PYTHON SEMINAR 2020

JENS HAHN

THEORETICAL BIOPHYSICS

TODAY

I Recap classes

II Data

III NumPy

IV SciPy

I. RECAP CLASSES

- Classes are blue prints for objects
- Classes can have attributes (class variables – *self* variables)
- Classes can have methods (class functions – *self* functions)
- Classes have magic methods which can be changed
- Classes need to be instantiated to gain an object

II. DATA

Data set: numerical values

100 measurements

300 replicates

	X_1	…	X_{100}
E_1	1,1	…	1,100
⋮	⋮		⋮
E_{300}	300,1	…	300,100

- Learn about NaN
- Calculate mean & median
- Calculate SD
- Normalisation
- Interpolation
- Linear regression

III. NUMPY - ARRAYS

Operations can be performed element-wise

```
import numpy as np

my_array = np.array([1,2,3])
```

III. NUMPY - ARRAYS

Operations can be performed element-wise

```
my_array * 3
array([3, 6, 9])
```

```
my_array * my_array
array([1, 4, 9])
```

array([False, False, False, False, False])

III. NUMPY - ARRAYS

"Mask" index arrays (Boolean indexing)

```
my_array >= 3
array([False, False, True])
```

```
my_array[my_array >= 3]
array([3])
```

III. NUMPY - NAN

- Numpy's NaN (*not a number)*
  - Very important place-holder in data sets
  - ***np.nan*** is of type ***float***
  - ***np.nan*** can be difficult to handle

III. LOAD THE DATA

```
import pickle as pkl

with open(“./assignment_data/20190527_data.pkl”, “rb”) as data_file:

    data = pkl.load(data_file)
```

III. LOAD THE DATA

Check **type** of the data `type(data)`

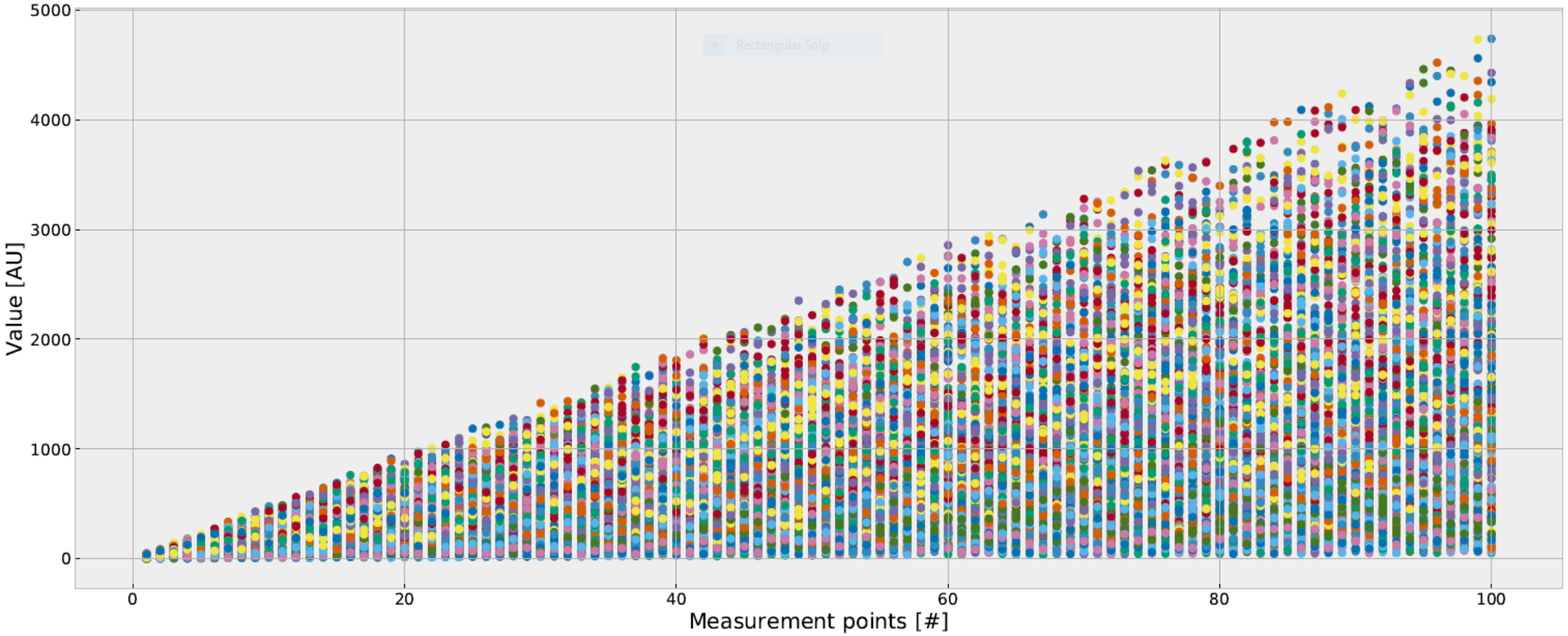
Load numpy `import numpy as np`

Check dimension of the data `data.shape`

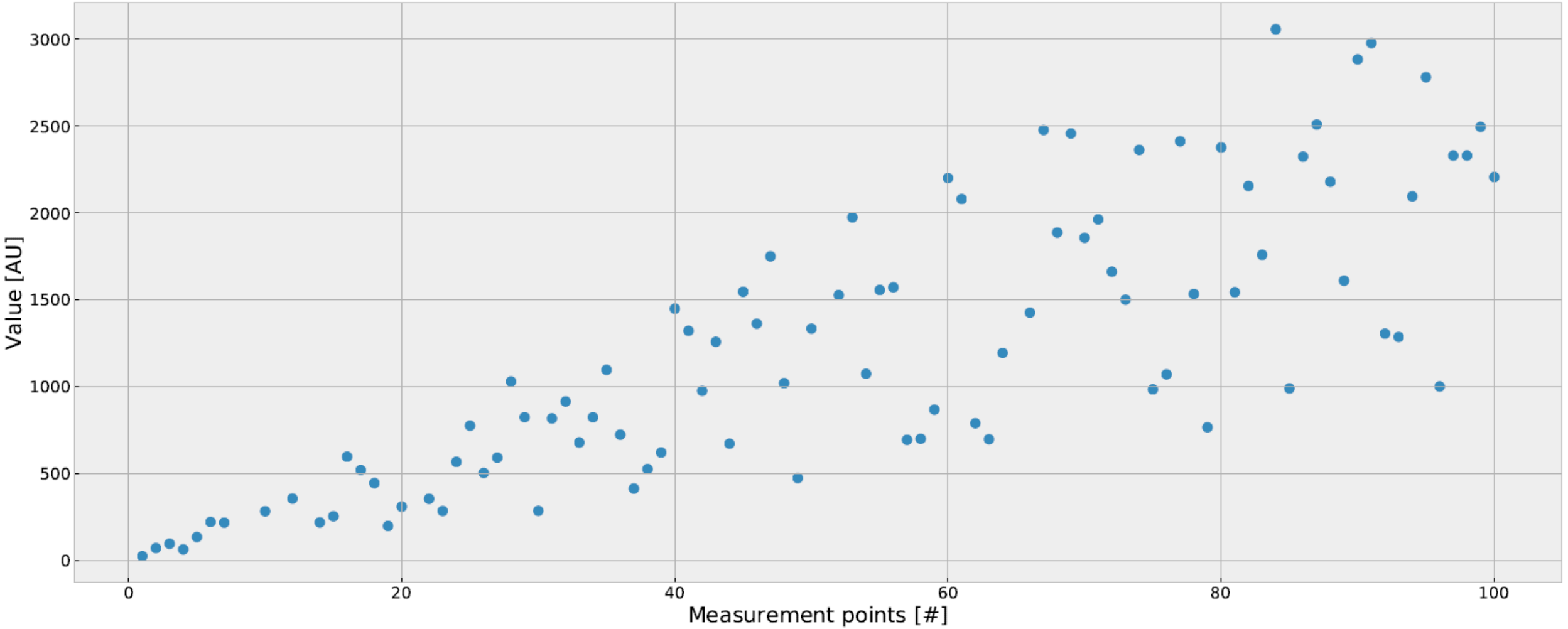
Look at the data

- First row: `data[0, :]`
- First column: `data[:, 0]`

III. LOAD THE DATA



III. LOAD THE DATA

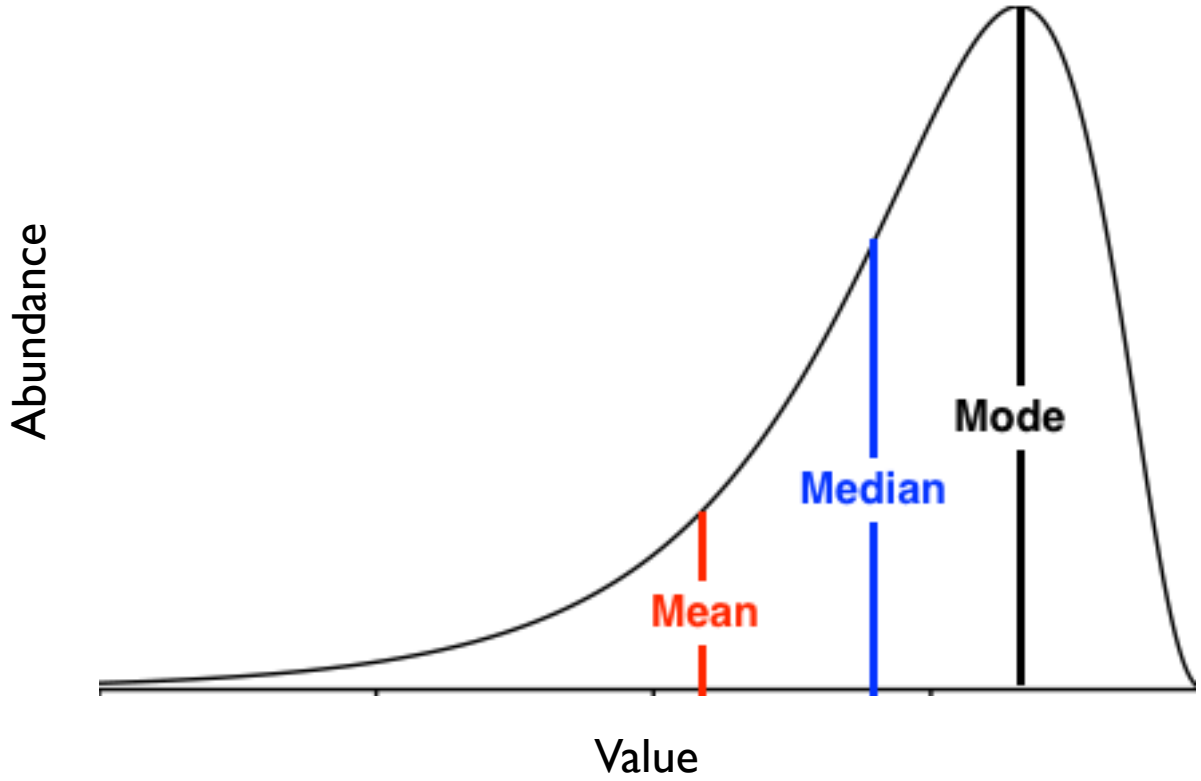


III. CHECK FOR NAN

Use ***np.isnan*** and ***np.count_nonzero*** to get the number of ***nans***

```
number_nans =
np.count_nonzero(np.isnan(data))
```

III. MEAN & MEDIAN



http://statisticshelper.com/mean-median-mode-calculator

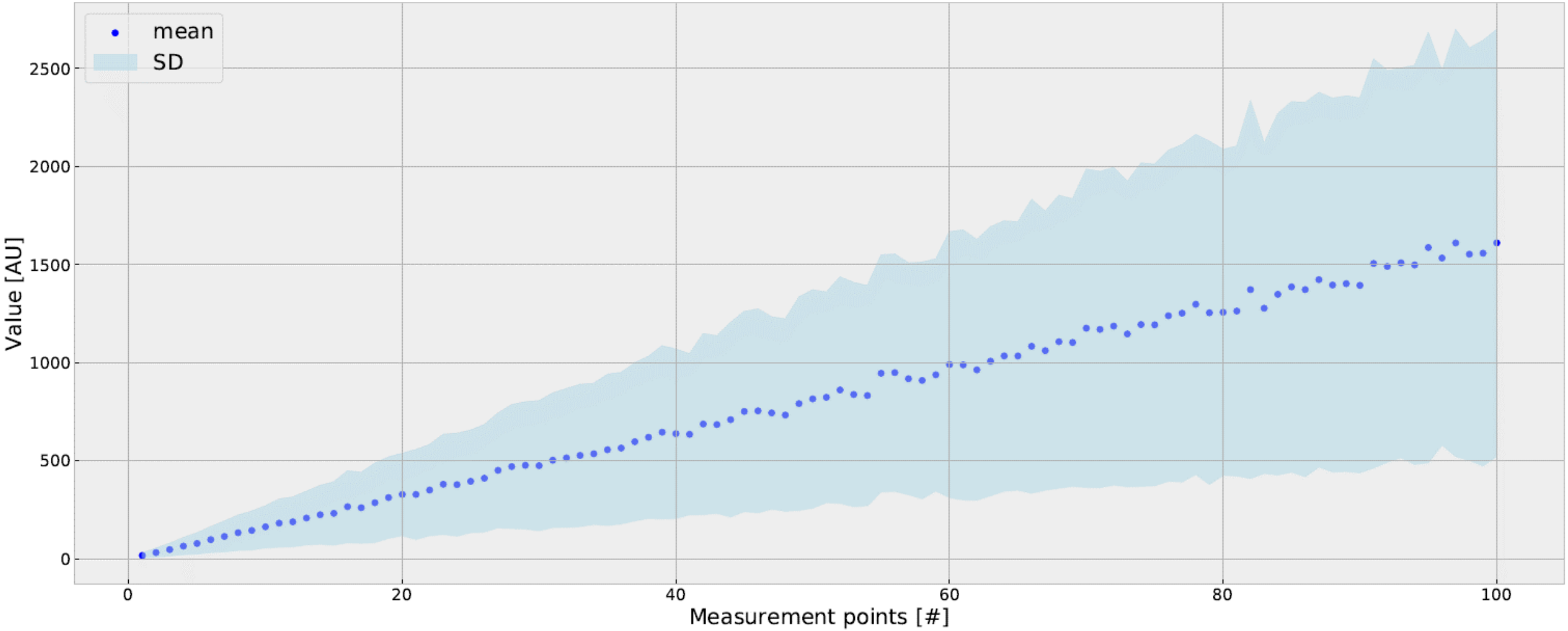
III. MEAN & MEDIAN

Calculate the **SD, mean,** and **median** per measurement

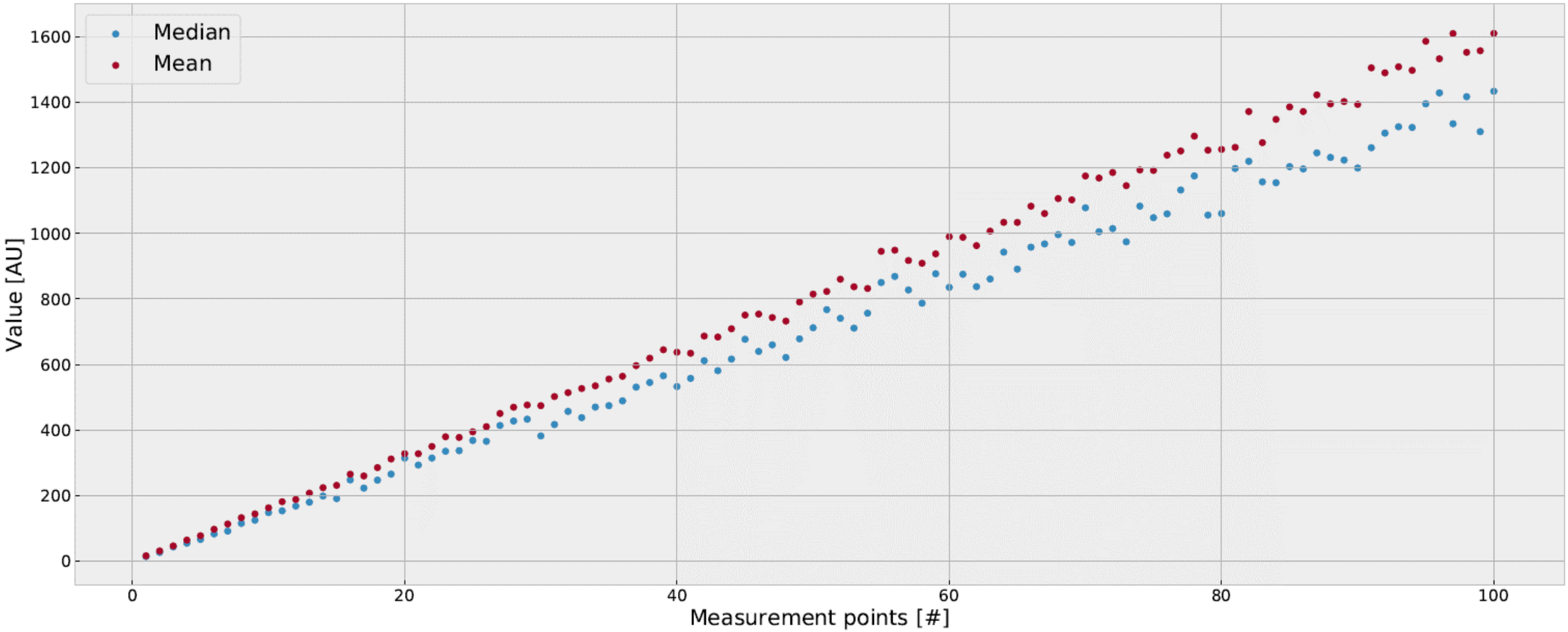
np.std, np.mean, and **np.median** cannot deal with **nans**!

```
mean_data = []
for column_index in range(data.shape[1]):
  mean_data.append(np.nanmean(data[:,column_index]))
```

III. MEAN & MEDIAN



III. MEAN & MEDIAN

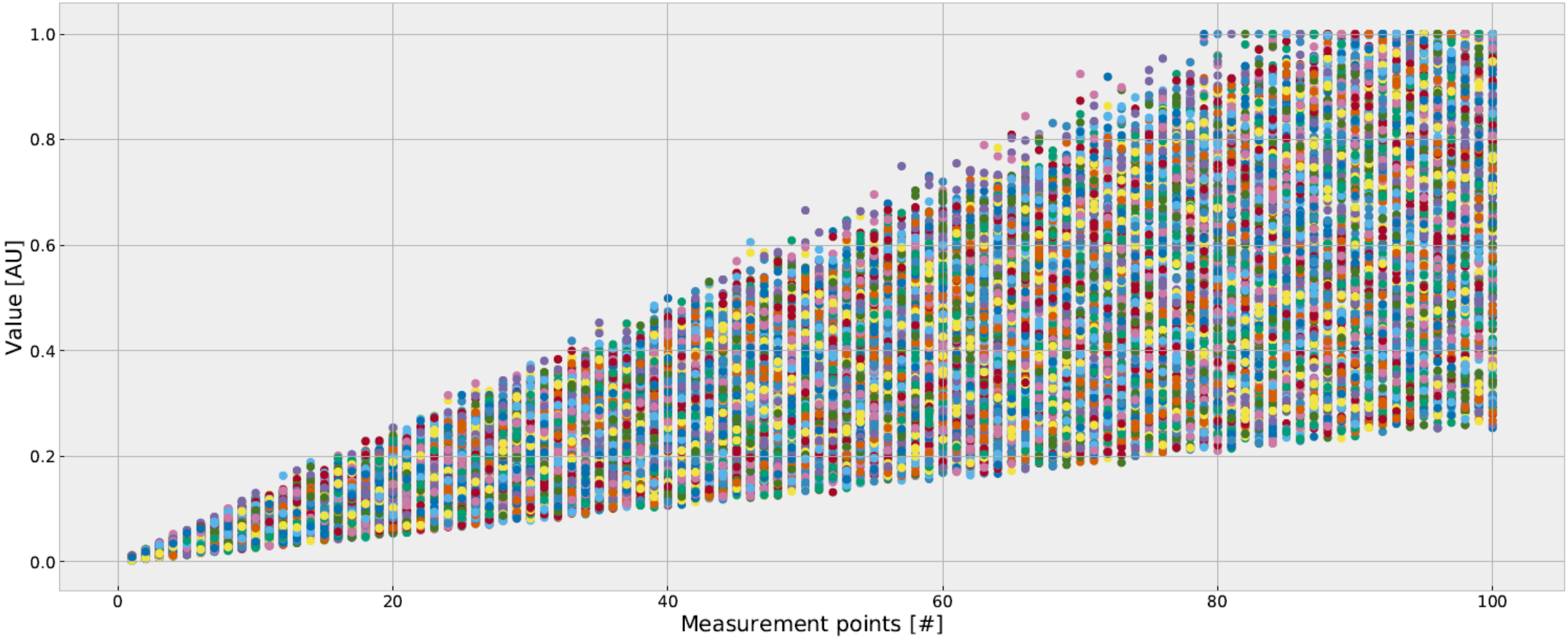


III. NORMALISATION

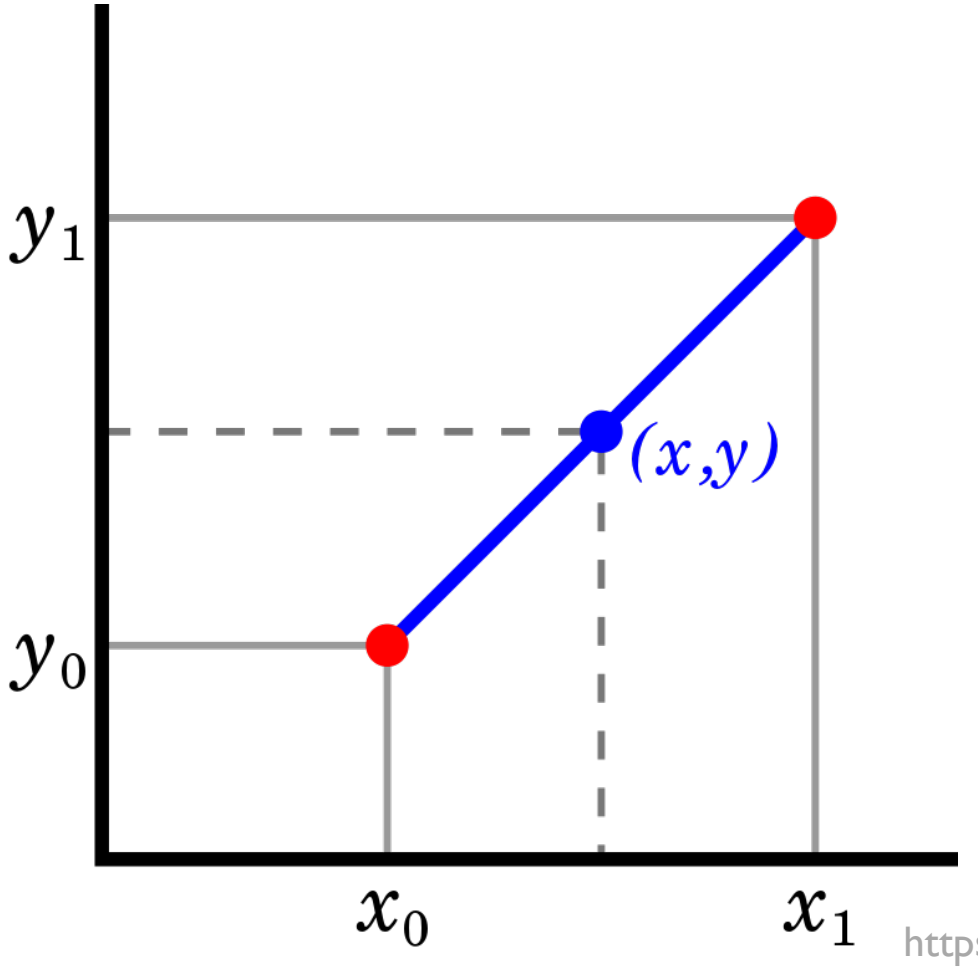
Normalise every replicate to be between 0 and 1

```
for row in range(data.shape[0]):
    data[row, :] /= np.nanmax(data[row, :])
```

III. NORMALISATION



III. INTERPOLATION



https://en.wikipedia.org/wiki/Linear_interpolation

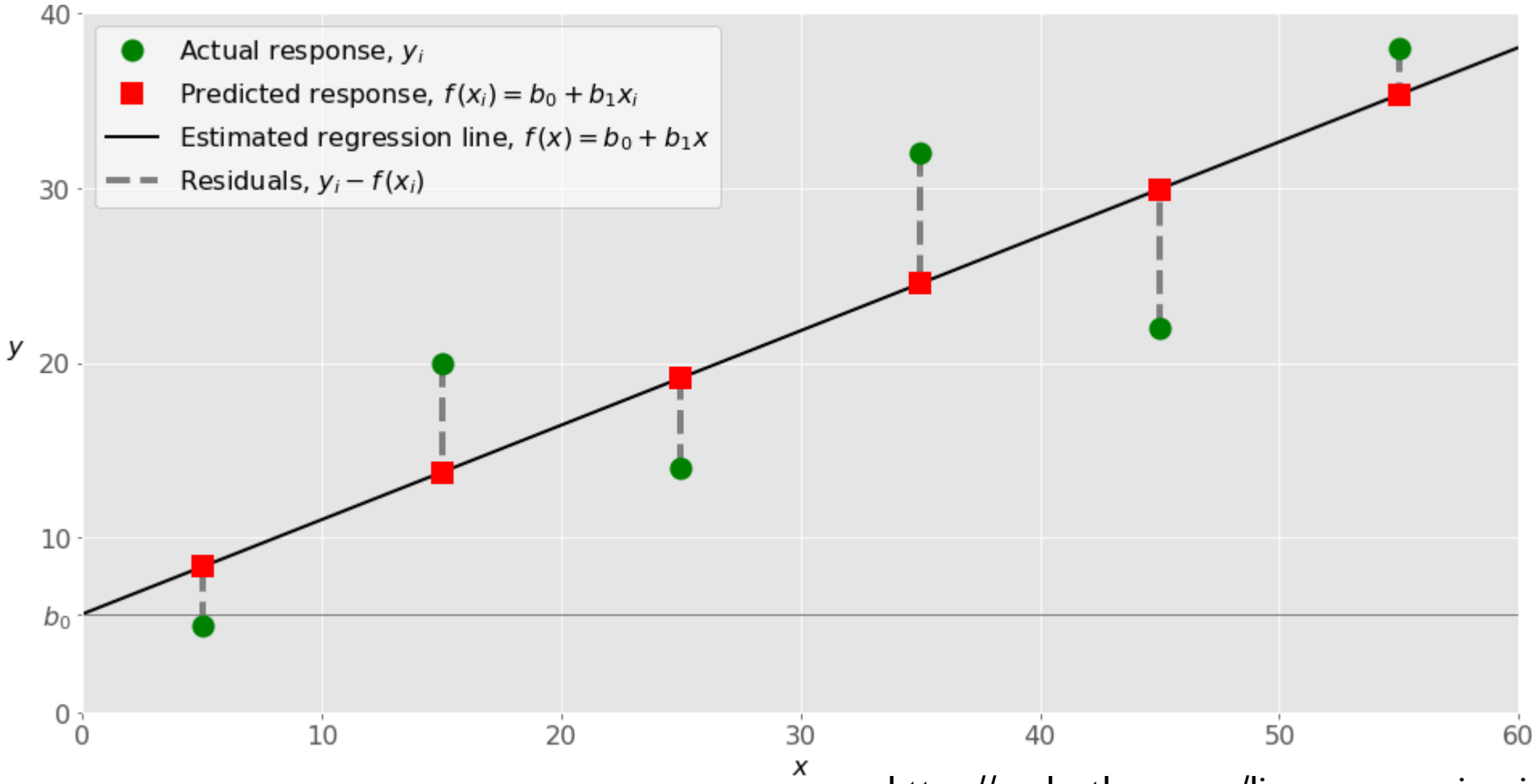
III. INTERPOLATION

Row **56** has two **NaN.**

Interpolate the values for these **NaN** using **np.interp** and replace it

```
mask = ~np.isnan(data[56,:])
x_val = np.array(range(100))[mask]
interp_value =
  np.interp([1,98], x_val, data[56,:][mask])
```

III. LINEAR REGRESSION



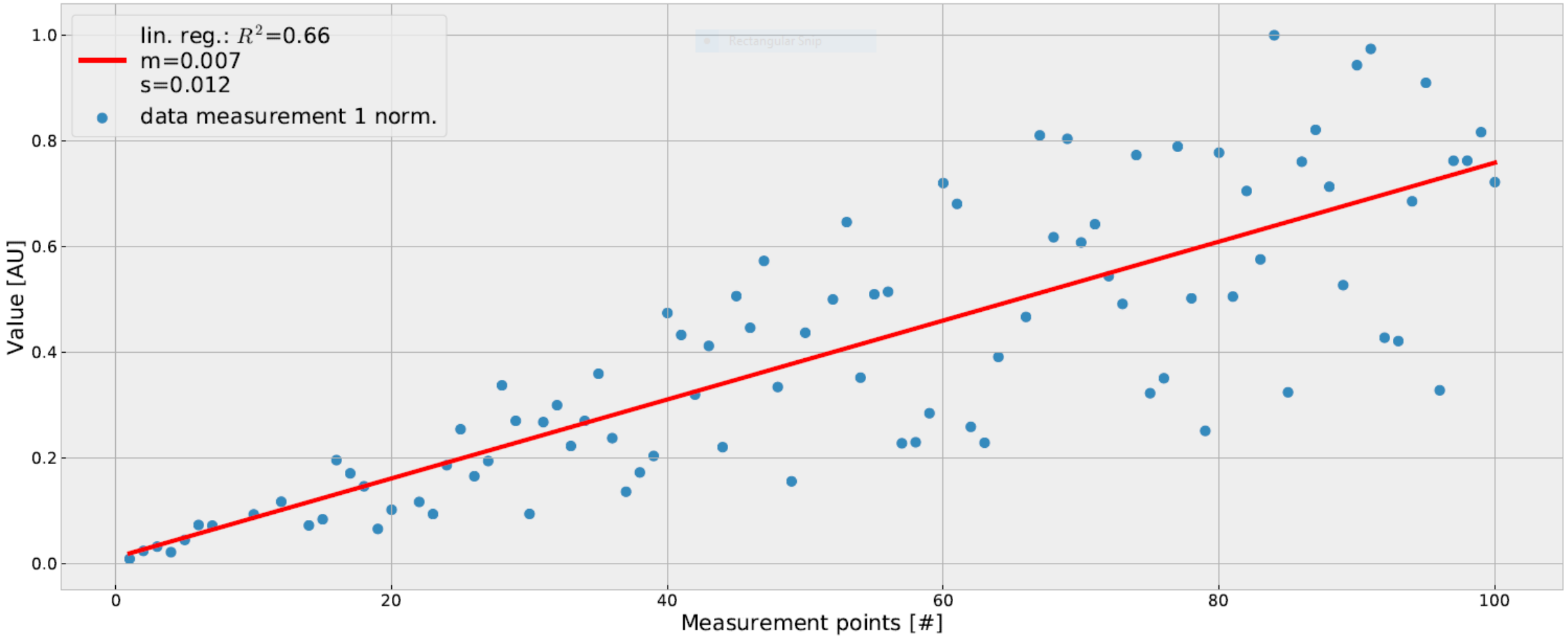
https://realpython.com/linear-regression-in-python/

III. LINEAR REGRESSION

Make a linear regression on the 1st replicate

```
from scipy.stats import linregress
mask = ~np.isnan(data[0, :])
x_val = np.array(range(100))[mask]
slope, intercept, r_value, p_value, std_err =
  linregress(x_val,data[0,:][mask])
```

III. LINEAR REGRESSION



IV. ASSIGNMENT – TURN IT INTO A CLASS

- Write a class DataAnalysis
- Write methods for every task

- Should work on numerical spreadsheet data
- Individual experiments or studies are in the rows

- Import data
- Calculate mean & median
- Calculate SD
- Normalisation
- (Interpolation)
- (Linear regression)

V. FURTHER READING

Python scikit learn

- Linear regression in Python

  https://realpython.com/linear-regression-in-python/

- scikit-learn documentation

  https://scikit-learn.org/stable/

Python numpy

- NumPy tutorial

  https://www.grund-wissen.de/informatik/python/scipy/numpy.html

- NumPy & SciPy documentation

  https://docs.scipy.org/doc/