In [1]:
```
from demo_support import *
import scikits.statsmodels.api as sm
import scikits.statsmodels.tsa.api as tsa
```

Ordinary least squares

In [2]:
```
data = sm.datasets.stackloss.load()
X = DataFrame(data.exog, columns=data.exog_name)
X['intercept'] = 1.
Y = Series(data.endog)

model = sm.OLS(Y, X)
results = model.fit()
print results.summary()
```

```
     Summary of Regression Results
=====================================
| Dependent Variable:            ['y']|
| Model:                           OLS|
| Method:                Least Squares|
| Date:               Thu, 15 Sep 2011|
| Time:                       11:05:37|
| # obs:                          21.0|
| Df residuals:                   17.0|
| Df model:                        3.0|
================================================================================
|                  coefficient     std. error    t-statistic          prob. |
--------------------------------------------------------------------------------
| AIRFLOW               0.7156         0.1349         5.3066         0.0001 |
| WATERTEMP              1.295         0.3680         3.5196         0.0026 |
| ACIDCONC             -0.1521         0.1563        -0.9733         0.3440 |
| intercept             -39.92          11.90        -3.3557         0.0038 |
================================================================================
|                         Models stats                      Residual stats  |
--------------------------------------------------------------------------------
| R-squared:                     0.9136   Durbin-Watson:              1.485  |
| Adjusted R-squared:            0.8983   Omnibus:                   0.7134  |
| F-statistic:                    59.90   Prob(Omnibus):             0.7000  |
| Prob (F-statistic):         3.016e-09   JB:                        0.1116  |
| Log likelihood:                -52.29   Prob(JB):                  0.9457  |
| AIC criterion:                  112.6   Skew:                     -0.1928  |
| BIC criterion:                  116.8   Kurtosis:                   3.107  |
--------------------------------------------------------------------------------
```

In [3]:
```
results.params
```

Out[3]:
```
AIRFLOW      0.715640200485
WATERTEMP    1.29528612439
ACIDCONC     -0.152122519149
intercept    -39.9196744201
```

In [4]:
```
results.cov_params()
```

Out[4]:

	AIRFLOW	WATERTEMP	ACIDCONC	intercept
AIRFLOW	0.01819	-0.03651	-0.007144	0.2876
WATERTEMP	-0.03651	0.1354	1.048e-05	-0.6518
ACIDCONC	-0.007144	1.048e-05	0.02443	-1.676
intercept	0.2876	-0.6518	-1.676	141.5

In [4]:

Robust linear model (RLM)

In [5]:
```
huber_t = sm.RLM(Y, X, M=sm.robust.norms.HuberT())
hub_results = huber_t.fit()
hub_results.params
```

Out[5]:
```
AIRFLOW      0.829384334499
WATERTEMP    0.926065966587
ACIDCONC     -0.127846724968
intercept    -41.0264983525
```

In [6]:
```
hub_results.bse
```

Out[6]:
```
AIRFLOW      0.111005213373
WATERTEMP    0.302930163159
ACIDCONC     0.128649614957
intercept    9.79189854298
```

In [7]:
```
hub_results.bcov_scaled
```

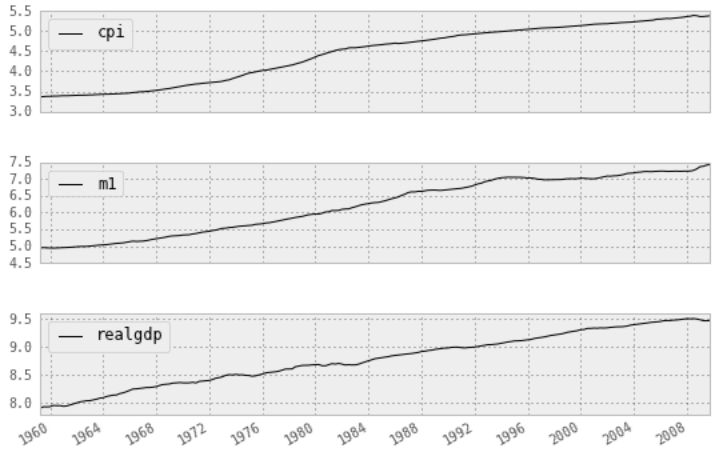
Out[7]:

	AIRFLOW	WATERTEMP	ACIDCONC	intercept
AIRFLOW	0.01232	-0.02474	-0.00484	0.1949
WATERTEMP	-0.02474	0.09177	7.098e-06	-0.4416
ACIDCONC	-0.00484	7.098e-06	0.01655	-1.136
intercept	0.1949	-0.4416	-1.136	95.88

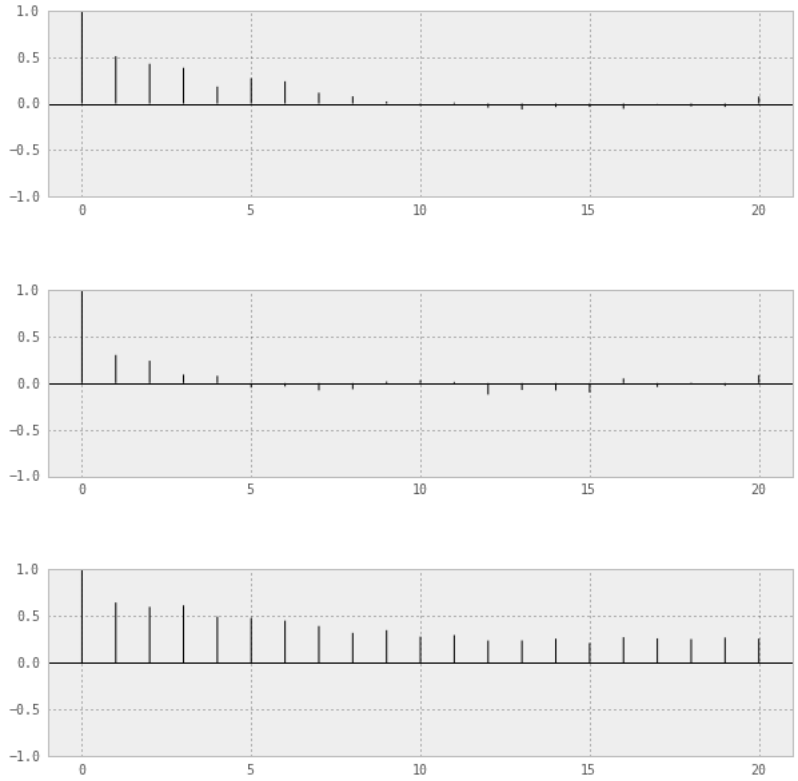
Time series analysis

In [8]:
```
data = sm.datasets.macrodata.load()
mdata = data.data

df = DataFrame.from_records(mdata)
quarter_end = datetools.BQuarterEnd()
df.index = [quarter_end.rollforward(datetime(int(y), int(q) * 3, 1))
            for y, q in zip(df.pop('year'), df.pop('quarter'))]
logged = np.log(df.ix[:, ['m1', 'realgdp', 'cpi']])
logged.plot(subplots=True)
```



In [9]:
```
log_difference = logged.diff().dropna()
plot_acf_multiple(log_difference.values)
```



Vector Autoregressive (VAR) Process

In [10]:
```
model = tsa.VAR(log_difference)
model.select_order()
```

```
                 VAR Order Selection
=====================================================
            aic          bic          fpe         hqic
-----------------------------------------------------
0        -27.80       -27.75    8.470e-13       -27.78
1        -28.70       -28.49    3.436e-13       -28.62
2        -28.92      -28.56*    2.748e-13       -28.78
3        -29.02       -28.50    2.504e-13      -28.81*
4        -29.01       -28.34    2.520e-13       -28.74
5       -29.06*       -28.23   2.408e-13*       -28.72
6        -29.06       -28.07    2.412e-13       -28.66
7        -28.98       -27.84    2.608e-13       -28.52
8        -28.98       -27.69    2.603e-13       -28.46
9        -28.94       -27.49    2.720e-13       -28.35
10       -28.93       -27.33    2.762e-13       -28.28
11       -28.90       -27.15    2.835e-13       -28.19
12       -28.88       -26.97    2.900e-13       -28.11
13       -28.82       -26.75    3.113e-13       -27.98
14       -28.75       -26.53    3.348e-13       -27.85
=====================================================
* Minimum


/Users/wesm/code/pandas/pandas/core/frame.py:2885: FutureWarning: dropEmptyRows is deprecated. Use dropna()
  FutureWarning)
/Users/wesm/code/statsmodels/main-statsmodels/scikits/statsmodels/tools/tools.py:257: FutureWarning: The default of `prepend` wil
  "next release, use explicit prepend", FutureWarning)
```

Out[10]:
```
{'aic': 5, 'bic': 2, 'fpe': 5, 'hqic': 3}
```

In [11]:
```
res = model.fit(2)
res.summary()
```

Out[11]:
```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Thu, 15, Sep, 2011
Time:                     11:05:38
--------------------------------------------------------------------
No. of Equations:         3.00000    BIC:                   -28.6281
Nobs:                     200.000    HQIC:                  -28.8342
Log likelihood:           2067.08    FPE:                2.60988e-13
AIC:                     -28.9744    Det(Omega_mle):     2.35396e-13
--------------------------------------------------------------------
Results for equation m1
==============================================================================
                coefficient       std. error           t-stat            prob
------------------------------------------------------------------------------
const              0.004968         0.001850            2.685           0.008
L1.m1              0.363636         0.071307            5.100           0.000
L1.realgdp        -0.077460         0.092975           -0.833           0.406
L1.cpi            -0.052387         0.128161           -0.409           0.683
L2.m1              0.250589         0.072050            3.478           0.001
L2.realgdp        -0.085874         0.092032           -0.933           0.352
L2.cpi             0.169803         0.128376            1.323           0.188
==============================================================================

Results for equation realgdp
==============================================================================
                coefficient       std. error           t-stat            prob
------------------------------------------------------------------------------
const              0.006121         0.001373            4.457           0.000
L1.m1             -0.043270         0.052928           -0.818           0.415
L1.realgdp         0.243380         0.069010            3.527           0.001
L1.cpi            -0.041436         0.095127           -0.436           0.664
L2.m1              0.102152         0.053479            1.910           0.058
L2.realgdp         0.155456         0.068310            2.276           0.024
L2.cpi            -0.179065         0.095287           -1.879           0.062
==============================================================================

Results for equation cpi
==============================================================================
                coefficient       std. error           t-stat            prob
------------------------------------------------------------------------------
const              0.001323         0.001009            1.311           0.191
L1.m1              0.022428         0.038888            0.577           0.565
L1.realgdp         0.046992         0.050705            0.927           0.355
L1.cpi             0.420292         0.069893            6.013           0.000
L2.m1              0.069411         0.039293            1.766           0.079
L2.realgdp        -0.044371         0.050190           -0.884           0.378
```

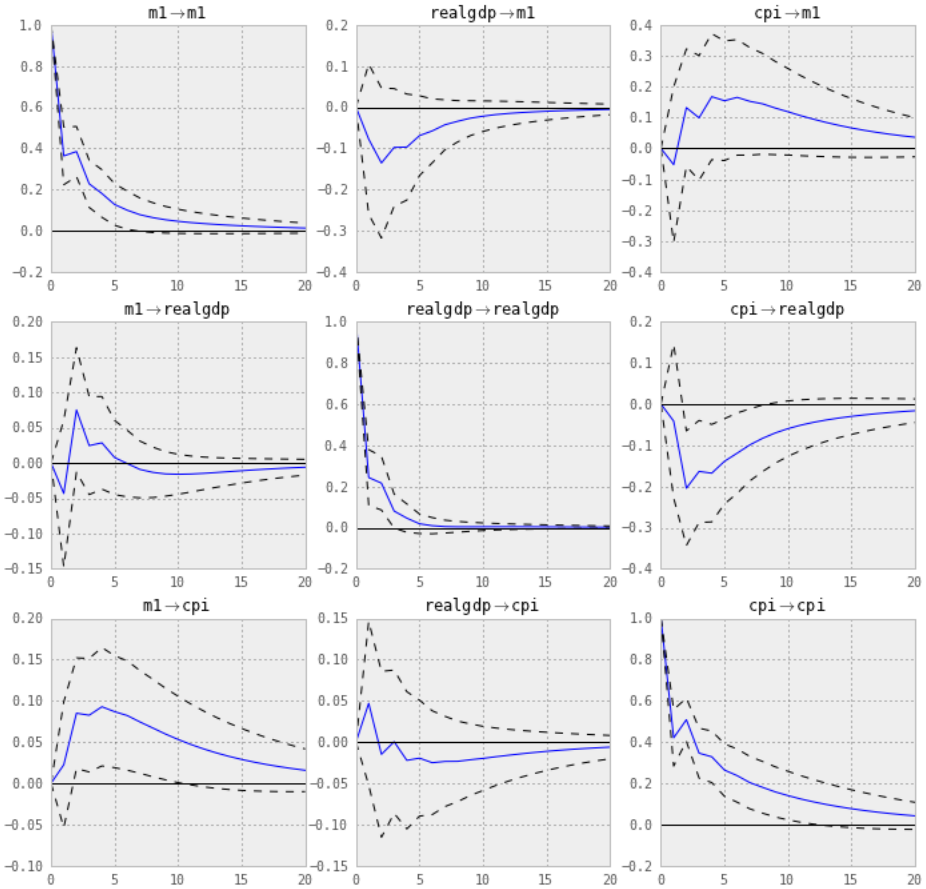
```
L2.cpi              0.334890         0.070011            4.783           0.000
==============================================================================

Correlation matrix of residuals
                    m1   realgdp       cpi
m1           1.000000 -0.055690 -0.297494
realgdp     -0.055690  1.000000  0.115597
cpi         -0.297494  0.115597  1.000000
```

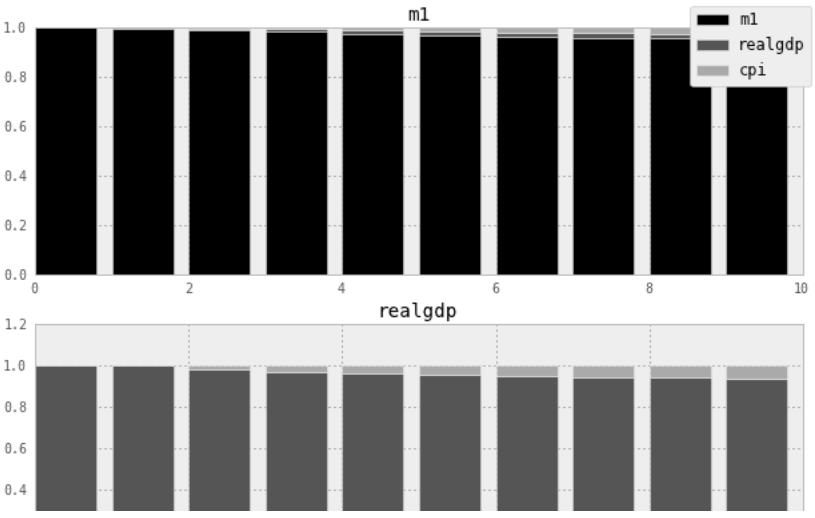
In [12]:
```
res.is_stable()
```

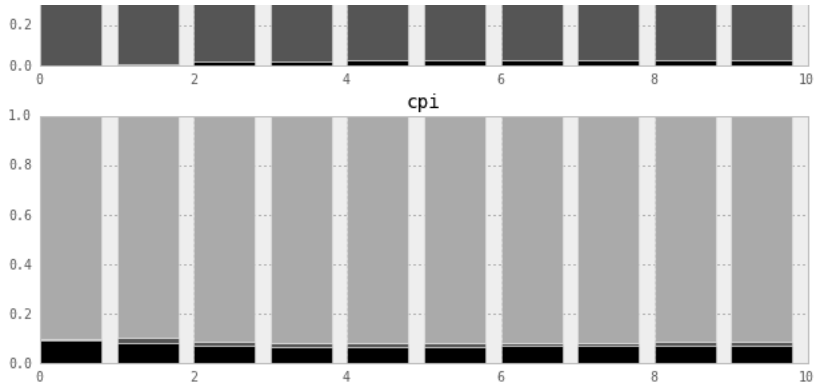
Out[12]:
```
True
```

In [13]:
```
# impulse response function
irf = res.irf(20)
irf.plot()
```



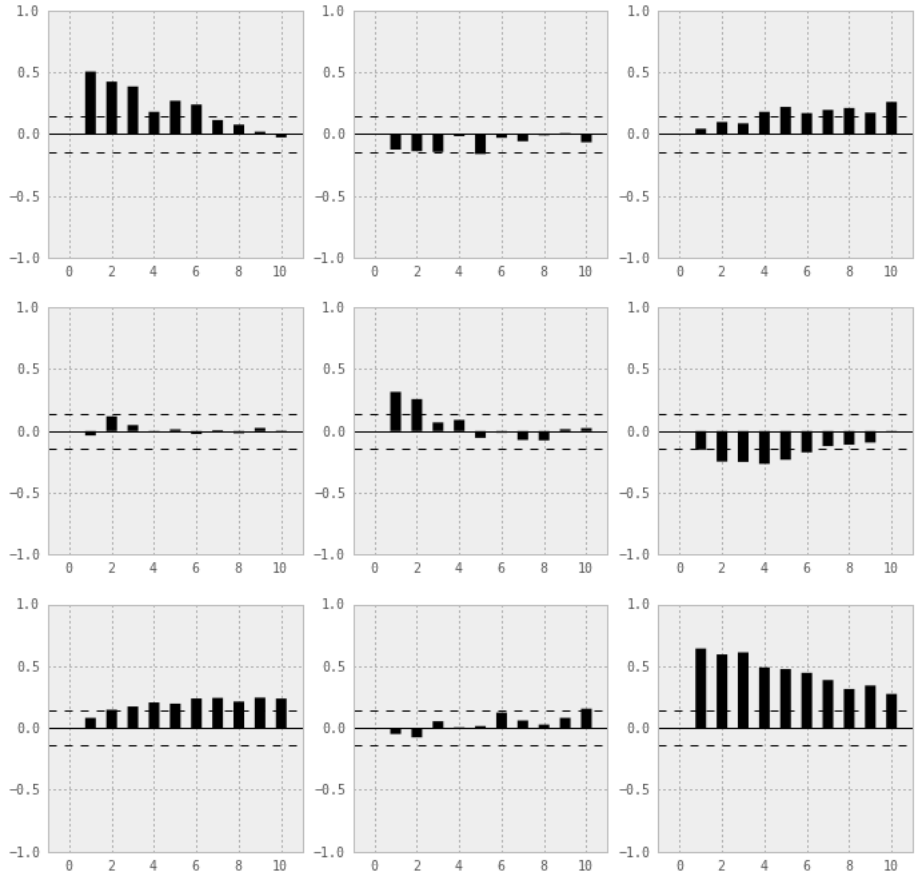
In [14]:
```
# Forecast error variance decomposition
fevd = res.fevd()
fevd.plot()
```





In [15]:
```
res.test_whiteness()
```

```
FAIL: Some autocorrelations exceed 0.1414 bound. See plot
```



In [16]:
```
res.test_causality('m1', 'realgdp')
```

```
Granger causality f-test
============================================================
   Test statistic   Critical Value          p-value        df
------------------------------------------------------------
         1.110648         3.011286            0.330  (2, 579)
============================================================
H_0: ['realgdp'] do not Granger-cause m1
Conclusion: fail to reject H_0 at 5.00% significance level
```

Out[16]:
```
{'conclusion': 'fail to reject',
 'crit_value': 3.0112857238108273,
 'df': (2, 579),
 'pvalue': 0.33004595119395758,
 'signif': 0.05,
 'statistic': 1.1106484210458074}
```

In [17]:
```
res.test_normality()
```

```
Normality skew/kurtosis Chi^2-test
=========================================================
   Test statistic   Critical Value          p-value  df
```

```
-------------------------------------------------------
        57.225730        12.591587            0.000   6
=======================================================
H_0: data generated by normally-distributed process
Conclusion: reject H_0 at 5.00% significance level
```

Out[17]:
```
{'conclusion': 'reject',
 'crit_value': 12.591587243743977,
 'df': 6,
 'pvalue': 1.6444341095846466e-10,
 'signif': 0.05,
 'statistic': 57.225729837244664}
```

Autoregressive-Moving Average (ARMA) Process

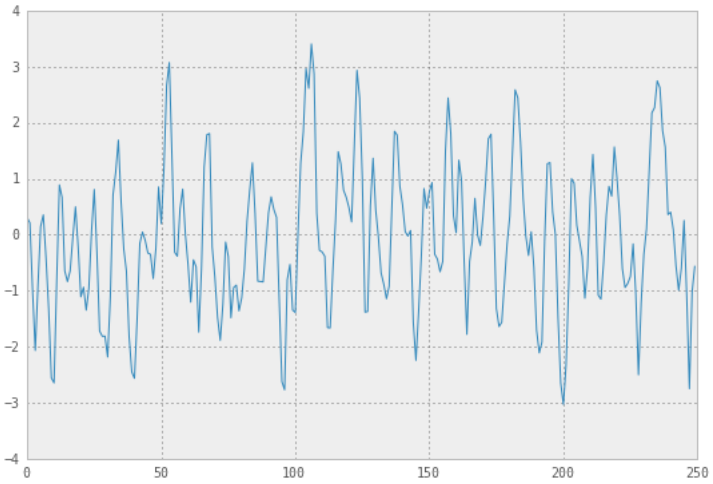
In [18]:
```
import numpy as np
import scikits.statsmodels.api as sm

# Generate some data from an ARMA process
from scikits.statsmodels.tsa.arima_process import arma_generate_sample

arparams = np.array([.75, -.25])
maparams = np.array([.65, .35])

# The conventions of the arma_generate function require that we specify a
# 1 for the zero-lag of the AR and MA parameters and that the AR parameters
# be negated.
arparams = np.r_[1, -arparams]
maparam = np.r_[1, maparams]
nobs = 250
y = arma_generate_sample(arparams, maparams, nobs)
plot(y)
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x10ec3f650>]
```



In [19]:
```
# Now, optionally, we can add some dates information. For this example,
# we'll use a pandas time series.
import pandas
dates = sm.tsa.datetools.dates_from_range('1980m1', length=nobs)
y = pandas.Series(y, index=dates)
arma_mod = sm.tsa.ARMA(y, freq='M')
arma_res = arma_mod.fit(order=(2,2), trend='nc', disp=-1)
arma_res.params
```

Out[19]:
```
ar.L1.y    1.12886907699
ar.L2.y    -0.463882785362
ma.L1.y    0.153265297539
ma.L2.y    -0.201761509133
```

Teaser: Formulas

In [20]:
```
from formula.terms import fromrec
from formula.ancova import *

df = read_table('epigen.dat', index_col=None)
recs = df.to_records(index=False)

terms = fromrec(recs)
```

```
race = terms['race']
edu = terms['edu']
smoke = terms['smoke']
ancova = ANCOVA(race, edu, (1, (race, edu)))
formula = ancova.formula + smoke

df.head().T
```

Out[20]:

	0	1	2	3	4
age	30to39	lt30	30to39	30to39	30to39
BMI	0	0	1	1	0
smoke	0	0	0	0	0
gestage	0	0	0	0	0
gender	0	0	1	0	1
edu	geCollege	geCollege	geCollege	geCollege	geCollege
race	EA	EA	EA	EA	EA
methyl1	38.36	37.85	38.57	39.75	43.83
methyl2	38.54	33.67	38.94	41.93	44.04
plate1	I	I	L	A	L
row1	A	B	D	A	G
column1	1	1	1	1	1
well1	A1	B1	D1	A1	G1
plate2	I	I	O	D	O
row2	G	H	D	A	G
column2	5	5	1	1	1
well2	G5	H5	D1	A1	G1

In [21]:
```
design = DataFrame.from_records(formula.design(recs))
print sm.OLS(recs['methyl1'],
             design).fit().summary()
```

```
     Summary of Regression Results
=====================================
| Dependent Variable:            ['y']|
| Model:                           OLS|
| Method:                Least Squares|
| Date:               Thu, 15 Sep 2011|
| Time:                       11:05:44|
| # obs:                         314.0|
| Df residuals:                  304.0|
| Df model:                        9.0|
=============================================================================================
|                              coefficient     std. error    t-statistic          prob. |
---------------------------------------------------------------------------------------------
| 1                                 46.64          1.465        31.8392         0.0000 |
| edu_ltCollege*race_EA             -3.250          2.317        -1.4024         0.1618 |
| edu_ltCollege*race_Other         -0.09743          3.739        -0.0261         0.9792 |
| edu_ltHS*race_EA                  0.08199          2.202         0.0372         0.9703 |
| edu_ltHS*race_Other                 1.542          3.448         0.4472         0.6550 |
| edu_ltCollege                       1.326          1.715         0.7734         0.4399 |
| edu_ltHS                          -0.1423          1.648        -0.0864         0.9312 |
| race_EA                            0.7005          1.627         0.4305         0.6671 |
| race_Other                         0.6537          2.455         0.2662         0.7902 |
| smoke                              3.079          1.033         2.9821         0.0031 |
=============================================================================================
|                         Models stats                        Residual stats  |
---------------------------------------------------------------------------------
| R-squared:                  0.03826   Durbin-Watson:              1.206  |
| Adjusted R-squared:        0.009784   Omnibus:                    42.88  |
| F-statistic:                1.344   Prob(Omnibus):          4.876e-10  |
| Prob (F-statistic):        0.2137   JB:                          29.20  |
| Log likelihood:            -1030.   Prob(JB):                4.556e-07  |
| AIC criterion:              2080.   Skew:                       0.7494  |
| BIC criterion:              2117.   Kurtosis:                    4.991  |
---------------------------------------------------------------------------
```

In [21]: