Explanatory Model for Car2go Demand

D

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
import datetime as dt
import scipy.stats as stats
from scipy.stats import norm
import numpy as np
import math
import seaborn as sns
from  InvarianceTestEllipsoid import InvarianceTestEllipsoid
from autocorrelation import autocorrelation
import statsmodels.api as sm
from statsmodels.sandbox.regression.predstd import wls_prediction_std
import statsmodels.tsa.ar_model as ar_model
import pickle
%matplotlib inline
```

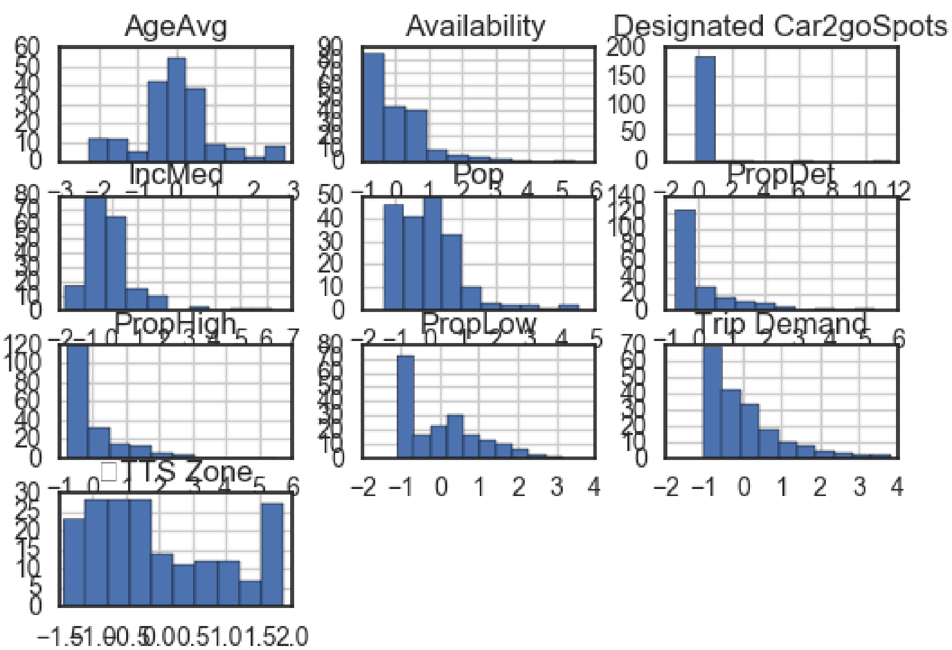
0. The model:

The model that will be used to demand of zone i

$T_i = B_{k=1:n} X_{ki} + \epsilon_i$

In [3]:
```
T = pd.read_csv("DDMFactors.csv")
T.drop('Total Seconds Available', axis = 1,inplace=True)
T = T.fillna(0)
means = T.mean()
stds = T.std()
T = (T-means)/stds
T.hist()
```

Out[3]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000000000C360898>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000C4DBB38>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000C4E84A8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000000000C595C50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000C5E1C88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000D5F1828>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000000000D63D780>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000D676E10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000D6C2F60>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000000000D704438>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000D74A8D0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000D785CF8>]], dtype=ob
ject)
```



1. Regression to obtain $m(t)$

In [4]:
```
Y = T["Trip Demand"]
#generating the factor space
X = T[T.columns[1:-2]]
```

1.1 Using Lasso Regression to shrink factors to zero

The plot below varies the magnitude of the lasso regularization to see which parameters go to zero

Training data: (x_t, y_t)

Model Specification: $Y = \beta X + C$

Lasso regularization: $\underset{\beta}{\operatorname{argmin}} \sum_t (y_t - (\beta x_t + C))^2 + \lambda ||\beta||_{l1}$

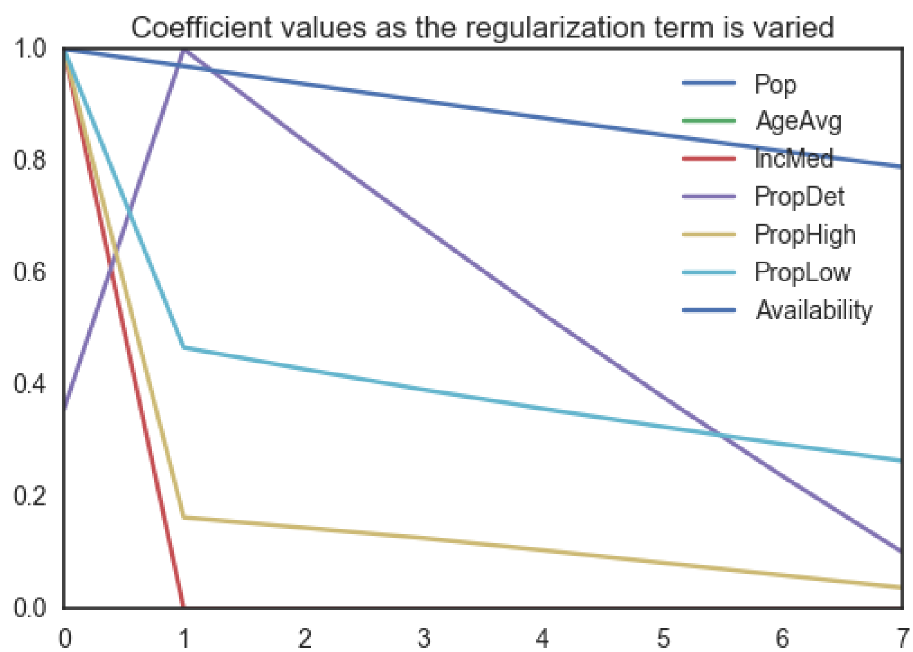
Depending on the value of λ, the coefficients in beta will shrink to zero

In [5]:
```
y = Y
L = []
model = sm.OLS(y, X)
for i in range(8):
    results = model.fit_regularized(method = 'elastic_net',alpha=i/30, L1_wt=0.5)
    L.append(results.params)
```

In [7]:
```
L = pd.DataFrame(L)
L = L/L.max(axis=0)
L.plot(title = "Coefficient values as the regularization term is varied")
L
```

Out[7]:

	Pop	AgeAvg	IncMed	PropDet	PropHigh	PropLow	Availability
0	NaN	NaN	1.0	0.356305	1.000000	1.000000	1.000000
1	NaN	NaN	0.0	1.000000	0.163107	0.466551	0.969481
2	NaN	NaN	0.0	0.836421	0.144647	0.428054	0.937837
3	NaN	NaN	0.0	0.680647	0.126056	0.391166	0.907141
4	NaN	NaN	0.0	0.526726	0.104324	0.357239	0.876519
5	NaN	NaN	0.0	0.378367	0.081853	0.325015	0.846604
6	NaN	NaN	0.0	0.236644	0.059536	0.293993	0.817586
7	NaN	NaN	0.0	0.101129	0.037394	0.264099	0.789423



In [8]:
```
cols = L.columns[L.ix[len(L)-1] > 0.001]
Xs = X[cols]
```

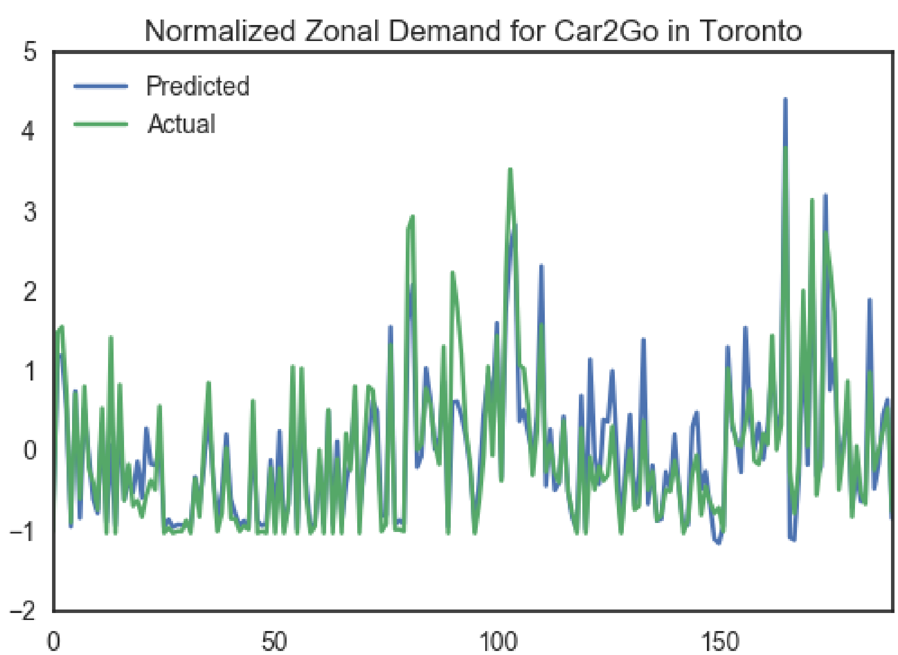
1.2 Mean Regression Results (p-values, coefficients ....)

In [9]:
```
model = sm.OLS(y,Xs)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            Trip Demand   R-squared:                       0.826
Model:                            OLS   Adj. R-squared:                  0.823
Method:                 Least Squares   F-statistic:                     221.2
Date:                Sun, 05 Nov 2017   Prob (F-statistic):           1.59e-69
Time:                        23:20:20   Log-Likelihood:                -102.82
No. Observations:                 190   AIC:                             213.6
Df Residuals:                     186   BIC:                             226.6
Df Model:                           4
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
PropDet         -0.2258      0.034     -6.717      0.000      -0.292      -0.159
PropHigh         0.0851      0.032      2.667      0.008       0.022       0.148
PropLow          0.2876      0.037      7.777      0.000       0.215       0.361
Availability     0.7653      0.037     20.924      0.000       0.693       0.837
==============================================================================
Omnibus:                       30.643   Durbin-Watson:                   1.607
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               73.865
Skew:                           0.697   Prob(JB):                     9.13e-17
Kurtosis:                       5.718   Cond. No.                         1.98
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

In [10]:
```
Comparison = pd.DataFrame(results.predict(Xs))
Comparison["Actual"] = y
Comparison.rename(columns={Comparison.columns[0]: 'Predicted'}, inplace=True)
Comparison.ix[len(y)-365:len(y)].plot(title = "Normalized Zonal Demand for Car2Go in Tor
onto")
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0xe565470>



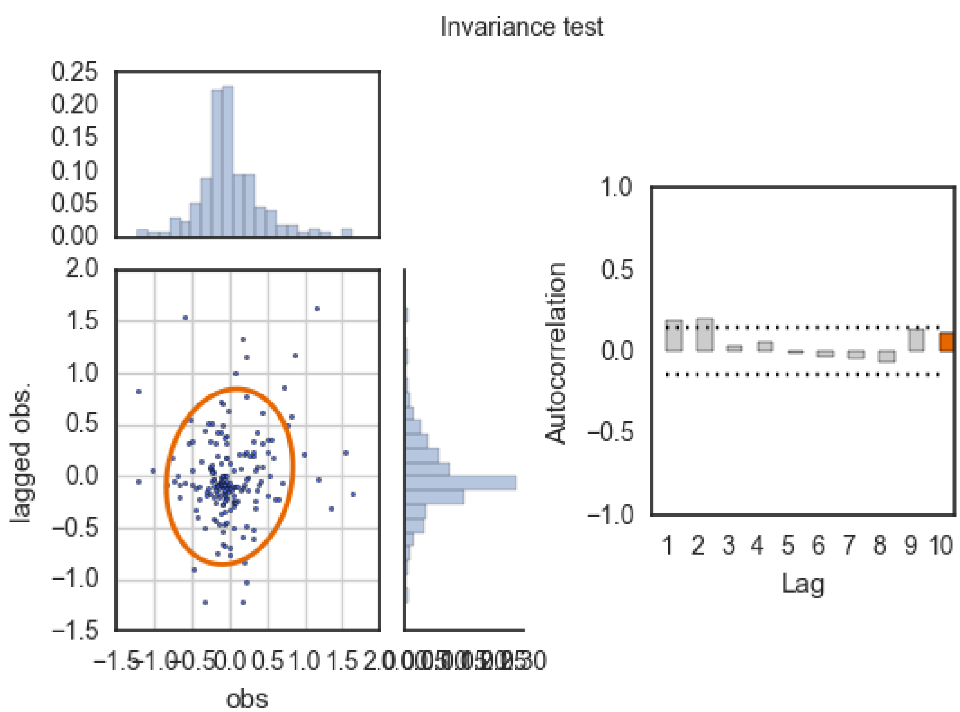
2. AR(1) Process for the Residuals

Lets check the residuals to make sure that they are approximately iid

$$X_{t+1} = \alpha X_t + \sigma(t)\epsilon$$

2.1 The code below is motivation for an AR(1) process for the residuals obtained from above: we see that there is significant correlation among the residuals from the mean process

In [11]:
```
epsi = Comparison['Actual'] - Comparison['Predicted']
epsi = np.array(epsi)
epsi = np.expand_dims(epsi, axis=0)

lag_ = 10  # number of lags (for auto correlation test)
acf = autocorrelation(epsi, lag_)

lag = 10 # lag to be printed
ell_scale = 2  # ellipsoid radius coefficient
fit = 0  # normal fitting
InvarianceTestEllipsoid(epsi, acf[0,1:], lag, fit, ell_scale);
```



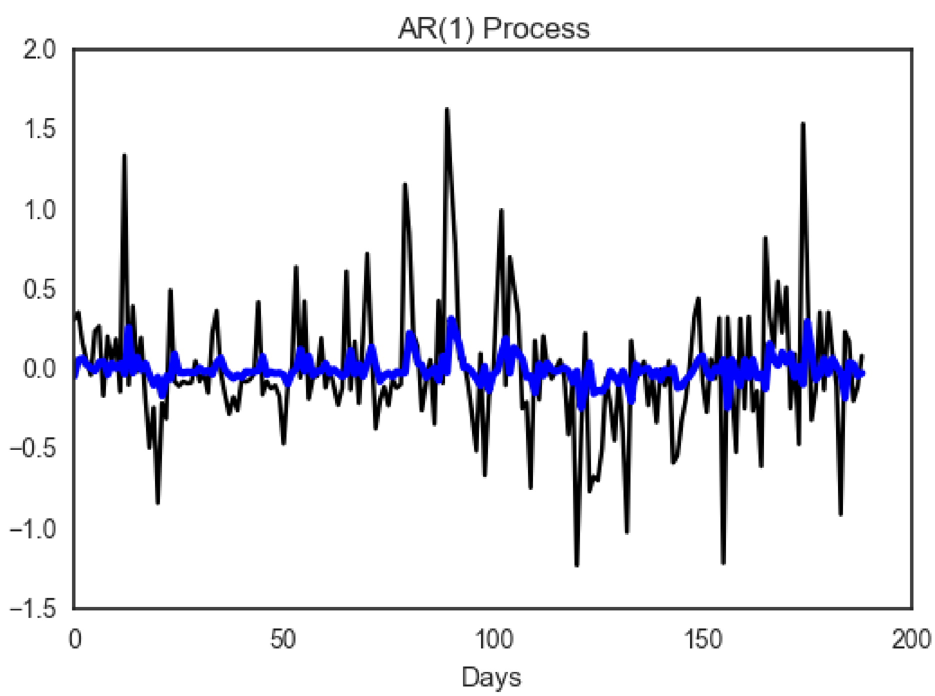
In [12]:
```
epsi = Comparison['Actual'] - Comparison['Predicted']
epsi = np.array(epsi)
model = sm.tsa.AR(epsi)
AResults= model.fit(maxlag = 30, ic = "bic",method = 'cmle')
print("The maximum number of required lags for the residuals above according to the Baye
s Information Criterion is:")
sm.tsa.AR(epsi).select_order(maxlag = 10, ic = 'bic',method='cmle')
```

```
The maximum number of required lags for the residuals above according to the Bayes Infor
mation Criterion is:
```

Out[12]: 2

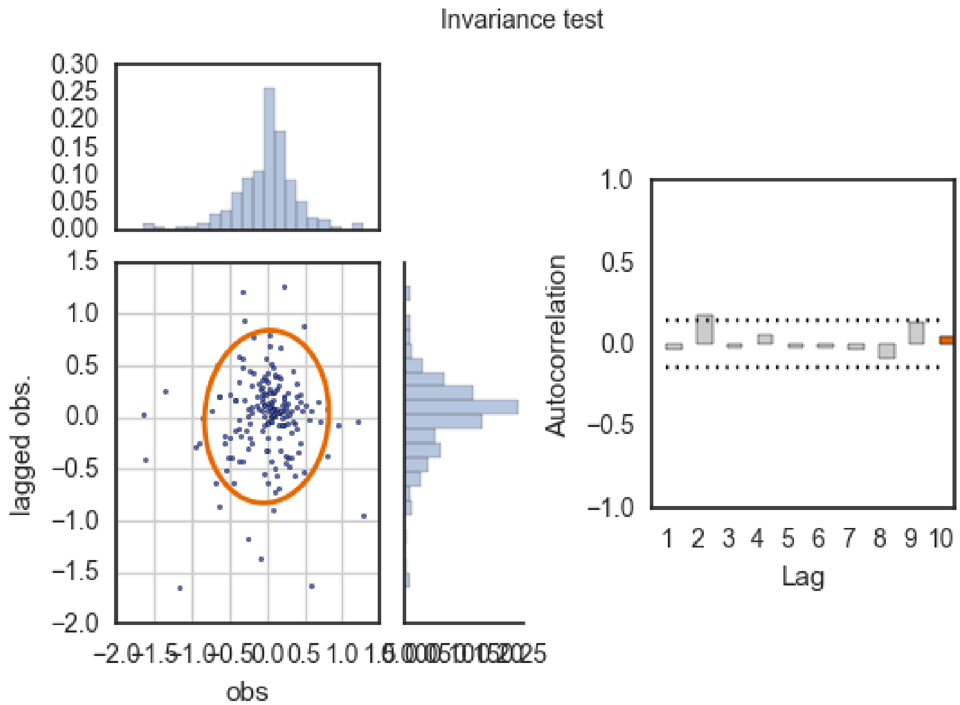
In [14]:
```
ar_mod = sm.OLS(epsi[1:], epsi[:-1])
ar_res = ar_mod.fit()
print(ar_res.summary())

ep = ar_res.predict()
print(len(ep),len(epsi))
z = ep - epsi[1:]

plt.plot(epsi[1:],  color='black')
plt.plot(ep, color='blue',linewidth=3)
plt.title('AR(1) Process')
plt.ylabel(" ")
plt.xlabel("Days")
plt.legend()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.038
Model:                            OLS   Adj. R-squared:                  0.033
Method:                 Least Squares   F-statistic:                     7.496
Date:                Sun, 05 Nov 2017   Prob (F-statistic):            0.00678
Time:                        23:21:11   Log-Likelihood:                -98.964
No. Observations:                 189   AIC:                             199.9
Df Residuals:                     188   BIC:                             203.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             0.1957      0.071      2.738      0.007       0.055       0.337
==============================================================================
Omnibus:                       33.733   Durbin-Watson:                   2.058
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               92.163
Skew:                           0.726   Prob(JB):                     9.71e-21
Kurtosis:                       6.098   Cond. No.                         1.00
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
189 190

C:\Program Files\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:531: UserWarning:
 No labelled objects found. Use label='...' kwarg on individual plots.
  warnings.warn("No labelled objects found. "
```



2.2 Invariance check for the residuals of the AR(1) process

In [15]:
```
z = np.expand_dims(z, axis=0)

lag_ = 10  # number of lags (for auto correlation test)
acf = autocorrelation(z, lag_)

lag = 10  # lag to be printed
ell_scale = 2  # ellipsoid radius coefficient
fit = 0  # normal fitting
InvarianceTestEllipsoid(z, acf[0,1:], lag, fit, ell_scale);
```



2.3 As per Benth lets see what the residuals of the AR(1) process are doing...

In [16]:
```
z = ep - epsi[1:]
plt.plot(z**2)
```

Out[16]: [<matplotlib.lines.Line2D at 0xe48f3c8>]