
(http://www.pieriandata.com)

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 reason we will cover it for use in this course, is that you may find it very useful later on when discussing time series data (typical of quantitative financial analysis).

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

In [17]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [18]:
```
# You can safely ignore the warning:
#  Please use the pandas.tseries module instead. from pandas.core import datetoo
ls
import statsmodels.api as sm
```

In [19]:
```
df = sm.datasets.macrodata.load_pandas().data
```

In [20]:

```
print(sm.datasets.macrodata.NOTE)
```

```
::
    Number of Observations - 203

    Number of Variables - 14

    Variable name definitions::

        year      - 1959q1 - 2009q3
        quarter   - 1-4
        realgdp   - Real gross domestic product (Bil. of chained 200
5 US$,
                    seasonally adjusted annual rate)
        realcons  - Real personal consumption expenditures (Bil. of
chained
                    2005 US$, seasonally adjusted annual rate)
        realinv   - Real gross private domestic investment (Bil. of
chained
                    2005 US$, seasonally adjusted annual rate)
        realgovt  - Real federal consumption expenditures & gross in
vestment
                    (Bil. of chained 2005 US$, seasonally adjusted a
nnual rate)
        realdpi   - Real private disposable income (Bil. of chained
2005
                    US$, seasonally adjusted annual rate)
        cpi       - End of the quarter consumer price index for all
urban
                    consumers: all items (1982-84 = 100, seasonally
adjusted).
        m1        - End of the quarter M1 nominal money stock (Seaso
nally
                    adjusted)
        tbilrate  - Quarterly monthly average of the monthly 3-month
                    treasury bill: secondary market rate
        unemp     - Seasonally adjusted unemployment rate (%)
        pop       - End of the quarter total population: all ages in
cl. armed
                    forces over seas
        infl      - Inflation rate (ln(cpi_{t}/cpi_{t-1}) * 400)
        realint   - Real interest rate (tbilrate - infl)
```

In [21]:

```
df.head()
```

Out[21]:

	year	quarter	realgdp	realcons	realinv	realgovt	realdpi	cpi	m1	tbilrate	unen
0	1959.0	1.0	2710.349	1707.4	286.898	470.045	1886.9	28.98	139.7	2.82	5
1	1959.0	2.0	2778.801	1733.7	310.859	481.301	1919.7	29.15	141.7	3.08	5
2	1959.0	3.0	2775.488	1751.8	289.226	491.260	1916.4	29.35	140.5	3.82	5
3	1959.0	4.0	2785.204	1753.7	299.356	484.052	1931.3	29.37	140.0	4.33	5
4	1960.0	1.0	2847.699	1770.5	331.722	462.199	1955.5	29.54	139.6	3.50	5

In [22]:

```
index = pd.Index(sm.tsa.datetools.dates_from_range('1959Q1', '2009Q3'))
```

In [23]:

```
df.index = index
```

In [24]:

```
df.head()
```

Out[24]:

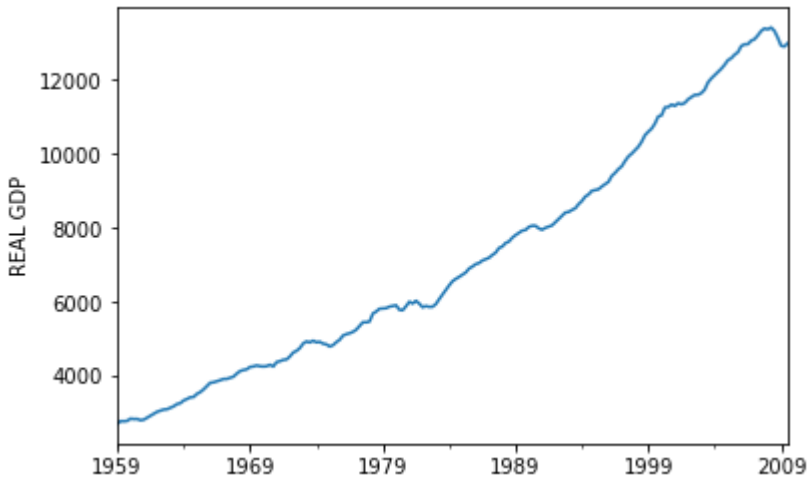
	year	quarter	realgdp	realcons	realinv	realgovt	realdpi	cpi	m1	tbilrate
1959-03-31	1959.0	1.0	2710.349	1707.4	286.898	470.045	1886.9	28.98	139.7	2.82
1959-06-30	1959.0	2.0	2778.801	1733.7	310.859	481.301	1919.7	29.15	141.7	3.08
1959-09-30	1959.0	3.0	2775.488	1751.8	289.226	491.260	1916.4	29.35	140.5	3.82
1959-12-31	1959.0	4.0	2785.204	1753.7	299.356	484.052	1931.3	29.37	140.0	4.33
1960-03-31	1960.0	1.0	2847.699	1770.5	331.722	462.199	1955.5	29.54	139.6	3.50

In [26]:

```
df['realgdp'].plot()
plt.ylabel("REAL GDP")
```

Out[26]:

```
<matplotlib.text.Text at 0x21b98304860>
```



Using Statsmodels to get the trend

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

$y_t = \tau_t + \zeta_t$

The components are determined by minimizing the following quadratic loss function

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

In [30]:

```
# Tuple unpacking
gdp_cycle, gdp_trend = sm.tsa.filters.hpfilter(df.realgdp)
```

In [31]:

```
gdp_cycle
```

Out[31]:

```
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
```

In [29]:

```
type(gdp_cycle)
```

Out[29]:

```
pandas.core.series.Series
```

In [36]:

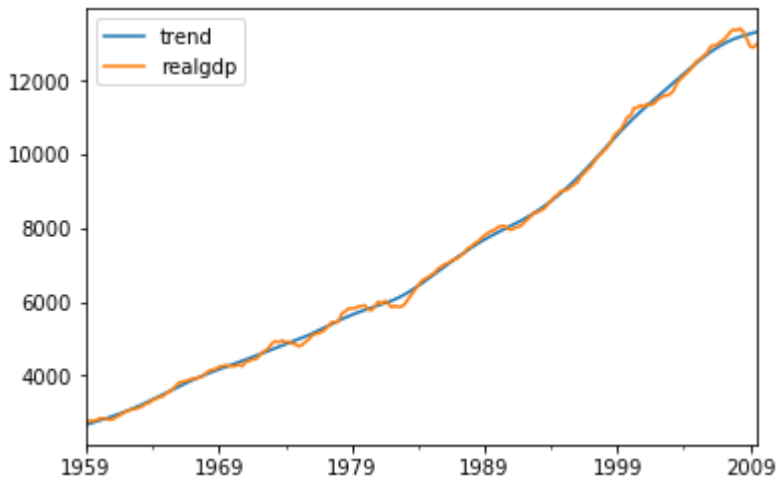
```
df["trend"] = gdp_trend
```

In [37]:

```
df[['trend','realgdp']].plot()
```

Out[37]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x21b98541080>
```

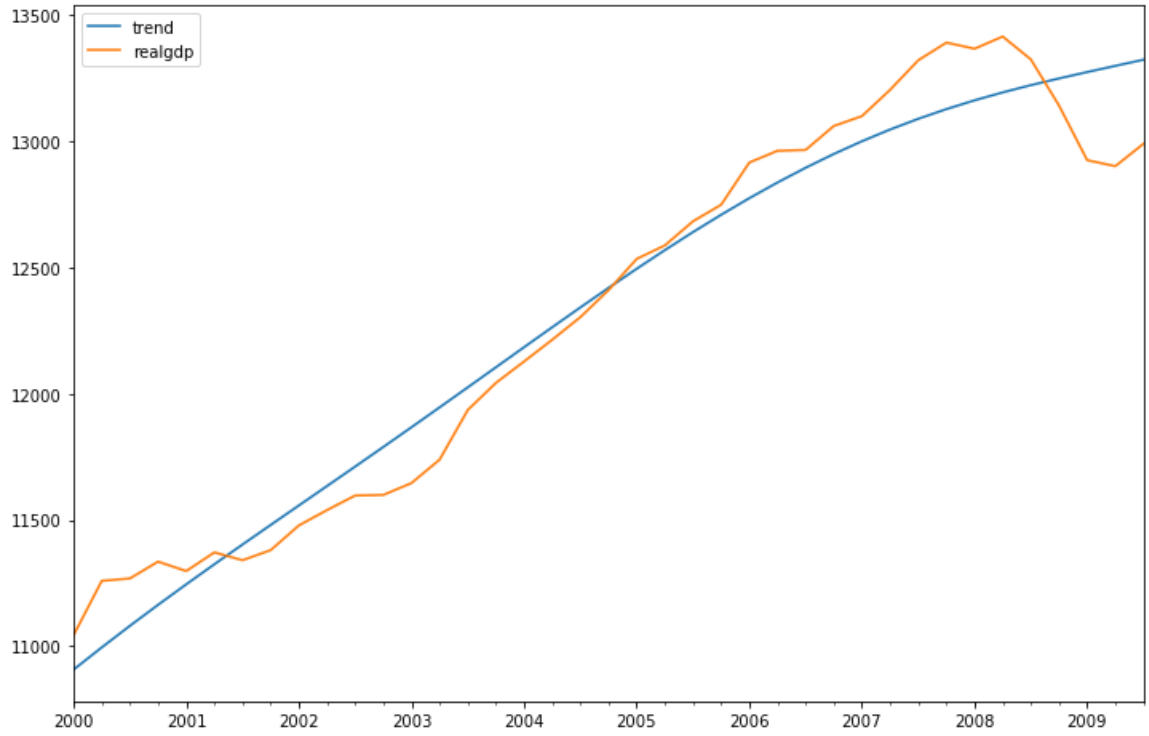


In [41]:

```
df[['trend','realgdp']]["2000-03-31":].plot(figsize=(12,8))
```

Out[41]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x21b98785390>
```



Great job!