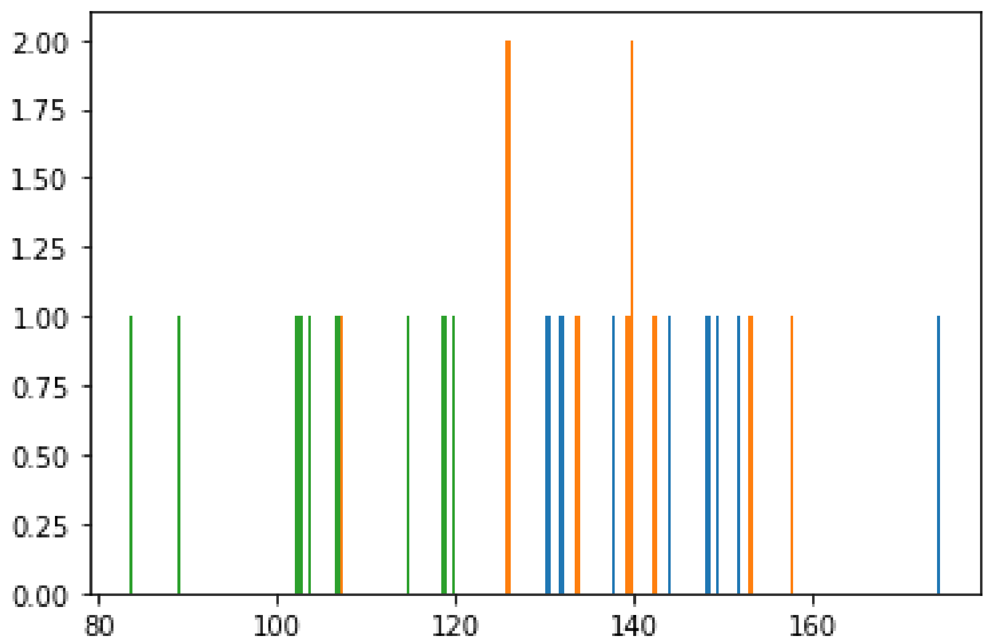


Extract the last depth values

[ ]:
```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 = [[4.478293750000001, 3.85711875, 4.165350000000001, 3.7627, 2.
↪0638625000000004, 2.5025312499999997],
             [5.79563125, 3.2915750000000004, 3.23479375, 3.6237749999999997],
              [14.488743750000001, 5.205331250000001, 13.42079375, 7.6122, 8.
↪576387500000001, 7.5819624999999995]]
group_AR25 = [[3.7700125, 3.4090937500000003, 3.9712187500000002, 2.94235, 2.
↪2349124999999996, 2.28149375, 2.7633875000000003],
             [5.092081249999999, 2.9481375, 2.3913375, 3.0161249999999993],
            [13.544818750000001, 5.876668749999999, 4.24003125, 4.
↪8320187500000005, 5.6397125]]
group_AR4 = [[3.7211749999999997, 4.508993749999999, 4.787049999999999, 2.
↪8526875, 2.0131437500000002, 2.841612499999999, 2.1451625],
            [6.5362937500000005, 4.2832375, 4.519093750000001, 4.
↪265725000000001, 6.7435, 3.0842750000000003, 2.9183250000000003, 3.
↪5413937500000006, 5.81105, 3.0568437499999996, 3.130125, 3.4471562500000004],
            [11.30195, 2.8212375000000005, 4.3357375000000005, 3.
↪7234812500000003]]

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("----------")
```

```
print("Comparing same tips at different depths")
print("AR1 1 vs 2")
result = stats.ttest_ind(group_AR1[0], group_AR1[1],␣
 ↪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 2 vs 3")
result = stats.ttest_ind(group_AR1[1], group_AR1[2],␣
 ↪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("----------")

print("AR25 1 vs 2")
result = stats.ttest_ind(group_AR25[0], group_AR25[1],␣
 ↪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("AR25 2 vs 3")
result = stats.ttest_ind(group_AR25[1], group_AR25[2],␣
 ↪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("----------")

print("AR4 1 vs 2")
```

```
result = stats.ttest_ind(group_AR4[0], group_AR4[1],␣
↪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("AR4 2 vs 3")
result = stats.ttest_ind(group_AR4[1], group_AR4[2],␣
↪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.
         0.3985513059435344
AR1 vs AR4 120
        H0: The two independent samples have identical average (expected)
values.
         0.7275363084218764
----------
AR1 vs AR25 220
        H0: The two independent samples have identical average (expected)
values.
         0.4904385586893891
AR1 vs AR4 220
        H0: The two independent samples have identical average (expected)
values.
         0.7028498930405361
----------
AR1 vs AR25 320
        H0: The two independent samples have identical average (expected)
values.
         0.27317721245430543
AR1 vs AR4 320
        H0: The two independent samples have identical average (expected)
values.
         0.15728336091132192
```

```
----------
Comparing same tips at different depths
AR1 1 vs 2
        H0: The two independent samples have identical average (expected)
values.
         0.5069928997827517
AR1 2 vs 3
        H1: The means of the distributions underlying the samples are unequal.
         0.012651297225629864
----------
AR25 1 vs 2
        H0: The two independent samples have identical average (expected)
values.
         0.6575911936308831
AR25 2 vs 3
        H0: The two independent samples have identical average (expected)
values.
         0.1137917055638882
----------
AR4 1 vs 2
        H0: The two independent samples have identical average (expected)
values.
         0.09900805583446837
AR4 2 vs 3
        H0: The two independent samples have identical average (expected)
values.
         0.5650679374616405
----------
```

Compare asymetric orientation

```
[ ]: group_AR1 = [4.478293750000001, 3.85711875, 4.165350000000001, 3.7627, 2.
     ↪0638625000000004, 2.5025312499999997]
     group_AR1sym_UP = [4.56710625, 4.630962500000001, 4.4870937500000005, 2.
     ↪95395625, 3.7320312499999995, 2.7121499999999994, 2.56056875, 2.
     ↪8769687499999996, 2.8425437500000004, 1.9451249999999998]
     group_AR1sym_DOWN = [2.740825, 1.840125, 1.76305625]

     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 indepppendent 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 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 DOWN
        H1: The means of the distributions underlying the samples are unequal.
0.03237051806118191
AR1 vs UP
        H0: The two indepppendent samples have identical average (expected)
values. 0.7820231117277838
UP vs DOWN
        H1: The means of the distributions underlying the samples are unequal.
0.03024950164915398
```

2 Active Strategies

2.0.1 Air blowing system

[ ]:
```
group_AR1 = [7.0503125, 9.3308125, 7.895068750000001]
group_AR1_AIR = [9.127122617599925, 6.710253867599925, 7.509710117599925, 7.
 ↪421297617599926]

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. 0.6578573459461617
```

2.0.2 Vacuuming system

[ ]:
```
group_AR1 = np.array([16.240403600000004, 18.608489900000002, 9.
↪457926200000001, 11.76311885])
group_AR1_AIR = np.array([5.37210625, 3.5309750000000006])

print("AR1 vs VACUUM")
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 VACUUM
        H1: The means of the distributions underlying the samples are unequal.
0.014782227276510938
```