lab2_number_estimation_weber_fraction

October 2, 2020

0.1 Lab 2 - Number estimation (Weber fraction)

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

[79]:
```
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.

[80]:
```
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)*.

[81]:
```
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()`.

[82]:
```
# Write your code here

mn = df.mean()
sd = df.std()

print('The mean is {}'.format(mn))
print('The std is {}'.format(sd))
```

```
The mean is 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
The std is 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
```

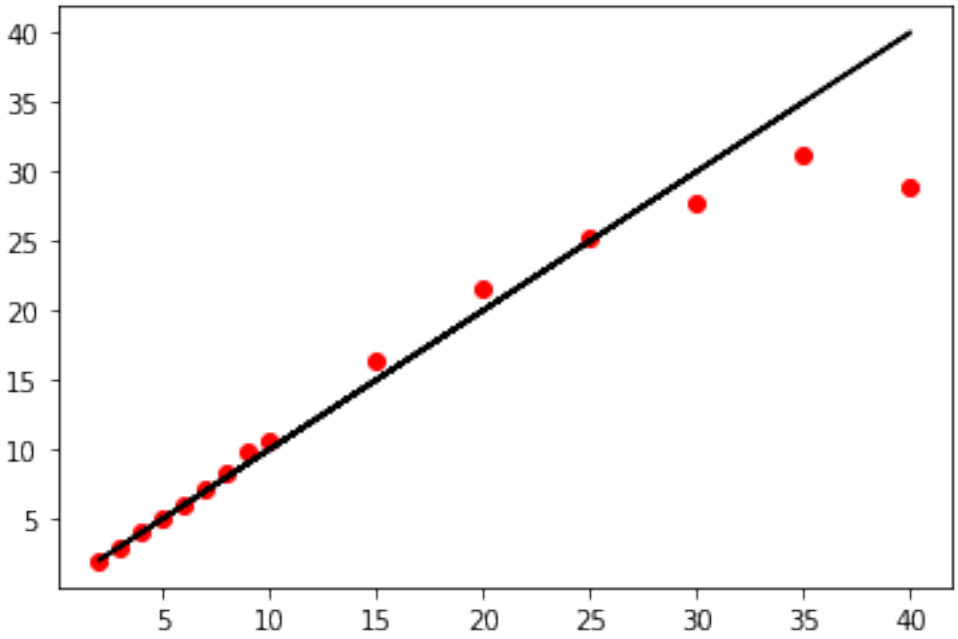
0.1.1 Figure 1 [2pts]

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()`.

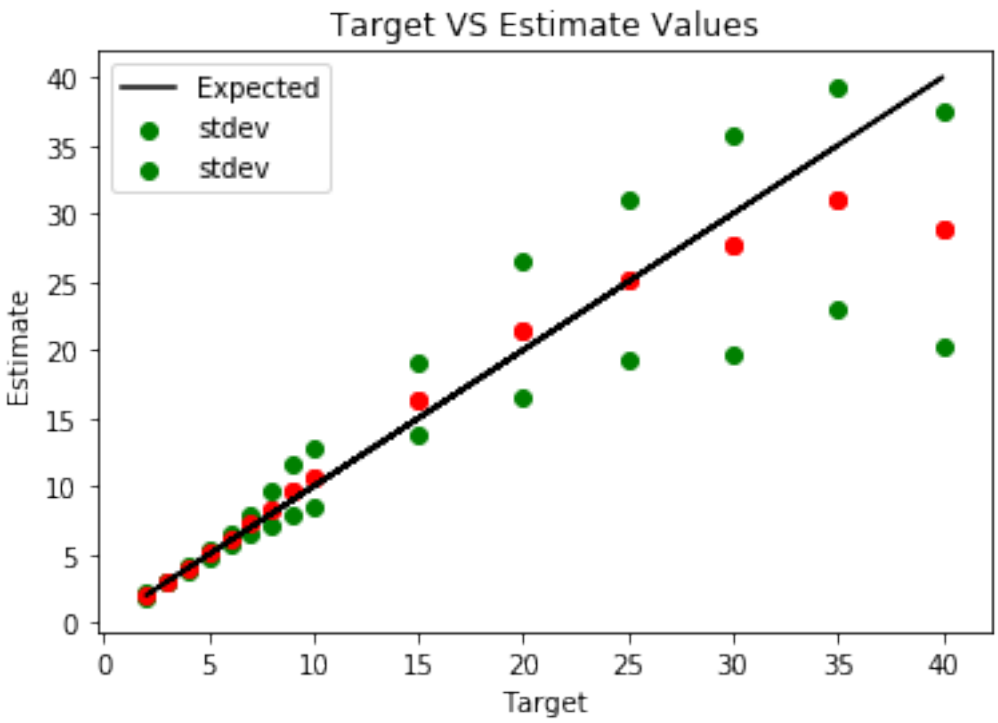
[83]:
```
# Task 1.1
plt.scatter(targets, mn, color = 'red')
expected = plt.plot(targets, targets, color = 'black', label = 'Expected')
```



[84]:
```
# Task 1.2
plt.scatter(targets, mn, color = "red")
add_sd = plt.scatter(targets, np.add(mn, sd), color = 'green', label = 'stdev')
subtract_sd = plt.scatter(targets, np.subtract(mn, sd), color = 'green', label␣
↪= 'stdev')
plt.scatter(targets, mn, color = "red")
expected = plt.plot(targets, targets, color = 'black', label = 'Expected')


# Task 1.3
plt.xlabel("Target")
plt.ylabel("Estimate")
plt.title("Target VS Estimate Values")
plt.legend()
```

[84]:
```
<matplotlib.legend.Legend at 0x118052240>
```



0.1.2 Figure 2 [2pts]

Divide *sd* by *mean* for each trial.

Hint: Use `np.divide()`.

[85]:
```
sd_over_mean = np.divide(sd, mn)
print('Dividing sd by mn equals: {}'.format(sd_over_mean))
```

```
Dividing sd by mn equals: Trial 1     0.000000
Trial 2     0.150608
Trial 3     0.299307
Trial 4     0.073717
Trial 5     0.068978
Trial 6     0.291170
Trial 7     0.104774
Trial 8     0.262073
Trial 9     0.067059
Trial 10    0.162443
Trial 11    0.198232
Trial 12    0.230819
Trial 13    0.198974
Trial 14    0.236254
```

```
Trial 15    0.052129
dtype: float64
```

Uncomment the following line to start a new figure.

[86]:
```
plt.figure()
```

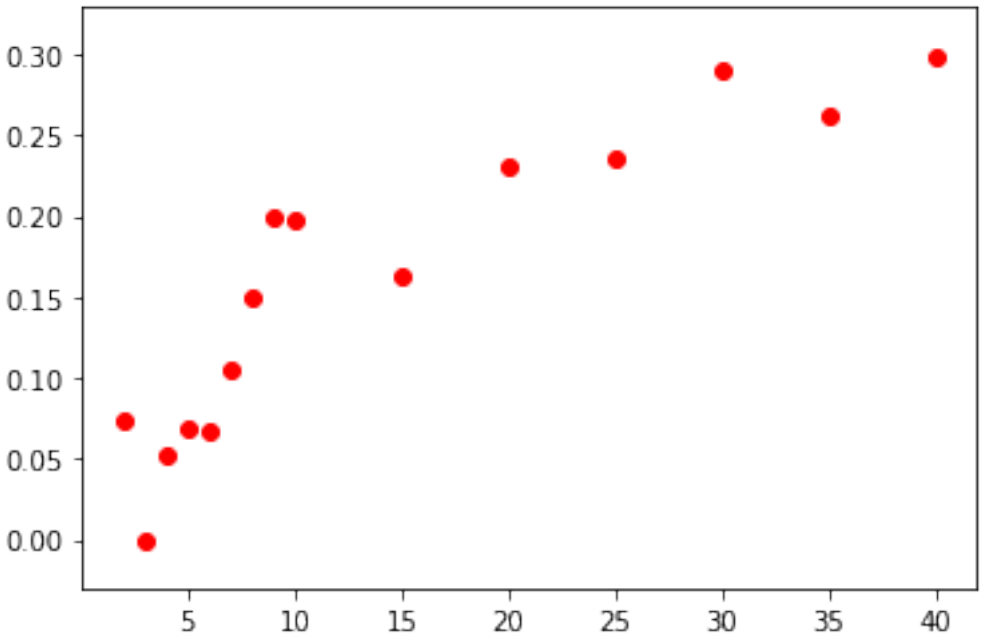
[86]:
```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

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

[87]:
```
# Task 1.2
plt.scatter(targets, sd_over_mean, color = "red")
```

[87]:
```
<matplotlib.collections.PathCollection at 0x118e0cb70>
```



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]**

[88]:
```
thres = 15
```

I pick the threshold as 15 because after that certain point, the error rate increases in a much stable fashion than other points. Degree of uncertainty scales with target nuber, so more targets means lower weber fraction (this means lesser error and greater acuity).

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()`.

[89]:
```
# plot consists of mean plots
# get only means where bigger or equal to than thresh
# calculate weber frac

idx_bigger_thresh = np.asarray(sd_over_mean)[np.where(df.mean() >= thres)]
webfrac = np.mean(idx_bigger_thresh)
print(webfrac)
```

```
0.24701118916625828
```

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]**