# Lab 9

January 11, 2021

## 1 Introduction

For this assignment, you will write a code which does a basic data processing pipeline having the following steps:

- Taking data from a file and cleaning it.
- Modifying this data.
- Creating a predictor on future based on this data.
- Plotting the data, the predictor and its prediction.

The deadline for the submission is Monday, 11 January 2021 (today) 23:00.

**NOTE that this is NOT a group-work. You should write your solution yourself and without getting help from anyone.**

## 2 Dataset

The dataset you will be using is created using USD to TL exchange rate for the last 5 years starting from a given date, namely 01.01.2015, with the following modifications to make things more interesting:

- Some points are corrupted and replaced with random values which are called noise.
- All given points are deformed with some transformations beyond recognition.

In other words, the data to be provided to you is just a very noisy 2D point cloud with **X** and **Y** values (columns). You are required to clean the noise and reverse all the deformations on the data before using it.

Note that your submission **will be tested with another dataset**. Therefore, in your code, you should not make any assumptions based on the current dataset. You must implement your code for the general case as outlined below.

## 3 Testing and Submission

Your task is to fill in the “TODO” bits provided in the code given in the following subsection. You can copy this code or download the file named “lab.py” through the assignment named “Lab 9” on Class3 server.

You will submit this “lab.py” file after completing the implementation through the same activity on Class3 server.

If you have necessary modules installed on your own computer, you can work locally. Otherwise, the external emulator is still available for you with its setup for these modules. Another alternative might be using Google Colab for this purposes.

A sample file for testing purposes is available for you under the files of assignment activity.

The same file is included in the external emulator (https://ceng240.github.io/pyeditor/) as well. Hence, you have the opportunity to test your code there, if you prefer. (*If the figure you plot is not shown automatically, click the icon left to the title "Browser Python" in the emulator page*)

Do not forget that this is not the file which will be used during the evaluation of your codes. Avoid any assumptions based on the given sample data.

Suggested version of Python is 3.7 since there may be minor incompabilities between the versions. Also here are the official versions of modules to help you to avoid any conflicts:

- pandas –> 0.23.4
- numpy –> 1.15.1
- matplotlib –> 2.2.3

### 3.1 Important Notes

- For this assignment you will implement the functions with empty bodies(`read_and_filter`, `fix_deformation`,`fit_and_predict` and `plot`). Anything you write outside of these functions will not be graded.
- The `test` function is just to show you an example run of the application. You can change it as you like. It will not be graded.
- Do not change the signature (function names and parameters) of the functions and be careful of their return types since any function which does not comply with the specifications will get 0.
- You should implement the functions in the given order. In case, the results of an earlier function are incorrect, the following functions will also work incorrectly. In other words, your mistakes will propagate to the remaining implementation.

## 4 Explanations of the Functions

In this section, we give a detailed description for every function (arguments, return type and its task) that appears in the problem.

### 4.1 `read_and_filter(filename, filter_limit)`

It takes a `str` argument which is called `filename` (a CSV file) and a `float` argument which is called `filter_limit`. It returns a Pandas `Dataframe`.

It should read the CSV file with the given filename. The retrieved data should have these two columns in this order: **X** and **Y**. Furthermore, this function should remove any row from the dataset where the corresponding Y value is bigger than the given `filter_limit`. Lastly, it should return the filtered dataset.

## 4.2 `fix_deformation(dataframe)`

This function takes the `dataframe` (which is a Pandas `DataFrame`) which will be the output of the `read_and_filter` function. It should return a NumPy array with two columns. The first column will store the X values and the second column will store the Y values.

The following deformations were performed on the "original" data in this order:

- Update X values by -1500.
- Scale X values by 0.01 (shrink 100 times).
- Rotate points 45 degrees clockwise around the origin.
- Update Y values by 5.
- Scale Y values by 0.5.

In function `fix_deformation()`, you should reverse all these deformations by **applying inverse of these transformations in the reverse order**. The inversion of deformations should proceed like this: Inverting the scaling Y by 0.5; inverting the update of Y by 5; inverting rotation of points by 45 degrees clockwise around the origin, and so on. You will find the explanations of these transformations below.

### 4.2.1 Updating X or Y values

Here, you must only update the values with the given amount on the given axis. For example, updating the X value of point `(1,0)` by 100 will yield point `(101,0)`. The inverse of this transformation will just be updating the point in the opposite direction. Therefore, the inverse operation is just updating the X value by -100.

### 4.2.2 Scaling X or Y values

For this transformation, you need to scale the values with the given factor on the given axis. For example, scaling the point `(0.5,0)` on the X axis by 7 will result in `(3.5,0)`. Its inverse is scaling on X axis by 1/7.

### 4.2.3 Rotation

Rotation is a little more complex than the others since the resulting values depend on both X and Y values. We can rotate a point $(X, Y)$ **counter-clockwise** around the origin with angle $\theta$ as follows to obtain new point $(\hat{X}, \hat{Y})$:

$$\hat{X} = X \times \cos\theta - Y \times \sin\theta,$$

$$\hat{Y} = X \times \sin\theta + Y \times \cos\theta.$$

To take an inverse of the rotation, you should calculate the rotation in the reverse direction. For example, the inverse of rotating 27 degrees in counter-clockwise is just 27 degrees in clockwise hence -27 degrees in counter-clockwise.

A very crucial point is that the arguments of trigonometric functions in NumPy are in **radians, not degrees**. So, you should convert degrees ($\theta_d$) into radians ($\theta_r$) as follows:

$$\theta_r = \theta_d \times \frac{\pi}{180}.$$

#### 4.2.4 Imprecisions due to floating point precision loss

At this point, after fixing every deformation, we are very close to the original data. X is the days passed since 01.01.2015 and Y is the exchange rate on that day. There is just one little thing we should fix before returning the result. If you inspect days (X values) you'll notice that they have very small decimal parts (maybe even less than $10^{-6}$). This is due to the limitations of floating-point numbers and imprecisions arisen due to those limitations. Since days with a decimal part does not make any sense, you need to round the days to the closest integer value before returning.

Note that the dataset you are given are not equally spaced, so you can see sequences like (0,1,4) in X when you fixed deformations.

### 4.3 `fit_and_predict(dataset, day)`

This function takes a NumPy array (`dataset` – see the Solution Template how this is linked to the other functions) and an `int` (the day of the wanted prediction). It returns a `tuple` with 3 elements all of which are `float`.

The aim of this function is to find a line which "best" describes the given dataset, so that we can make some predictions about future values. In other words, we are interested in finding the following function:

$$y = f(x) = \alpha + \beta x.$$

With this function, we can predict the $y$ value given an $x$ value.

Finding this linear function can be formulated as finding the $\alpha$ and $\beta$ values that minimize the following error:

$$\sum_{i=1}^{n} \left[y_i - (\alpha + \beta x_i)\right]^2 .$$

The minimizing values $\hat{\alpha}$ and $\hat{\beta}$ are as follows:

$$\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x},$$

$$\hat{\beta} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

where $\bar{x}$ and $\bar{y}$ are respectively the averages (mean) of $x$ and $y$ values.

With these estimated $\hat{\alpha}$ and $\hat{\beta}$ values, our model (predictor) becomes:

$$y = \hat{\alpha} + \hat{\beta}x.$$

After finding this model, you should make a prediction for the given '`day`'. Therefore, your return value should be the tuple ($\hat{\alpha}$, $\hat{\beta}$, predicted value $y$).

### 4.4 `plot(dataset, alpha, beta, prediction, day)`

This function takes the `dataset` output of `fix_deformation()`; $\alpha$, $\beta$, and prediction output from the `fit_and_predict()` function and the `day` of the prediction. It does not return anything. Instead, it draws the dataset points, the line of the model using $\alpha$ and $\beta$, and the prediction point. When plotting:

- Label of X-axis should be 'Day'.
- Label of Y-axis should be 'USD'.
- The title of the plot should be 'Exchange Rate'.
- The line using alpha and beta arguments which starts from point 0 towards the prediction point; while including both end-points.

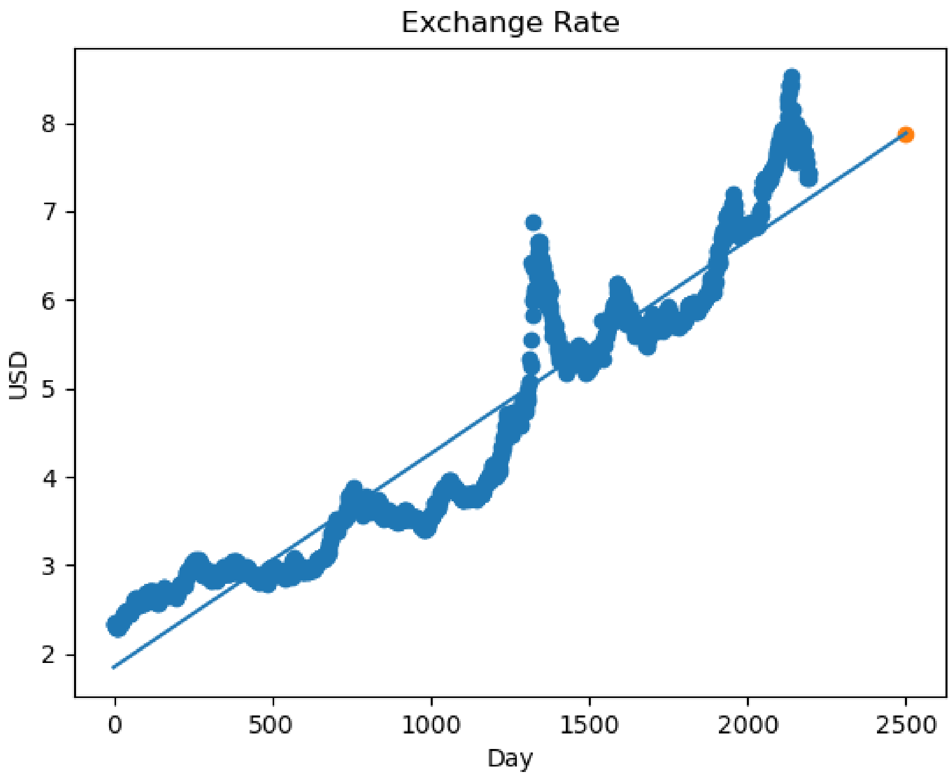
You can find the example output under the `test()` section below.

### 4.5 `test()`

The test function **will not be graded**. It is just to show an example usage of the other functions and to ease your testing/debugging process. You can change the body of this function as you like to test the individual functions different arguments etc., but make sure you do not introduce any errors preventing the code from running since it will result in **0 as your grade** .

The example function reads data from `data.csv` and filters anything whose Y value is bigger than 11. Then, it fixes deformation, creates a predictor based on the result of the deformation operation, and finally makes a prediction about the exchange rate on day 2500 (since 01.01.2015) and plots all on the screen.

You can see the result below. Note that there may be some differences between the appearance of your plot (color etc.) and the example below, but it should be almost identical in terms of the positions of the points in the dataset, the position of the prediction and the line of the predictor. Note that the example test function **will raise an error** until all functions are implemented.


Exchange Rate
USD
Day
2
3
4
5
6
7
8
0
500
1000
1500
2000
2500