exp

November 8, 2018

```
In [15]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         sns.set()
```

Setup

```
In [3]: data = pd.read_msgpack("data_2018-11-06-22-59-57.msgpack")
        df = pd.DataFrame.from_dict(data)
        df.columns = ["f", "mdia", "msgtype", "pwm", "rc", "rw", "t", "t0", "timestamp"]
        if(df.iloc[0].timestamp > 0):
            df.timestamp -= df.iloc[0].timestamp
        df.pwm.replace(to_replace=0, value=np.NaN, inplace=True)
        df.f = -df.f
```

Augment w/ minima, rolling mean

```
In [6]: from scipy.signal import argrelmin, argrelmax, argrelextrema
        minima = argrelmin(df.f.values, order=5000)[0]
        maxima = argrelmax(df.f.values, order=5000)[0]
        df['f_ra'] = df.f.rolling(window=300).mean()
```

```
In [12]: df = df.fillna(0)
```

Mean, variance stationarity

```
In [21]: df.f_ra.mean(), df.f_ra.var(), df.f_ra.min(), df.f_ra.max()
```

```
Out[21]: (21.684030983816214,
          103.2979611757999,
          -1.1003944220642248,
          40.441713879903155)
```

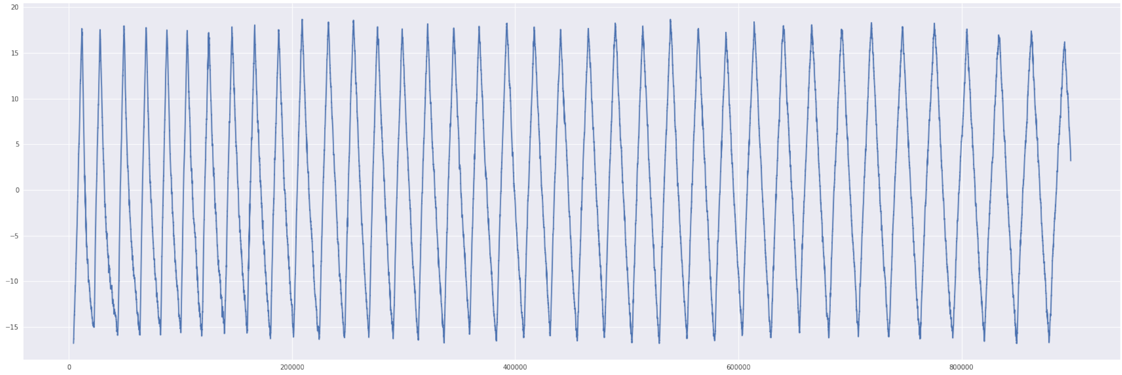
Detrend

```
In [35]: df = df[df.f_ra > 5]
```

```
In [36]: from scipy.signal import detrend
         df['f_ra_det'] = detrend(df.f_ra, axis=-1, type='constant', bp=0)
```

```
In [37]: fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(30, 10))
         plt.plot(df.f_ra_det)
```

```
Out[37]: [<matplotlib.lines.Line2D at 0x7fc4a0595320>]
```



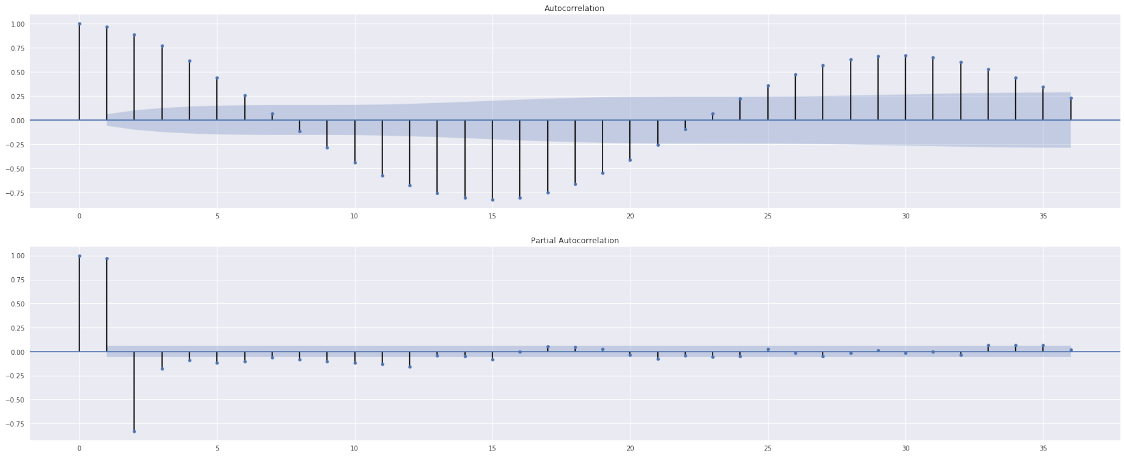
These inform # of lags:

```
In [44]: minima.shape, maxima.shape
```

```
Out[44]: ((36,), (38,))
```

From now on `f_ra_det` is detrended rolling average force. AR terms come from high autocorrelation:

```
In [47]: import statsmodels.api as sm
         fig = plt.figure(figsize=(30,12))
         ax1 = fig.add_subplot(211)
         fig = sm.graphics.tsa.plot_acf(df.f_ra_det.iloc[::800], lags=36, ax=ax1)
         ax2 = fig.add_subplot(212)
         fig = sm.graphics.tsa.plot_pacf(df.f_ra_det[::800], lags=36, ax=ax2)
         plt.show()
```



```
In [96]: df['dt'] = df.timestamp * 100000
         df.dt = pd.to_datetime(df.dt, unit='ms')
         df = df.set_index('dt')
```

```
In [97]: df.head()
```

```
Out[97]:                                       f  mdia  msgtype    pwm    rc    rw  \
         dt
         1970-01-01 02:00:23.610877991  5.464552   1.0        2  150.0  27.5  0.35
         1970-01-01 02:00:24.607276917  4.343770   1.0        2  150.0  27.5  0.35
         1970-01-01 02:00:26.510810852  6.094062   1.0        2  150.0  27.5  0.35
         1970-01-01 02:00:28.495216370  5.250987   1.0        2  150.0  27.5  0.35
         1970-01-01 02:00:30.391883850  6.265777   1.0        2  150.0  27.5  0.35

                                                t         t0  timestamp      f_ra  \
         dt
         1970-01-01 02:00:23.610877991  34.183258  24.727188  72.236109  5.005800
         1970-01-01 02:00:24.607276917  34.183258  24.727188  72.246073  5.006797
         1970-01-01 02:00:26.510810852  34.189365  24.727188  72.265108  5.016541
         1970-01-01 02:00:28.495216370  34.189365  24.727188  72.284952  5.017539
         1970-01-01 02:00:30.391883850  34.193535  24.728174  72.303919  5.021741

                                         f_ra_det
         dt
         1970-01-01 02:00:23.610877991 -16.782868
         1970-01-01 02:00:24.607276917 -16.781871
         1970-01-01 02:00:26.510810852 -16.772127
         1970-01-01 02:00:28.495216370 -16.771130
         1970-01-01 02:00:30.391883850 -16.766927
```

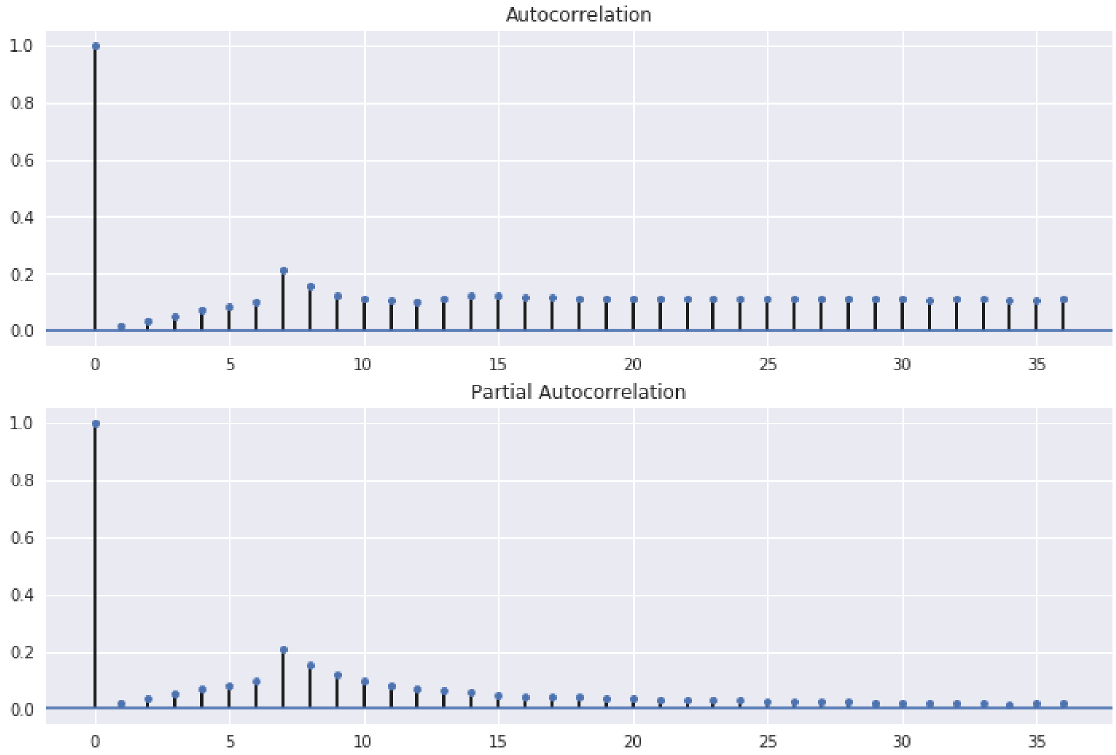
```
In [98]: from statsmodels.tsa.arima_model import ARIMA as ARIMA
```

```
In [100]: model = ARIMA(endog=df.f_ra_det, order=(0,1,6))
          results = model.fit()
```

```
/usr/local/lib/python3.5/dist-packages/statsmodels/tsa/kalmanf/kalmanfilter.py:646: FutureWarnin
  if issubdtype(paramsdtype, float):
/usr/local/lib/python3.5/dist-packages/statsmodels/tsa/kalmanf/kalmanfilter.py:650: FutureWarnin
  elif issubdtype(paramsdtype, complex):
```

```
In [101]: fig = plt.figure(figsize=(12,8))
          ax1 = fig.add_subplot(211)
          fig = sm.graphics.tsa.plot_acf(results.resid, lags=36, ax=ax1)
          ax2 = fig.add_subplot(212)
          fig = sm.graphics.tsa.plot_pacf(results.resid, lags=36, ax=ax2)
          plt.show()
```

```
/usr/local/lib/python3.5/dist-packages/statsmodels/tsa/kalmanf/kalmanfilter.py:577: FutureWarnin
  if issubdtype(paramsdtype, float):
```



```
In [102]: results.summary()
```

```
/usr/local/lib/python3.5/dist-packages/statsmodels/tsa/kalmanf/kalmanfilter.py:646: FutureWarnin
  if issubdtype(paramsdtype, float):
/usr/local/lib/python3.5/dist-packages/statsmodels/tsa/kalmanf/kalmanfilter.py:650: FutureWarnin
  elif issubdtype(paramsdtype, complex):
```

```
Out[102]: <class 'statsmodels.iolib.summary.Summary'>
          """
                                       ARIMA Model Results
          ==============================================================================
          Dep. Variable:             D.f_ra_det   No. Observations:               893408
          Model:                 ARIMA(0, 1, 6)   Log Likelihood             3575244.631
          Method:                       css-mle   S.D. of innovations              0.004
          Date:                Thu, 08 Nov 2018   AIC                     -7150473.263
          Time:                        16:31:28   BIC                     -7150379.640
          Sample:                    01-01-1970   HQIC                    -7150447.381
                                   - 01-19-1970
```

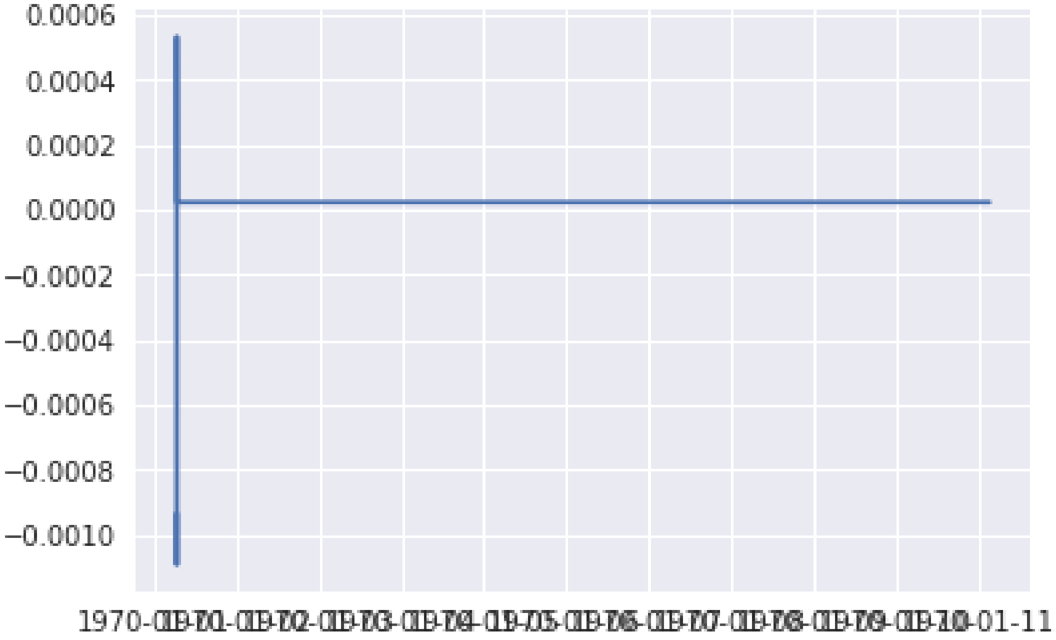
```
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
const              2.234e-05   1.08e-05      2.069      0.039    1.18e-06    4.35e-05
ma.L1.D.f_ra_det      0.2958      0.001    274.120      0.000       0.294       0.298
ma.L2.D.f_ra_det      0.2563      0.001    228.327      0.000       0.254       0.259
ma.L3.D.f_ra_det      0.2454      0.001    218.518      0.000       0.243       0.248
ma.L4.D.f_ra_det      0.2123      0.001    212.506      0.000       0.210       0.214
ma.L5.D.f_ra_det      0.1700      0.001    170.429      0.000       0.168       0.172
ma.L6.D.f_ra_det      0.1277      0.001    127.519      0.000       0.126       0.130
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
MA.1            0.8853            -0.9181j            1.2754           -0.1279
MA.2            0.8853            +0.9181j            1.2754            0.1279
MA.3           -0.2126            -1.4442j            1.4597           -0.2733
MA.4           -0.2126            +1.4442j            1.4597            0.2733
MA.5           -1.3383            -0.6842j            1.5031           -0.4248
MA.6           -1.3383            +0.6842j            1.5031            0.4248
-----------------------------------------------------------------------------
"""
```

```
In [113]: r = results.predict(start = 10000, end = 500000, dynamic= True)
```

```
/usr/local/lib/python3.5/dist-packages/statsmodels/tsa/kalmanf/kalmanfilter.py:577: FutureWarnin
  if issubdtype(paramsdtype, float):
```

```
In [111]: plt.plot(r)
```

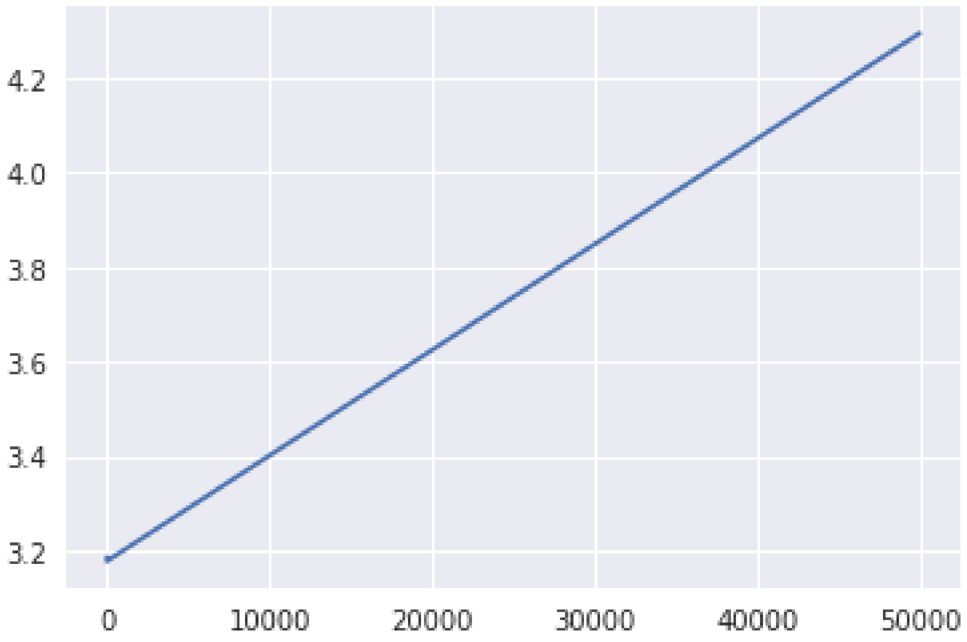
```
Out[111]: [<matplotlib.lines.Line2D at 0x7fc49abe9d68>]
```



```
In [106]: forecast, std, conf = results.forecast(50000)
          plt.plot(forecast)
```

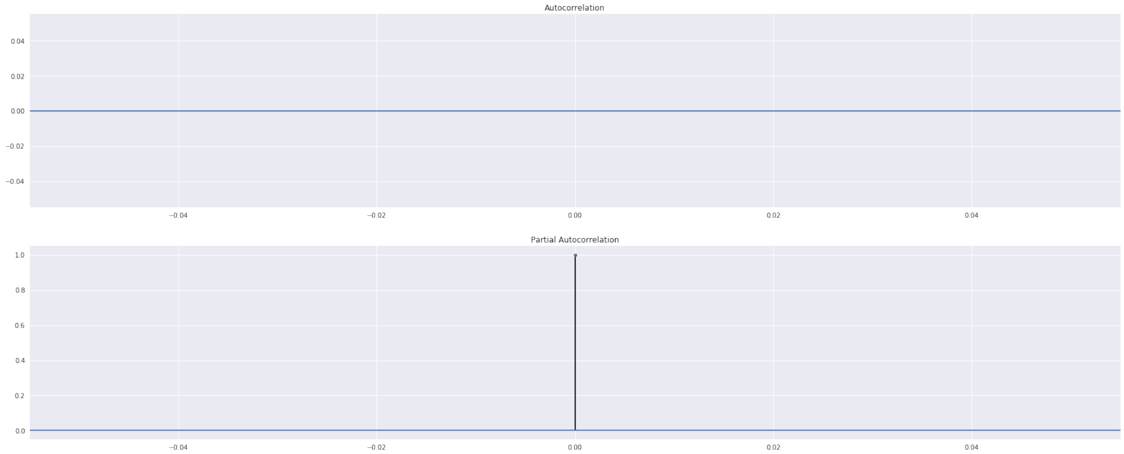
```
/home/nwchen/.local/lib/python3.5/site-packages/scipy/signal/signaltools.py:1336: FutureWarning:
  out = out_full[ind]
```

```
Out[106]: [<matplotlib.lines.Line2D at 0x7fc49accbeb8>]
```



1 GARBAGE BELOW

```
In [41]: fig = plt.figure(figsize=(30,12))
         ax1 = fig.add_subplot(211)
         fig = sm.graphics.tsa.plot_acf(df.f.iloc[::10].diff(), lags=40, ax=ax1)
         ax2 = fig.add_subplot(212)
         fig = sm.graphics.tsa.plot_pacf(df.f.iloc[::10].diff(), lags=40, ax=ax2)
         plt.show()
```



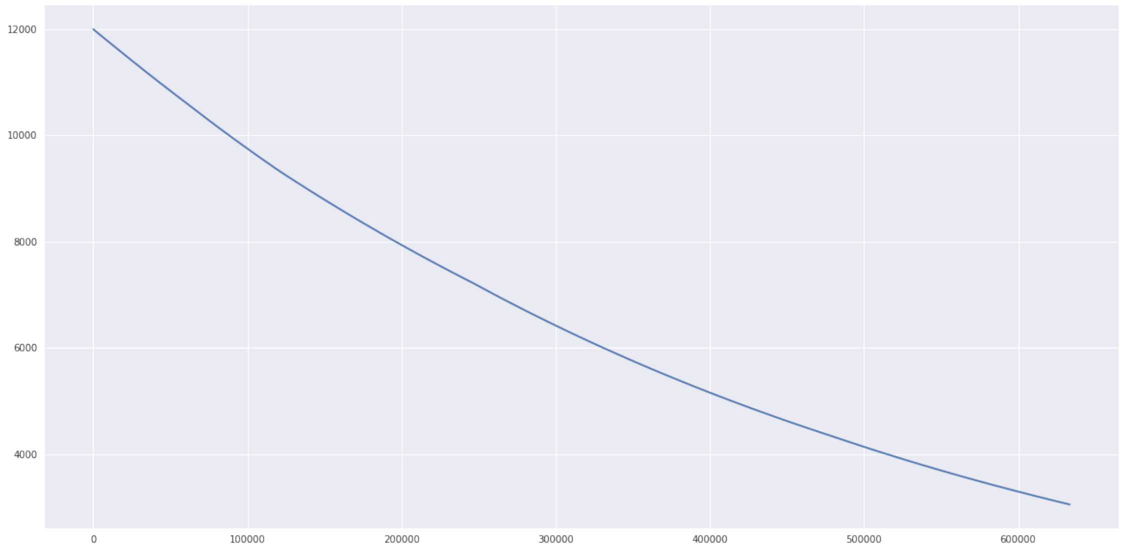
```
In [9]: # t: point in time
        # tc: time constant
        # a: amplitude
        def LOOSE(t, t0, tc, a, p0, is_heating):
            k = 1 if is_heating else -1
            return p0 * (1 + a*(np.exp(k*(t-t0)/(tc)))) # * np.exp(t/tc)
```

```
In [10]: a = df.f.max()
         t0 = 0
         p0 = 1
         tc = 8000
         is_heating = True
```

```
In [11]: loosies = df.timestamp.apply(LOOSE, args=(t0, 2000, a, p0, is_heating))
```

```
In [12]: loosies = df.timestamp.apply(LOOSE, args=(t0, tc, 12000, p0, not is_heating))
         fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(20, 10))
         plt.plot(loosies)
```

```
Out[12]: [<matplotlib.lines.Line2D at 0x7f6318d905c0>]
```



```
In [13]: from sklearn import linear_model
         from sklearn.linear_model import LinearRegression
         from sklearn.preprocessing import PolynomialFeatures
         from sklearn.pipeline import Pipeline
```

```
In [14]: ma = df.f.rolling(window=300).mean().dropna()
         tsma = df.timestamp.rolling(window=300).mean().dropna()
         X = pd.DataFrame({'ma': ma, 'tsma': tsma}) #df.timestamp.values.reshape(df.timestamp.si
         y = ma
```

```
In [15]: X.head()
```

```
Out[15]:            ma      tsma
         299  0.628544  2.456122
         300  0.624040  2.472461
         301  0.622513  2.488817
         302  0.617728  2.505203
         303  0.611173  2.521591
```

```
In [28]: poly = PolynomialFeatures(degree=2)
         poly.fit_transform(X) #x1^0x2^0, ^1^0, ^0^1, ^2^0, ^1^1, ^0^2
```

```
Out[28]: array([[1.00000000e+00, 6.28543919e-01, 2.45612203e+00, 3.95067458e-01,
                 1.54378057e+00, 6.03253542e+00],
                [1.00000000e+00, 6.24040213e-01, 2.47246071e+00, 3.89426188e-01,
                 1.54291491e+00, 6.11306196e+00],
                [1.00000000e+00, 6.22513496e-01, 2.48881742e+00, 3.87523053e-01,
                 1.54932243e+00, 6.19421214e+00],
                ...,
          [1.00000000e+00, 1.01081709e+00, 1.09392940e+04, 1.02175120e+00,
           1.10576253e+04, 1.19668153e+08],
          [1.00000000e+00, 1.00851773e+00, 1.09393123e+04, 1.01710802e+00,
           1.10324904e+04, 1.19668553e+08],
          [1.00000000e+00, 1.00561858e+00, 1.09393306e+04, 1.01126872e+00,
           1.10007941e+04, 1.19668954e+08]])
```

```
In [29]: model = Pipeline([('poly', PolynomialFeatures(degree=2)), ('linear', LinearRegression(f
```

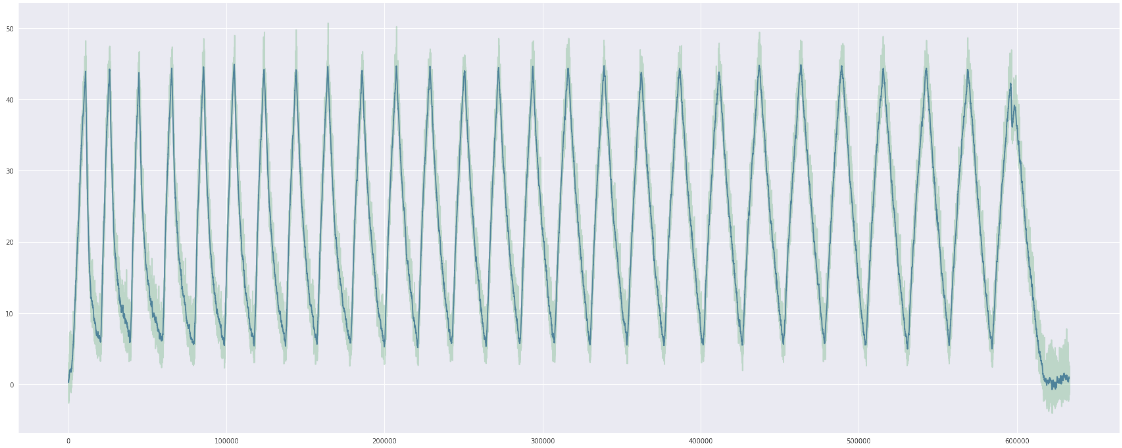
```
In [30]: model = model.fit(X, y)
```

```
In [31]: model.named_steps['linear'].coef_, model.named_steps['linear'].intercept_
```

```
Out[31]: (array([ 5.61427682e-09,  1.00000000e+00, -4.04517457e-14,  1.34544678e-14,
                  5.55978874e-16,  0.00000000e+00]), 0.0)
```

```
In [32]: fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(30,12))
         ax.plot(model.predict(X))
         ax.plot(df.f, alpha=.3)
```

```
Out[32]: [<matplotlib.lines.Line2D at 0x7f2507e3fd30>]
```

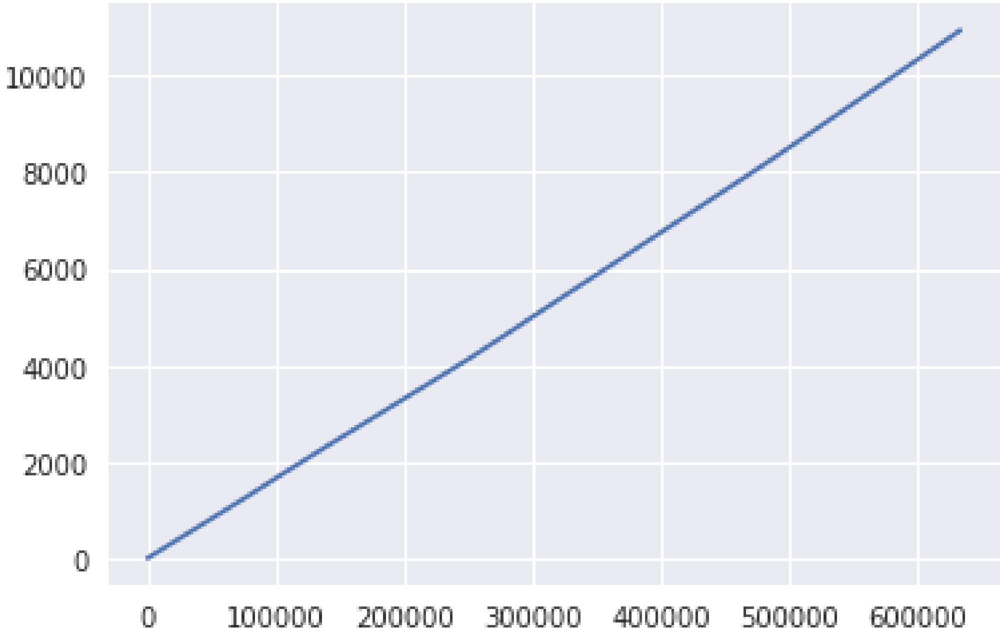


```
In [48]: def EZ_MODEL(x):
             xp = x.values
             coefs = [ 5.61427682e-09,  1.00000000e+00, -4.04517457e-14,  1.34544678e-14, 5.5597
             #coefs = [1, 1, 1, 1, 1]
             yhat = coefs[0]*xp[:,0] + coefs[1]*xp[:,1] + coefs[2]*(xp[:,0]**2) + coefs[3]*(xp[:
             return yhat
```

```
In [49]: yhat = EZ_MODEL(X)
```

```
In [50]: plt.plot(yhat)
```

```
Out[50]: [<matplotlib.lines.Line2D at 0x7f62f36440f0>]
```



1.0.1 single cycle

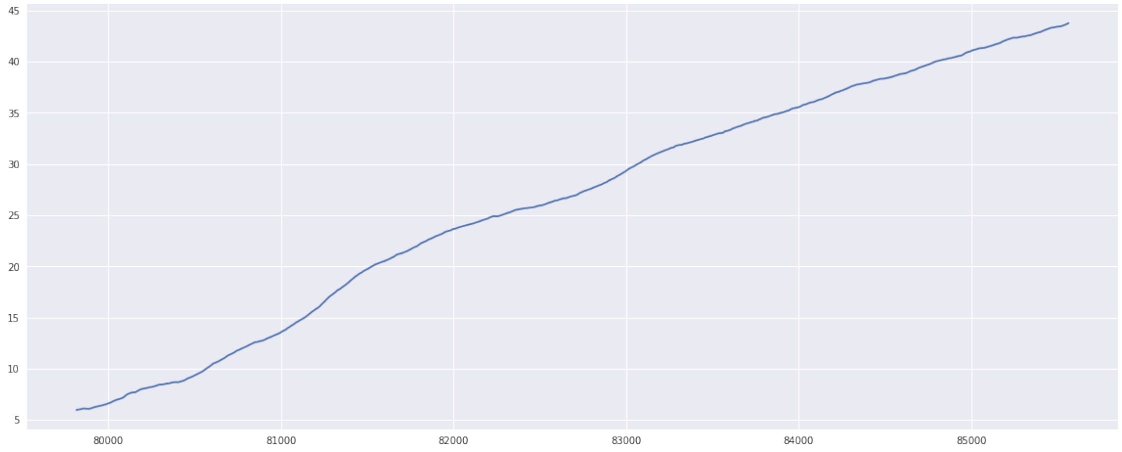
```
In [98]:

/home/nwchen/.local/lib/python3.5/site-packages/scipy/signal/_peak_finding.py:68: RuntimeWarning
  results &= comparator(main, plus)
/home/nwchen/.local/lib/python3.5/site-packages/scipy/signal/_peak_finding.py:69: RuntimeWarning
  results &= comparator(main, minus)
/home/nwchen/.local/lib/python3.5/site-packages/scipy/signal/_peak_finding.py:68: RuntimeWarning
  results &= comparator(main, plus)
/home/nwchen/.local/lib/python3.5/site-packages/scipy/signal/_peak_finding.py:69: RuntimeWarning
  results &= comparator(main, minus)
```

```
In [99]: minima, maxima

Out[99]: (array([   214,  20174,  38313,  79155,  98694, 117433, 136368, 156336,
                 178013, 199924, 221258, 241686, 264319, 285324, 307736, 330042,
                 377151, 400604, 426503, 452401, 477609, 504812, 530607, 557268,
                 584535, 622356]),
          array([ 10975,  26153,  45021,  65632,  85677, 105194, 123991, 144138,
                 164291, 186173, 207596, 228875, 250710, 272266, 294110, 316578,
                 338806, 361751, 387943, 411694, 437241, 463216, 490004, 515464,
                 542936, 569062, 596704, 631609]))
```

1.1 rising front

```
In [115]: plt.figure(figsize=(20,8))
          plt.plot(X.f)
```

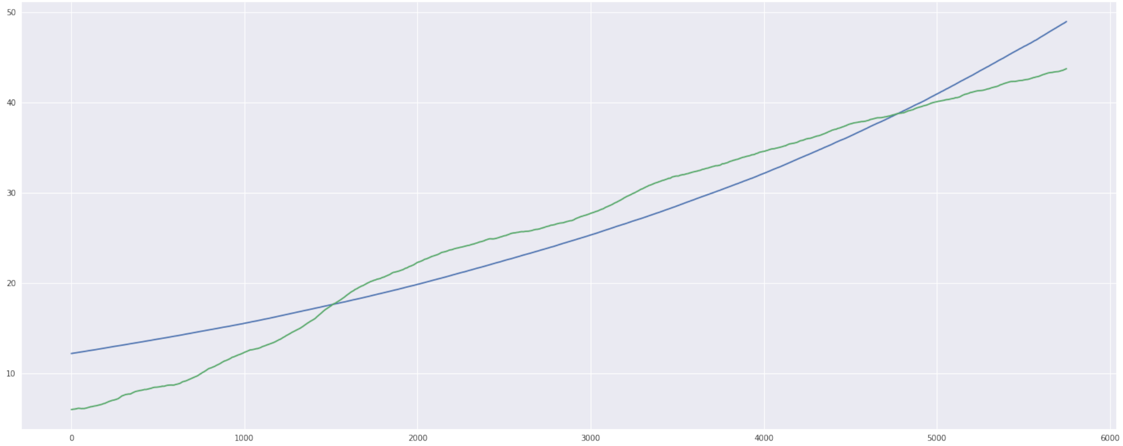
```
Out[115]: [<matplotlib.lines.Line2D at 0x7f62f988e630>]
```



```
In [121]: print(np.where(~np.isfinite(X.f.values)))
```

```
(array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
         13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
         26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
         39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
         52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
         65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
         78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
         91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
        104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
        117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
        130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
        143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
        156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
        169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
        182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
        195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
        208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
        221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
        234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246,
        247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259,
        260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272,
        273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285,
        286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298]),)
```

```
In [122]: np.polyfit(X.timestamp.values[299:], X.f.values[299:], 1)

Out[122]: array([ 4.04202640e-01, -5.25391322e+02])

In [125]: from scipy.optimize import curve_fit

In [154]: curve_fit(lambda t,a,b: a*np.exp(b*t),  X.timestamp.values[299:]-X.timestamp.values[29

Out[154]: (array([12.18783049,  0.01428122]), array([[ 2.99690222e-03, -3.19125356e-06],
                 [-3.19125356e-06,  3.79438365e-09]]))

In [157]: fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(25,10))
          ax.plot(12.18781192*np.exp(0.01428125*(X.timestamp.values[299:]-X.timestamp.values[299
          ax.plot(X.f.values[299:])

Out[157]: [<matplotlib.lines.Line2D at 0x7f62f93a7748>]
```

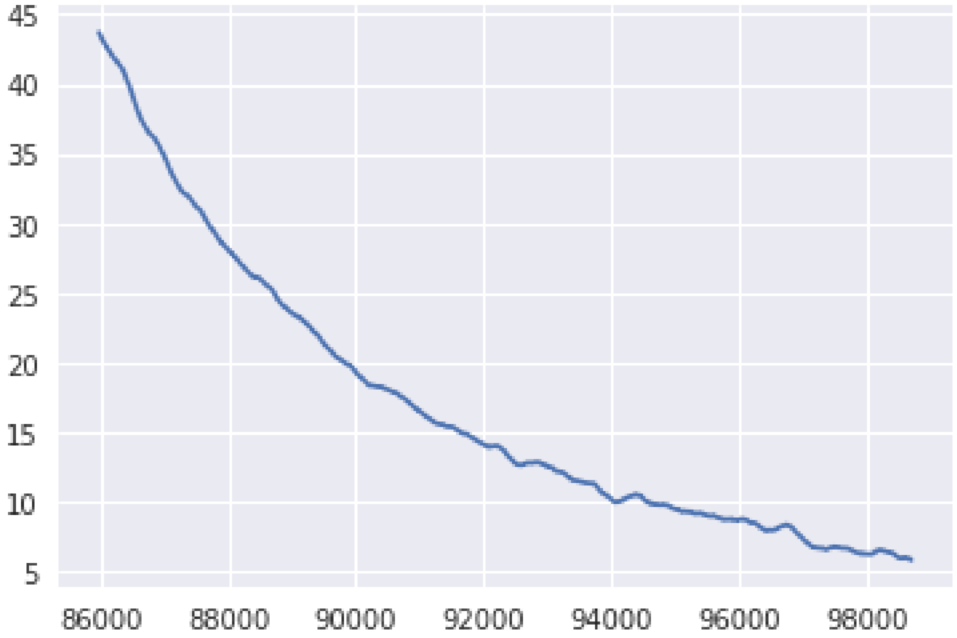


1.2 falling front

```
In [167]: maxima[4], minima[4]

Out[167]: (85677, 98694)

In [174]: X = df.iloc[maxima[4]:minima[4],]
          X.f = X.f.rolling(window=300).mean()

/home/nwchen/.local/lib/python3.5/site-packages/pandas/core/generic.py:4405: SettingWithCopyWarn
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  self[name] = value
```

```
In [175]: plt.plot(X.f)
```

```
Out[175]: [<matplotlib.lines.Line2D at 0x7f62f921ae48>]
```



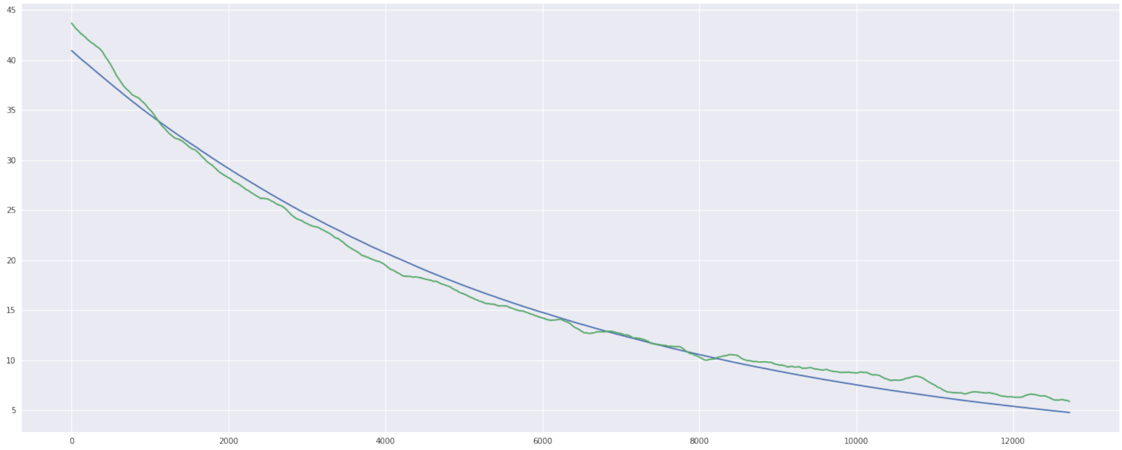
```
In [176]: curve_fit(lambda t,a,b: a*np.exp(b*t),  X.timestamp.values[299:]-X.timestamp.values[29
```

```
Out[176]: (array([ 4.09413046e+01, -9.93004214e-03]),
           array([[ 7.38156710e-04, -2.06443828e-07],
                  [-2.06443828e-07,  1.06211435e-10]]))
```

$$\hat{F} = F(t)_A = Ae^{\frac{t-t_0}{\tau}} = 4.09e^{-0.00993t}$$

```
In [178]: fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(25,10))
          ax.plot(4.09413046e+01*np.exp(-9.93004214e-03*(X.timestamp.values[299:]-X.timestamp.va
          ax.plot(X.f.values[299:])
```

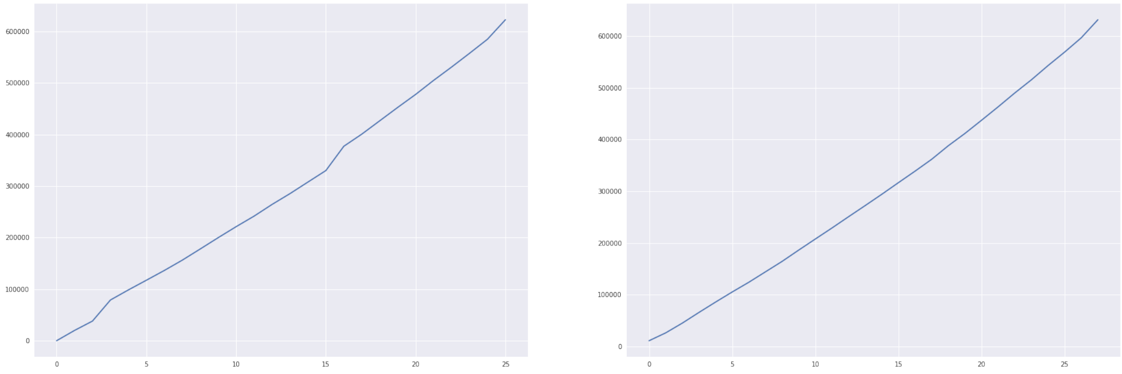
```
Out[178]: [<matplotlib.lines.Line2D at 0x7f62f91a3da0>]
```



1.2.1 deriving `tc` from period trend (`a` update)

```
In [183]: fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(30, 10))
          ax.flat[0].plot(minima)
          ax.flat[1].plot(maxima)
```

```
Out[183]: [<matplotlib.lines.Line2D at 0x7f62f9050390>]
```



```
In [191]: def PERIODS(extrema, data):
              p = np.empty(extrema.shape)
              for i in range(1, extrema.shape[0]):
                  p[i] = data[extrema[i]] - data[extrema[i-1]]
              return p
```

```
In [194]: pmin = PERIODS(minima, df.timestamp.values)
          pmax = PERIODS(maxima, df.timestamp.values)
```

```
In [212]: rangemin = np.arange(pmin.shape[0]).reshape(-1, 1)
          rangemax = np.arange(pmax.shape[0]).reshape(-1, 1)
```

```
In [235]: from sklearn.linear_model import Ridge
```

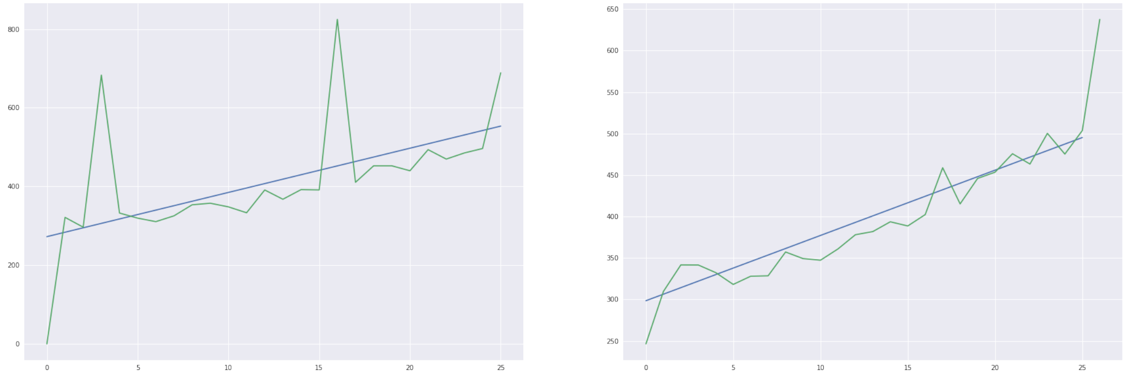
```
In [240]: regmin = Ridge(alpha=10).fit(rangemin, pmin)
          regmax = Ridge(alpha=10).fit(rangemax[3:-2], pmax[3:-2])
```

$$\Delta\text{period} = |t^{i}_{min} - t^{i-1}_{min}|$$

over time

```
In [241]: fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(30, 10))
          ax.flat[0].plot(regmin.predict(rangemin))
          ax.flat[0].plot(pmin)
          ax.flat[1].plot(regmax.predict(rangemax[2:]))
          ax.flat[1].plot(pmax[1:])[0]
```

```
Out[241]: <matplotlib.lines.Line2D at 0x7f62f5852048>
```



```
In [244]: def regmin.coef_, regmin.intercept_, regmax.coef_, regmax.intercept_
```

```
Out[244]: (array([11.23701196]), 272.51819267038803, array([7.8670855]), 282.77006235401)
```

```
In [246]: def SRM(t0, t, pwm):
              if pwm==0:
                  return 4.09413046e+01*np.exp(-9.93004214e-03*(t-t0))
              if pwm==1:
                  return 12.18781192*np.exp(0.01428125*(t-t0))
              return -1
```

```
In [250]: df.pwm.fillna(value=0.0, inplace=True)
```

```
In [252]: df
```

```
Out[252]:                 f  mdia  msgtype  pwm    rc   rw          t         t0  \
          0        1.924732   1.0        2  0.0  29.1  0.4  29.597118  26.686825
          1        1.903733   1.0        2  0.0  29.1  0.4  29.597118  26.686825
          2        1.318297   1.0        2  0.0  29.1  0.4  29.596083  26.689442
          3        0.999092   1.0        2  0.0  29.1  0.4  29.596083  26.689442
          4        2.489463   1.0        2  0.0  29.1  0.4  29.599459  26.689442
          5        0.499448   1.0        2  0.0  29.1  0.4  29.599459  26.689442
          6        0.733154   1.0        2  0.0  29.1  0.4  29.596083  26.689442
          7        3.075488   1.0        2  0.0  29.1  0.4  29.596083  26.689442
          8        1.956969   1.0        2  0.0  29.1  0.4  29.597118  26.686825
          9        1.211885   1.0        2  0.0  29.1  0.4  29.597118  26.686825
          10       0.679974   1.0        2  0.0  29.1  0.4  29.597118  26.686825
          11       2.542726   1.0        2  0.0  29.1  0.4  29.593742  26.686825
          12       1.637589   1.0        2  0.0  29.1  0.4  29.593742  26.686825
          13       0.520448   1.0        2  0.0  29.1  0.4  29.593742  26.686825
          14       3.022201   1.0        2  0.0  29.1  0.4  29.596083  26.689442
          15       1.105484   1.0        2  0.0  29.1  0.4  29.596083  26.689442
          16       1.424718   1.0        2  0.0  29.1  0.4  29.597118  26.686825
          17       1.765039   1.0        2  0.0  29.1  0.4  29.597118  26.686825
          18       1.318297   1.0        2  0.0  29.1  0.4  29.597118  26.683172
          19       2.116691   1.0        2  0.0  29.1  0.4  29.597118  26.683172
          20       2.084448   1.0        2  0.0  29.1  0.4  29.596083  26.685804
          21       2.084448   1.0        2  0.0  29.1  0.4  29.596083  26.685804
          22       2.095691   1.0        2  0.0  29.1  0.4  29.596083  26.685804
          23       1.126484   1.0        2  0.0  29.1  0.4  29.596083  26.689442
          24       0.807337   1.0        2  0.0  29.1  0.4  29.596083  26.689442
          25       0.978092   1.0        2  0.0  29.1  0.4  29.596083  26.685804
          26       1.818268   1.0        2  0.0  29.1  0.4  29.596083  26.685804
          27       2.084448   1.0        2  0.0  29.1  0.4  29.596083  26.685804
          28       0.892709   1.0        2  0.0  29.1  0.4  29.596083  26.685804
          29       1.126484   1.0        2  0.0  29.1  0.4  29.596083  26.685804
          ...           ...   ...      ...  ...   ...  ...        ...        ...
          633318   0.387371   1.0        2  0.0  29.1  0.4  62.283234  26.961542
          633319   0.938646   1.0        2  0.0  29.1  0.4  62.283234  26.961542
          633320   1.628362   1.0        2  0.0  29.1  0.4  62.277489  26.963123
          633321   0.917646   1.0        2  0.0  29.1  0.4  62.277489  26.963123
          633322   1.819663   1.0        2  0.0  29.1  0.4  62.275467  26.927490
          633323  -0.195656   1.0        2  0.0  29.1  0.4  62.275467  26.927490
          633324   0.493407   1.0        2  0.0  29.1  0.4  62.271317  26.948685
          633325   1.798663   1.0        2  0.0  29.1  0.4  62.271317  26.948685
          633326   1.108824   1.0        2  0.0  29.1  0.4  62.273129  26.939804
          633327   0.440388   1.0        2  0.0  29.1  0.4  62.273129  26.939804
          633328   0.440388   1.0        2  0.0  29.1  0.4  62.271530  26.949389
          633329   1.766584   1.0        2  0.0  29.1  0.4  62.271530  26.949389
          633330   1.872745   1.0        2  0.0  29.1  0.4  62.271530  26.949389
          633331  -1.096132   1.0        2  0.0  29.1  0.4  62.271530  26.949389
          633332   0.917646   1.0        2  0.0  29.1  0.4  62.271530  26.949389
          633333  -0.778395   1.0        2  0.0  29.1  0.4  62.270428  26.984764
```

```
633334  0.281345   1.0         2  0.0  29.1  0.4  62.265339  26.979065
633335  0.069321   1.0         2  0.0  29.1  0.4  62.265339  26.979065
633336  1.044730   1.0         2  0.0  29.1  0.4  62.265339  26.979065
633337 -0.301630   1.0         2  0.0  29.1  0.4  62.263035  26.947157
633338  1.182874   1.0         2  0.0  29.1  0.4  62.263035  26.947157
633339  0.334357   1.0         2  0.0  29.1  0.4  62.260593  26.946453
633340  1.660434   1.0         2  0.0  29.1  0.4  62.260593  26.946453
633341  0.493407   1.0         2  0.0  29.1  0.4  62.261169  26.962215
633342  0.440388   1.0         2  0.0  29.1  0.4  62.261169  26.962215
633343  0.970687   1.0         2  0.0  29.1  0.4  62.260235  26.956993
633344  0.334357   1.0         2  0.0  29.1  0.4  62.260235  26.956993
633345  1.342039   1.0         2  0.0  29.1  0.4  62.255348  26.944622
633346  1.055776   1.0         2  0.0  29.1  0.4  62.255348  26.944622
633347  0.610479   1.0         2  0.0  29.1  0.4  62.255562  26.948936

           timestamp
0           0.000000
1           0.013638
2           0.023941
3           0.042946
4           0.062157
5           0.081833
6           0.100975
7           0.120380
8           0.139790
9           0.149719
10          0.168900
11          0.188614
12          0.198624
13          0.208733
14          0.227680
15          0.246845
16          0.265938
17          0.285717
18          0.304783
19          0.324562
20          0.343571
21          0.353577
22          0.373623
23          0.392094
24          0.402576
25          0.421684
26          0.440735
27          0.460625
28          0.479805
29          0.498833
...              ...
633318  10941.518303
```

```
633319  10941.536858
633320  10941.557365
633321  10941.575920
633322  10941.595451
633323  10941.614983
633324  10941.634514
633325  10941.654045
633326  10941.672600
633327  10941.693108
633328  10941.711662
633329  10941.731194
633330  10941.750725
633331  10941.770256
633332  10941.789787
633333  10941.809319
633334  10941.828850
633335  10941.847404
633336  10941.867912
633337  10941.886467
633338  10941.905998
633339  10941.925529
633340  10941.945061
633341  10941.964592
633342  10941.984123
633343  10942.002678
633344  10942.022209
633345  10942.042717
633346  10942.061272
633347  10942.080803

[633348 rows x 9 columns]
```

```
In [251]: SRM(0, df.timestamp.values, df.pwm)
```

```
    ---------------------------------------------------------------------------

    ValueError                                Traceback (most recent call last)

    <ipython-input-251-35ce38ec35a9> in <module>()
----> 1 SRM(0, df.timestamp.values, df.pwm)


    <ipython-input-246-af7403ee2876> in SRM(t0, t, pwm)
      1 def SRM(t0, t, pwm):
----> 2     if pwm==0:
      3         return 4.09413046e+01*np.exp(-9.93004214e-03*(t-t0))
      4     if pwm==1:
```

```
      5         return 12.18781192*np.exp(0.01428125*(t-t0))


    ~/.local/lib/python3.5/site-packages/pandas/core/generic.py in __nonzero__(self)
   1574         raise ValueError("The truth value of a {0} is ambiguous. "
   1575                          "Use a.empty, a.bool(), a.item(), a.any() or a.all()."
-> 1576                          .format(self.__class__.__name__))
   1577 
   1578     __bool__ = __nonzero__


    ValueError: The truth value of a Series is ambiguous. Use a.empty, a.bool(), a.item(), a
```