Lab 2 - Number estimation (Weber fraction)

This lab must be done **individually**. The required packages have been imported for you below.

In [22]:

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

The target number (i.e. ground truth) for each experimental trial is provided in the following python array.

In [23]:

```
targets = np.array([3, 8, 40, 2, 5, 30, 7, 35, 6, 15, 10, 20, 9, 25, 4
]);
```

Read in the experimental data---these are identical to what you've analyzed in Lab 1. `df` is a dataframe of size *(Participants x Trials)*.

In [24]:

```
df = pd.read_csv('data-number-estimation.csv')
```

Compute the $mean$ and standard deviation (sd) for each trial (do not use a `for` loop). **[1pt]**

Hint: Use `df.mean()` and `df.std()` .

In [25]:

```
# Write your code here
mn = df.mean()
mn
```

Out[25]:

```
Trial 1      3.000000
Trial 2      8.319149
Trial 3     28.872340
Trial 4      1.978723
Trial 5      5.085106
Trial 6     27.723404
Trial 7      7.234043
Trial 8     31.127660
Trial 9      6.085106
Trial 10    16.425532
Trial 11    10.680851
Trial 12    21.489362
Trial 13     9.744681
Trial 14    25.170213
Trial 15     4.000000
dtype: float64
```

In [26]:

```
sd = df.std()
sd
```

Out[26]:

```
Trial 1     0.000000
Trial 2     1.252934
Trial 3     8.641701
Trial 4     0.145865
Trial 5     0.350762
Trial 6     8.072234
Trial 7     0.757937
Trial 8     8.157730
Trial 9     0.408059
Trial 10    2.668208
Trial 11    2.117283
Trial 12    4.960156
Trial 13    1.938938
Trial 14    5.946571
Trial 15    0.208514
dtype: float64
```

Figure 1 [2pts]

Task 1.1: Plot $mean$ responses against target numbers and add a reference line for the ground truth. **Hint**: Use `plt.plot()` .

Task 1.2: Plot $mean + sd$ and $mean - sd$. **Hint**: Use `np.add()` and `np.subtract()` .

Task 1.3: Annotate the graph and axes. **Hint**: Use `plt.legend()` and `plt.xlabel()` and `plt.ylabel()` .

In [27]:

```
# creating a dataframe
mean_of_trail = [3.000000, 8.319149, 28.872340,1.978723, 5.085106, 27.
723404,7.234043, 31.127660, 6.085106,
                 16.425532, 10.680851,21.489362, 9.744681,  25.170213
,4.000000 ]
sd_of_trail = [sd]
data_accuracy = [targets, mean_of_trail, sd]
summary_df = pd.DataFrame(data_accuracy
                           ,columns = ['Trial 1', 'Trial 2','Trial 3'
                                       , 'Trial 4','Trial 5', 'Trial 6
'
                                       ,'Trial 7', 'Trial 8','Trial 9'
                                       , 'Trial 10','Trial 11', 'Trial
12'
                                       ,'Trial 13', 'Trial 14', 'Trial
15']
                           ,index =['Target', 'mean','sd'])
summary_df
# organized by the values of targets
summary_df = summary_df.sort_values(by ='Target', axis=1)
summary_df.head()
```

Out[27]:

	Trial 4	Trial 1	Trial 15	Trial 5	Trial 9	Trial 7	Trial 2	Trial 13	Trial 11
Target	2.000000	3.0	4.000000	5.000000	6.000000	7.000000	8.000000	9.000000	10.000000
mean	1.978723	3.0	4.000000	5.085106	6.085106	7.234043	8.319149	9.744681	10.680851
sd	0.145865	0.0	0.208514	0.350762	0.408059	0.757937	1.252934	1.938938	2.117283

In [28]:

```
# extract vectors from this dataframe
sd_sorted = summary_df.iloc[2]
mean_sorted = summary_df.iloc[1]
```

In [29]:

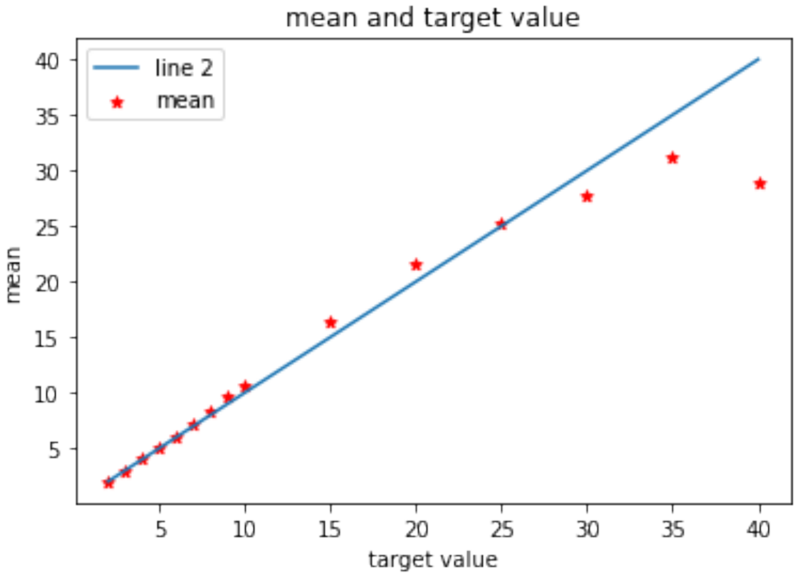
```
# sort targets
targets = np.array([3, 8, 40, 2, 5, 30, 7, 35, 6, 15, 10, 20, 9, 25, 4
]);
targets.sort()
targets
```

Out[29]:

```
array([ 2,  3,  4,  5,  6,  7,  8,  9, 10, 15, 20, 25, 30, 35, 40])
```

In [30]:

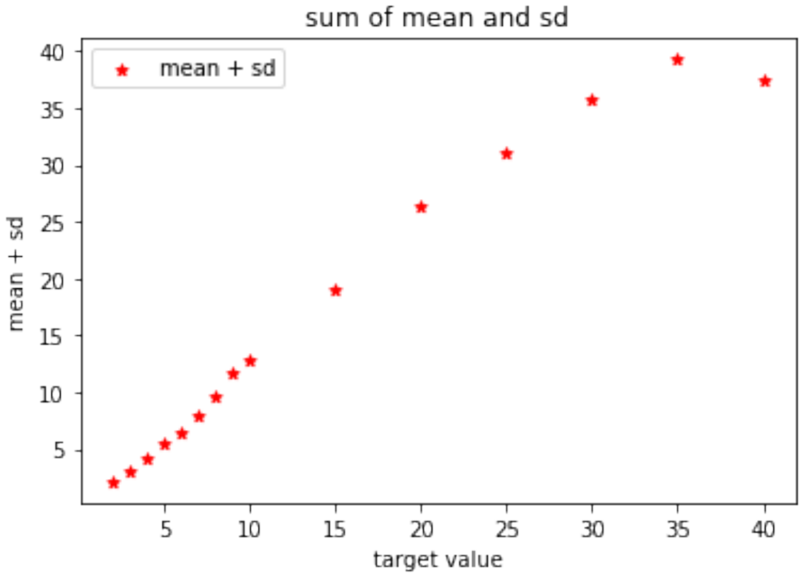
```
# Task 1.1



##  line 1 points
x1 = targets
y1 = mean_sorted
# plotting the line 1 points
plt.scatter(x1, y1, label= "mean", color= "red",
            marker= "*", s=30)

# line 2 points
x2 = targets
y2 = targets
# plotting the line 2 points
plt.plot(x2, y2, label = "line 2")

# naming the x axis
plt.xlabel('target value')
# naming the y axis
plt.ylabel('mean')
# giving a title to my graph
plt.title('mean and target value')

# show a legend on the plot
plt.legend()

# function to show the plot
plt.show()
```



In [31]:

```python
# Task 1.2 Plot mean + sd and mean - sd

in_arr1 = np.array(mean_sorted)
in_arr2 = np.array(sd_sorted)

print ("1st Input array : ", in_arr1)
print ("2nd Input array : ", in_arr2)

out_arr = np.add(in_arr1, in_arr2)
print ("output added array : ", out_arr)

## plot
x1 = targets
y1 = out_arr
# plotting the line 1 points
plt.scatter(x1, y1, label= "mean + sd", color= "red",
            marker= "*", s=30)

# naming the x axis
plt.xlabel('target value')
# naming the y axis
plt.ylabel('mean + sd')
# giving a title to my graph
plt.title('sum of mean and sd')

# show a legend on the plot
plt.legend()

# function to show the plot
plt.show()
```

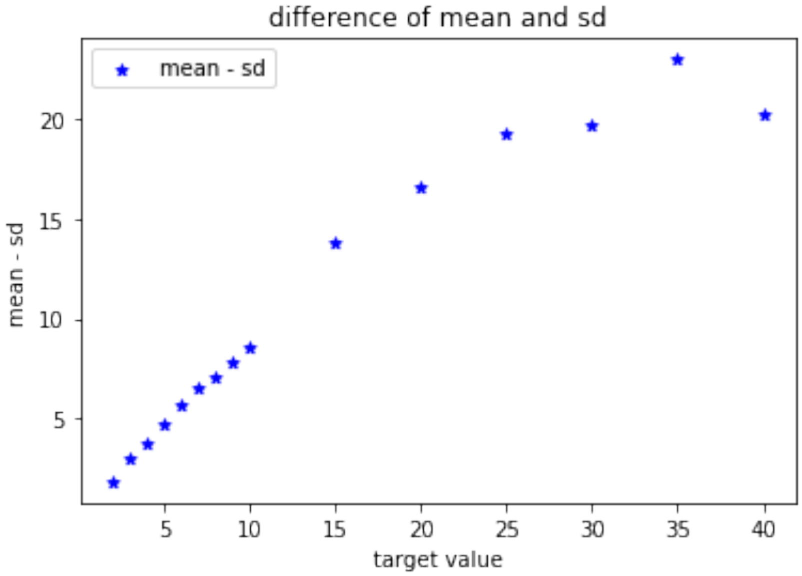
```
1st Input array :  [ 1.978723  3.        4.        5.085106  6.08510
6  7.234043  8.319149
  9.744681 10.680851 16.425532 21.489362 25.170213 27.723404 31.1276
6
 28.87234 ]
2nd Input array :  [0.14586499 0.        0.20851441 0.35076235 0.40
805942 0.75793673
 1.25293365 1.93893836 2.11728274 2.668208   4.96015577 5.94657062
 8.07223402 8.15773045 8.64170127]
output added array :  [ 2.12458799  3.          4.20851441  5.435868
35  6.49316542  7.99197973
  9.57208265 11.68361936 12.79813374 19.09374    26.44951777 31.1167
8362
 35.79563802 39.28539045 37.51404127]
```



In [32]:

```
# Task 1.2 plot mean - sd

in_arr1 = np.array(mean_sorted)
in_arr2 = np.array(sd_sorted)

print ("1st Input array : ", in_arr1)
print ("2nd Input array : ", in_arr2)

out_arr = np.subtract(in_arr1, in_arr2)
print ("output subtract array : ", out_arr)

## plot
x1 = targets
y1 = out_arr
# plotting the line 1 points
plt.scatter(x1, y1, label= "mean - sd", color= "blue",
            marker= "*", s=30)

# naming the x axis
plt.xlabel('target value')
# naming the y axis
plt.ylabel('mean - sd')
# giving a title to my graph
plt.title('difference of mean and sd')

# show a legend on the plot
plt.legend()

# function to show the plot
plt.show()
```

```
1st Input array :  [ 1.978723  3.         4.         5.085106  6.08510
6  7.234043  8.319149
  9.744681 10.680851 16.425532 21.489362 25.170213 27.723404 31.1276
6
 28.87234 ]
2nd Input array :  [0.14586499 0.         0.20851441 0.35076235 0.40
805942 0.75793673
 1.25293365 1.93893836 2.11728274 2.668208   4.96015577 5.94657062
 8.07223402 8.15773045 8.64170127]
output subtract array :  [ 1.83285801  3.          3.79148559  4.734
34365  5.67704658  6.47610627
  7.06621535  7.80574264  8.56356826 13.757324   16.52920623 19.2236
4238
 19.65116998 22.96992955 20.23063873]
```



In [33]:

```
# Task 1.3
# annotation and labels are included in previous graphs
```

Figure 2 [2pts]

Divide sd by $mean$ for each trial.

Hint: Use `np.divide()` .

In [34]:

```
in_arr1 = np.array(sd_sorted)
in_arr2 = np.array(mean_sorted)


divided = np.divide(in_arr1, in_arr2)
print ("sd/mean : ", divided)
```

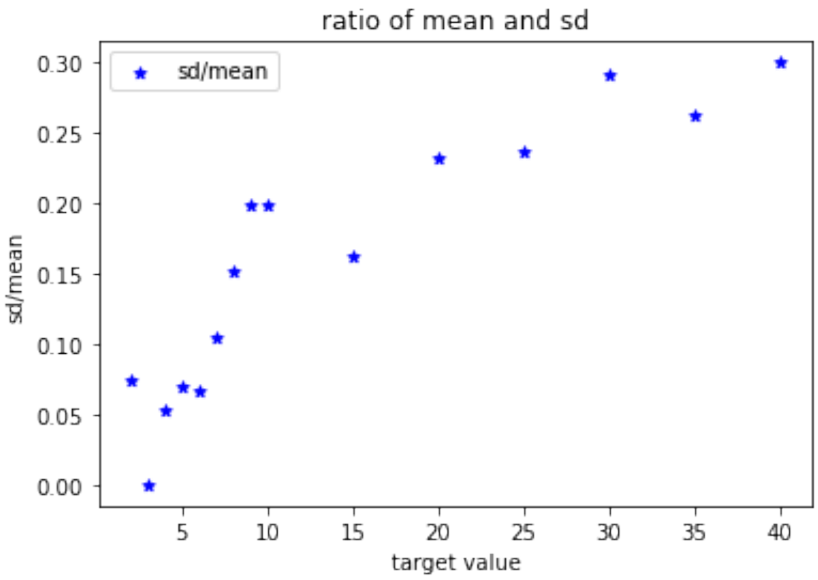
```
sd/mean :  [0.07371673 0.         0.0521286  0.06897838 0.06705872 0
.1047736
 0.15060839 0.19897402 0.19823165 0.16244271 0.23081913 0.23625428
 0.29117038 0.26207336 0.29930727]
```

Uncomment the following line to start a new figure.

Plot $\frac{sd}{mean}$ ratios against the target numbers.

In [35]:

```
plt.figure()
x1 = targets
y1 = divided
# plotting the line 1 points
plt.scatter(x1, y1, label= "sd/mean", color= "blue",
            marker= "*", s=30)

# naming the x axis
plt.xlabel('target value')
# naming the y axis
plt.ylabel('sd/mean')
# giving a title to my graph
plt.title('ratio of mean and sd')

# show a legend on the plot
plt.legend()

# function to show the plot
plt.show()
```



Estimate Weber's fraction in two steps. 1) First choose an appropriate threshold target number (given the plot you've made above) and justify your choice. **[2pts]**

In [36]:

```
thres = 8
```

Justification: The threshold target number was chosen to be 8. The ratio of stadard deviation and mean increased significantly starting from 9 as the target value. Target values included 8 and below have relatively small error rate. Referenced from the mean v.s. target value graph

2) Then calculate Weber fraction by averaging $\frac{sd}{mean}$ ratios across trials that have targets greater than the threshold you've chosen. **[1pt]**

Hint: Use `np.where()` and `np.mean()` .

In [39]:

```
# find the index of target value = 9
index = np.where(targets == 9)
print (index) #7, this means that we have to select sd/mean ratio starting from index 7
```

```
(array([7]),)
```

In [40]:

```
# sd/mean vector for target values above 9, from index 7 as
divided_for_target_above8 = divided[7:]

# calculate weber fraction
wb = divided_for_target_above8.mean()
print ("Weber fraction : ", wb)
```

```
Weber fraction :  0.23490910068496382
```

Export and submit a **fully executable** Python Jupyter Notebook along with a PDF export of your notebook showing all results you've obtained. Please follow the naming convention as suggested in Lab 1.**[2pts]**

In [ ]: