# Lectures 2 & 3: Time Series, ARIMA Models

This lesson is based on material by [Robert Nau, Duke University]

# Time Series Data

A time series consists of repeated observations of a single variable, $y$, at various times, $t$.

$$\mathbf{y} = \{y_1, y_2, y_3, \ldots, y_t\}$$

We seek to predict $y_{t+1}$ using the information from previous observations $\mathbf{y}$.

# Time Series Data

In order to estimate $y_{t+1}$, we need to find the effect of previous observations of $y$ on the upcoming period. We might write this model as

$$y_{t+1} = \alpha + \sum_{s=1}^{t} \beta_s \cdot y_s + \epsilon$$

# Time Series Data

If we choose to base our model solely on the previous period, then the model would be written

$$y_{t+1} = \alpha + \beta_t \cdot y_t + \epsilon$$

Critically, OLS estimates of this model are invalid.

# Autocorrelation

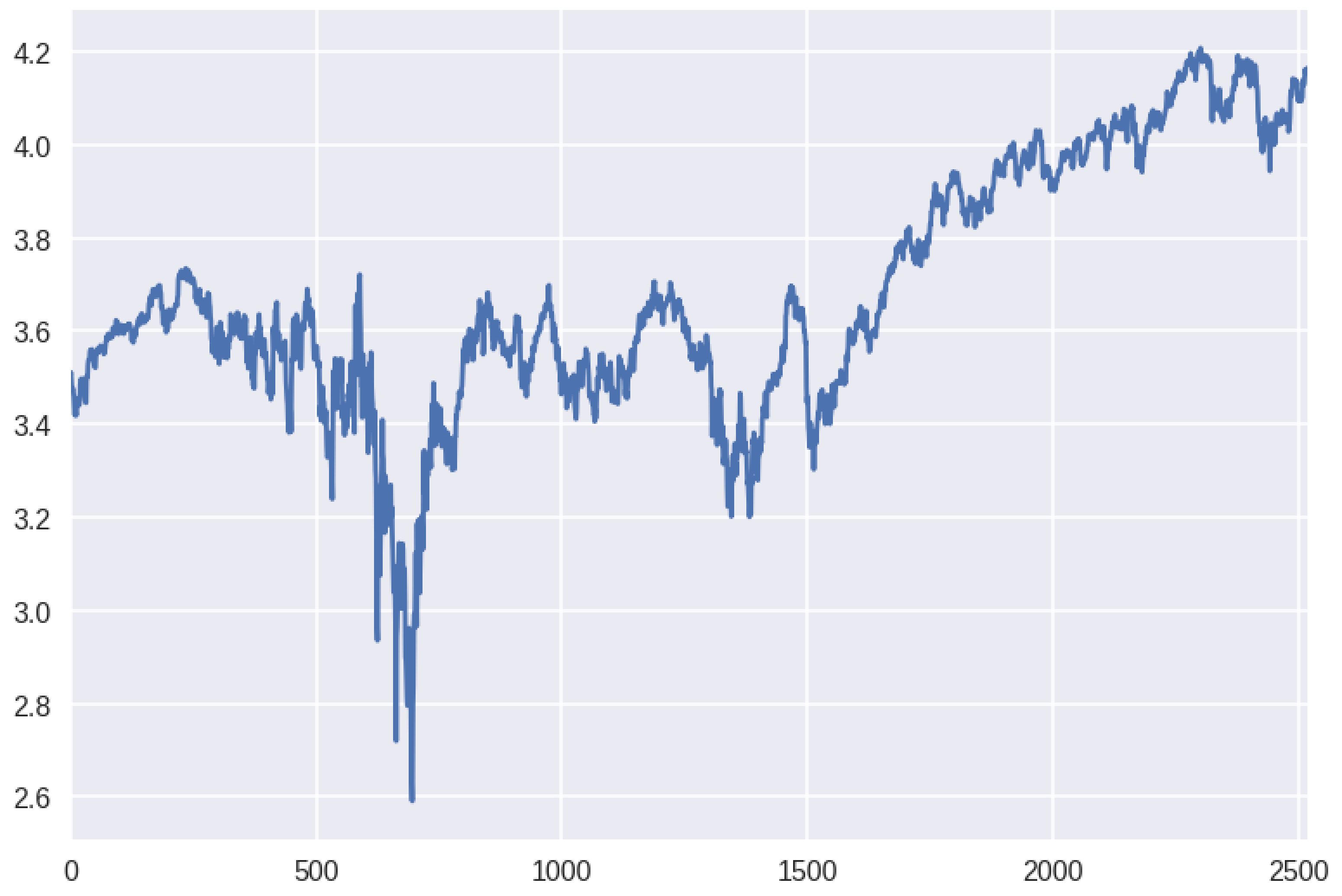
One of the primary assumptions of the OLS model is that

$$Cov(\epsilon_t, \epsilon_s) = 0,\ \forall\ t \neq s$$

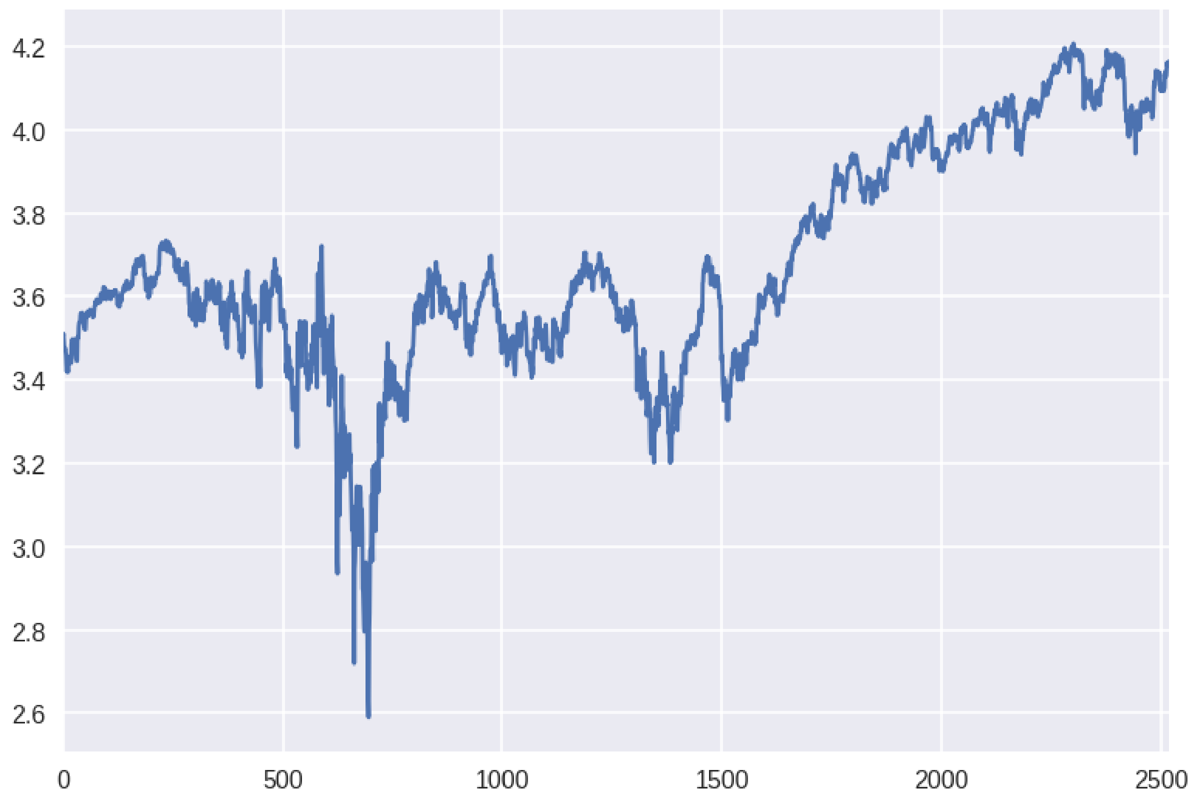
This assumption is clearly **not** valid in the case of time series data.

Let's look at some data to find out why.

# Autocorrelation



# Autocorrelation



We need to find a model that can eliminate the autocorrelation almost always seen in time series data.

# Autoregressive Models

AR models are based on the premise that deviation from the underlying trend in the data persists in all future observations.

$$y_t = \alpha + \sum_{i=1}^{p} \rho_i \cdot y_{t-i} + \epsilon_t$$

Where $\rho$ is the correlation term between periods and $\epsilon$ is an error (shock) term

# AR Models

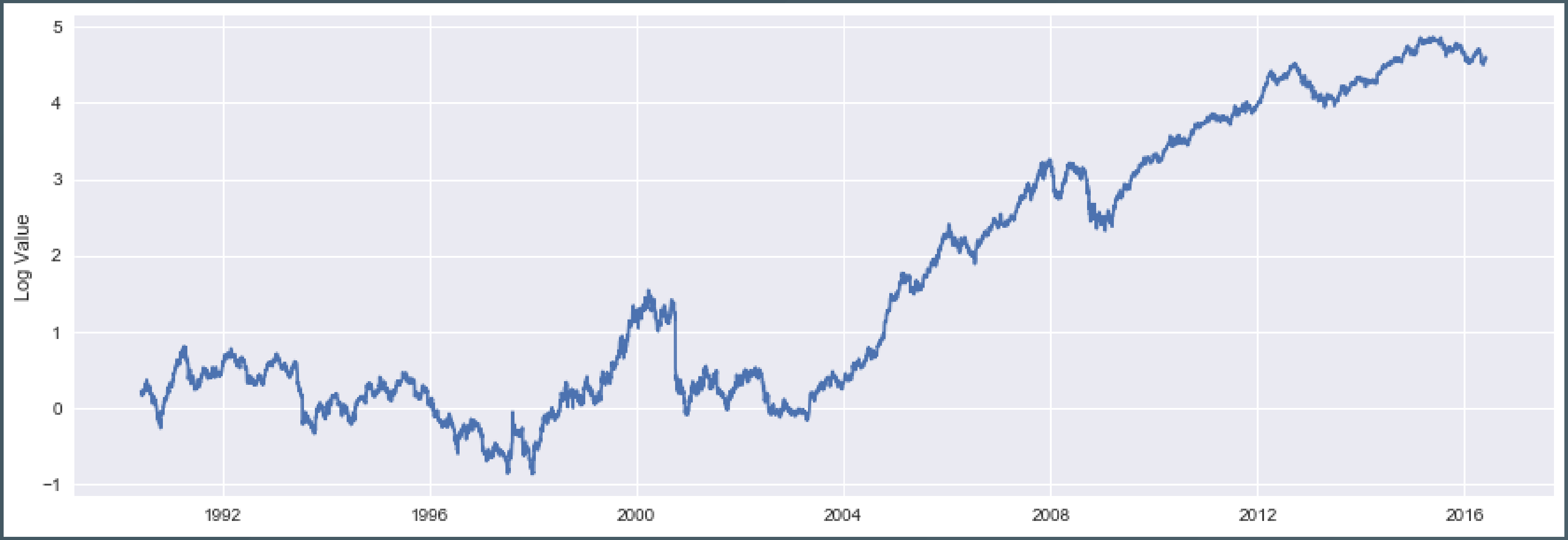
- We need to consider lagged observations of $y$ in order to predict future outcomes
- The number of lags that we include is the **order** of our AR model
  - The model is an AR(p) Model, where p is the order of the model

# AR Models

- The AR coefficients tell us how quickly a model returns to its mean
    - If the coefficients on AR variables add up to close to 1, then the model reverts to its mean **slowly**
    - If the coefficients sum to near zero, then the model reverts to its mean **quickly**

# Integrated Models

Integration occurs when a process is non-stationary. A non-stationary process is one that contains a linear time trend. One example might be a long-term series of stock prices:



# Integrated Models

We need to ensure that our data is stationary. To do so, we need to remove the time-trend from the data.

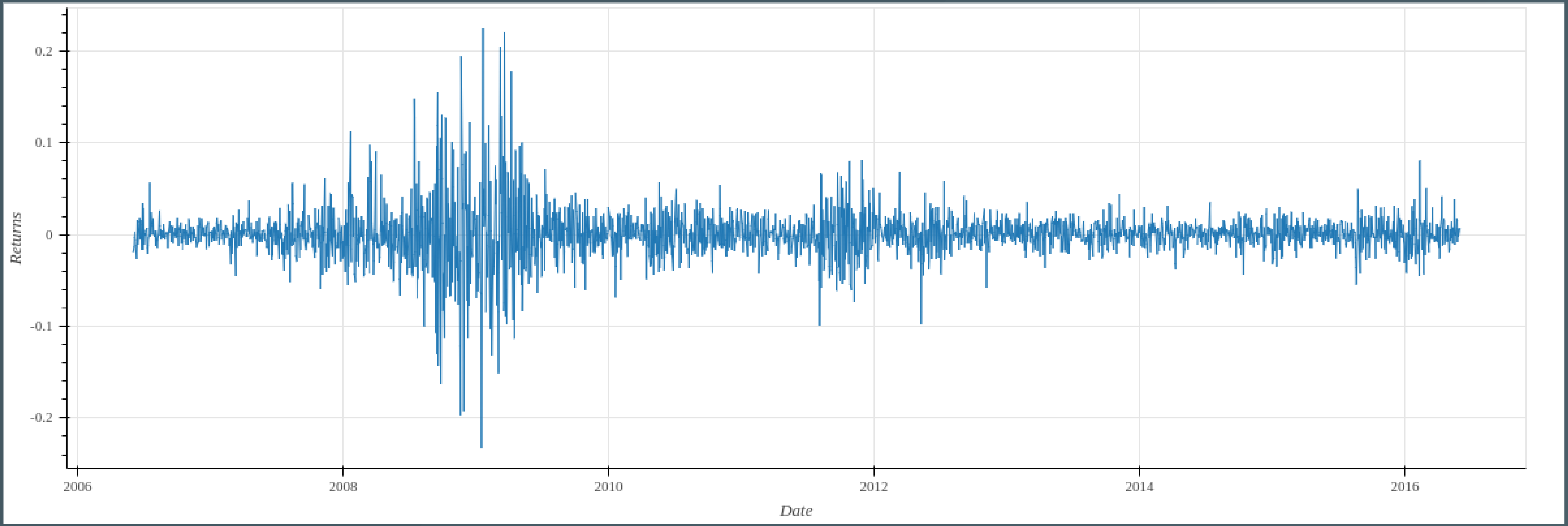
- This is typically done through differencing

$$y_i^s = y_i - y_{i-1}$$

where $y_t^s$ is the stationary time series based on the original series $y_t$

# Integrated Models

Here, the time trend has been differenced out of the data from the previous plot



# Integrated Models

The Integration term $d$ represents the number of differencing operations performed on the data:

- I(1): $ y^s_t = y_t - y_{t-1} $
- I(2): $ y^s_t = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) $

Where an I(2) model is analogous to a standard difference-in-differences model applied to time-series data.

# Moving Average Models

While an AR(·) model accounts for previous values of the dependent variable, MA(·) models account for previous values of the **error** terms:

$$AR(p) = \alpha + \sum_{i=1}^{p} \rho_i \cdot y_{t-i} + \epsilon_t$$

$$MA(q) = \alpha + \sum_{i=1}^{q} \theta_i \cdot \epsilon_{t-i} + \epsilon_t$$

# Moving Average Models

An MA model suggests that the current value of a time-series depends linearly on previous error terms.

- Current value depends on how far away from the underlying trend previous periods fell
- The larger $\theta$ becomes, the more persistent those error terms are

# Moving Average Models

- AR models' effects last infinitely far into the future
  - Each observation is dependent on the observation before
- In an MA model, the effect of previous periods only persists $q$ periods into the past
  - Each error is uncorrelated with previous errors

# Putting it Together

In order to account for all the problems that we might encounter in time series data, we can make use of ARIMA models.

**A**uto**R**egressive **I**ntegrated **M**oving **A**verage models allow us to

- Include lags of the dependent variable
- Take differences to eliminate trends
- Include lagged error terms

# Putting it Together

Even better, we can use ARIMA**X** models to include exogenous regressors in our estimations!

Now we just need to understand how to decide on the correct specifications for our model.

# The ARIMA(X) Model

ARIMA models are often referred to as ARIMA($p$, $d$, $q$) models, where $p$, $d$, and $q$ are the parameters denoting the order of the autoregressive terms, integration terms, and moving average terms, respectively.

- It is often a matter of guessing and checking to find the correct specification for a model
- We can use the ACF and PACF graphs to visually determine the order of our model

# The Autocorrelation Function (ACF)

The ACF illustrates the correlation between a dependent variable and its lags.

- Choose how many lags to explore (based on nature of data)
- **Reminder**: correlations will vary between -1 and 1, with 1 being perfect correlation, and -1 being perfect inverse correlation
- Correlation can be cyclical!

# The Autocorrelation Function (ACF)

# The Partial Autocorrelation Function

The PACF illustrates the correlation between a dependent variable and its lags, **after controlling for lower-order lags**.

- Choose how many lags to explore (based on nature of data)

# The Partial Autocorrelation Function (PACF)

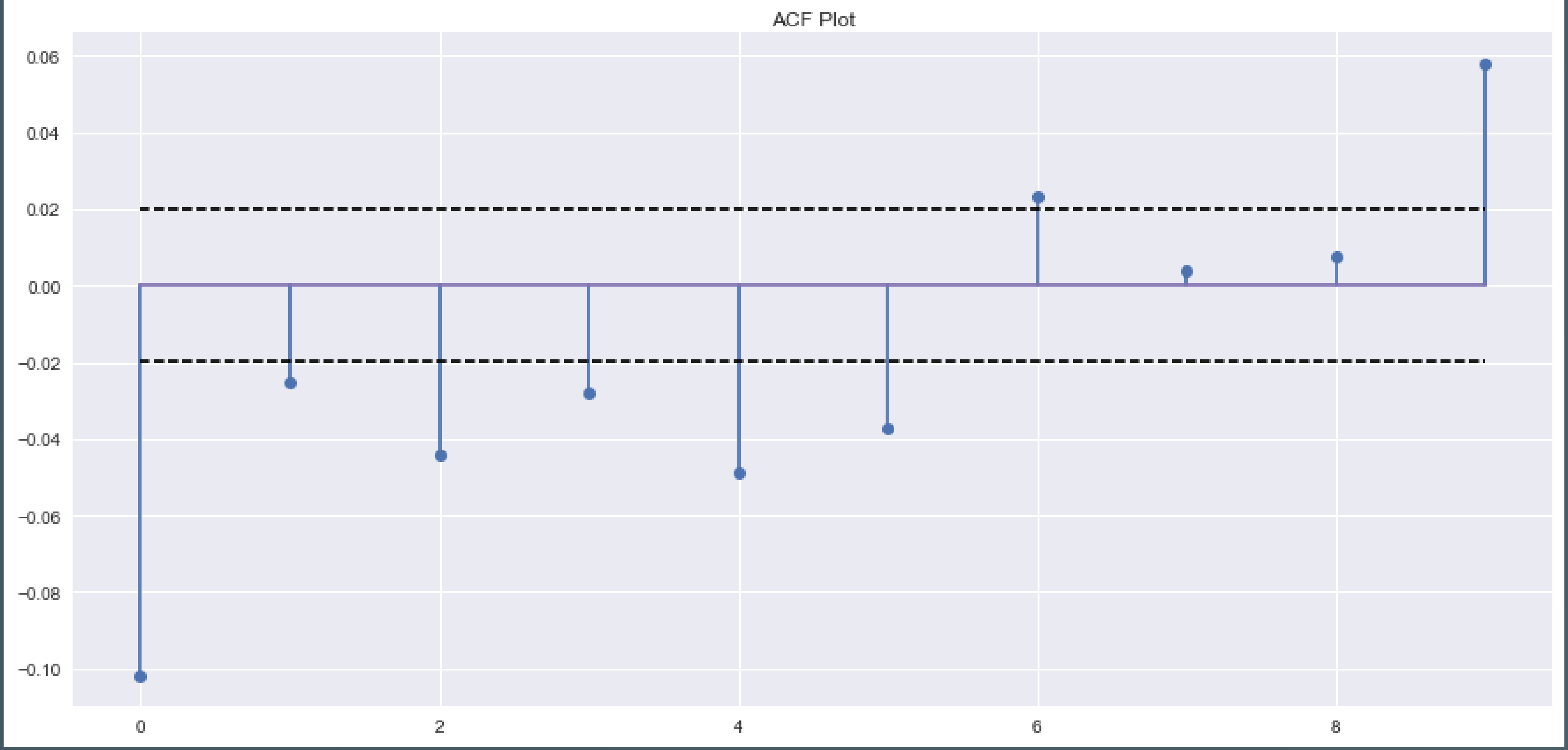
# Building the Model

1. Make the series **stationary**
    - When the ACF falls "quickly" to zero at higher lags, the series is stationary
    - Can also use a **unit root test** to check for stationarity

# Building the Model

# Building the Model

Stationary:



# Building the Model

1. Make the series **stationary**
2. Use ACF and PACF plots to decide if you should include **AR** or **MA** terms in your model
   - Typically, we do not use both in the same model

# Building the Model

Signatures of **AR** and **MA** models:

**AR** Model: ACF dies out gradually, and the PACF cuts off sharply after a few lags

**MA** Model: ACF cuts off sharply, and PACF dies off more gradually (remember that **MA** models are based on previous *errors*)

# Building the Model

1. Make the series **stationary**
2. Use ACF and PACF plots to decide if you should include **AR** or **MA** terms in your model
3. Fit the model, and check residual ACF and PACF for lingering significance
4. If there are significant terms in residual ACF or PACF, add **AR** or **MA** terms, and try again

# ARIMA(X) in Python

```
# Import pandas, numpy, and libraries for ARIMA models,
#     for tools such as ACF and PACF functions, plotting,
#     and for using datetime formatting
import pandas as pd
import numpy as np
from statsmodels.tsa.arima_model import ARIMA
import statsmodels.tsa.stattools as st
import matplotlib.pyplot as plt
from datetime import datetime

# Import the pandas datareader function
from pandas_datareader.data import DataReader

# Collect data
a = DataReader('AAPL', 'yahoo', datetime(1990,6,1),
                datetime(2016,6,1))
```

# ARIMA(X) in Python

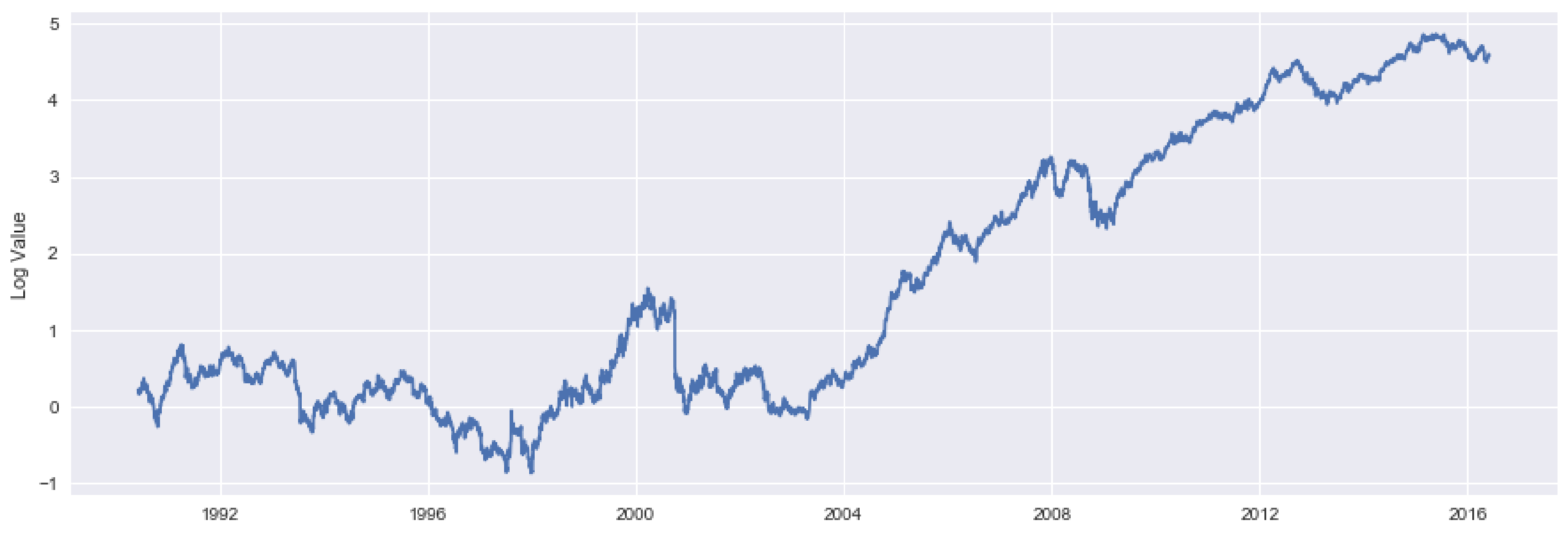
```python
# Generate DataFrames from raw data
a_ts = pd.DataFrame(np.log(a['Adj Close'].values))
a_ts.columns = ["Index"]
a_ts['date'] = a.index.values
```

Here, we generate the time-series data that we will work with as we explore our ARIMA(X) models

- Take the logged price data, and put it into a separate DataFrame for analysis

# ARIMA(X) in Python

```python
# Plot the data
p = figure(plot_width = 1200, plot_height=400,
        y_axis_label="Log Value",
        x_axis_label="Date",
        x_axis_type="datetime")
p.line(a_ts['date'], a_ts['Index'])
show(p)
```



# ARIMA(X) in Python

```
# Generate plot from ACF
acf, aint=st.acf(a_ts['Index'], nlags=10, alpha=.05)
# Create figure, add ACF values
p = figure(plot_width = 800, plot_height = 600)
p.vbar(x = list(range(1,11)), width = 0.5, top = acf[1:],
        bottom = 0)
# Confidence Intervals
p.line(list(range(1,11)), [1/np.sqrt(len(a_ts))]*10,
        color = 'black', line_dash = "dashed")
p.line(list(range(1,11)), [-1/np.sqrt(len(a_ts))]*10,
        color = 'black', line_dash = "dashed")
show(p)
```

# ACF Plot

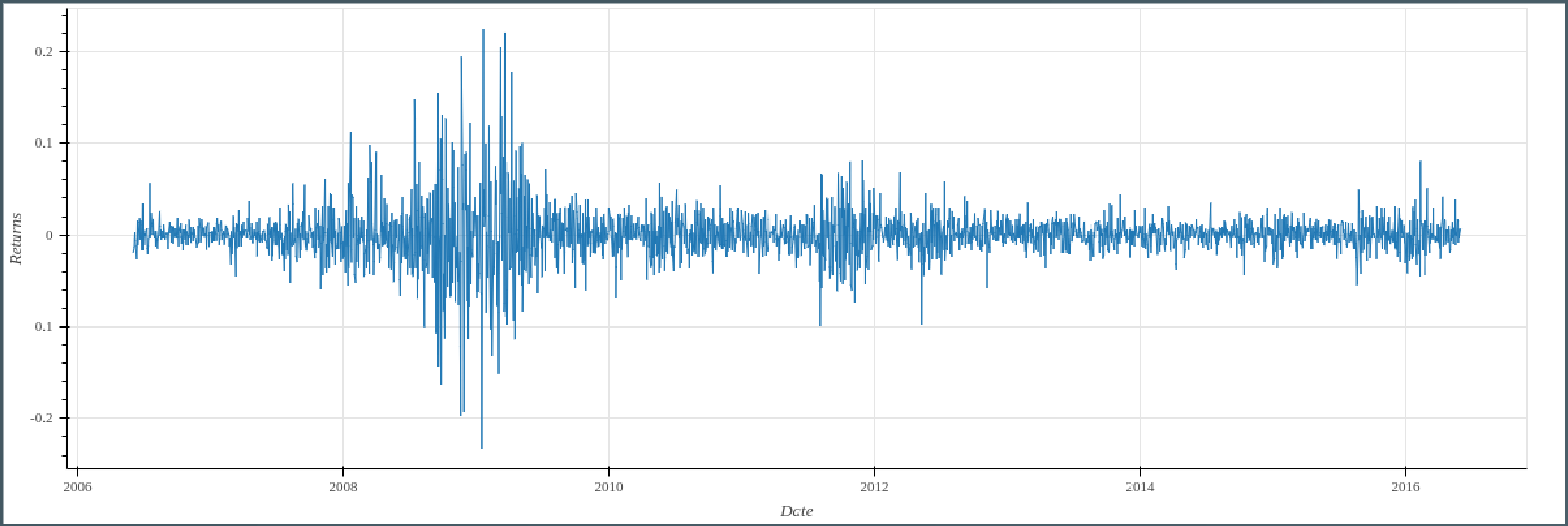
# ARIMA(X) in Python

```
# Generate plot from PACF
pacf, paint=st.pacf(a_ts['Index'], nlags=10, alpha=.05)
# Create figure, add ACF values
p = figure(plot_width = 800, plot_height = 600)
p.vbar(x = list(range(1,11)), width = 0.5, top = pacf[1:],
        bottom = 0)
# Confidence Intervals
p.line(list(range(1,11)), [1/np.sqrt(len(a_ts))]*10,
        color = 'black', line_dash = "dashed")
p.line(list(range(1,11)), [-1/np.sqrt(len(a_ts))]*10,
        color = 'black', line_dash = "dashed")
show(p)
```

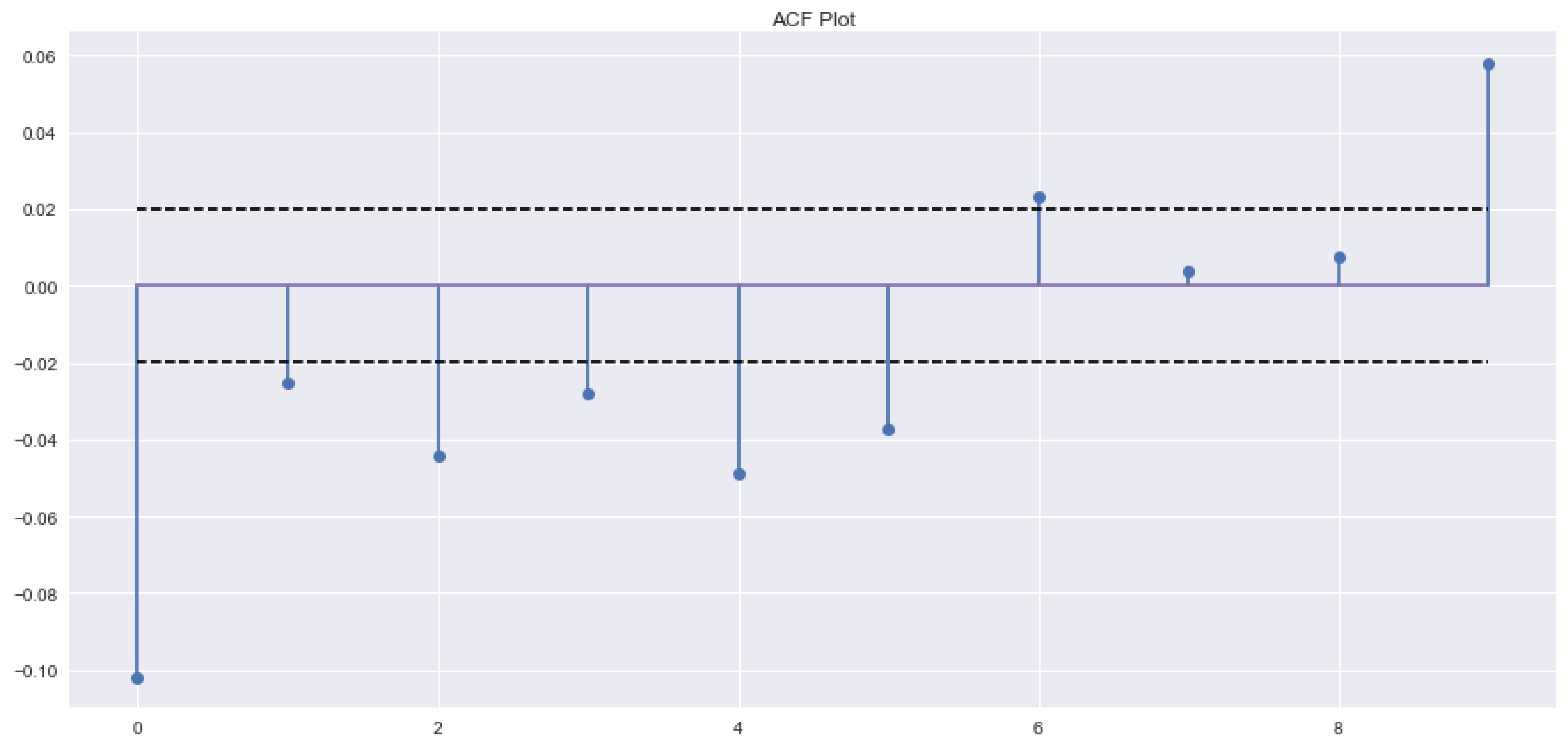
# PACF Plot

# ARIMA(X) in Python

```
# Plot first differences
plt.figure(figsize=(15, 5))
plt.ylabel("Returns")
plt.plot(np.diff(a_ts["Index"])[1:]) # plot differenced
plt.show()                           # log price series
```



# Differenced ACF Plot



This looks a lot more like white noise than the undifferenced ACF plot!

# Finding the Right Fit

- Time series models are unique in Econometrics: we need to **visually** diagnose the proper specifications for our model
    - This takes practice
    - This takes repetition and iteration for any given model

# Fitting the ARIMA(X) model

```python
from statsmodels.tsa.arima_model import ARIMA

model = ARIMA(a_ts, (1,1,1)) # Use a_ts data to fit an
                             # ARIMA(1,1,1) model
reg = model.fit() # Fit the model using standard params
res = reg.resid # store the residuals as res
```

Once we fit the ARIMA model using our selected specification, we can then explore the residual ACF and PACF of the model.

# Fitting the ARIMA(X) model

Residual ACF (the text is dark)

# Fitting the ARIMA(X) model

Residual PACF (the text is dark) - nearly identical to the ACF plot (what does that mean?)

# Looking Ahead

Now that we have a fitted model, we can start to make predictions

```
fcst = reg.forecast(steps=10) # Generate forecast
future = pd.DatetimeIndex(start=datetime(2016,6,2),
                          freq='D', periods=10) # Index
predicted = pd.DataFrame(fcst[0], columns = ['Index'],
                         index = future) # Map forecast
upper = fcst[2][:,1] # Specify upper 95% CI
lower = fcst[2][:,0] # Specify lower 95% CI
```

We make our out-of-sample forecast, and store it as a DataFrame, with dates as index values

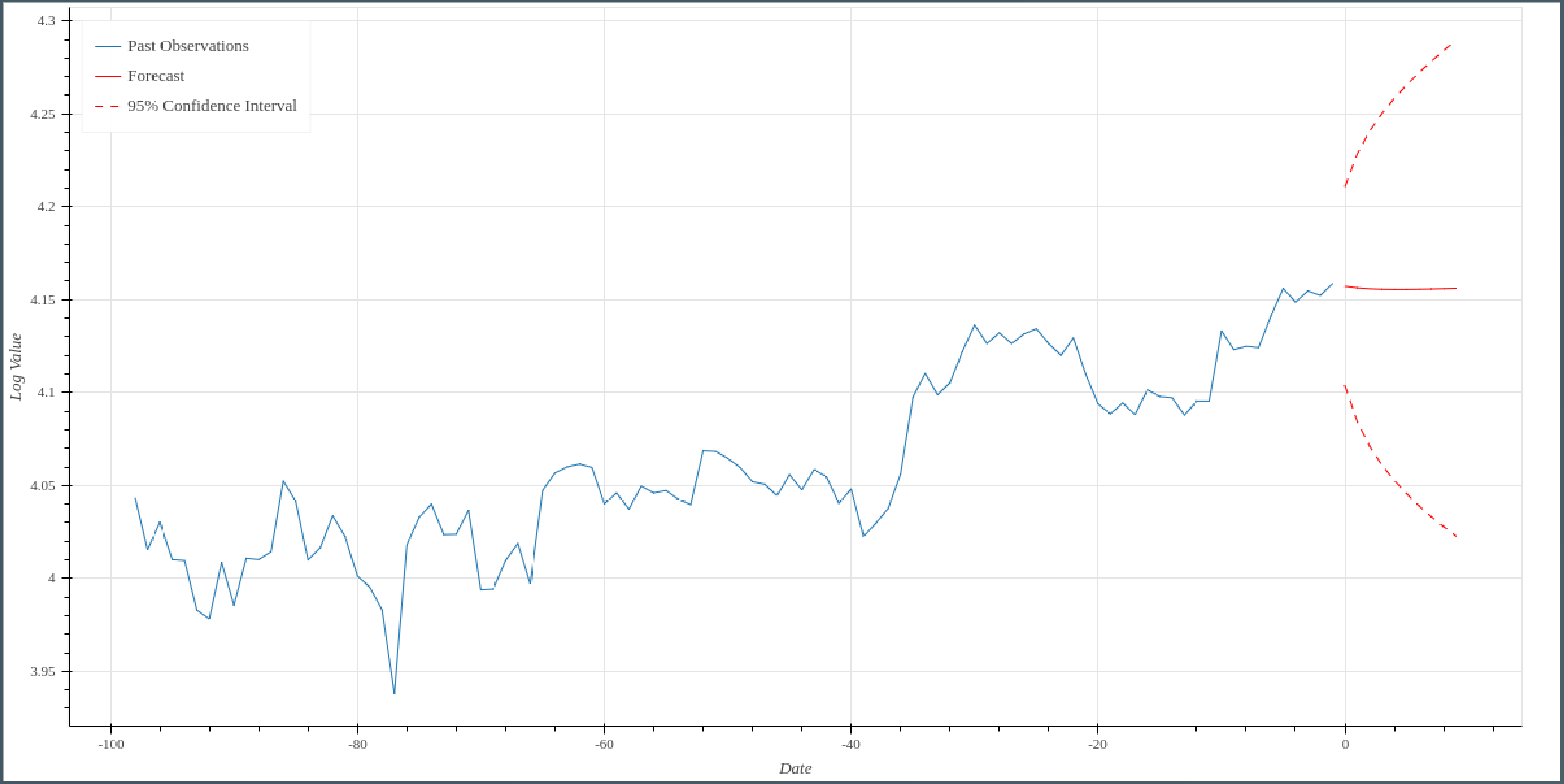
# Looking Ahead

```
p = figure(plot_width = 1200, plot_height=400,
        y_axis_label="Log Value",
        x_axis_label="Date")
p.line(list(range(-98,0)), a_ts['Index'][-98:],
        legend="Past Observations")
rng = list(range(0,10))
p.line(rng, predicted['Index'], color = 'red',
        legend="Forecast")
p.line(rng, upper, color = 'red', line_dash = 'dashed',
        legend="95% Confidence Interval")
p.line(rng, lower, color = 'red', line_dash = 'dashed')
p.legend.location="top_left"
show(p)
```

We can then take a look at how our prediction follows the pattern from our time series

# Looking Ahead

Plotting the forecast,



# Finding the Right Fit

In order to more carefully choose the proper specification, we will actually select a preliminary model, and then investigate the **residual** ACF and PACF plots

- We can start with an ARIMA(0,1,0) model, or perhaps an ARIMA(1,0,0) model
- Base our starting point on the shape of our time series, or intuition about the data

# For lab today:

Working with your group, use the Omaha historic weather data (using all but the final 10 days) to:

- Plot the data
- Make the data stationary
- Fit an ARIMA model
- Validate the model by plotting residuals
- Forecast temperature 10 days into the future, and send me your forecast, so we can compare to the real temperature