In [59]:

```
import numpy as np
import pandas as pd
import pyflux as pf
from datetime import datetime
import matplotlib.pyplot as plt
from pylab import rcParams
import matplotlib.patches as mpatches

%matplotlib inline
import trustedanalytics as ta
ta.connect()
```

```
Already connected.  This client instance connected to server http://localhost:9099/v1 (version=TheReneN
umber) as user test_api_key_1 at 2016-09-16 11:32:38.688631.
```

Create a frame with data that we'll use to train the ARIMAX model

The frame has columns for the observed value "VALUE" and several other columns that contain exogenous variables (converusd, petrole, petrole_sin, petrole_log).

In [35]:

```
schema = [("SUPPLIER", str),("REF", ta.float64),("DATE", str),("VALUE", ta.float64),("converusd", ta.fl
oat64), ("petrole", ta.float64),("petrole_sin", ta.float64),("petrole_log", ta.float64)]
csv = ta.CsvFile("/datasets/arimax_train_pov.csv", schema=schema, skip_header_lines=1)
train_frame = ta.Frame(csv)
```

```
Done [=========================] 100.00% Time 00:00:03
```

In [60]:

```
train_frame.inspect()
```

Out[60]:

```
[#]  SUPPLIER  REF            DATE        VALUE    converusd  petrole
=============================================================================
[0]  123PN     3528707186409  2015-09-02  38.3333     1.1255    46.25
[1]  123PN     3528707186409  2015-09-03  38.3333     1.1229    46.75
[2]  123PN     3528707186409  2015-09-04  39.0833     1.1138    46.05
[3]  123PN     3528707186409  2015-09-05  39.0833     1.1138    46.05
[4]  123PN     3528707186409  2015-09-06  39.0833     1.1138    46.05
[5]  123PN     3528707186409  2015-09-07  39.0833     1.1146    46.05
[6]  123PN     3528707186409  2015-09-08     39.5     1.1162    45.94
[7]  123PN     3528707186409  2015-09-09     39.5     1.1139    44.15
[8]  123PN     3528707186409  2015-09-10     39.5     1.1185    45.92
[9]  123PN     3528707186409  2015-09-11     39.5     1.1268    44.63

[#]  petrole_sin  petrole_log
============================
[0]  0.766831397     3.834061
[1]  0.365239258     3.844814
[2]  0.879061452     3.829728
[3]  0.879061452     3.829728
[4]  0.879061452     3.829728
[5]  0.879061452     3.829728
[6]  0.926080737     3.827336
[7]  0.166917868     3.787593
[8]  0.933441532     3.826901
[9]  0.603356086     3.798406
```

Create and train the model

Create an ARIMAX model, and then train the model by providing the frame of data, the "VALUE" column, a list of exogenous columns, p, d, q, x max lag, and a boolean flag indicating if the intercept should be included.

The ARIMAX model train() return a list of coefficients ('c' (an intercept term), ar terms, ma terms and) and (1 + xMagLag) coefficients for each "x" column.

In [37]:

```
arimax = ta.ArimaxModel()

timeseriesColumn = "VALUE"
xColumns = ["converusd", "petrole", "petrole_sin","petrole_log"]
p = 1
d = 1
q = 1
xMaxLag = 0
includeOriginalXreg = True
includeIntercept = True

arimax.train(train_frame, timeseriesColumn, xColumns, p, d, q, xMaxLag, includeOriginalXreg, includeInt
ercept)
```

```
Done [=========================] 100.00% Time 00:00:01
Done [=========================] 100.00% Time 00:00:04
```

Out[37]:

```
{u'ar': [0.7331745545680962],
 u'c': 0.003380661917224556,
 u'ma': [-0.9638139784105879],
 u'xreg': [-3.312077729440245,
  0.0692571988623542,
  0.05650501047060423,
  -3.078919944223733]}
```

So, in this example the coefficients are:

term	coefficient
c	0.003380661917224556
ar	0.7331745545680962
ma	-0.9638139784105879
converusd	-3.312077729440245
petrole	0.0692571988623542
petrole_sin	0.05650501047060423
petrole_log	-3.078919944223733

Create a frame that contains test data

In [38]:

```
csv_test = ta.CsvFile("/datasets/arimax_test_pov.csv", schema=schema, skip_header_lines=1)

test_frame = ta.Frame(csv_test)

test_frame.inspect()
```

```
Done [=========================] 100.00% Time 00:00:03
```

Out[38]:

```
[#]  SUPPLIER  REF            DATE        VALUE    converusd  petrole
=================================================================
[0]  123PN     3528707186409  2016-05-19   42.313   1.132481  47.73716
[1]  123PN     3528707186409  2016-05-20  42.0379   1.132507  47.74288
[2]  123PN     3528707186409  2016-05-21  42.1398   1.132534   47.7486
[3]  123PN     3528707186409  2016-05-22  42.2136   1.132561  47.75432
[4]  123PN     3528707186409  2016-05-23   42.199   1.132588  47.76004
[5]  123PN     3528707186409  2016-05-24  42.2237   1.132615  47.76576
[6]  123PN     3528707186409  2016-05-25  42.5559   1.132642  47.77148
[7]  123PN     3528707186409  2016-05-26  42.617    1.132668  47.7772
```

```
[7]  123PN     3528707186409  2016-05-26   42.617   1.132668   47.7772
[8]  123PN     3528707186409  2016-05-27  42.6005   1.132695  47.78292
[9]  123PN     3528707186409  2016-05-28  42.5602   1.132722  47.78864

[#]  petrole_sin  petrole_log
==============================
[0]   -0.5755444      3.86571
[1]   -0.5802125      3.86583
[2]   -0.5848616      3.86595
[3]   -0.5894916      3.86607
[4]   -0.5941022     3.866189
[5]   -0.5986935     3.866309
[6]   -0.6032651     3.866429
[7]    -0.607817     3.866549
[8]   -0.6123491     3.866668
[9]   -0.6168611     3.866788
```

Predict

Using the frame of test data, run ARIMAX predict().

In [39]:

```
p = arimax.predict(test_frame, timeseriesColumn, xColumns)

p.inspect()
```

```
Done [=========================] 100.00% Time 00:00:04
```

Out[39]:

```
[#]  SUPPLIER  REF            DATE        VALUE    converusd  petrole
=================================================================================
[0]  123PN     3528707186409  2016-05-19   42.313   1.132481  47.73716
[1]  123PN     3528707186409  2016-05-20  42.0379   1.132507  47.74288
[2]  123PN     3528707186409  2016-05-21  42.1398   1.132534   47.7486
[3]  123PN     3528707186409  2016-05-22  42.2136   1.132561  47.75432
[4]  123PN     3528707186409  2016-05-23   42.199   1.132588  47.76004
[5]  123PN     3528707186409  2016-05-24  42.2237   1.132615  47.76576
[6]  123PN     3528707186409  2016-05-25  42.5559   1.132642  47.77148
[7]  123PN     3528707186409  2016-05-26   42.617   1.132668   47.7772
[8]  123PN     3528707186409  2016-05-27  42.6005   1.132695  47.78292
[9]  123PN     3528707186409  2016-05-28  42.5602   1.132722  47.78864

[#]  petrole_sin  petrole_log  predicted_y
=============================================
[0]   -0.5755444      3.86571  42.6222607204
[1]   -0.5802125      3.86583  42.5743607991
[2]   -0.5848616      3.86595  42.5422529871
[3]   -0.5894916      3.86607  42.5217235479
[4]   -0.5941022     3.866189  42.5096861558
[5]   -0.5986935     3.866309  42.5038718378
[6]   -0.6032651     3.866429  42.5026201193
[7]    -0.607817     3.866549  42.5047135827
[8]   -0.6123491     3.866668  42.5092627272
[9]   -0.6168611     3.866788  42.5156092358
```

In [40]:

```
p.inspect(n=p.row_count, columns=["VALUE", "predicted_y"])
```

Out[40]:

```
[##]  VALUE    predicted_y
============================
[0]    42.313  42.6222607204
[1]   42.0379  42.5743607991
[2]   42.1398  42.5422529871
[3]   42.2136  42.5217235479
[4]    42.199  42.5096861558
[5]   42.2237  42.5038718378
[6]   42.5559  42.5026201193
[7]    42.617  42.5047135827
```

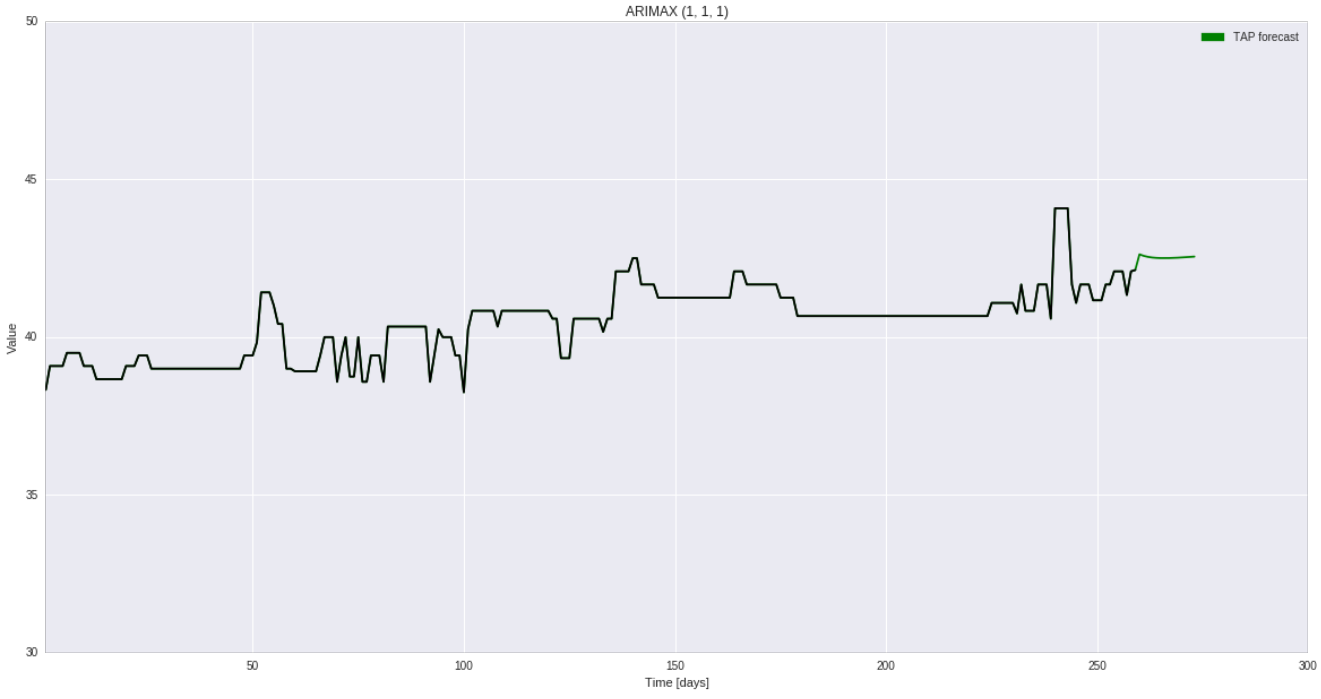
```
[8]   42.6005  42.5092627272
[9]   42.5602  42.5156092358
[10]  42.7652  42.5232735259
[11]  42.7496  42.5319070588
[12]  42.6925   42.541248137
[13]  42.6917  42.5511079693
```

In [55]:

```
r = [[42.01537], [41.93893], [41.88609], [41.84954], [41.82422], [41.80667], [41.79446], [41.78596], [41.78000], [41.77581], [41.77283], [41.77069], [41.76913], [41.76797]]

tsTrain = train_frame.take(n=train_frame.row_count, columns=["VALUE"])
predictedTAP = timeseries + p.take(n=p.row_count, columns=["predicted_y"])
predictedR = tsTrain + r
tsTest = tsTrain + p.take(n=p.row_count, columns=["VALUE"])

rcParams['figure.figsize'] = 20, 10
plt.ylabel('Value');
plt.axis([1, 300, 30, 50])
plt.plot(predictedTAP, 'g')

plt.plot(tsTrain, 'black')
plt.xlabel("Time [days]")
plt.title("ARIMAX (1, 1, 1)")


green_patch = mpatches.Patch(color='green', label='TAP forecast')
red_patch = mpatches.Patch(color='red', label='R forecast')
blue_patch = mpatches.Patch(color='blue', label='Test data')

plt.legend(handles=[green_patch])
```

Out[55]:

```
<matplotlib.legend.Legend at 0xc0398d0>
```



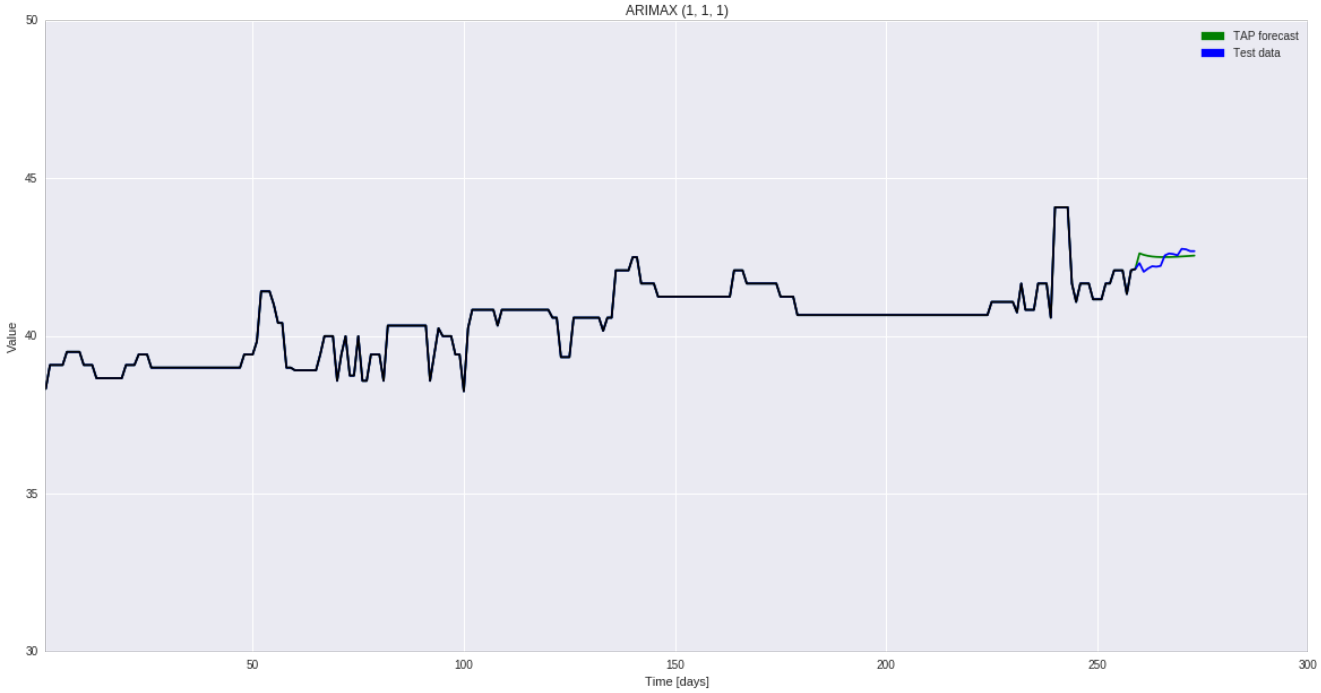
In [56]:

```
r = [[42.01537], [41.93893], [41.88609], [41.84954], [41.82422], [41.80667], [41.79446], [41.78596], [41.78000], [41.77581], [41.77283], [41.77069], [41.76913], [41.76797]]

tsTrain = train_frame.take(n=train_frame.row_count, columns=["VALUE"])
predictedTAP = timeseries + p.take(n=p.row_count, columns=["predicted_y"])
predictedR = tsTrain + r
tsTest = tsTrain + p.take(n=p.row_count, columns=["VALUE"])

rcParams['figure.figsize'] = 20, 10
```

```
plt.ylabel('Value');
plt.axis([1, 300, 30, 50])
plt.plot(predictedTAP, 'g')
plt.plot(tsTest, 'blue')

plt.plot(tsTrain, 'black')
plt.xlabel("Time [days]")
plt.title("ARIMAX (1, 1, 1)")


plt.legend(handles=[green_patch, blue_patch])
```

Out[56]:

```
<matplotlib.legend.Legend at 0xd2d0910>
```

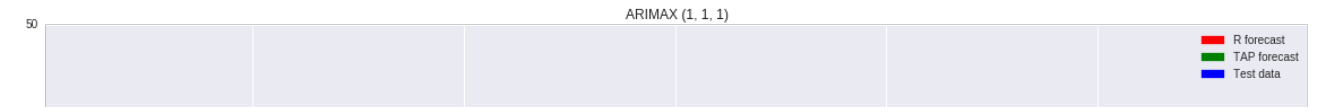


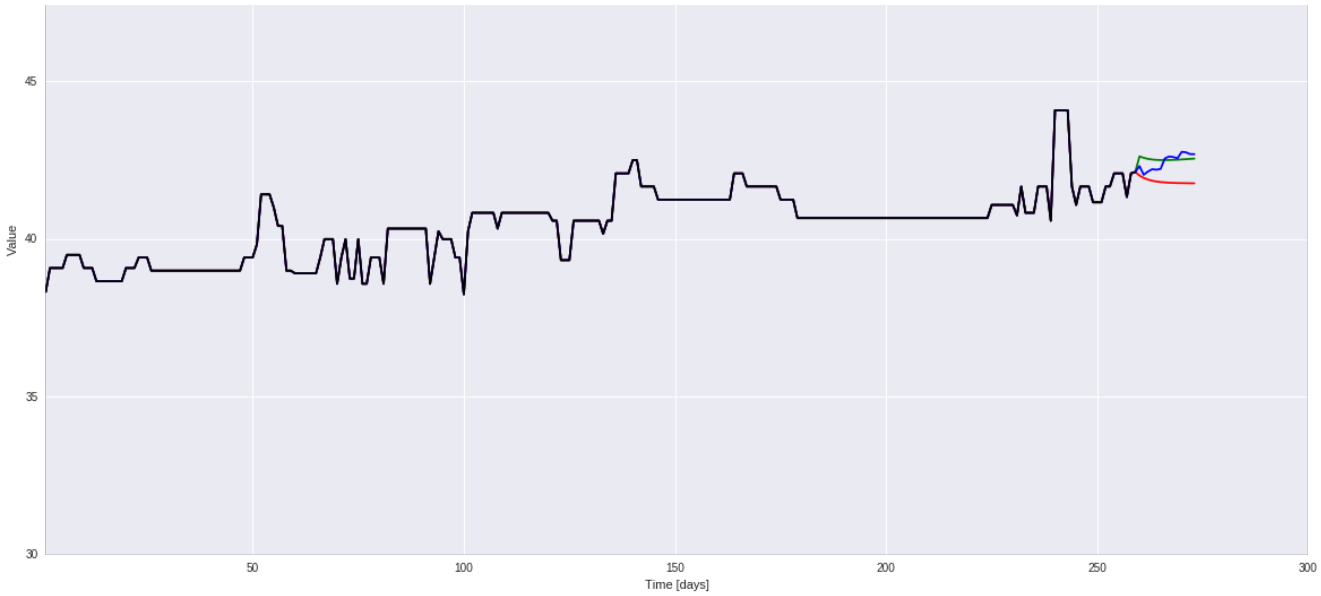
In [57]:

```
r = [[42.01537], [41.93893], [41.88609], [41.84954], [41.82422], [41.80667], [41.79446], [41.78596], [4
1.78000], [41.77581], [41.77283], [41.77069], [41.76913], [41.76797]]

tsTrain = train_frame.take(n=train_frame.row_count, columns=["VALUE"])
predictedTAP = timeseries + p.take(n=p.row_count, columns=["predicted_y"])
predictedR = tsTrain + r
tsTest = tsTrain + p.take(n=p.row_count, columns=["VALUE"])

rcParams['figure.figsize'] = 20, 10
plt.ylabel('Value');
plt.axis([1, 300, 30, 50])
plt.plot(predictedTAP, 'g')
plt.plot(predictedR, 'r')
plt.plot(tsTest, 'blue')

plt.plot(tsTrain, 'black')
plt.xlabel("Time [days]")
plt.title("ARIMAX (1, 1, 1)")


plt.legend(handles=[red_patch, green_patch, blue_patch])
```

Out[57]:

```
<matplotlib.legend.Legend at 0xdf2ac10>
```





In [58]:

```
arimax.publish()
```

```
Done [========================] 100.00% Time 00:00:01
```

Out[58]:

```
{u'category': u'model',
 u'dataSample': u'',
 u'format': u'tar',
 u'isPublic': False,
 u'recordCount': 0,
 u'size': 173209600,
 u'sourceUri': u'hdfs://ekot.jf.intel.com:8020/user/ekot/models_8f707cce3c5b47b3b6808d6c4b59cec8.tar',
 u'targetUri': u'hdfs://ekot.jf.intel.com:8020/user/ekot/models_8f707cce3c5b47b3b6808d6c4b59cec8.tar',
 u'title': u'arimax_model'}
```

In [ ]:

```

```