GENERAL ASSEMBLY

# TIME SERIES MODELING

*Sri Kanajan*

# LEARNING OBJECTIVES

- Model and predict from time series data using AR, ARMA, or ARIMA models
- Specifically, coding these models in `statsmodels`

COURSE

# PRE-WORK

# PRE-WORK REVIEW

- Prior exposure to linear regression with discussion of coefficients and residuals

OPENING

# TIME SERIES MODELING

# TIME SERIES MODELING

- In the last class, we focused on exploring time series data and common statistics for time series analysis.
- In this class, we will advance those techniques to show how to predict or forecast forward from time series data.
- With a sequence of values (a time series), we will use the techniques in this class to predict a future value.

# TIME SERIES MODELING

- There are many times when you may want to use a series of values to predict a future value.
  - The number of sales in a future month
  - Anticipated website traffic when buying a server
  - Financial forecasting
  - The number of visitors to your store during the holidays

## INTRODUCTION

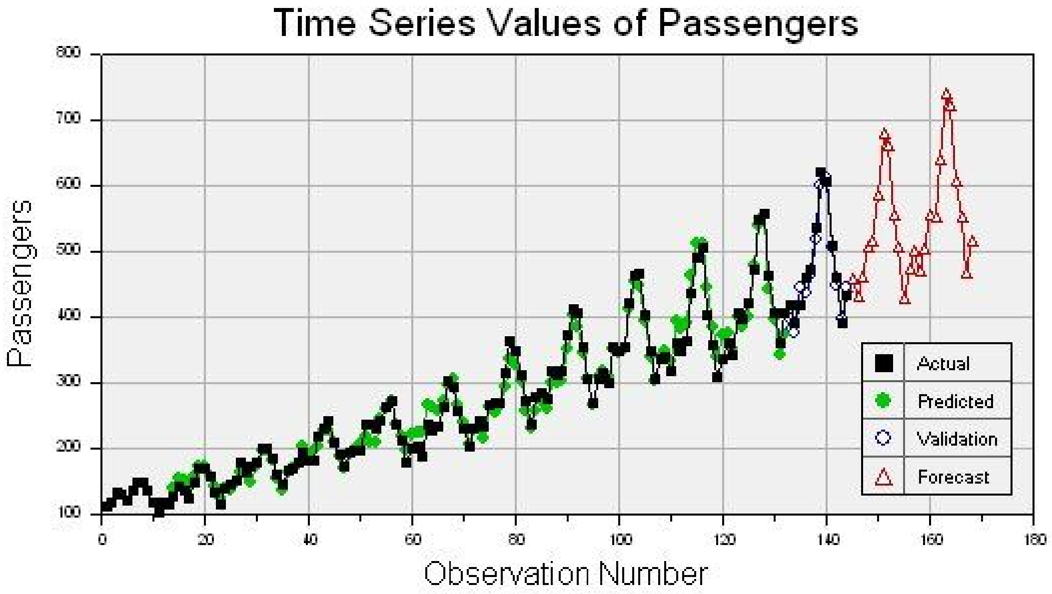
# WHAT ARE TIME SERIES MODELS?

# WHAT ARE TIME SERIES MODELS?

- Time series models are models that will be used to predict a future value in the time series.

- **Like** other predictive models, we will use prior history to predict the future.

- **Unlike** previous models, we will use the earlier in time *outcome* variables as *inputs* for predictions.



# PROPERTIES FOR TIME-SERIES PREDICTION

- A *moving average* is an average of $p$ surrounding data points in time.

... to $t$ - $p$ + $1$. This includes t, t + 1, t + 2, ..., t-p, t-p+1.

$$F_t = \frac{1}{p} \sum_{k=t}^{t-p+1} Y_k$$

Divide by the $p$ surrounding data points

Get the $p$ points from $t$ ...

# PROPERTIES FOR TIME-SERIES PREDICTION

- *Autocorrelation* is how correlated a variable is with itself. Specifically, how related are variables earlier in time with variables later in time.

$$r_k = \frac{\sum_{t=k+1}^{n} (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{n} (y_t - \bar{y})^2}$$

- We fix a *lag*, k, which is how many time points earlier we should use to compute the correlation.

# PROPERTIES FOR TIME-SERIES PREDICTION

- We can use these values to assess how we plan to model our time series.
- Typically, for a high quality model, we require some autocorrelation in our data.
- We can compute autocorrelation at various lag values to determine how far back in time we need to go.

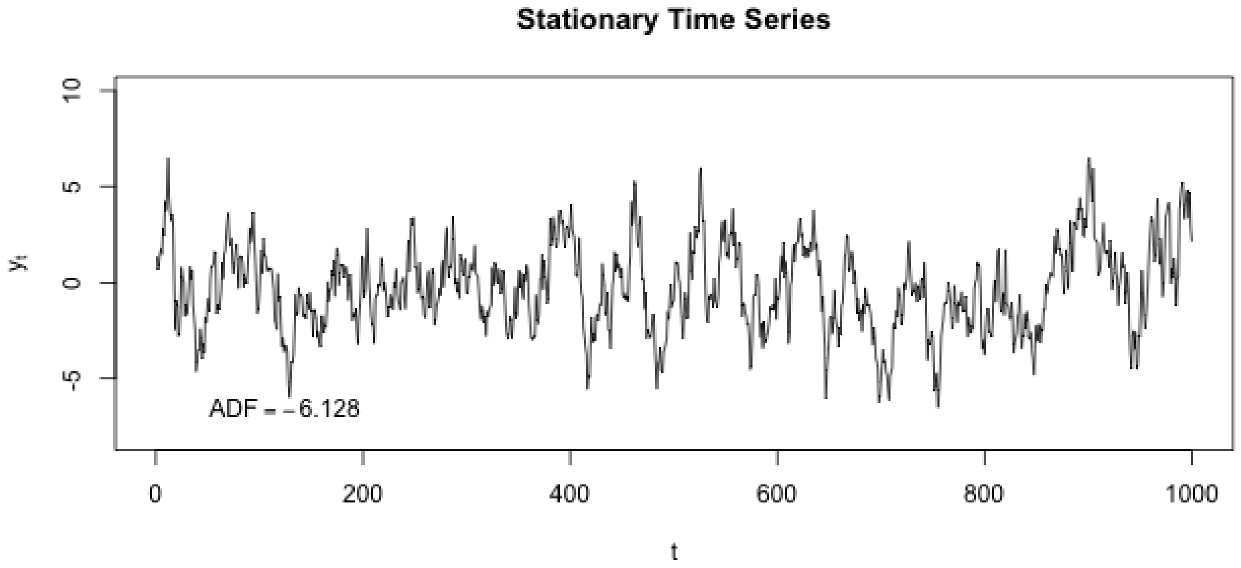
# PROPERTIES FOR TIME-SERIES PREDICTION

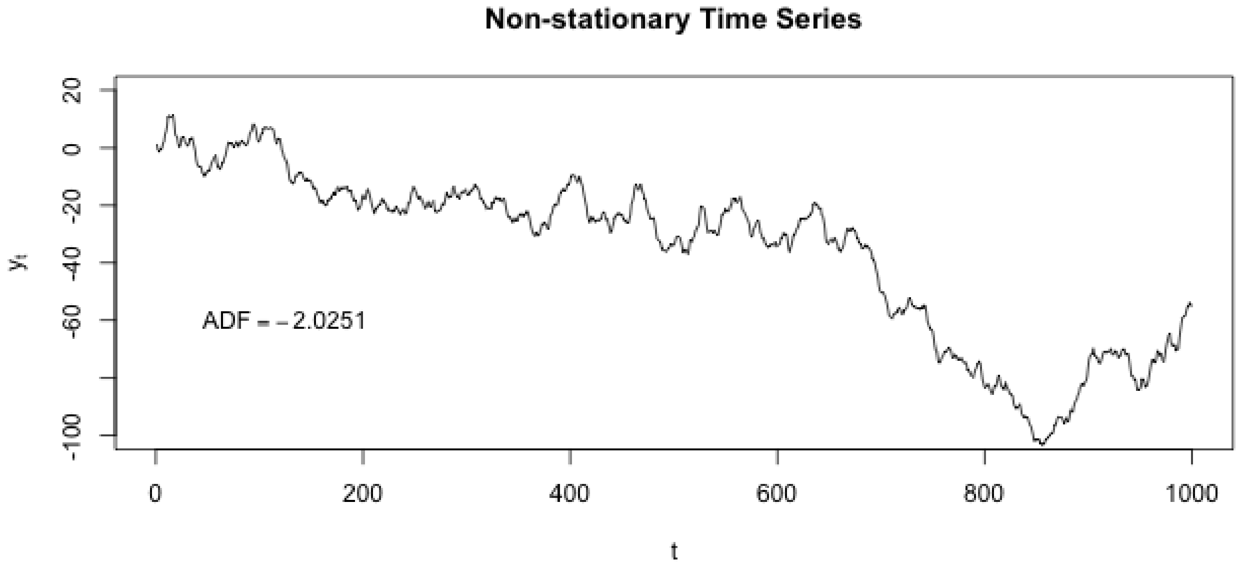
- Many models make an assumption of *stationarity*, assuming the mean and variance of our values is the *same* throughout.

- While the values (e.g. of sales) may shift up or down over time, the mean and variance of sales is constant (i.e. there aren't many dramatic swings up or down).

- These assumptions may not represent real world data; we must be aware of that when we are breaking the assumptions of our model.

# PROPERTIES FOR TIME-SERIES PREDICTION



# PROPERTIES FOR TIME-SERIES PREDICTION

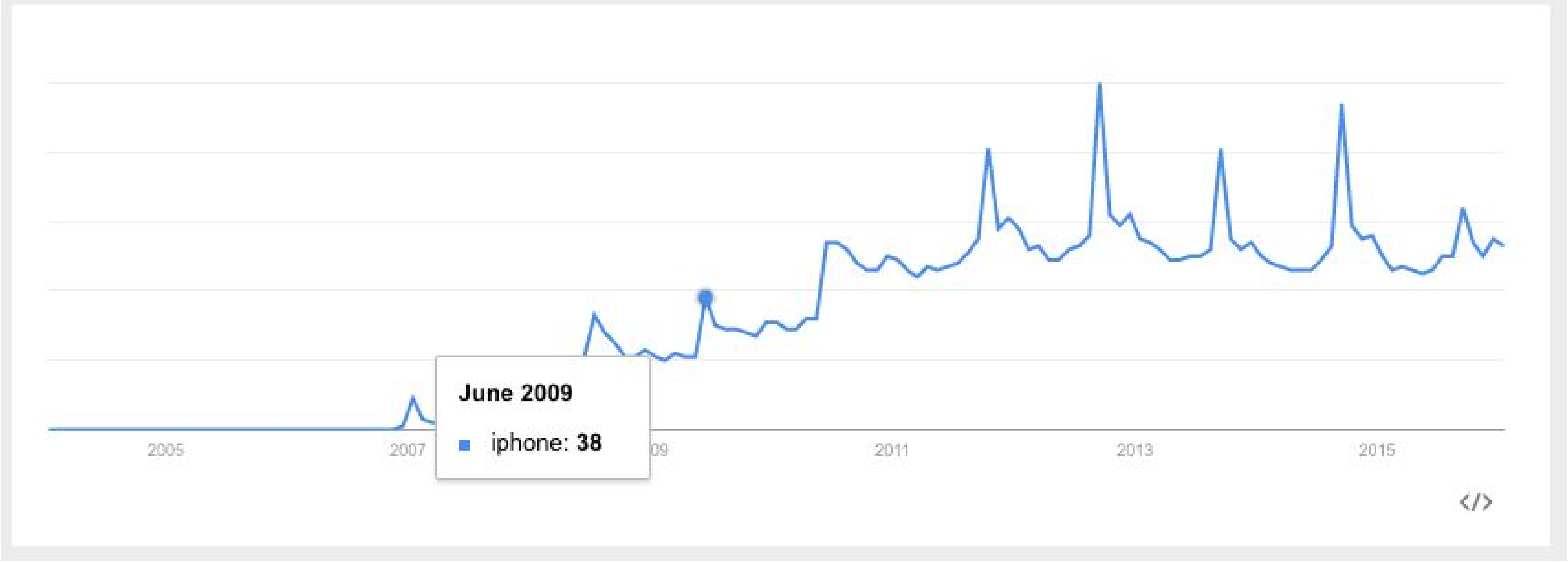
- Often, if these assumptions don't hold, we can alter our data to make them true. Two common methods are *detrending* and *differencing*.

- *Detrending* would mean to remove any major trends in our data.

- We could do this is many ways, but the simplest is to fit a line to the trend and make a new series that is the difference between the line and the true series.

# PROPERTIES FOR TIME-SERIES PREDICTION

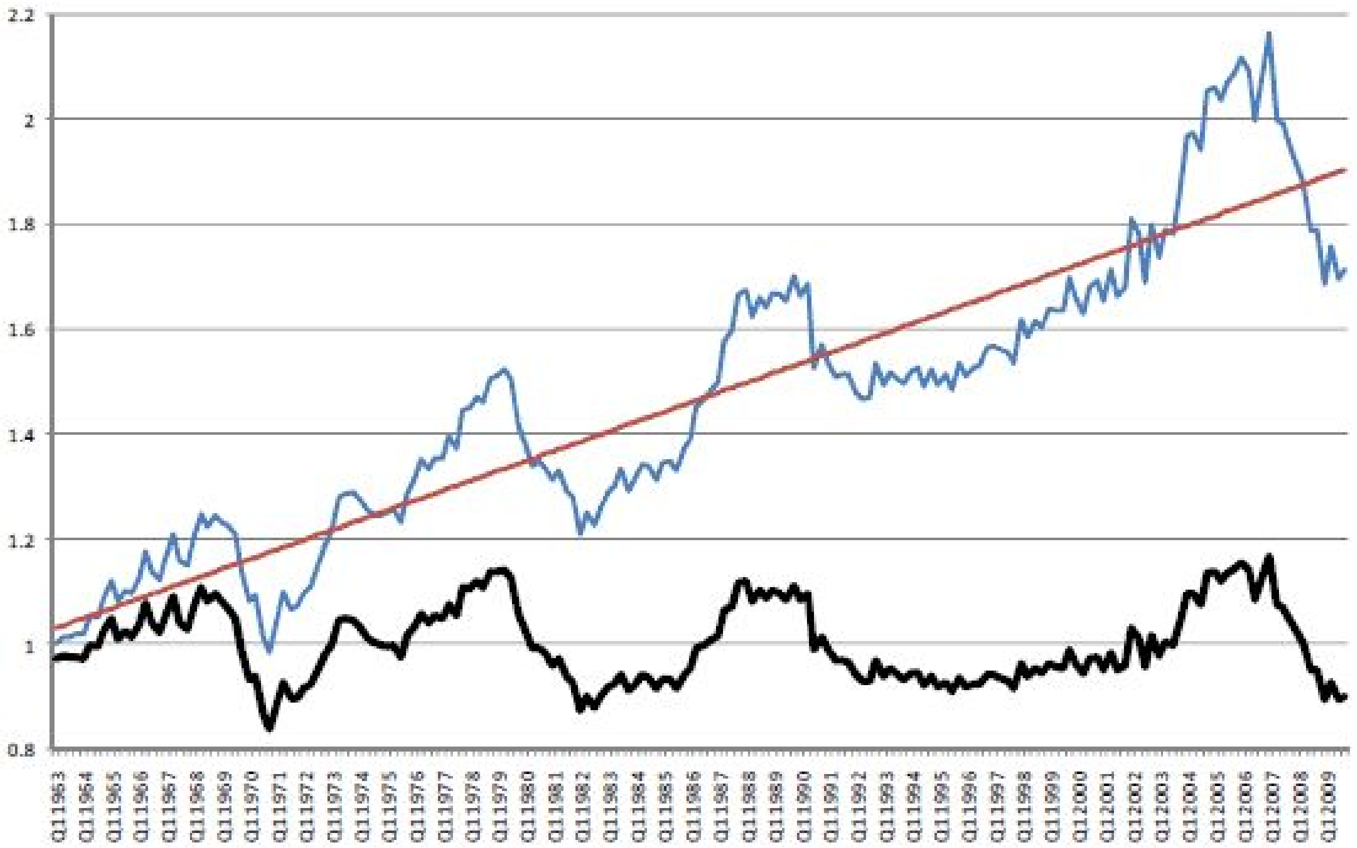
- For example, there is a clear upward (non-stationary) trend in google searches for “iphone”.




- If we fit a line to this data first, we can create a new series that is the difference between the true number of searches and the predicted searches.

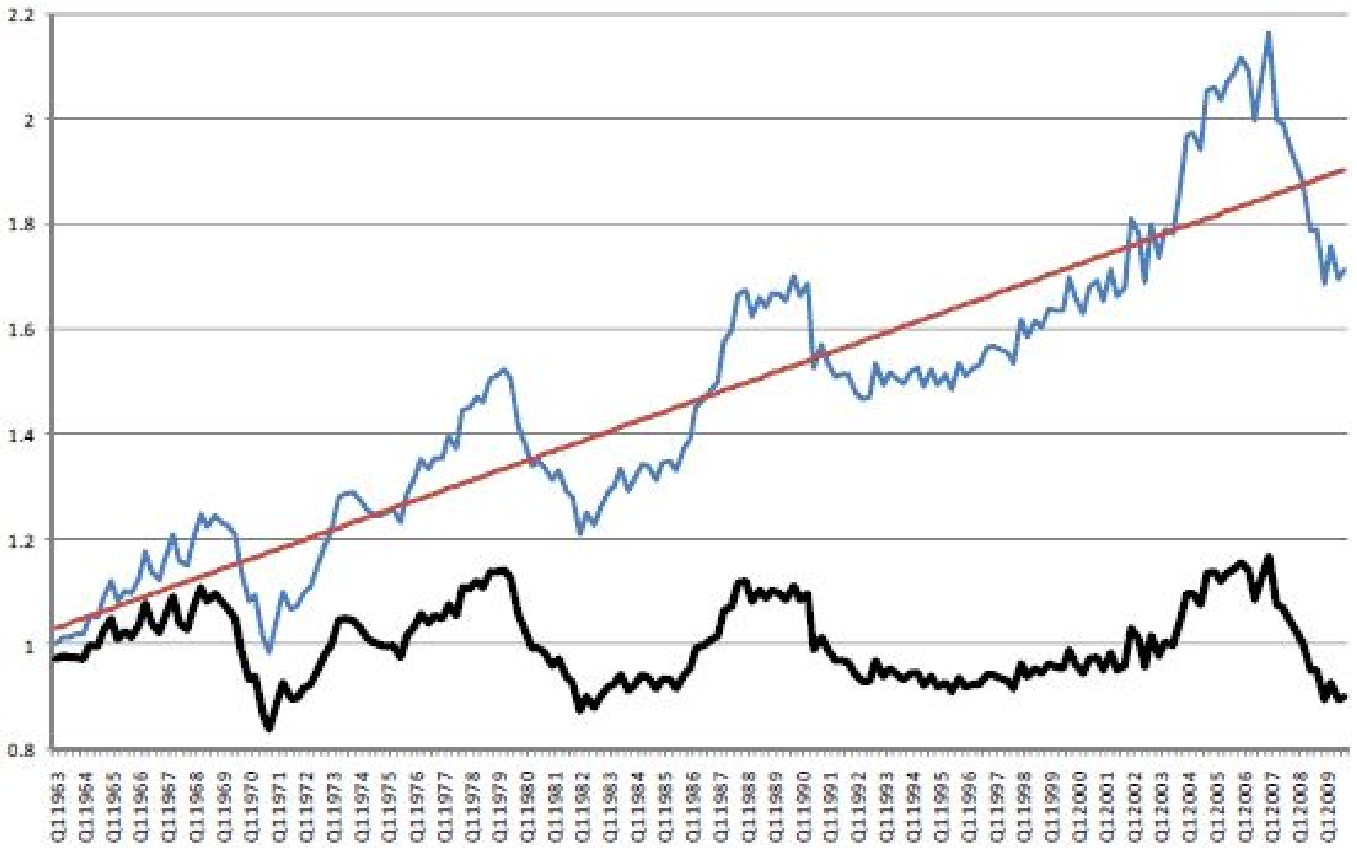
# PROPERTIES FOR TIME-SERIES PREDICTION

- Below is an example where we look at US housing prices over time. Clearly, there is an upward trend, making the time series non-stationary (ie: the mean house price is increasing).



# PROPERTIES FOR TIME-SERIES PREDICTION

- We can fit a line that represents the trend. With our trend line, we can subtract the trend line value from the original value to get the bottom figure.

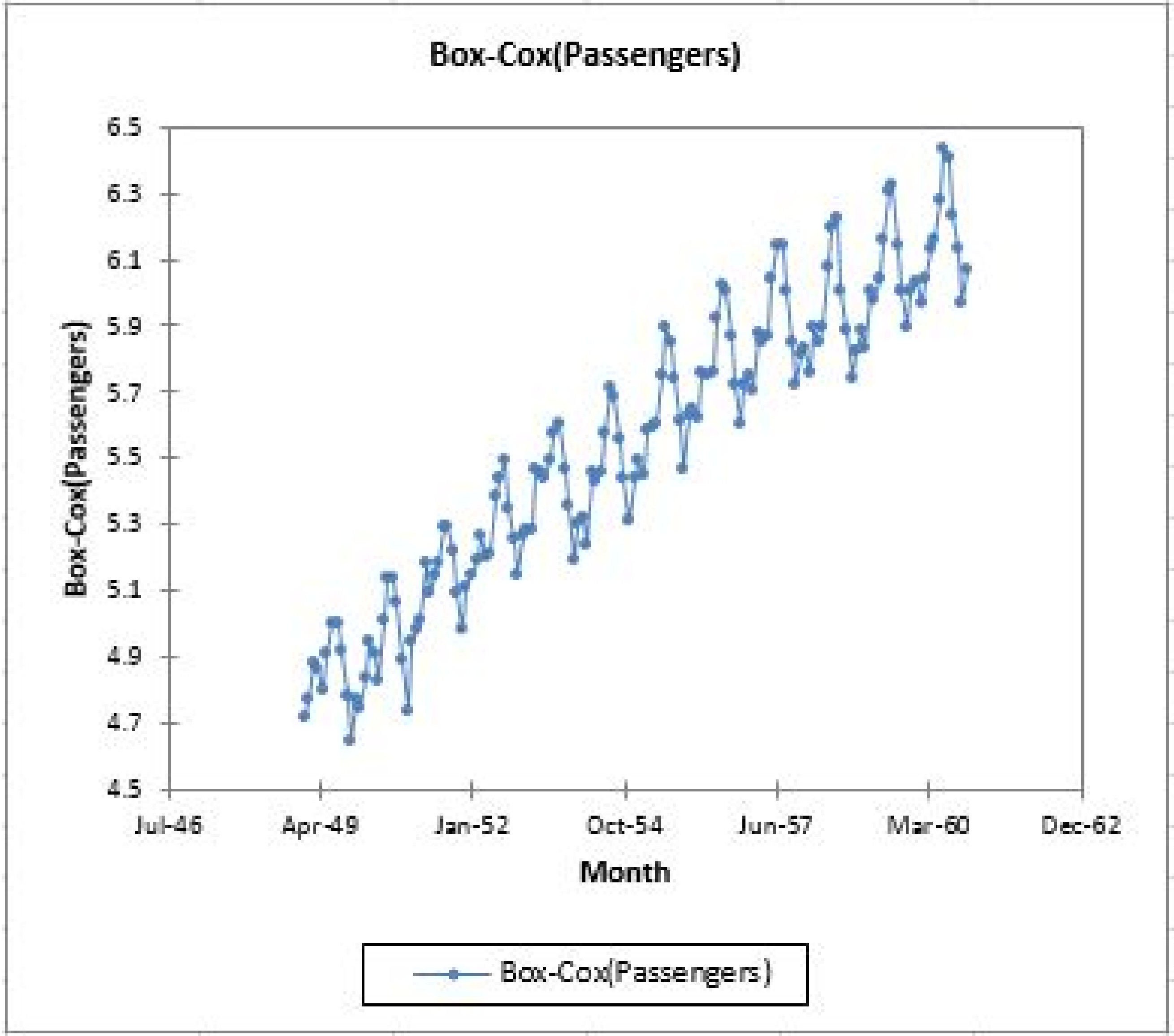


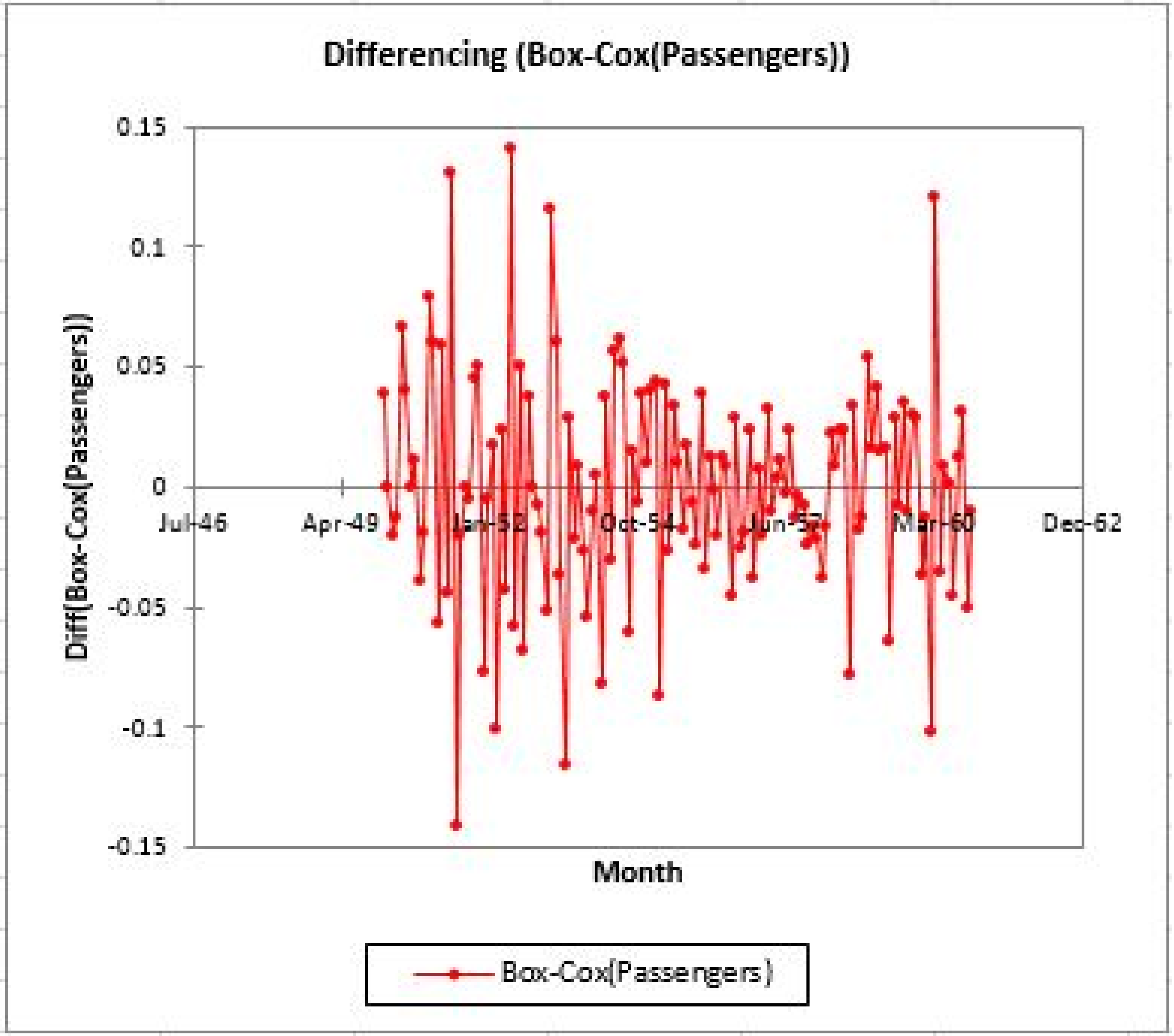
## PROPERTIES FOR TIME-SERIES PREDICTION

- A simpler method is *differencing*. This is very closely related to the `diff` function we saw in the last class.

- Instead of predicting the series (again our non-stationary series), we can predict the difference between two consecutive values.

# PROPERTIES FOR TIME-SERIES PREDICTION



# TIME SERIES MODELS

- In the rest of this lesson, we are going to build up to the **ARIMA** time series model.

- This model combines the ideas of differencing and two models we will see.

    - **AR** - autoregressive models

    - **MA** - moving average models

## AR MODELS

- Autoregressive (AR) models are those that use data from previous time points to predict the next.
- This is very similar to previous regression models, except as input, we take the previous outcome.
- If we are attempting to predict weekly sales, we use the sales from a previous week as input.
- Typically, AR models are notes AR(p) where *p* indicates the number of previous time points to incorporate, with AR(1) being the most common.

# AR MODELS

- In an autoregressive model, similar to standard regression, we are learning regression coefficients for each of the *p* previous values. Therefore, we will learn *p* coefficients or **β** values.

- If we have a time series of sales per week, $y_i$, we can regress each yi from the last *p* values.

$$y_i = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 y_{i-1} + \boldsymbol{\beta}_2 y_{i-2} + \ldots + \boldsymbol{\beta}_p y_{i-p} + \varepsilon$$

- As with standard regression, our model assumes that each outcome variable is a linear combination of the inputs and a random error term.

# AR MODELS

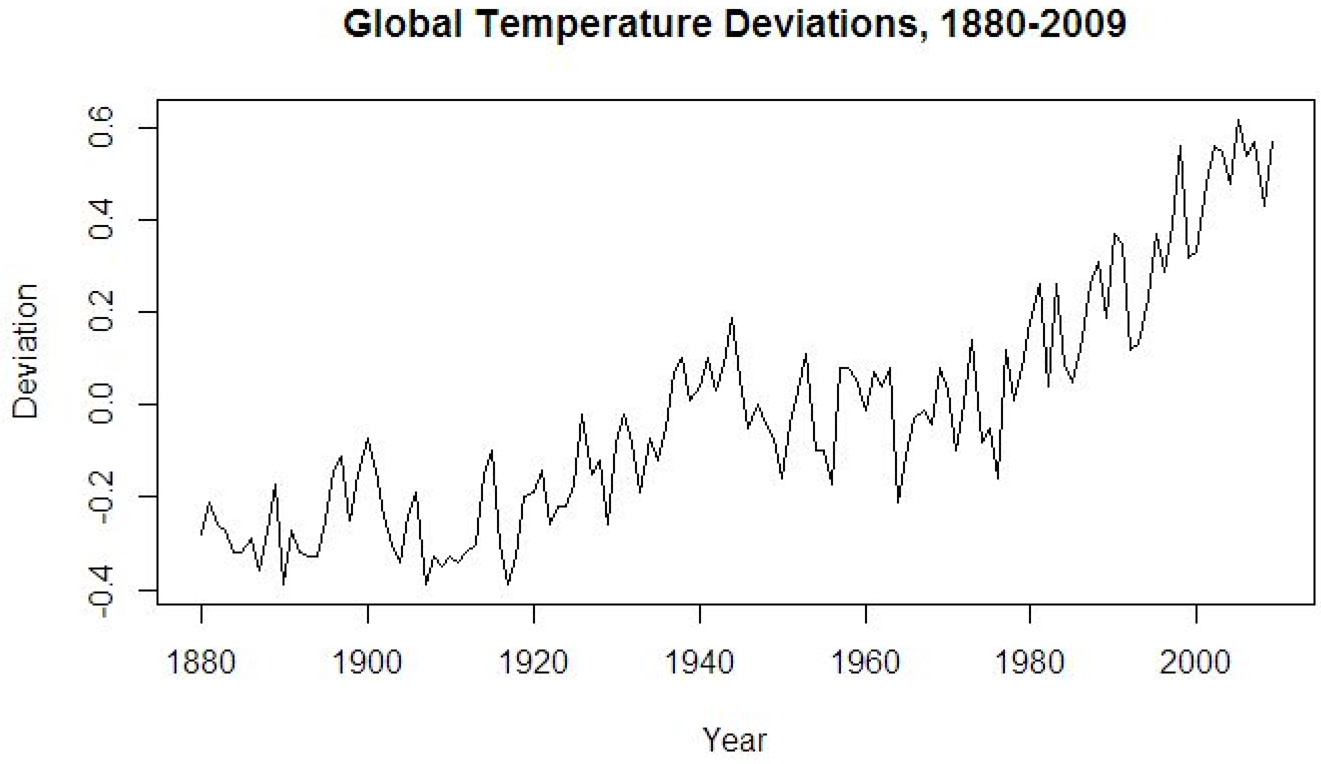
- For an AR(1) model, we will learn a single coefficient.

- This coefficient, $\boldsymbol{\beta}$, will tell us the relationship between the previous value, $Y_{t-1}$, and the next value, $Y_t$.

$$Y_t = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 \cdot Y_{t-1}$$

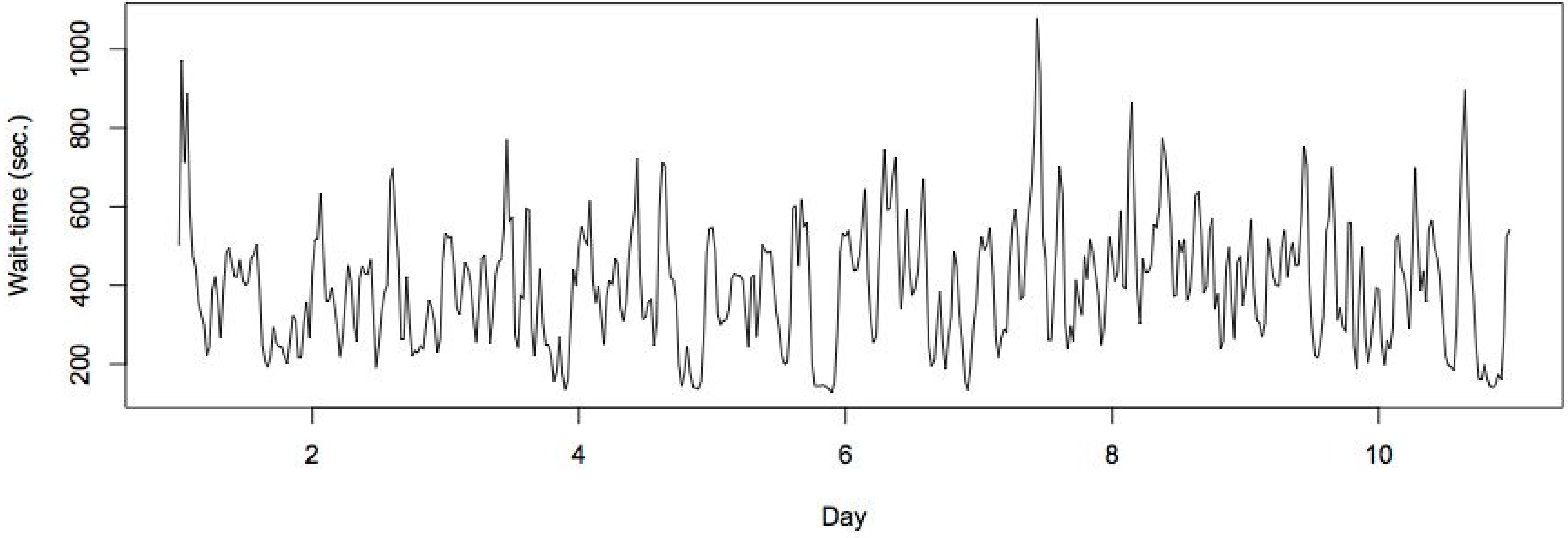
# AR MODELS

- A value > 1 would indicate a growth over previous values.  This would typically represent non-stationary data, since if we compound the increases, the values are continually increasing.



# AR MODELS

- Values between 1 and -1 represent increasing and decreasing patterns from previous patterns.



# AR MODELS

- As with other models, interpretation of the model becomes more complex as we add more factors.
- Going from AR(1) to AR(2) can add significant *multi-collinearity*.

## AR MODELS

- Recall that *autocorrelation* is the correlation of a value with its series *lagged* behind.
- A model with high correlation implies that the data is highly dependent on previous values and an autoregressive model would perform well.

# AR MODELS

- Autoregressive models are useful for learning falls or rises in our series.
- This will weight together the last few values to make a future prediction.
- Typically, this model type is useful for small-scale trends such as an increase in demand or change in tastes that will gradually increase or decrease the series.
- This model does not capture the seasonal element
- You can identify the number of AR components by observing the partial autocorrelation plot

# ACTIVITY: KNOWLEDGE CHECK

EXERCISE

## ANSWER THE FOLLOWING QUESTIONS

1. If we observe an autocorrelation near 1 for lag 1, what do we expect the single coefficient in an AR(1) model to be? >1, between 0 and 1, or <1?
2. What if we observe an autocorrelation of 0?

## DELIVERABLE

Answers to the above questions

## MA MODELS

- **Moving average (MA) models**, as opposed to AR models, do not take the previous outputs (or values) as inputs.  They take the previous error terms.

- We will attempt to predict the next value based on the overall average and how off our previous predictions were.

- The intuition is that the MA model captures the high frequency element

# MA MODELS

- This model is useful for handling specific or abrupt changes in a system.
- AR models slowly incorporate changes in the system by combining previous values; MA models use prior errors to quickly incorporate changes.
- This is useful for modeling a sudden occurrence - something going out of stock or a sudden rise in popularity affecting sales.

# MA MODELS

- As in AR models, we have an order term, *q*, and we refer to our model as MA(q). The moving average model is dependent on the last *q* errors.

- If we have a time series of sales per week, $y_i$, we can regress each $y_i$ from the last *q* error terms.

$$y_i = \text{mean of series} + \varepsilon_i + \boldsymbol{\beta}_1\varepsilon_{i-1} + \boldsymbol{\beta}_2\varepsilon_{i-2} + \ldots + \boldsymbol{\beta}_q\varepsilon_{i-q}$$

- We include the mean of the time series (that's why it's called a moving average) as we assume the model takes the mean value of the series and randomly jumps around it.

# MA MODELS

- Of course, we don't have error terms when we start - where do they come from?
- This requires a more complex fitting procedure than we have seen previously.
- We need to iteratively fit a model (perhaps with random error terms), compute the errors and then refit, again and again.

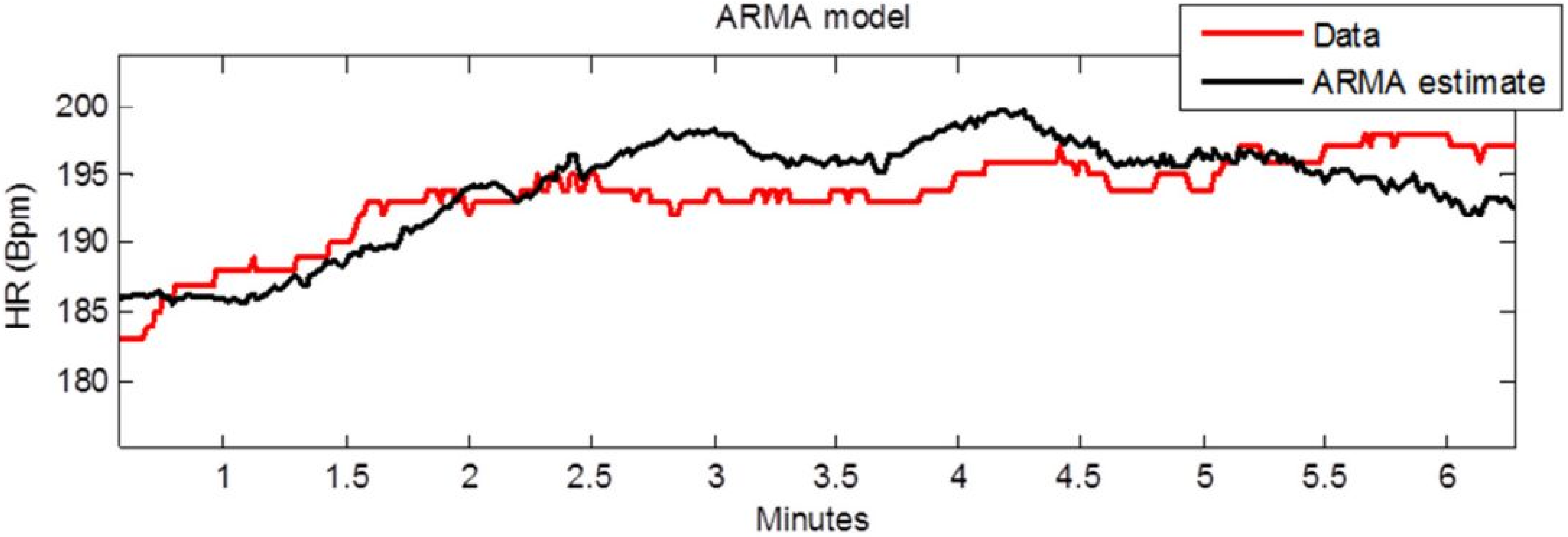
# MA MODELS

- In this model, we learn $q$ coefficients.
- In an MA(1) model, we learn one coefficient.
- This value indicates the impact of how our previous error term on the next prediction.

# ARMA MODELS

- **ARMA** (pronounced 'R-mah') models combine the autoregressive and moving average models.

- An ARMA(p,q) model is simply a combination (sum) of an AR(p) model and MA(q) model.



## ARMA MODELS

- We specify two model settings, *p* and *q*, which correspond to combining an AR(p) model with an MA(q) model.

- Incorporating both models allows us to mix two types of effects.

    - AR models slowly incorporate changes in preferences, tastes, and patterns.

    - Moving average models base their prediction on the prior error, allowing to correct sudden changes based on random events - supply, popularity spikes, etc.

## CONCLUSION

# TOPIC REVIEW

# CONCLUSION

- Time-series models use previous values to predict future values, also known as forecasting.
- AR and MA model are simple models on previous values or previous errors respectively.
- ARMA combines these two types of models to account for both gradual shifts (due to AR models) and abrupt changes (MA models).

# CONCLUSION

- Note that none of these models may perform well for data that has more random variation.

- For example, for something like iphone sales (or searches) which may be sporadic, with short periods of increases, these models may not work well.

COURSE

# BEFORE NEXT CLASS

LESSON

# Q & A

LESSON

# EXIT TICKET

DON’T FORGET TO FILL OUT YOUR EXIT TICKET

# DUE DATE

- Project: Final Project, Part 3

DEMO

# TIME SERIES MODELING IN STATSMODELS

# TIME SERIES MODELING IN STATSMODELS

- To explore time series models, we will use the Rossmann sales data.

- This dataset has sales data for every Rossmann store for a 3-year period and indicators for holidays and basic store information.

# TIME SERIES MODELING IN STATSMODELS

- In the last class, we saw that we could plot the sales data at a particular store to identify how the sales changed over time.
- We also computed autocorrelation for the data at varying lag periods. This helps us identify if previous timepoints are predictive of future data and which time points are most important - the previous day, week, or month.

# TIME SERIES MODELING IN STATSMODELS

- In this class, we will use `statsmodels` to code AR, MA, ARMA, and ARIMA models.

- `statsmodels` provides a nice summary utility to help us diagnose models.

# ARIMA MODELS IN STATSMODELS

- We can adjust the AR component of the model to adjust for a piece of this. Let's increase the lag to 7.

```
model = ARIMA(store1_sales_data, (7, 1, 2)).fit()
model.summary()

plot_acf(model.resid, lags=50)
```

- This removes some of the autocorrelation in the residuals but large discrepancies still exist.

- However, they exist where we are breaking our model assumptions.

# ARIMA MODELS IN STATSMODELS

- Increasing p increases the dependency on previous values further (longer lag). But our autocorrelation plots show this isn't necessary past a certain point.

- Increasing q increases the likelihood of an unexpected jump at a handful of points. The autocorrelation plots show this doesn't help past a certain point.

- Increasing d increases differencing, but d=1 moves our data towards stationarity (other than a few points). d=2 would imply an exponential trend which we don't have here.

INDEPENDENT PRACTICE

# WALMART SALES DATA

# ACTIVITY: WALMART SALES DATA

EXERCISE

## DIRECTIONS (50 minutes)

We will analyze the weekly sales data from Walmart over a two year period from 2010 to 2012. The data is separated by store and department, but we will focus on analyzing one store for simplicity.

To read in the data

```
import pandas as pd
import numpy as np

%matplotlib inline

data =
pd.read_csv('lessons/lesson-16/assets/data/train.csv')
data.set_index('Date', inplace=True)
data.head()
```

# ACTIVITY: WALMART SALES DATA

EXERCISE

## DIRECTIONS

Complete the following tasks:

1. Filter the dataframe to Store 1 sales and aggregate over departments to compute the total sales per store.
2. Plot the `rolling_mean` for `Weekly_Sales`. What general trends do you observe?
3. Compute the 1, 2, 52 autocorrelations for `Weekly_Sales` and/or create an autocorrelation plot.
4. What does the autocorrelation plot say about the type of model you want to build?

# ACTIVITY: WALMART SALES DATA

EXERCISE

## DIRECTIONS

5. Split the weekly sales data in a training and test set - using 75% of the data for training.
6. Create an AR(1) model on the training data and compute the mean absolute error of the predictions.
7. Plot the residuals - where are their significant errors?
8. Compute and AR(2) model and an ARMA(2, 2) model - does this improve your mean absolute error on the held out set?
9. Finally, compute an ARIMA model to improve your prediction error - iterate on the p, q, and parameters comparing the model's performance..