00-Introduction-to-Statsmodels

October 19, 2022

For more information, visit us at www.pieriandata.com

1 Introduction to Statsmodels

Statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. An extensive list of result statistics are available for each estimator. The results are tested against existing statistical packages to ensure that they are correct. The package is released under the open source Modified BSD (3-clause) license. The online documentation is hosted at statsmodels.org. The statsmodels version used in the development of this course is 0.9.0.

For Further Reading:

Statsmodels Tutorial: Time Series Analysis

Let's walk through a very simple example of using statsmodels!

1.0.1 Perform standard imports and load the dataset

For these exercises we'll be using a statsmodels built-in macroeconomics dataset:

NOTE: Although we've provided a .csv file in the Data folder, you can also build this DataFrame with the following code: import pandas as pd import statsmodels.api as sm df = sm.datasets.macrodata.load_pandas().data df.index = pd.Index(sm.tsa.datetools.dates_from_range('1959Q1', '2009Q3')) print(sm.datasets.macrodata.NOTE)

[1]:
```
import numpy as np
import pandas as pd
%matplotlib inline

df = pd.read_csv('../Data/macrodata.csv',index_col=0,parse_dates=True)
df.head()
```

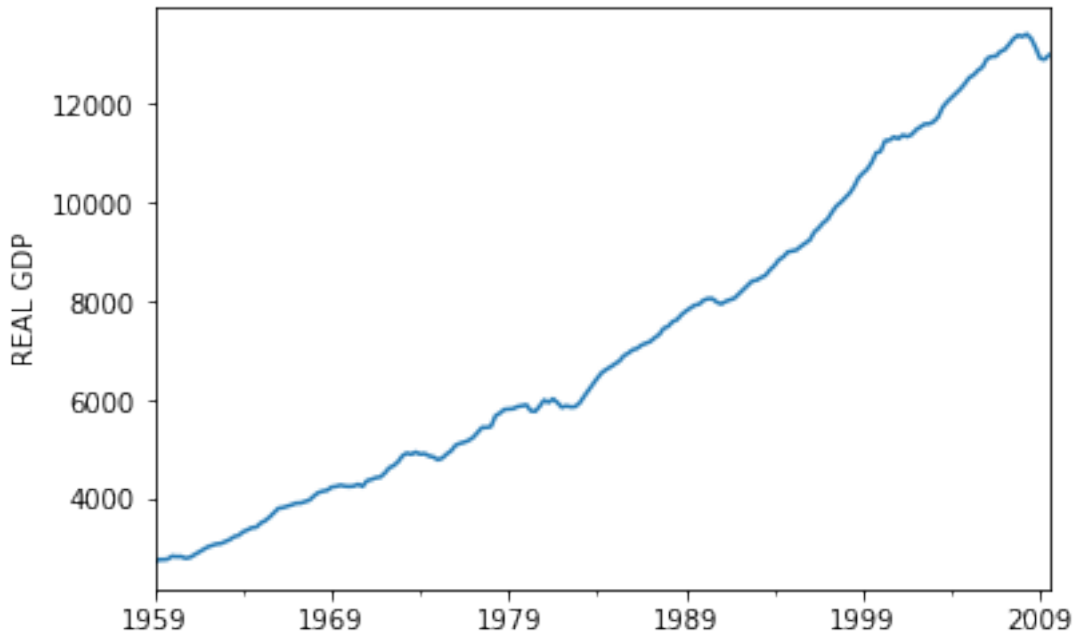
[1]:

	year	quarter	realgdp	realcons	realinv	realgovt	realdpi	\
1959-03-31	1959	1	2710.349	1707.4	286.898	470.045	1886.9	
1959-06-30	1959	2	2778.801	1733.7	310.859	481.301	1919.7	
1959-09-30	1959	3	2775.488	1751.8	289.226	491.260	1916.4	
1959-12-31	1959	4	2785.204	1753.7	299.356	484.052	1931.3	
1960-03-31	1960	1	2847.699	1770.5	331.722	462.199	1955.5	

	cpi	m1	tbilrate	unemp	pop	infl	realint
1959-03-31	28.98	139.7	2.82	5.8	177.146	0.00	0.00
1959-06-30	29.15	141.7	3.08	5.1	177.830	2.34	0.74
1959-09-30	29.35	140.5	3.82	5.3	178.657	2.74	1.09
1959-12-31	29.37	140.0	4.33	5.6	179.386	0.27	4.06
1960-03-31	29.54	139.6	3.50	5.2	180.007	2.31	1.19

1.0.2 Plot the dataset

[2]:
```
ax = df['realgdp'].plot()
ax.autoscale(axis='x',tight=True)
ax.set(ylabel='REAL GDP');
```



1.1 Using Statsmodels to get the trend

Related Function:

statsmodels.tsa.filters.hp_filter.hpfilter(X, lamb=1600) Hodrick-Prescott filter

The Hodrick-Prescott filter separates a time-series y_t into a trend component τ_t and a cyclical component c_t

$y_t = \tau_t + c_t$

The components are determined by minimizing the following quadratic loss function, where λ is a smoothing parameter:

$\min_{\tau_t} \sum_{t=1}^{T} c_t^2 + \lambda \sum_{t=1}^{T} [(\tau_t - \tau_{t-1}) - (\tau_{t-1} - \tau_{t-2})]^2$

The λ value above handles variations in the growth rate of the trend component.When analyzing quarterly data, the default lambda value of 1600 is recommended. Use 6.25 for annual data, and 129,600 for monthly data.

[3]:
```
from statsmodels.tsa.filters.hp_filter import hpfilter

# Tuple unpacking
gdp_cycle, gdp_trend = hpfilter(df['realgdp'], lamb=1600)
```

[4]:
```
gdp_cycle
```

[4]:
```
1959-03-31     39.511915
1959-06-30     80.088532
1959-09-30     48.875455
1959-12-31     30.591933
1960-03-31     64.882667
1960-06-30     23.040242
1960-09-30     -1.355312
1960-12-31    -67.462365
1961-03-31    -81.367438
1961-06-30    -60.167890
1961-09-30    -46.369224
1961-12-31    -20.695339
1962-03-31     -2.162153
1962-06-30     -4.718648
1962-09-30    -13.556457
1962-12-31    -44.369262
1963-03-31    -43.320274
1963-06-30    -44.546971
1963-09-30    -26.298758
1963-12-31    -44.261196
1964-03-31    -14.434412
1964-06-30    -20.266867
1964-09-30    -19.137001
1964-12-31    -54.824590
1965-03-31    -15.962445
1965-06-30    -13.740115
1965-09-30     13.254828
```

```
1965-12-31     56.030402
1966-03-31    103.074337
1966-06-30     72.175348
                  …
2002-06-30    -95.260035
2002-09-30   -114.798768
2002-12-31   -190.025905
2003-03-31   -221.225647
2003-06-30   -207.139428
2003-09-30    -89.685415
2003-12-31    -61.895316
2004-03-31    -56.628782
2004-06-30    -49.616781
2004-09-30    -38.362890
2004-12-31     -8.956672
2005-03-31     39.070285
2005-06-30     18.652990
2005-09-30     42.798035
2005-12-31     39.627354
2006-03-31    141.269129
2006-06-30    125.653779
2006-09-30     70.676428
2006-12-31    110.887665
2007-03-31     99.564908
2007-06-30    157.161271
2007-09-30    231.874638
2007-12-31    263.554667
2008-03-31    204.422097
2008-06-30    221.373942
2008-09-30    102.018455
2008-12-31   -107.269472
2009-03-31   -349.047706
2009-06-30   -397.557073
2009-09-30   -333.115243
Name: realgdp, Length: 203, dtype: float64
```

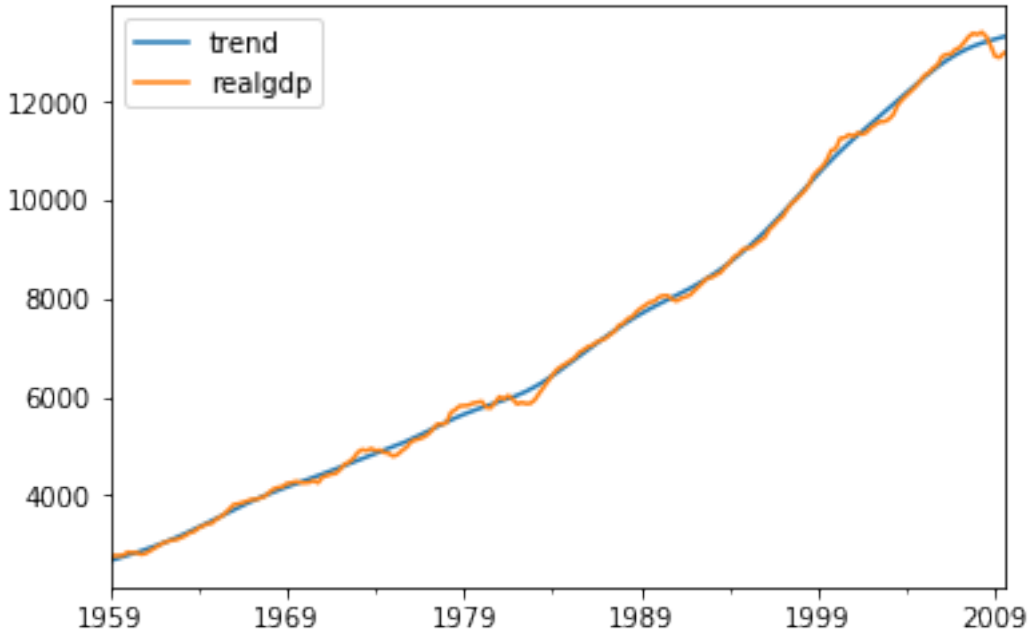
We see from these numbers that for the period from 1960-09-30 to 1965-06-30 actual values fall below the trendline.

[5]:
```
type(gdp_cycle)
```

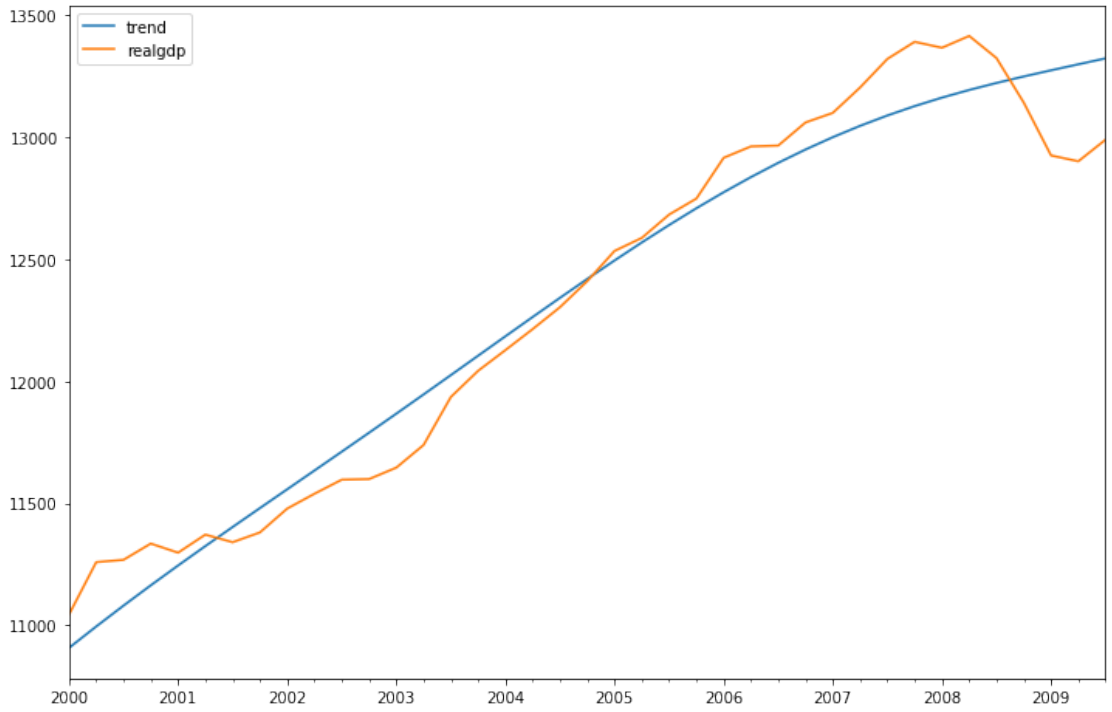
[5]:
```
pandas.core.series.Series
```

[6]:
```
df['trend'] = gdp_trend
```

[7]:
```
df[['trend','realgdp']].plot().autoscale(axis='x',tight=True);
```



[8]:
```
df[['trend','realgdp']]['2000-03-31':].plot(figsize=(12,8)).
↪autoscale(axis='x',tight=True);
```



1.2 Great job!