HH-2-2_soln

January 16, 2022

0.0.1 Hughes and Hase, Problem 2.2

[1]:
```
import numpy as np
```

Twelve data points given. Enter them into a `numpy` array:

[2]:
```
data = np.array([5.33, 4.95, 4.93, 5.08, 4.95, 4.96, 5.02, 4.99, 5.24, 5.25, 5.
↪23, 5.01])
```

i) Calculating the mean: $\mu = \frac{1}{N}\sum_i x_i$

[3]:
```
print('mean = ', np.sum(data)/len(data))
```

```
mean =  5.078333333333334
```

OR

[4]:
```
print('mean = ', np.mean(data))
```

```
mean =  5.078333333333334
```

ii) Calculating standard deviation: $\sigma_{N-1} = \sqrt{\frac{1}{N-1}\sum_i (x_i - \mu)^2}$

[5]:
```
print("standard deviation = ",np.sqrt(np.sum((data-np.mean(data))**2)/
↪(len(data)-1)))
```

```
standard deviation =  0.14357977404617803
```

OR

[6]:
```
print("standard deviation = ",np.std(data))
```

```
standard deviation =  0.13746716779734078
```

These results do not agree!!

By default the numpy `std` function calculates σ_N, which is similar to the σ_{N-1} given in Eq. (2.3) of H&H, except the denominator is N instead of $N-1$. The difference doesn't usually matter, and we won't go into this in any depth now. But if we set the `ddof=1` option, `numpy` will calculate σ_{N-1}.

Remember: you can see all the details of `np.std` by typing `np.std?`.

[7]:
```
print("standard deviation = ", np.std(data, ddof=1))
```

```
standard deviation =  0.14357977404617803
```

iii) Standard error, or standard deviation of the mean Use Eq. (2.7):

$$\alpha = \frac{\sigma_{N-1}}{\sqrt{N}}.$$

[8]:
```
print("standard error =", np.std(data, ddof=1)/np.sqrt(len(data)))
```

```
standard error = 0.04144791059787326
```

iv) Formatted result: Sensitivity $= 5.08 \pm 0.04\,\mathrm{A/W}$

Version information `version_information` is from J.R. Johansson (jrjohansson at gmail.com); see Introduction to scientific computing with Python for more information and instructions for package installation.

`version_information` is installed on the linux network at Bucknell

[9]:
```
%load_ext version_information
```

[10]:
```
%version_information numpy
```

[10]:

Software	Version
Python	3.7.8 64bit [GCC 7.5.0]
IPython	7.17.0
OS	Linux 3.10.0 1127.19.1.el7.x86_64 x86_64 with centos 7.9.2009 Core
numpy	1.19.1
Sun Jan 16 14:26:17 2022 EST	

[ ]:
```

```

HH-2-3_soln

January 16, 2022

0.0.1 Hughes & Hase Problem 2.3

The standard error, or standard deviation of the mean, is given by Eq.(2.7):

$$\alpha = \frac{\sigma_{N-1}}{\sqrt{N}}.$$

To decrease α by a factor of 10, the denominator must be increased by the same factor, which means that N must increase by a factor of 100. Translating to the described experiment, this means that data should be collected for 100 minutes (assuming that everything in the experiment is stable for that length of time).

```
[ ]:
```

pendulum_soln

January 16, 2022

0.0.1 Pendulum problem

[1]:
```
import numpy as np
```

Data Data for pendulum swings: + Standard deviation for any set of timing measurements = 0.04 s + Experiment A: 12 sets of 10 swings; average time for 10 swings $T_{10} = 28.39\,\mathrm{s}$ + Experiment B: 1 set of 120 swings; time for 120 swings $T_{120} = 340.61\,\mathrm{s}$

Period from Experiment A: The standard error (standard deviation of the mean) for the time for 10 swings is

$$\alpha = \frac{\sigma}{\sqrt{N}} = \frac{0.04}{\sqrt{12}}$$

The time for one swing is the time for 10 swings divided by 10:

$$\begin{aligned} T_1 &= \frac{T_{10}}{10} \\ &= \frac{28.39 \pm \frac{0.04}{\sqrt{12}}}{10} \\ &= 2.839 \pm \frac{0.04}{10\sqrt{12}} \\ &= 2.839 \pm 0.001155 \end{aligned}$$

The presentation form of this result is

$$T_1 = 2.839 \pm 0.001\,\mathrm{s}.$$

[2]:
```
t10 = 28.39
alpha10 = 0.04/np.sqrt(12)
t1 = t10/10
alpha1 = alpha10/10

print('T1 =',t1,'+/-',alpha1)
```

```
T1 = 2.839 +/- 0.0011547005383792516
```

Period from Experiment B: The standard error (standard deviation of the mean) for the time for 120 swings is

$$\alpha = \frac{\sigma}{\sqrt{N}} = \frac{0.04}{\sqrt{1}}$$

The time for one swing is the time for 120 swings divided by 120:

$$\begin{aligned} T_1 &= \frac{T_{120}}{120} \\ &= \frac{340.61 \pm \frac{0.04}{1}}{120} \\ &= 2.838417 \pm \frac{0.04}{120\sqrt{1}} \\ &= 2.838417 \pm 0.000333 \end{aligned}$$

The presentation form of this result is

$$T_1 = 2.8384 \pm 0.0003\,\mathrm{s}.$$

[3]:
```
t120 = 340.61
alpha120 = 0.04/np.sqrt(1)
t1 = t120/120
alpha1 = alpha120/120

print('T1 =',t1,'+/-',alpha1)
```

```
T1 = 2.838416666666667 +/- 0.0003333333333333333
```

[ ]:
```
```

estimate_mean_sd_1_soln

January 16, 2022

0.0.1 Estimating mean and standard deviation from numerical data

One method is to look for the median. There are 15 data points. The middle value is 4.19365, meaning 7 values are larger than this and 7 are smaller. The median can be a poor estimate for the mean if the distribution is very asymmetric, but these numbers go up to 4.5 and down to 3.9, so it looks like they are fairly well centered on 4.2.

Using H&H's "rough and ready" estimate (p. 11) for the standard deviation, we find that the maximum value minus the mean is about 0.3. Taking 2/3 of that gives a estimated standard deviation of 0.2.

Quantitative check

[1]:
```
import numpy as np
```

[2]:
```
data = np.array([4.1075, 4.39831, 4.19365, 4.20259,
4.26921, 4.13037, 3.97548, 4.51314, 4.01286, 4.0101, 4.15578, 4.35153,
4.30801, 4.21082, 3.94315])
```

[3]:
```
np.std(data,ddof=1)
```

[3]:
```
0.1640200187259383
```

Not too bad!

[4]:
```
%load_ext version_information
%version_information numpy
```

[4]:

Software	Version
Python	3.7.8 64bit [GCC 7.5.0]
IPython	7.17.0
OS	Linux 3.10.0 1127.19.1.el7.x86_64 x86_64 with centos 7.9.2009 Core
numpy	1.19.1
Sun Jan 16 15:32:31 2022 EST	

[ ]:
```
```