STATS IN PYTHON

NOV 4, 2016

STATSMODEL.API

```
import numpy as np
import statsmodels.api as sm

# Fit regression model
results = sm.OLS(y, X).fit()

# Inspect the results
print results.summary()
```

http://statsmodels.sourceforge.net/stable/

STATSMODEL.API

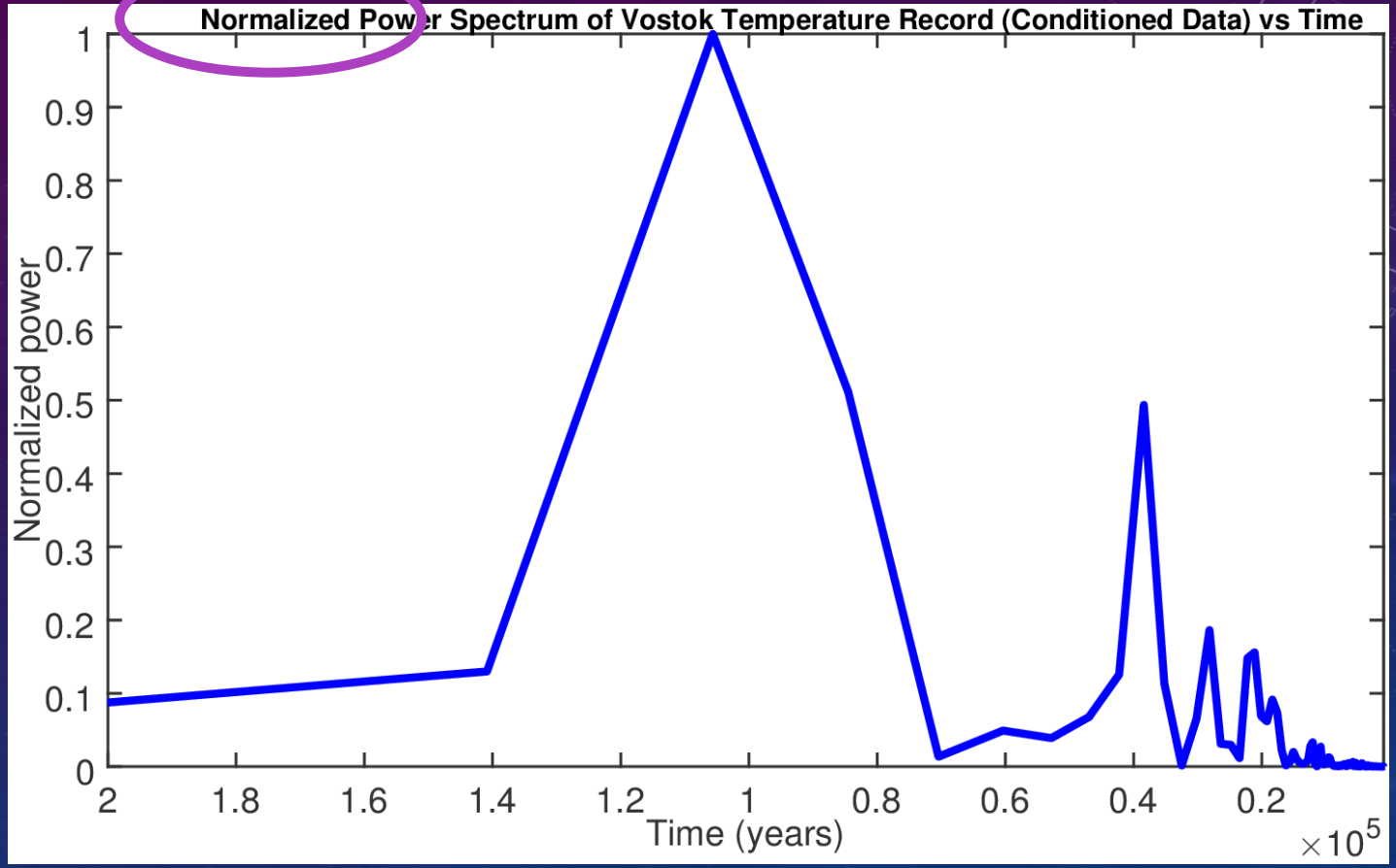
```
nobs = len(year) # Number of observations
nrgs = 5        # Number of regressors + 1 (1 is for the y-intercept const.)

 # Create a matrix of 1s with the dimentions defined above
# Populate the matrix columns with regressors (leaving column 0 as 1's)
regressors = np.ones((nobs,nrgs),dtype = np.float)

regressors[:,1] = tsi
regressors[:,2] = VAR

# Create an ordinary least squares model
model = sm.OLS(obs_o3,regressors)# Extract the constants
constants = model.fit().params
```

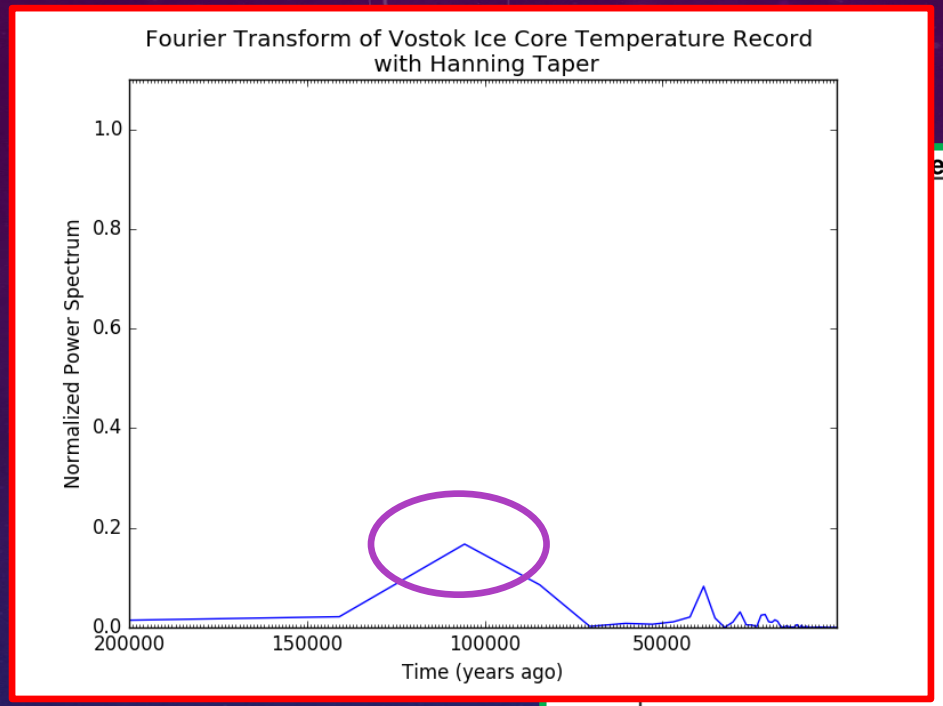
STATSMODEL.API

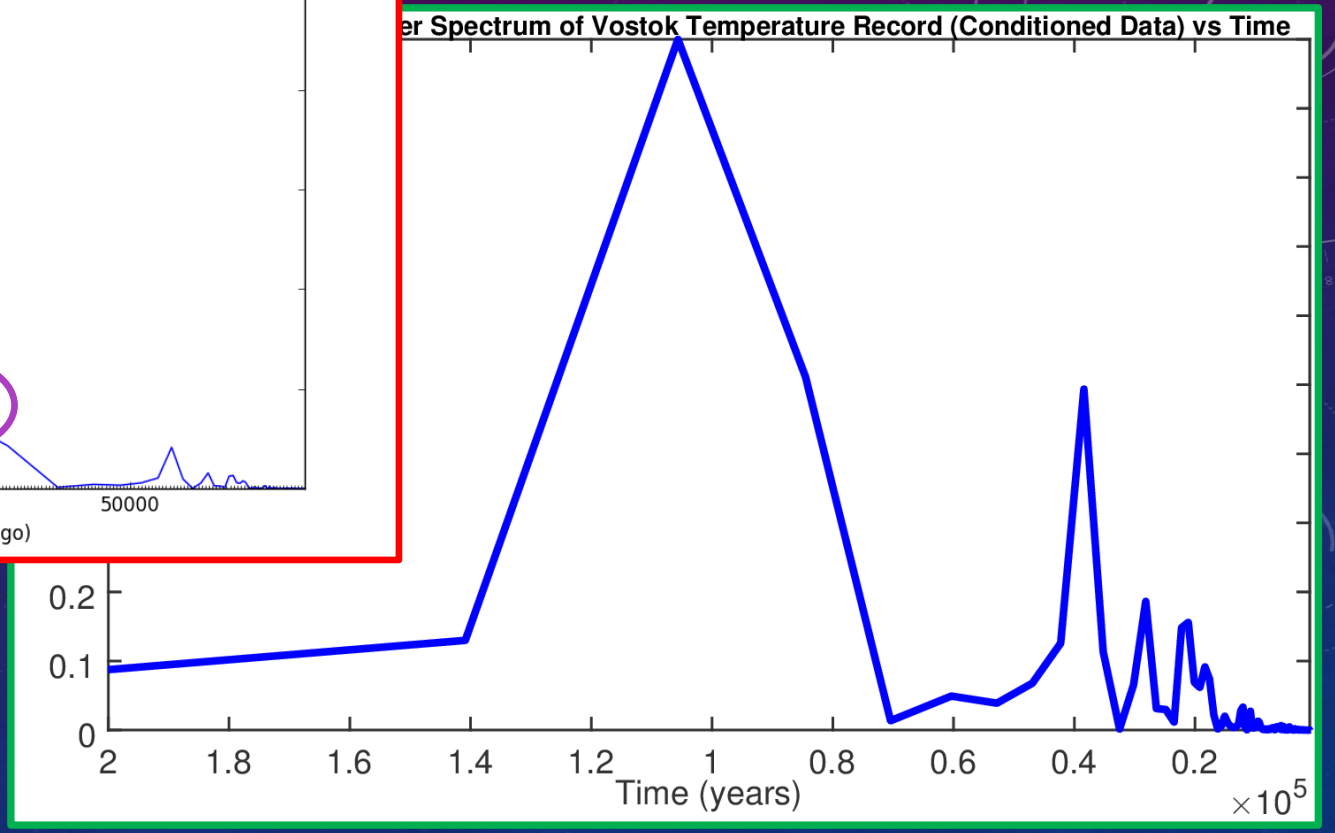
```
# Calculate the model fit result
result = constants[0] + constants[1]*tsi + constants[2]*halo + constants[3]*aero + constants[4]*qbo

plt.plot(year, result, label = 'MLR w/ Python')

print model.fit().summary()
```

FOURIER TRANSFORM HW REVIEW

NOV 4, 2016


Normalized Power Spectrum of Vostok Temperature Record (Conditioned Data) vs Time
Normalized power
Time (years)
×10^5
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2


Fourier Transform of Vostok Ice Core Temperature Record
with Hanning Taper
Normalized Power Spectrum
1.0
0.8
0.6
0.4
0.2
0.0
200000
150000
100000
50000
Time (years ago)


er Spectrum of Vostok Temperature Record (Conditioned Data) vs Time
0.2
0.1
0
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
Time (years)
×10^5

```
power = power/np.amax(power[0:n/2])

power = power/np.amax(power[1:n/2])

power = power/np.amax(power[2:n/2])

power = power/np.amax(power[4:n/2])
```