Although consecutive measurements may increase or decrease on an opposed direction, the overall behavior persists over time. An example of this is climate; in the Northern Hemisphere, temperatures might decrease by several degrees in May from one day to the next day, but the overall behavior is warming towards the summer. Understanding trends also enables predictive analysis; hence the warnings of global warming. But it’s not all about natural sciences; trends are very important in markets and business.  
In this entry, we will use the same dataset used in the last entry to detect trends in the data.  
First, we will load our dataset:

>>> import pandas as pd

>>> import matplotlib.pyplot as plt

>>> fcdata = pd.read\_csv('flotation-cell.csv', index\_col=0)

>>> print(fcdata.head())

Feed rate Upstream pH CuSO4 added Pulp level \

Date and time

15/12/2004 19:57:01 341.049347 10.820513 7.995605 24.443470

15/12/2004 19:57:31 274.270782 10.827351 7.786569 27.819294

15/12/2004 19:58:01 334.836761 10.854701 7.655922 30.335533

15/12/2004 19:58:32 323.605927 10.885470 7.838828 30.663738

15/12/2004 19:59:03 322.341309 10.851282 7.995605 30.288647

Air flow rate

Date and time

15/12/2004 19:57:01 2.802198

15/12/2004 19:57:31 2.798535

15/12/2004 19:58:01 2.805861

15/12/2004 19:58:32 2.802198

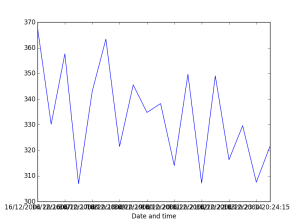
15/12/2004 19:59:03 2.805861

Next, we will select the data we want to adjust:

>>> selected = fcdata.loc[('16/12/2004 20:16:00' < fcdata.index) & (fcdata.index < '16/12/2004 20:25:00'),'Feed rate']

>>> selected.plot()

>>> plt.show()

[](https://i2.wp.com/www.emilkhatib.com/wp-content/uploads/2015/11/figure_2.png)

Original time series

So, how can we use Pandas to find trends in this series? Well, there are many ways, but we will be using an additional library (actually a library used by Pandas in its core): [NumPy](http://www.numpy.org/). NumPy has a lot of interesting mathematical functions, and you might want to have a serious look on it. In this case, we will be using a polynomial fitting function, [polyfit()](http://docs.scipy.org/doc/numpy/reference/generated/numpy.polyfit.html) to find the best adjusting first degree polynomial and its fitting error. A first degree polynomial has two parameters, that we can call the *slope* and the *offset*. If the slope is different from zero, then there is a trend in the data.polyfit() returns several values (see the [documentation](http://docs.scipy.org/doc/numpy/reference/generated/numpy.polyfit.html), but we only need two of them: the coefficients (an array containing the slope and the offset), and the residuals (that gives us a measure of the fitting error and that we will convert into the [Normalized Mean Squared Error (NRMSE)](https://en.wikipedia.org/wiki/Root-mean-square_deviation), that gives us a measurement of the error between 0 and 1).

>>> import numpy as np

>>> coefficients, residuals, \_, \_, \_ = np.polyfit(range(len(selected.index)),selected,1,full=True)

>>> mse = residuals[0]/(len(selected.index))

>>> nrmse = np.sqrt(mse)/(selected.max() - selected.min())

>>> print('Slope ' + str(coefficients[0]))

>>> print('NRMSE: ' + str(nrmse))

Slope -1.72979024566

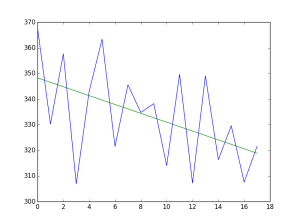
NMRSE: 0.274160734073

We can see that a negative trend is detected, of about -1.73 units of the *Feed rate* per time interval (in this dataset, the time interval is 30 seconds), with an error of about 27.4%. We can now draw the trend line. Remember that the trend line is a polynomial in the Ax+B form:

>>> plt.plot(selected)

>>> plt.plot([coefficients[0]\*x + coefficients[1] for x in range(len(selected))])

>>> plt.show()

[](https://i2.wp.com/www.emilkhatib.com/wp-content/uploads/2015/11/figure_3.png)

Data with the trend line