

Transformations Revisited

Last time

Transformations were applied using google sheets and pre written python code



This time

new transformations in python with numpy

The screenshot displays the Spyder Python IDE interface. The main editor shows a script named `intro_01.py` with the following code:

```
1 # import statement
2
3 import numpy as np
4
5 %%<- creates a code cell
6
7 # declare a variable
8
9
10 a = 10
11
12 %%<- create a list
13
14 b = [10, 5, -16, 23]
15
16 %%<- create numpy array
17
18 c = np.array(b)
19
20 %%<- data transformation
21
22 new_c = c * -1
23
24
25
```

The Variable Explorer on the right shows the state of the variables:

Name	Type	Size	Value
a	int	1	10
b	list	4	[10, 5, -16, 23]
c	Array of int64 (4,)		[10 5 -16 23]
new_c	Array of int64 (4,)		[-10 -5 16 -23]
test_variable	int	1	70

The Console window at the bottom shows the execution of the code, including the assignment of `test_variable` to 70 and the execution of `new_c = c * -1`.

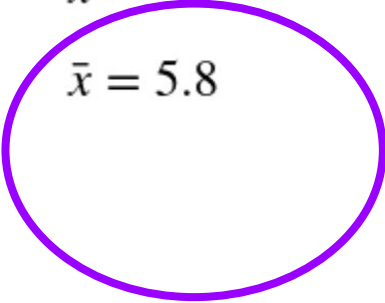
List of example transformations / computations

- given starting array, x
 - mean, \bar{x}
 - for every observation, i
 - error, $x_i - \bar{x}$
 - squared error, $(x_i - \bar{x})^2$
 - Sum over all squared error
 - Sum over all squared error normalized by number of observations - 1

Underlying math: mean

array: x = 3, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 8, 9

$$\bar{x} = \frac{3 + 4 + 4 + 5 + 5 + 5 + 6 + 6 + 6 + 7 + 7 + 8 + 9}{13}$$

$$\bar{x} = 5.8$$


Underlying math: error and squared error

x	\bar{x}	$(x_i - \bar{x})$	$(x_i - \bar{x})^2$
3	5.8	-2.8	7.84
4	5.8	-1.8	3.24
4	5.8	-1.8	3.24
5	5.8	-0.8	0.64
5	5.8	-0.8	0.64
5	5.8	-0.8	0.64
6	5.8	0.2	0.04
6	5.8	0.2	0.04
6	5.8	0.2	0.04
7	5.8	1.2	1.44
7	5.8	1.2	1.44
8	5.8	2.2	4.84
9	5.8	3.2	10.24

Underlying math: sum over all squared error

x	\bar{x}	$(x_i - \bar{x})$	$(x_i - \bar{x})^2$
3	5.8	-2.8	7.84
4	5.8	-1.8	3.24
4	5.8	-1.8	3.24
5	5.8	-0.8	0.64
5	5.8	-0.8	0.64
5	5.8	-0.8	0.64
6	5.8	0.2	0.04
6	5.8	0.2	0.04
6	5.8	0.2	0.04
7	5.8	1.2	1.44
7	5.8	1.2	1.44
8	5.8	2.2	4.84
9	5.8	3.2	10.24

Sum = 34.32

Underlying math: normalization by num observations - 1

x = 3, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 8, 9

number observations = 13

number observations - 1 = 12

Sum = 34.32

34.32 / 12

Code

```
import numpy as np

# create numpy array with original data
x = np.array([3, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 8, 9])

# find number of observations
n = np.size(x)

# mean
xbar = np.sum(x) / n

# error
error = x - xbar

# squared error
se = error**2

# sum of squared error over all observations
sse = np.sum(se)

# normalization by number observations - 1
result = sse/(n-1)

# look at the result
print(result)
```

2.858974358974359

Benefit of code

The size of the original array does not matter in code!

But imagine how annoying it would be to deal with **10 million** rows in google sheets...

```
import numpy as np

# create numpy array with 10 million numbers
x = np.arange(10000000)

# find number of observations
n = np.size(x)

# mean
xbar = np.sum(x) / n

# error
error = x - xbar

# squared error
se = error**2

# sum of squared error over all observations
sse = np.sum(se)

# normalization by number observations - 1
result = sse/(n-1)

# look at the result
print(result)
```

8333334166666.48

Exercise

Copy in the code shown in these slides into spyder and explore all the variables created!

What is the value of:

- n
- xbar
- error
- se
- sse
- result

Change x and repeat the exploration

fill in your results here: <https://forms.gle/CKWEnykPyyH3Xxch6>