Week 2 - Vector AutoRegressive Models

What is a VAR model?

VAR models are a more structural way that we can model time series data.

- VAR: **V**ector **A**uto**R**egressive model
- Makes use of multiple correlated time series
- Based on SUR (Seemingly Unrelated Regressions) models

Quick Overview of SUR models

Consider j regression equations:

$$Y_j = X_j \beta_j + \epsilon_j$$

where Y_j, and ϵ_j are $N \times 1$, X_j is $N \times K$, and β_j is $K \times 1$

Quick Overview of SUR models

Consider j regression equations:

$$Y_j = X_j\beta_j + \epsilon_j$$

Imagine that the outcomes Y_{ij} are correlated such that

$$Cov(\epsilon_{ij}, \epsilon_{ik}) = \sigma_{ij}$$

and

$$Cov(\epsilon_{ij}, \epsilon_{i'k}) = 0, \;\; \forall \; i \neq i'$$

Quick Overview of SUR models

We can stack our J regressions to get a single system of equations:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_J \end{bmatrix} = \begin{bmatrix} X & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & X & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & X \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{J \times K} \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_{J \times K} \end{bmatrix}$$

Where Y_j is a vector of length N, and X is an $N \times K$ matrix

Quick Overview of SUR models

Then the FGLS estimator of the system is

$$\hat{\beta}_{FGLS} = \left(X' \left(\hat{\Sigma} \otimes I_J\right) X\right)^{-1} X' \left(\hat{\Sigma} \otimes I_J\right) Y$$

Where $\hat{\Sigma} = [\hat{\sigma}_{ij}]$, and

$$\hat{\sigma}_{ij} = \frac{1}{N} (y_i - X_i \beta_i)' (y_j - X_j \beta_j)$$

Quick Overview of SUR models

So what does all this mean?

- SUR models relax the assumption that each regression is uncorrelated with the others
    - No longer modeling a bunch of separate time series
- Allows us to use some of our dependent variables in the X matrices for other regressions
    - Allow us to model simultaneous time series, where the errors across the series will be correlated

VAR Models

Just a SUR model where the multiple dependent variables are time series

- We can include lags of dependent variables as part of the X matrix of covariates
- VAR models are built to capture the interactions between variables as time passes

VAR Models

We can write the VAR model

$$\mathbf{y}_{i,t} = \mu_{\mathbf{i}} + \mathbf{\Gamma}_{i,1}\mathbf{y}_{i,t-1} + \sum_{j=1}^{J}\mathbf{\Gamma}_{j,1}\mathbf{y}_{i,t-1} + \epsilon_{i,t}$$

where $i \neq j$

Representing the ith equation relating lagged dependent variables to the dependent variable in time t.

Implementing a VAR Model

```
# Getting started by importing modules and data
import pandas as pd, numpy as np
from statsmodels.tsa.api import VAR
import statsmodels.tsa.stattools as st
import plotly.express as px
from datetime import datetime

# Collect data, set index
# Be sure to put the URL back onto a single line!!
data = pd.read_csv("https://github.com/dustywhite7/Econ8310/
raw/master/DataSets/pollutionBeijing.csv")
# Difference and log dep. var.
format = '%Y-%m-%d %H:%M:%S'
data['datetime'] = pd.to_datetime(data['datetime'],
        format=format)
data.set_index(pd.DatetimeIndex(data['datetime']),
        inplace=True)
```

Implementing a VAR Model

```
# Select variables for VAR model
varData = data[['pm2.5','TEMP','PRES',
                              'Iws']].dropna()[:-50]
```

- **REMEMBER: We need ALL stationary variables**
  - `st.adfuller` is the test to use on each variable
- We also need the terminal (or starting) values of each variable PRIOR to differencing to reconstruct the original time series

Implementing a VAR Model

```
model = VAR(varData) # define the model and data
model.select_order() # uses information criteria to
                     # select the model order
modelFit = model.fit(30)
                # order chosen based on BIC criterion
```

- Diagnostics like those from the ARIMA(p,d,q) models are not available to determine our model order
- Use information criteria to find the optimal order of the VAR model
    - You could also do this with your ARIMA models if you so choose

Forecasting with a VAR Model

```
# Forecasting
pred = modelFit.forecast(varData['2013-01-04':].values,
                         steps = 50)
```

- When using a trained VAR model, we must include enough observations from our dataset in order to provide the expected number of lags to the model
- The more lags we include in the model, the more observations we need before we can fit the model to the data

Forecasting with a VAR Model

Recall that our raw forecast is not always what we will observe in the real world

- If we have **differenced** our data to achieve stationarity, we need to undo that differencing in our model's forecasts

Forecasting with a VAR Model

```
# Make predictions a DataFrame
nextPer = pd.DataFrame(pred,
                # Use the index from the last 50 observations
                #   as the index for the predictions
                pd.DatetimeIndex(
                varData.iloc[-50:].index),
                columns=varData.columns)

# Isolate the testing data, last 50 periods
test = data.iloc[-50:]
```

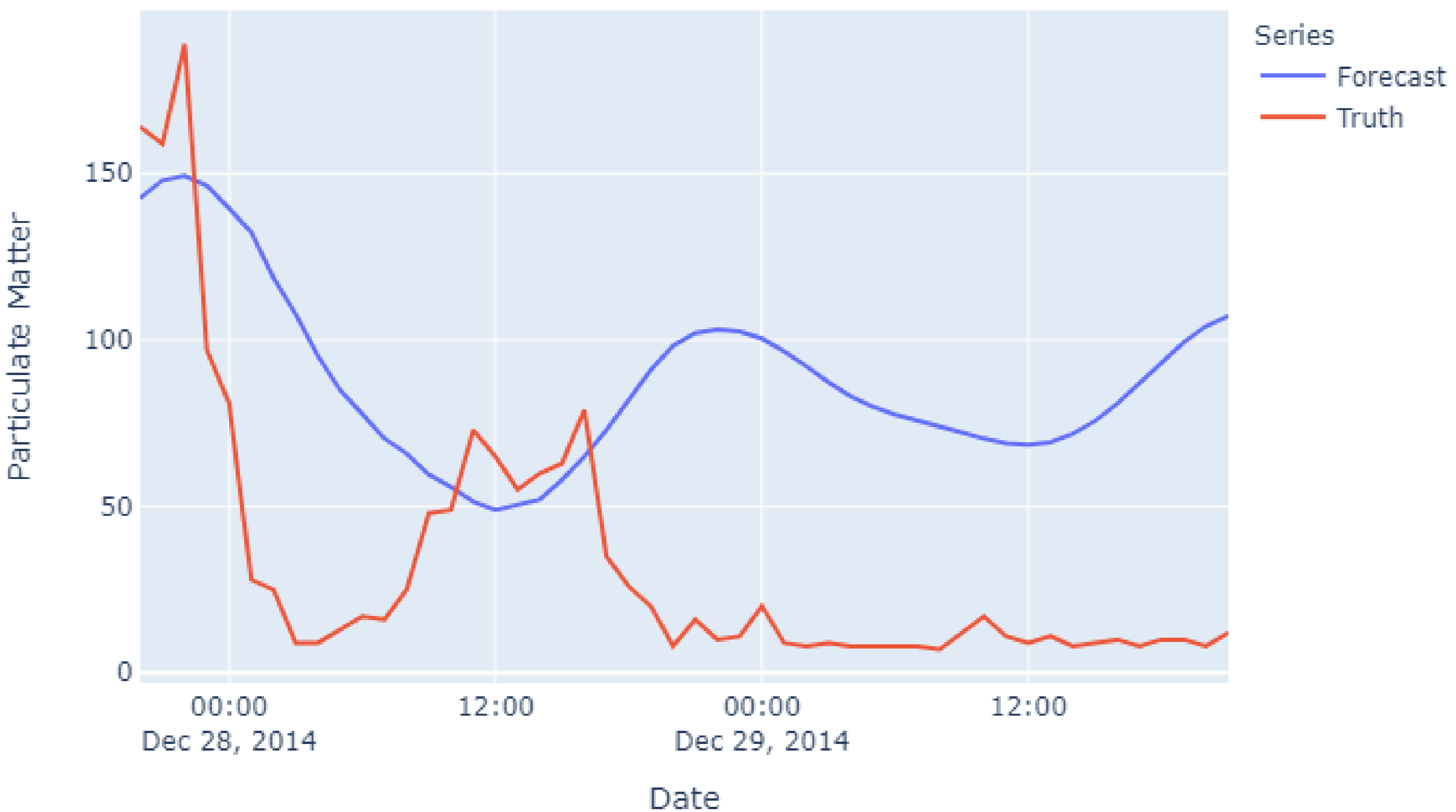
Here, we transform our predictions (and truth) into datetime formatted values, so that we can more easily plot them.

Forecasting with a VAR Model

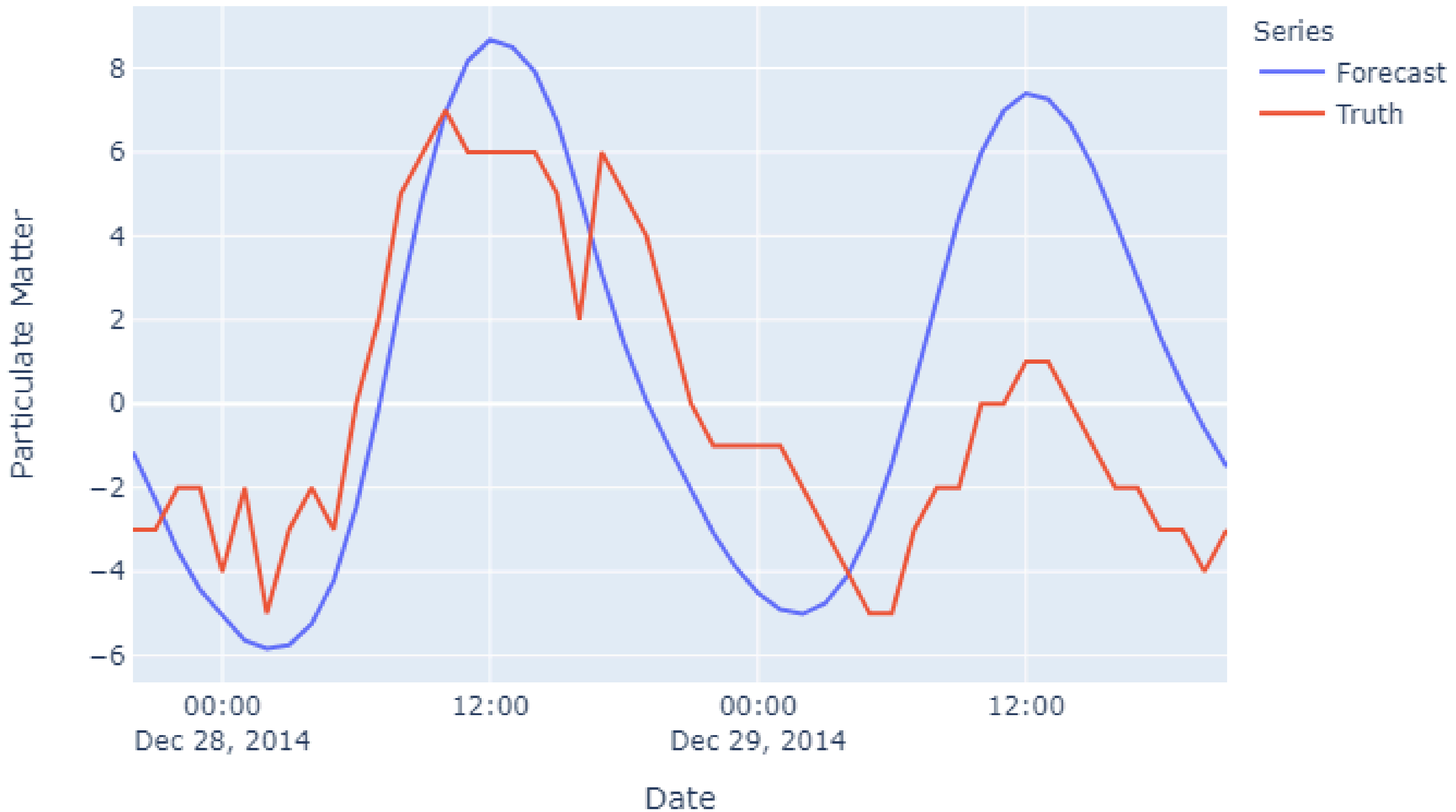
```
# Create and format the figure
fig = px.line(x = nextPer.index,
                y=[nextPer['pm2.5'], test['pm2.5']],
                labels = {
                        'value' : 'Particulate Matter',
                        'x' : 'Date',
                        'variable' : 'Series'
                })

# Renaming the series
fig.data[0].name = "Forecast"
fig.data[1].name = "Truth"

# Render the plot
fig.show()
```

Plotting prediction vs truth

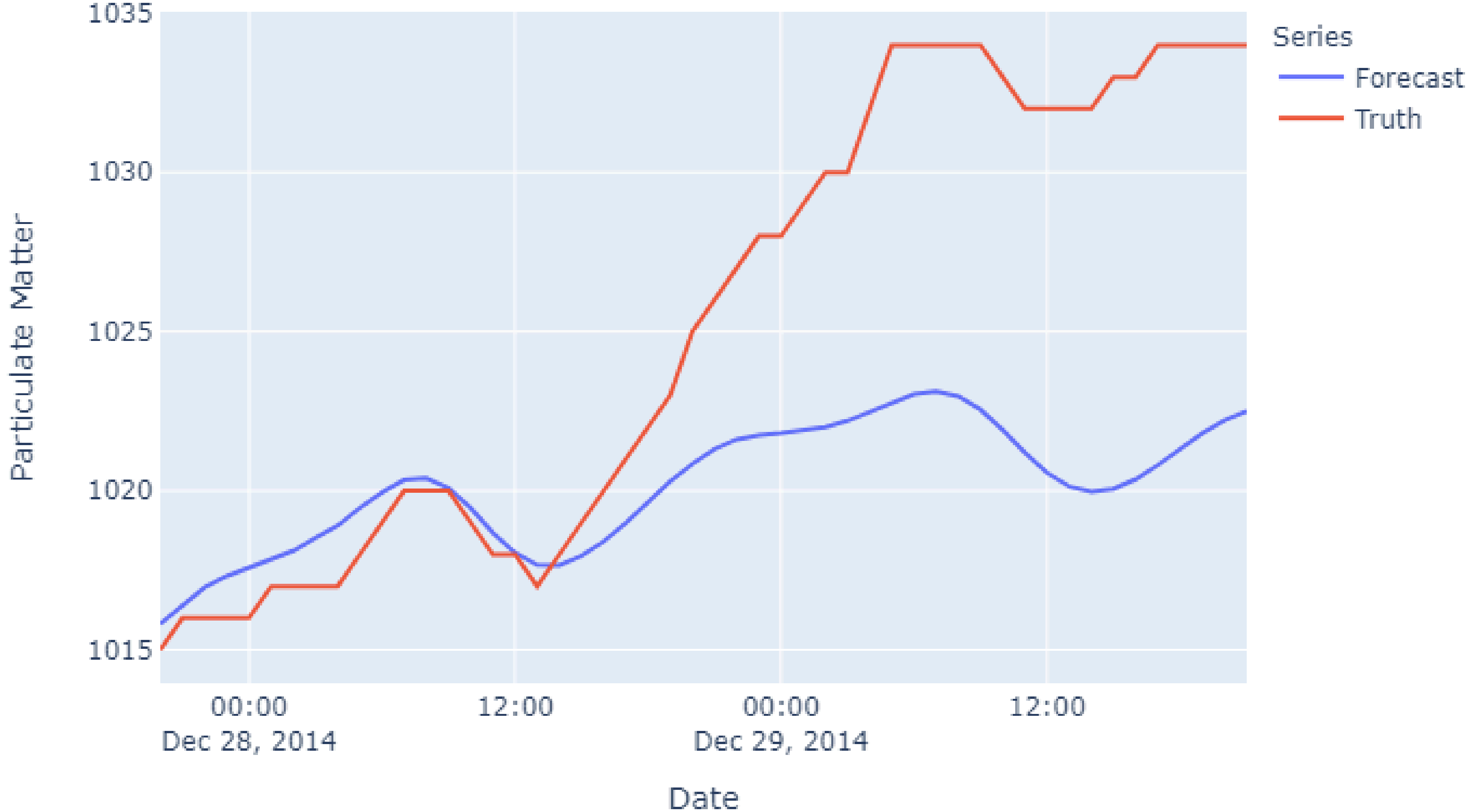
Particulate Matter



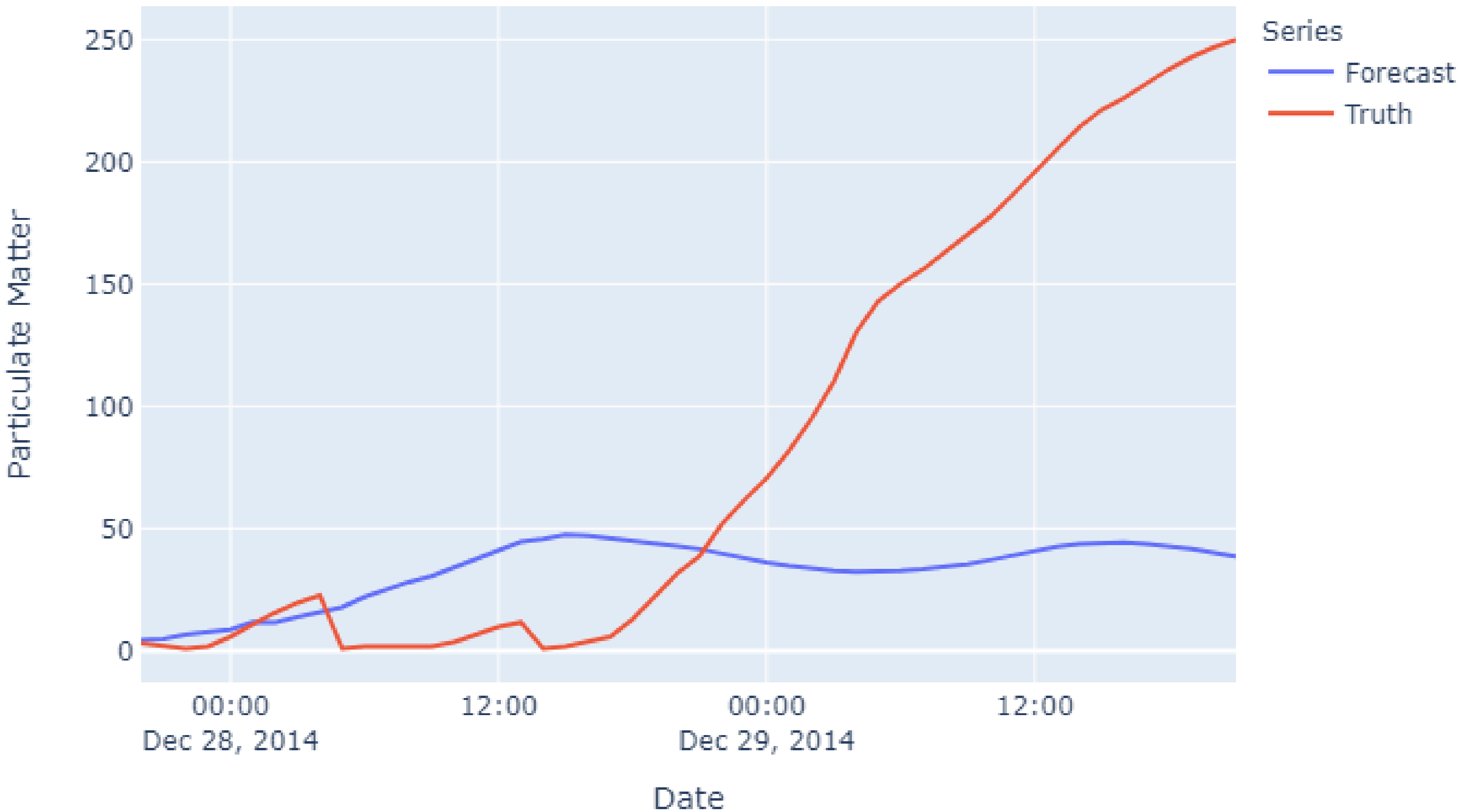
Temperature



Air Pressure



Wind Speed


Series
Forecast
Truth
250
200
150
100
50
0
Particulate Matter
00:00
Dec 28, 2014
12:00
00:00
Dec 29, 2014
12:00
Date

Forecasting Observations

- Repeated Forecasts are needed when data is updated
- Forecasts are not accurate far into the future
- They DO, however, manage to move in the right direction pre-emptively! (A bigger deal than it sounds)

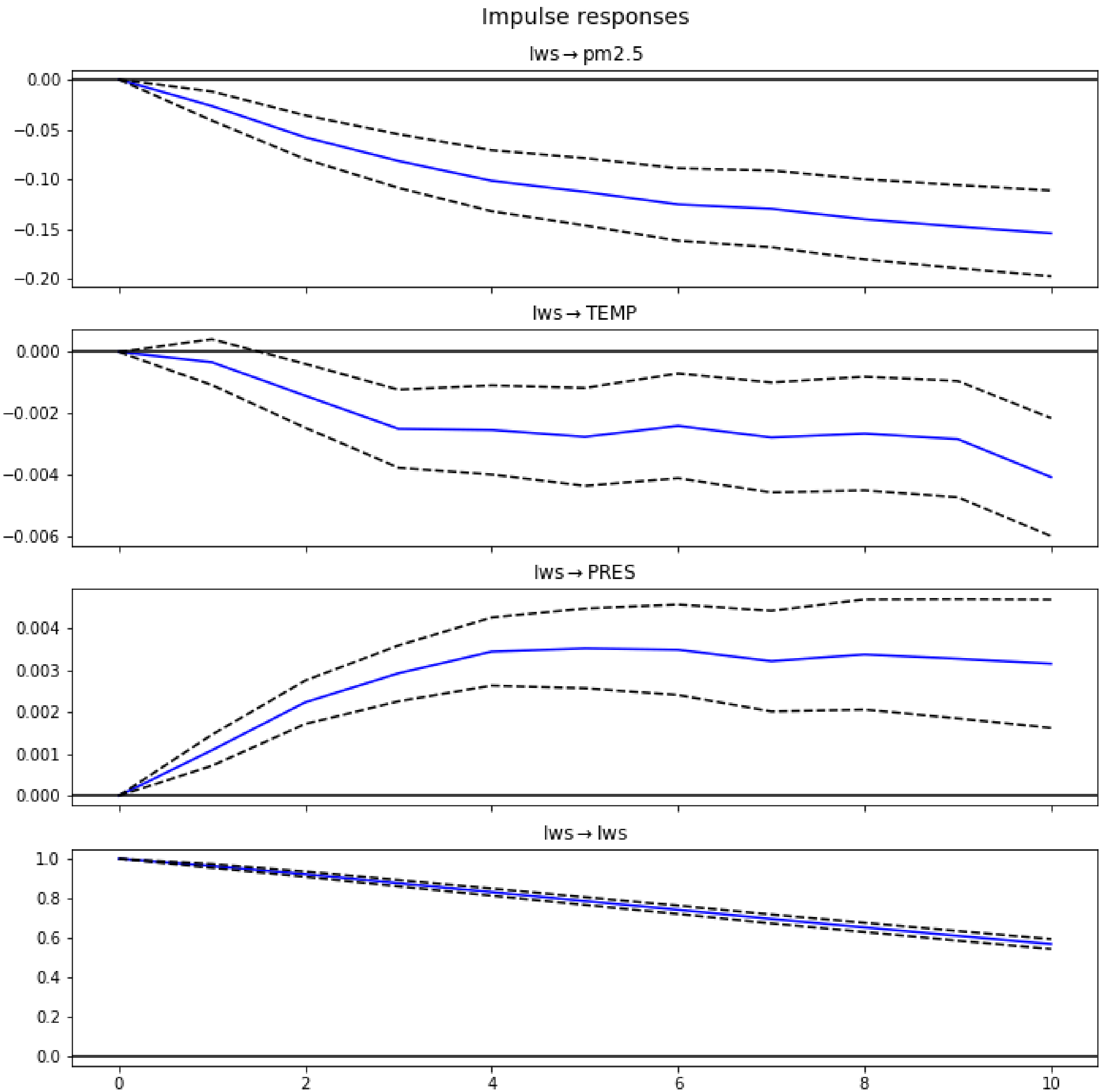
Impulse Response Functions

- VAR Models can show us how each variable responds to a shock in our system
- Frequently used to determine impact of policy changes or economic shocks in Macro models
- Give us insight into how our VAR model perceives the relationship between parameters over time

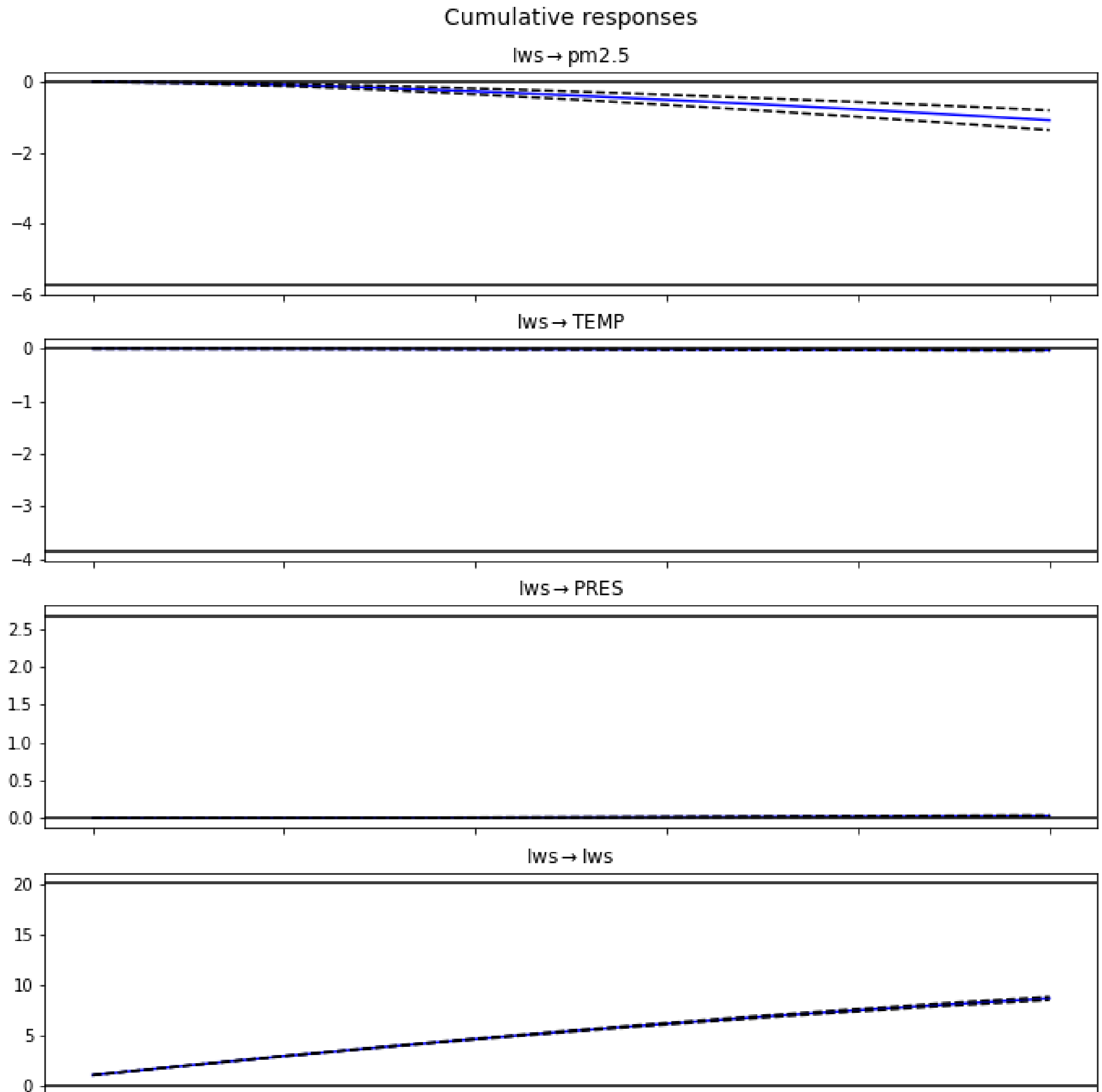
Impulse Response Functions

```
irf = modelFit.irf(10) # 10-period Impulse Response Fn
irf.plot(impulse = 'Iws') # Plot volume change impact
irf.plot_cum_effects(impulse = 'Iws') # Plot effects
```

- Generate a 10-period Impulse Response Function (IRF)
- Focus on plotting the effect of changes in trade volume on all variables (over 10 periods)
- Plot the cumulative effect over 10 periods



Impulse responses
lws → pm2.5
0.00
−0.05
−0.10
−0.15
−0.20
lws → TEMP
0.000
−0.002
−0.004
−0.006
lws → PRES
0.004
0.003
0.002
0.001
0.000
lws → lws
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10


Cumulative responses
lws → pm2.5
0
−2
−4
−6
lws → TEMP
0
−1
−2
−3
−4
lws → PRES
2.5
2.0
1.5
1.0
0.5
0.0
lws → lws
20
15
10
5
0

VARMAX

We can extend our by adding exogenous variables into the mix, as well as Moving Average terms (shocks based on last period's errors).

Using the Vector AutoRegressive Moving Average with eXogenous regressors (VARMAX) model, this is straightforward

VARMAX

```
# Prep our data
columns = ['pm2.5','TEMP','PRES','Iws', 'DEWP']
varData = data[columns].dropna()[-500:-50]
exog = varData['DEWP']
varData.drop('DEWP', axis=1, inplace=True)

# Create and fit model
model = sm.tsa.VARMAX(endog = varData.values,
        exog=exog.values,
        order=(10,0)) # define the order here for VARMAX!
modelFit = model.fit()
modelFit.summary()
```

VARMAX

This is a pretty robust time series model! We can explore many facets of our time series, as well as the interconnected system of variables that define local weather (or other systems), and generate figures to explain these relationships to stakeholders.

Lab Time!