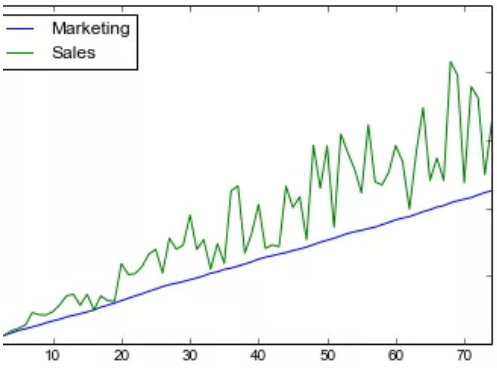
Analyzing Multivariate Time-Series using ARIMAX in Python with StatsModels



Okay, so this is my third tutorial about time-series in python. The first one was on univariate ARIMA models, and the second one was on univariate SARIMA models. Today is different, in that we are going to introduce another variable to the model. We'll assume that one is completely exogenous and is not affected by the ongoings of the other. In real-life I imagine that this is kind of doesn't exist very often, but it is worth noting that this sort of thing can happen from time to time, and the model is nonetheless useful for a short-term predictions.

In this post:

The common pitfalls associated with running ARIMAX models.
Generating forecasts from these models

The Mechanics of ARIMAX

Mechanically, ARIMAX and ARIMA do not differ. In fact in StatsModels (and other software), we don't even make a distinction between the two models. They are so similar that making a distinction is almost petty. Yet, the distinction exists, you can go look it up. So the question is why?

The answer is that by trying to combine two time-series in a regression opens you up to all kinds of new mistakes that you can make. Yeah, univariate time-series analysis has different things, like ensuring that your time-series is stationary. But multivariate time-series you start entering the weird world of causality bending. (Causality bending is my own term for what is going on here). Let's point out the basic rules of causality.

Here are the rules:

1. A cause can generate an effect at the same time that the cause happens. (Things can happen contemporaneously.)
2. A cause can generate an effect that happens after the cause. (The present can affect the future.)
3. A cause that has already happened can generate an affect in the future. (The past can affect the future)

4. An effect cannot precede a cause. (The future does not affect the present nor the past.)
5. Speeding weasels can not go faster than the speed of light. (If going faster than the speed of light is possible, all of these rules will cease to exist.)

These rules of causality form the basis of why ARIMAX deserves to be designated separately from ARIMA, at least in my mind. You see, if you violate any of these rules with your ARIMAX model, it automatically invalidates that model.

Let's say that you have a variable X which causes Y with a 1 period lag. In other words, X happens, and then 1 period later Y happens because of X. For concreteness, let's say the number of Republican Congressman elected in a year is X, and the number of tax cut bills passed by congress is Y. We'll just assume for now that there is a pretty stable positive relationship between these variables, I don't know if there is or not, that is for political scientists to figure out. Obviously, a Republican would need to be elected before they could influence the number of tax cuts they give out. But what if we got it wrong. What if we regressed X on the lag of Y. We literally just put the cart in front of the horse.

In mathese:

$$) + \alpha_1 Y_{t-1} + \epsilon$$

ıld imply that tax cuts cause republicans to get elected. We violated rule 4 or rule 5. That just won't do.

ıl, there is no way to avoid this situation except to use your intuition. However, there does exist a test, n help you to identify whether or not you are making this mistake. That test is a granger-causality test. I put too much stock into this test, mostly because it won't identify contemporaneous causality. But hey, it worth a look to see if we are making an obvious flaw.

ɔther Issue

her problem is there is no way to create a stationary time-series by adding up non-stationary time-rt of, there is a special case which would be fun to take a look at called cointegration. We'll build up to now, if I have a stationary time-series Y, and I regress it on a non-stationary time series, the math will let ıis, you end up with a contradiction. The math is forced to do some weird stuff to make it work out. In X variable must be cointegrated with your residuals. Uh-oh! That means that Y must be a linear ion of noise and X. Things, just got heteroscadastic, serial correlated, and biased. Biased? Really? Why?

ıt got biased because in order for X and the residuals to be cointegrated the residuals have to be trending, which means that they can not have an expected value of zero. Therefore, the standard proofs that OLS is not biased don't work, because they depend on the residuals having a mean of zero. So we have to make sure that our X is also stationary.

Enough Talk, Let's See Some Code

Alright, you want some code! Let's start with a dataset that you can download. Here is a dataset that you can play with: salesdata2.

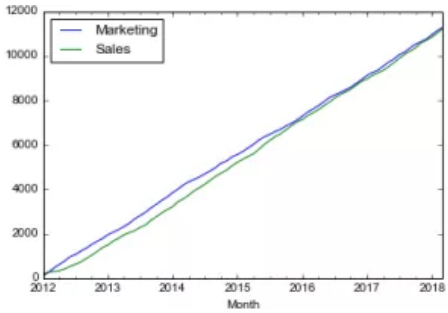
Run this code and you will see that we have 3 variables, month, marketing, and sales:

```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv('~/salesdata2.csv')
print(df)
```

We don't really care about the month variable. So let's see what these variables look like as time series. We'll use the following commands to visually inspect these variables.

```
df[['Marketing','Sales']].plot()
plt.show()
```

And when you do that you should see this graph pop-up:



With this you can clearly tell that both time-series are trending which is a fairly clear sign that both series have unit roots. We can check whether or not they have unit roots by using the augmented Dickey Fuller (ADF) test. Also, notice that Sales is more volatile than Marketing. Treating this will be a subject of a future post.

```
import statsmodels.api as sm
t(sm.tsa.stattools.adfuller(df['Marketing']))
t(sm.tsa.stattools.adfuller(df['Sales']))
```


luces the following output:

```
l1261902882013171, 0.9481670336582011, 0, 74, {'5%': -2.9014701097664504, '
l0318177272648931, 0.90956617826759134, 7, 67, {'5%': -2.9057551285231229,
```

ve can not reject the null-hypothesis that these series have a unit root. So we should difference both a first step.

ust run a naive regression to see what happens on the undifferenced time-series, and then again on the ed time-series, here's the code for the first regression.

```
const']=1
l1=sm.OLS(endog=df['Sales'],exog=df['Marketing','const'])
lts1=model1.fit()
print(results1.summary())
```

Which gives you the following output:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 3.943e+04
Date:                Tue, 27 Jun 2017   Prob (F-statistic):          1.59e-101
Time:                        22:18:35   Log-Likelihood:                -479.13
No. Observations:                  75   AIC:                             962.3
Df Residuals:                      73   BIC:                             966.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Marketing      1.0348      0.005    198.581      0.000       1.024       1.045
const       -498.4924     34.123    -14.609      0.000    -566.500    -430.485
==============================================================================
Omnibus:                       27.350   Durbin-Watson:                   0.131
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               62.288
Skew:                           1.203   Prob(JB):                     2.98e-14
Kurtosis:                       6.761   Cond. No.                     1.33e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
[2] The condition number is large, 1.33e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

And here is the code for a differenced time-series regression.

```
df['diffS']=df['Sales'].diff()
df['diffM']=df['Marketing'].diff()
```

```
model2=sm.OLS(endog=df['diffS'].dropna(),exog=df[['diffM','const']].dropna())
results2=model1.fit()
print(results2.summary())
```

The output for this regression looks like this:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  diffS   R-squared:                       0.016
Model:                            OLS   Adj. R-squared:                  0.003
Method:                 Least Squares   F-statistic:                     1.196
Date:                Tue, 27 Jun 2017   Prob (F-statistic):              0.278
Time:                        22:18:35   Log-Likelihood:                -374.58
No. Observations:                  74   AIC:                             753.2
Df Residuals:                      72   BIC:                             757.8
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
diffM         -0.1686      0.154     -1.094      0.278      -0.476       0.139
const        174.5690     23.685      7.370      0.000     127.354     221.784
==============================================================================
Omnibus:                        4.654   Durbin-Watson:                   1.182
Prob(Omnibus):                  0.098   Jarque-Bera (JB):                3.841
Skew:                          -0.520   Prob(JB):                        0.147
Kurtosis:                       3.405   Cond. No.                         808.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
```

Shares

There is a difference between these two models. One is just plain wrong. Well, let's think about this for a moment. We get a better fit with the first regression, but when we think about what is going on can remember that both time series are trending. There is a third variable in the mix. Essentially, time is d to both my input and output. What happens is that we get a very significant, and wrong, estimate our coefficient is the ratio of the trends, not the effect that X has on Y. That's why we get the inearity warning too. Uh-oh, that must mean that the first is wrong, and the second is right. But there isn't anything significant in the second, we can't say anything.

Well, what if we didn't difference marketing, but we did difference sales? Well, that leads to a consistency problem, we have a series without the unit-root regressed on a series with a unit-root, and that causes the problems with error terms that have to have a trend.

Okay, so at this point, we can start looking for the correct ARIMA model using ACF and PACF plots. I'm not going to do that since I covered that at length in my first tutorial. Let's turn our attention to something else. This issue of causality. How is it that I know that Marketing is driving sales and not vice-versa.

That's what a granger-causality test is for. The test is whether the first variable is caused by the second variable. Here's the code:

```
print(sm.tsa.stattools.grangercausalitytests(df[['Sales','Marketing']].dropna()
```

And the results:

```
Granger Causality
('number of lags (no zero)', 1)
ssr based F test:         F=0.5684  , p=0.4534  , df_denom=71, df_num=1
ssr based chi2 test:   chi2=0.5924  , p=0.4415  , df=1
likelihood ratio test: chi2=0.5900  , p=0.4424  , df=1
parameter F test:         F=0.5684  , p=0.4534  , df_denom=71, df_num=1
{1: ({'lrtest': (0.59002558961230989, 0.44240922354434226, 1), 'params_ftest':
```

You can also see by reversing the test that Sales does not cause Marketing:

```
print(sm.tsa.stattools.grangercausalitytests(df[['Marketing','Sales']].dropna()
```

And the results:

```
Granger Causality
('number of lags (no zero)', 1)
ssr based F test:         F=0.5684  , p=0.4534  , df_denom=71, df_num=1
ssr based chi2 test:   chi2=0.5924  , p=0.4415  , df=1
likelihood ratio test: chi2=0.5900  , p=0.4424  , df=1
parameter F test:         F=0.5684  , p=0.4534  , df_denom=71, df_num=1
{1: ({'lrtest': (0.59002558961230989, 0.44240922354434226, 1), 'params_ftest':
```

Finally, here is the full blown code for the correct ARIMAX model. I know this is the "correct" model because I generated the data. But there is more to talk about with this dataset than meets the eye. The biggest problem comes from the fact that I am using non-normal errors, which is throwing things off a bit.

```
df['lag']=df['diffM'].shift()
df.dropna(inplace=True)
model3=sm.tsa.ARIMA(endog=df['Sales'],exog=df[['lag']],order=[1,1,0])
results3=model3.fit()
print(results3.summary())
```

Shares

esults

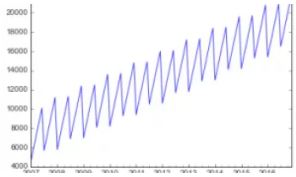
```
                             ARIMA Model Results
==============================================================================
 Variable:                 D.Sales   No. Observations:                   71
l:                 ARIMA(1, 1, 0)   Log Likelihood                -345.588
od:                       css-mle   S.D. of innovations             31.448
:                 Tue, 27 Jun 2017   AIC                            699.176
:                         08:00:20   BIC                            708.226
le:                   05-01-2012   HQIC                           702.775
                    - 03-01-2018
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.97
------------------------------------------------------------------------------
:           111.6228     19.506      5.723      0.000      73.393     149.8
              0.2771      0.126      2.194      0.032       0.030       0.5
l.D.Sales     0.1457      0.124      1.178      0.243      -0.097       0.3
                                    Roots
==============================================================================
                 Real           Imaginary           Modulus         Frequency
------------------------------------------------------------------------------
AR.1            6.8622           +0.0000j            6.8622            0.0000
------------------------------------------------------------------------------
```

So that's it. The basic take aways are that when introducing another variable into your time-series, you need to make sure that the time-series is consistent, in that the variables are integrated to the same order, and that the causal structure of your model makes sense not just intuitively, but statistically as well.

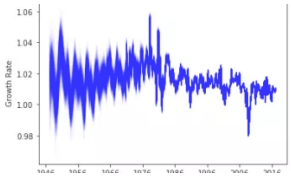
Like this:

Like

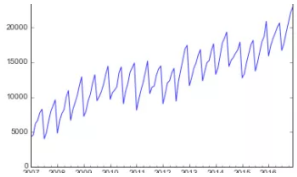
Be the first to like this.



Basics of ARIMA Models With Statsmodels in Python
June 14, 2017
In "Time Series"



Bayesian Time Varying Coefficients in PYMC3
October 10, 2017
In "Bayesian Statistics"



SARIMA models using Statsmodels in Python
June 21, 2017
In "Time Series"

Shares

es | Tags: ARIMA, ARIMAXitconsultant

IA models using Statsmodels in Python GARCH Models in Python →

thoughts on "Analyzing Multivariate ne-Series using ARIMAX in Python h StatsModels"

Tony on September 3, 2017

Hi,

s for sharing your knowledge !

ed an error in your code though, and I really don't know what you wanted to do, even if the code seems pretty clear to me.

df['const']

This line fails for me. I guess you wanted to created a new dataset, but is there something missing ?

Best regards

Reply

Ryan Barnes on September 6, 2017

Hey Tony,
Thanks for catching that. I updated the code in the article. It was supposed to be:

df['const']=1

Again thanks for pointing this out for me.

Reply

Luis on September 29, 2017

One important will be how to deal with exogenous variables when you have quite a lot candidates? Secondly, how one can be sure that there is not overfitting when including more than 2/3 exogenous variables in the model.
I have read about lasso regression, but not sure how to applied to an ARIMAX model.

Let´s say you have around 50 observations and end up with model with 8 variables (+ dependant Y). How will you evaluate that there is not overfitting on that model.

Reply

Ryan Barnes on September 29, 2017

Lasso regression would work in this case, typically statsmodels does not have an implementation though. You would have to build your own maximum likelihood estimator and then tack the regularization term on the end of the likelihood function. Unfortunately, that is very much a custom build, but it can be done, which is beyond the scope of what I was trying to do in this blog post.

Reply

Raj Thilak on November 13, 2017

This article saved my life. I've been trying to find something to explain implementation of multivariate time series regression in ARIMA. If we use the ARIMAX model with a test dataset to make out of sample predictions, does it work alright or is there anything we need to watch out for?
Also, just a small correction, the grangercausality results that you show are both identical. It would be helpful to see the second result so that we can appreciate the difference. Thanks a lot.

Reply

Ryan Barnes on November 13, 2017

Raj,

I'm so glad you found this tutorial useful!

The model should work just fine with out of sample data. The trick is to make sure that your autocorrelation is accurate. I believe the predict method on your statsmodel arima class implements this correctly (it's been been awhile since I read the documentation though).

· the granger causality test, I'm sure I just copied and pasted the wrong set. Thanks for the heads up.

serdar on November 15, 2017

Hello,
ew about python and data science and i am looking for releated python documents with statistics and eries model like arima ,ma,...
ɔe very happy if you would help me about where i can find that documents

> *Ryan Barnes* on November 15, 2017
>
> Serdar,
>
> ırd for me to point you in the direction of solid resources at your level without knowing your level nally. That being said the statsmodels documentation is generally a pretty good starting point /www.statsmodels.org/stable/index.html They have an entire section on time series models: /www.statsmodels.org/stable/tsa.html. Also, if you want some personalized one on one training you can ne a call at (801) 815-2922 or email me at ryan@barnesanalytics.com. I'd be happy to help.

Reply

Muddassir on July 31, 2018

Completely irrelevant!!!!Can u please post about how to do multivariate time series forecasting

Reply

Ryan Barnes on July 31, 2018

Muddassir, I'm sorry that you didn't find this post helpful. You can use this model to forecast, you just assume (with this model) that the variables are not endogenous, i.e. you have exogenous variables to regress with. I believe that the title of the post mentions that I'd be using ARIMAX, so I definitely didn't mean to mislead you. My guess is that you are looking for a model like VAR, VECM where the variables are completely endogenous with each other, right? I'll look into writing up a post about that, thanks for the suggestion.

Reply

Muddassir on July 31, 2018

Thanks for quick reply...i have 4 years of data 2007,2008,2009,2010 data with 10 predictor variables and 1 target variable .
Target variable is continous.
I have to forecast the target variable for 2011
Note: predictor variables are not given for 2011 data .
Please suggest me the appropriate model to apply

Reply

Bill on October 4, 2018

The two Granger Causality tests are stille both identical and as such, I have no idea of what to look for — can you post the correct test results?

Reply

Maya on December 2, 2018

Noob question: How do you read the results of the Granger causality test? The results look identical for both cases in your code

Reply

Leave a Reply

Enter your comment here...

Shares

uses Akismet to reduce spam. Learn how your comment data is processed.

arnes

nes has a PhD in economics
cus on econometrics.

Latest Tweets

Use twitter widget for twitter feed.

Connect with us

Facebook

Twitter

Google +

Linkedin

Contact info

Phone: 801-815-2922
Fax:

Email:
ryan@barnesanalytics.com
Website:
http://barnesanalytics.com