GENERAL ASSEMBLY

TIME SERIES MODELING

Trevor Lindsay

LEARNING OBJECTIVES

- Model and predict from time series data using AR, ARMA or ARIMA models
- Code 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 this class, we will focus on exploring time series data and common statistics for time series analysis
- We will then 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

WHAT IS TIME SERIES DATA?

WHAT IS TIME SERIES DATA?

- Time series data is any data where the individual data points change over time.
- This is fairly common in sales and other business cases where data would likely change according to seasons and trends.
- Time series data is also useful for studying social phenomena. For instance, there is statistically more crime in the summer, which is a seasonal trend.

WHAT IS TIME SERIES DATA?

- Most datasets are likely to have an important time component, but typically we assume that it's fairly minimal.
- For example, if we were analyzing salaries in an industry, it's clear that salaries shift over time and vary with the economic period.
- However, if we are examining the problem on a smaller scale (e.g. 3-5 years), the effect of time on salaries is much smaller than other factors, like industry or position.

WHAT IS TIME SERIES DATA?

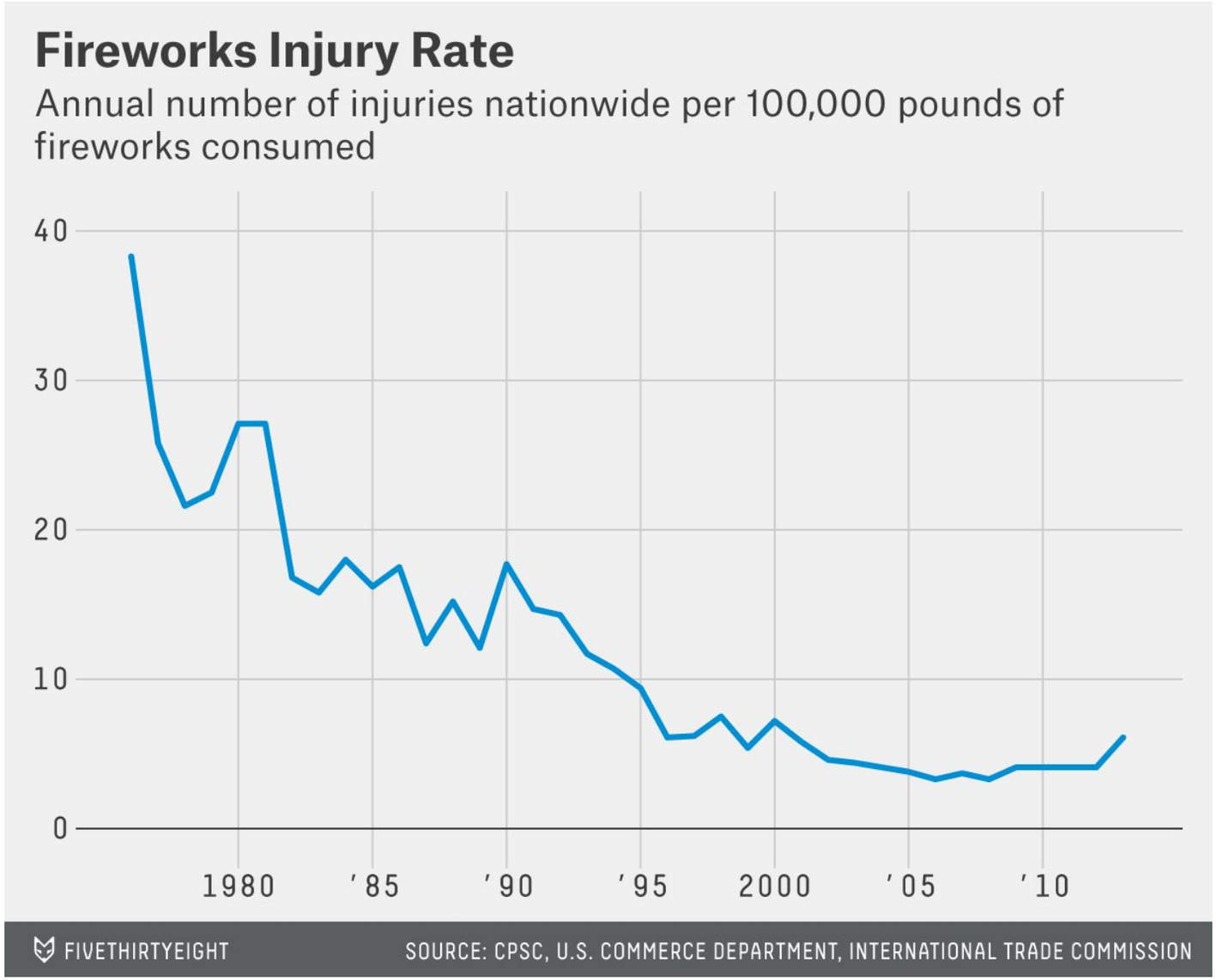
- When the time component is important, we need to focus on identifying the aspects of the data that are influenced by time and those that aren't.
- Typically, time series data will be a sequence of values. We will be interested in studying the changes to this series and how related individual values are.
- For example, how much does this week's sales affect next week's? How much does today's stock price affect tomorrow's?

WHAT IS TIME SERIES DATA?

- Typically, we are interested in separating the effects of time into two components:
  - Trends - significant increases or decreases over time
  - Seasonality - regularly repeating increases or decreases

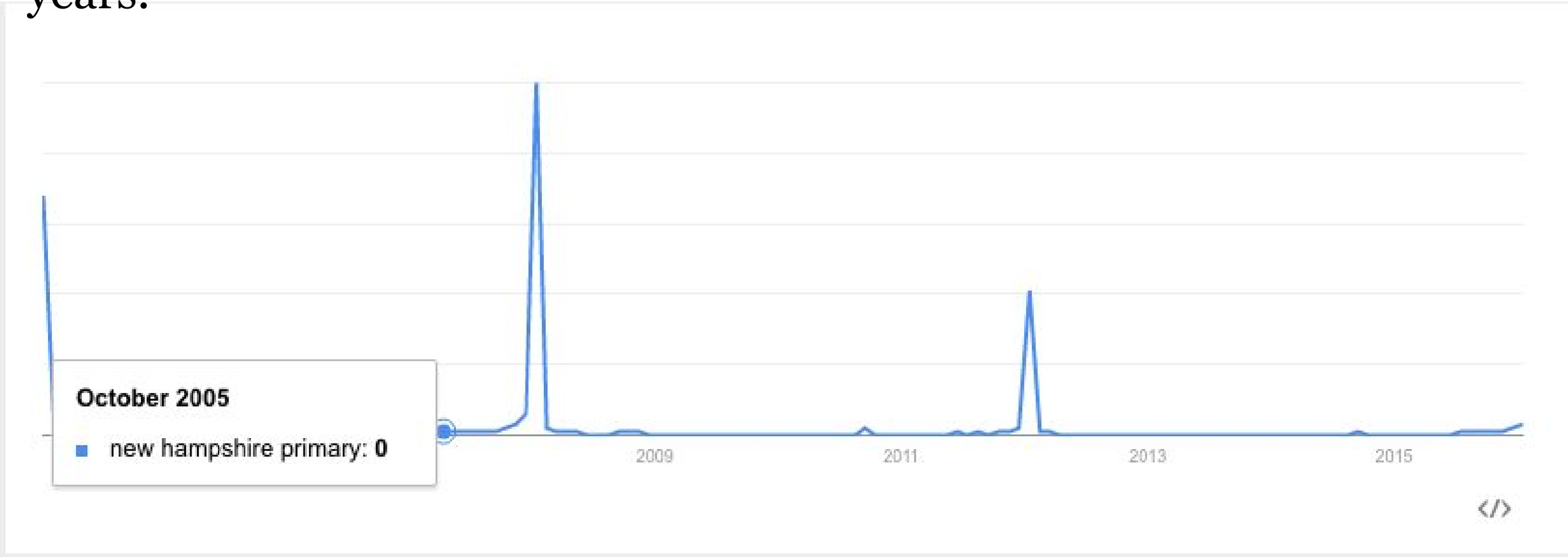
WHAT IS TIME SERIES DATA?

- This plot of fireworks injury rates has an overall *trend* of fewer injuries with no *seasonal* pattern.



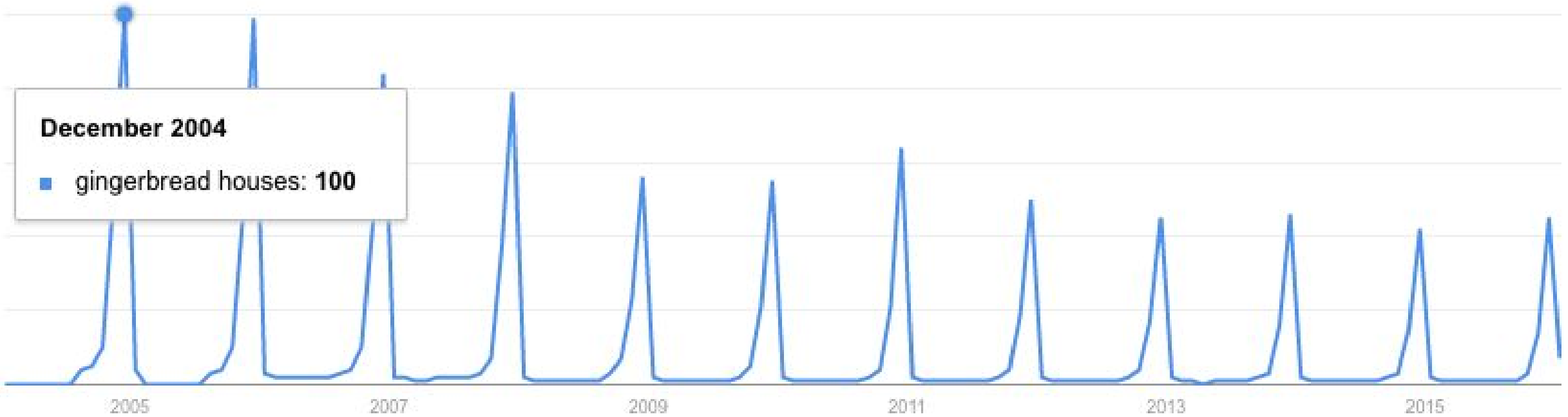
WHAT IS TIME SERIES DATA?

- Meanwhile, the number of searches for the New Hampshire Primary has a clear *seasonal* component - it peaks every four years and on election years.



WHAT IS TIME SERIES DATA?

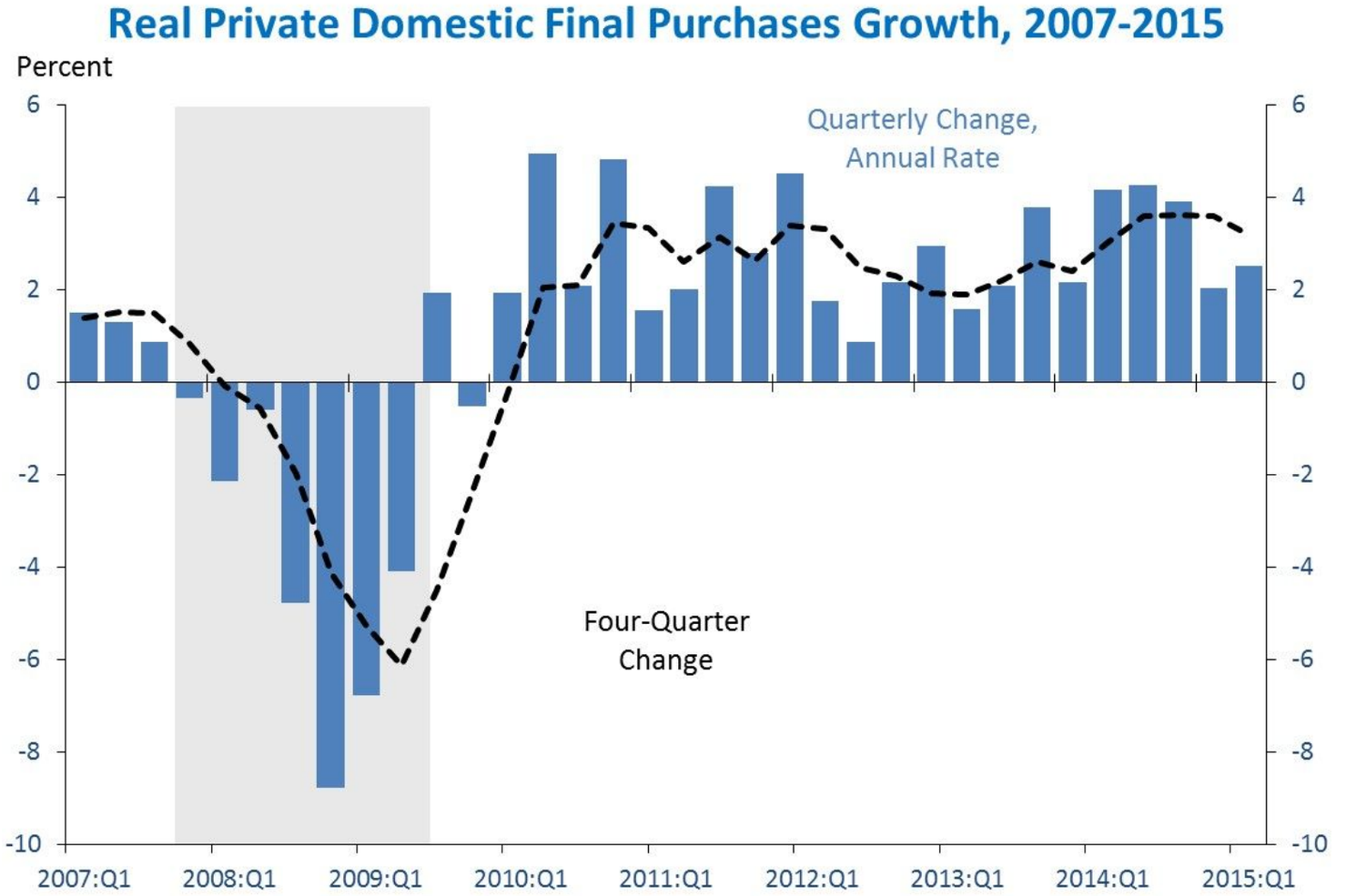
- Similarly, searches for ‘gingerbread houses’ spike every year around the holiday season.



- These spikes recur on a fixed time-scale, making them *seasonal* patterns.

WHAT IS TIME SERIES DATA?

- Many other types of regularly occurring up or down swings may occur without a fixed timescale or *period* (e.g. growth vs. recession for economic trends).

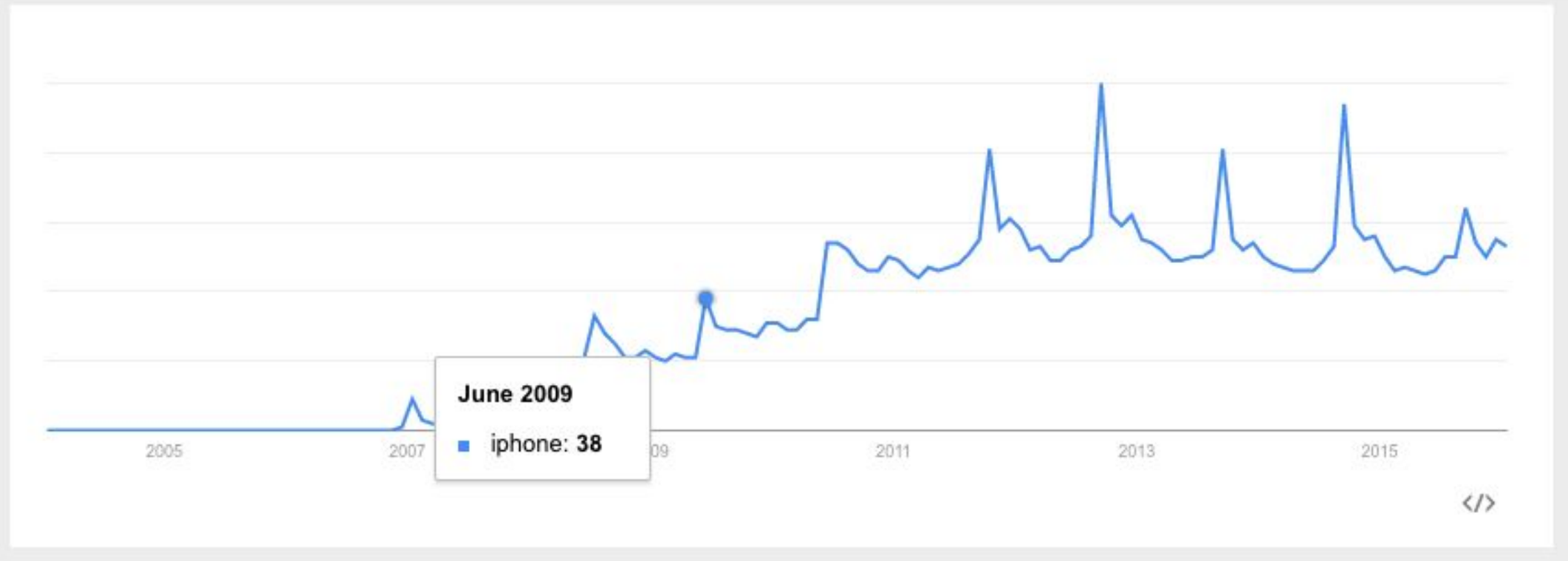


WHAT IS TIME SERIES DATA?

- These aperiodic patterns are called cycles.
- While identifying aperiodic cycles is important, they are often treated differently than seasonal effects. Seasonal effects are useful for their consistency, since prior data is useful as a predictor.

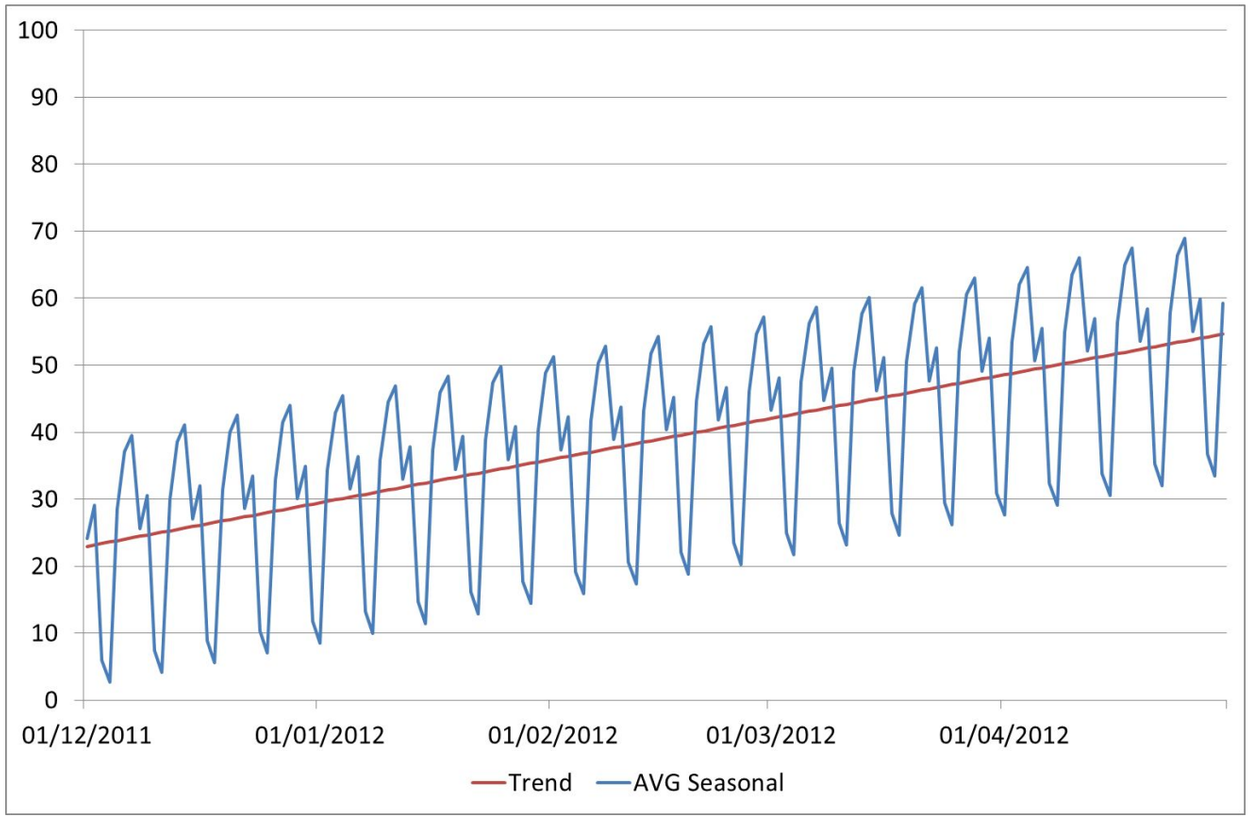
WHAT IS TIME SERIES DATA?

- Searches for “iphone” have both a general trend upwards (indicating more popularity for the phone) as well as a seasonal spike in September (which is when Apple typically announces new versions).



WHAT IS TIME SERIES DATA?

- Most often, we're interested in studying the *trend* and not the *seasonal* fluctuations.
- Therefore it is important to identify whether we think a change is due to an ongoing trend or seasonal change.



ACTIVITY: KNOWLEDGE CHECK

ANSWER THE FOLLOWING QUESTIONS

1. Discuss one or two more time series examples from Google Trends..
2. Identify relevant trends and seasonal patterns.

DELIVERABLE

Answers to the above questions

COMMON ANALYSIS FOR TIME SERIES DATA

MOVING AVERAGES

- A moving average replaces each data point with an average of k consecutive data points in time.
- Typically, this is k/2 data points prior to and following a given time point, but it could also be the k preceding points.
- These are often referred to as the “rolling” average.
- The measure of average could be mean or median.
- The formula for the rolling mean is:

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

AUTOCORRELATION

- In previous classes, we have been concerned with how two variables are correlated (e.g. height and weight, education and salary).
- Autocorrelation is how correlated a variable is with itself. Specifically, how related are variables earlier in time with variables later in time.

AUTOCORRELATION

- To compute autocorrelation, we fix a “lag” k. This is how many time points earlier we should use to compute the correlation.
- A lag of 1 computes how correlated a value is with the prior one. A lag of 10 computes how correlated a value is with one 10 time points earlier.

AUTOCORRELATION

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

DEMO

EXPLORING ROSSMANN DRUGSTORE SALES DATA

EXPLORING ROSSMAN DRUGSTORE SALES DATA

```
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)

df['Year'] = df.index.year
df['Month'] = df.index.month
```

ACTIVITY: KNOWLEDGE CHECK

COMPLETE THE FOLLOWING TASKs

1. There are over a million sales data points in this dataset, so for some analysis we will focus on just one store. Create a new dataframe with only data for Store 1 and assign it to the variable **store1**
2. Plot the sales over time. This requires filtering the dataframe to only include dates that Store 1 was open.

DELIVERABLE

New dataframe with only data for Store 1 and plot of sales over time

COMPUTING AUTOCORRELATION

- To measure how much the sales are correlated with each other, we want to compute the *autocorrelation* of the "Sales" column.

- In Pandas, we do this with the `autocorr` function. `autocorr` takes one argument, `lag`. This is how many points prior should be used to compute the correlation.

```
df['Sales'].resample('D', how='mean').autocorr(lag=1)
```

- As with correlation between different variables, as this number moves closer to 1, the data is more correlated.

AGGREGATES OF SALES OVER TIME

- If we want to investigate trends over time in sales, we will start by computing simple aggregations. What were the mean and median sales in each year and each month?

- In Pandas, this is performed using the `resample` function, which is very similar to the `groupby` function. It allows us to group over different time periods.

AGGREGATES OF SALES OVER TIME

- We can use `df.resample` and provide the following arguments.
  - A level on which to roll up to: ‘D’ for day, ‘W’ for week, ‘M’ for month, ‘A’ for year.
  - The aggregation to perform: ‘mean’, ‘median’, ‘sum’, etc.

AGGREGATES OF SALES OVER TIME

```
df[['Sales']].resample('A', how=['median', 'mean'])
```

```
df[['Sales']].resample('M', how=['median', 'mean'])
```

- Here we see that December 2013 and 2014 were the highest average sales months.

AGGREGATES OF SALES OVER TIME

- While identifying the monthly averages are useful, we often want to compare the sales data of a date to a smaller window.
- To understand holidays' sales, we want to compare the sales data of late December to a few days surrounding it.
- We can do this using rolling averages.

AGGREGATES OF SALES OVER TIME

- In Pandas, we can compute the rolling average using the `pd.rolling_mean` or `pd.rolling_median` functions.

```
pd.rolling_mean(df[['Sales']], window=3, center=True, freq='D')
```

- This computes a rolling mean of sales using the sales on each day, the day preceding, and the day following (`window=3` and `center=True`).

AGGREGATES OF SALES OVER TIME

- `rolling_mean` (as well as `rolling_median`) takes the series to aggregate and three important parameters:
  - `window` - the number of days to include in the average
  - `center` - whether the window should be centered on the date or use data prior to that date
  - `freq` - what level to roll up the averages to (as used in resample); `'D'` for day, `'W'` for week, `'M'` for month, `'A'` for year

AGGREGATES OF SALES OVER TIME

- We can use our index filtering to just look at 2015:

```
pd.rolling_mean(df[['Sales']], window=3, center=True, freq='D')['2015']
```

- Instead of plotting the full time series, we can plot the rolling mean instead. This smooths random changes in sales as well as removing outliers, helping us identify larger trends.

```
pd.rolling_mean(df[['Sales']], window=10, center=True, freq='D').plot()
```

PANDAS WINDOW FUNCTIONS

- Pandas `rolling_mean` and `rolling_median` are only two examples of Pandas window function capabilities.
- Window functions operate on a set of N consecutive rows (a window) and produce output.
- In addition, there are `rolling_sum`, `rolling_min`, `rolling_max`, and many more.

PANDAS WINDOW FUNCTIONS

- Another common window function is `diff`, which takes the difference over time.

- `pd.diff` takes one argument, `periods`, which is how many rows prior to use for the difference.

- For example, if we want to compute the difference in sales, day by day:

```
df['Sales'].diff(periods=1)
```

PANDAS WINDOW FUNCTIONS

- However, if we wanted to compare the same day in the prior week, we could set `periods=7`.

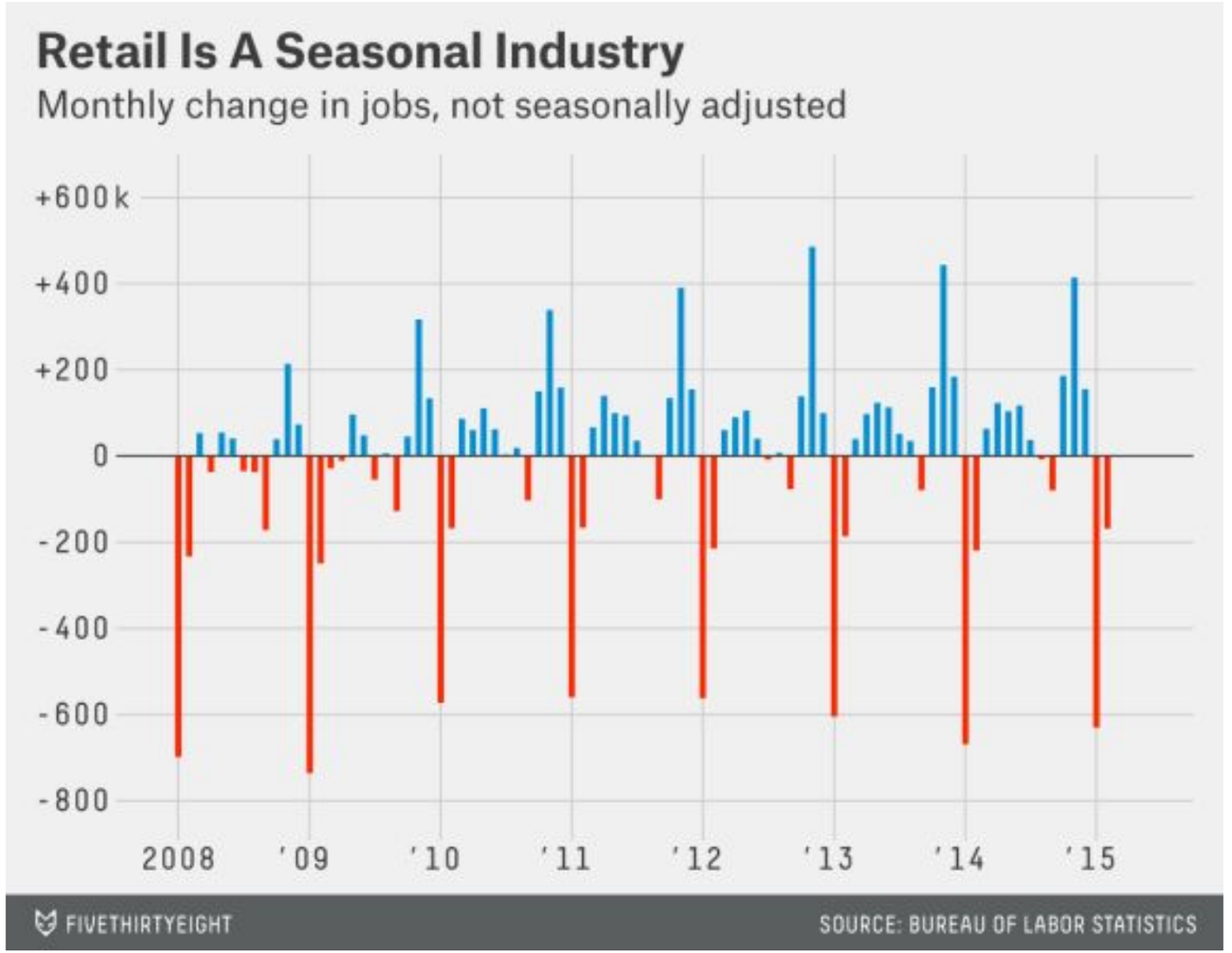
```
df['Sales'].diff(periods=7)
```

- This would compute the difference in sales, from every day to the same day in the previous week.

WHAT IS TIME SERIES DATA?

- Difference functions allow us to identify seasonal changes as we see repeated up or down swings.

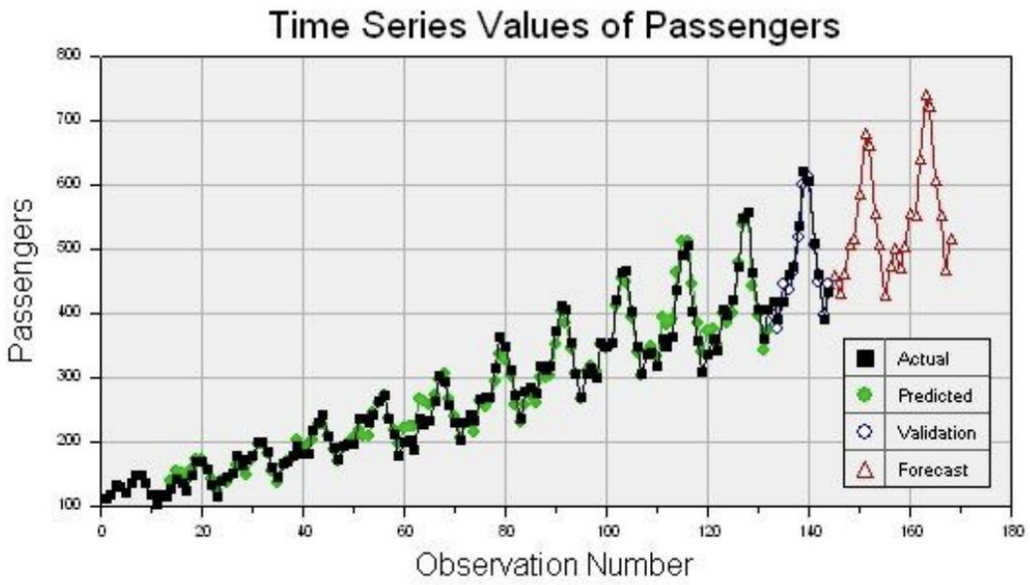
- This plot of the month to month change (`diff`) in retail jobs helps identify the seasonal component of the number of retail jobs.



WHAT ARE TIME SERIES MODELS?

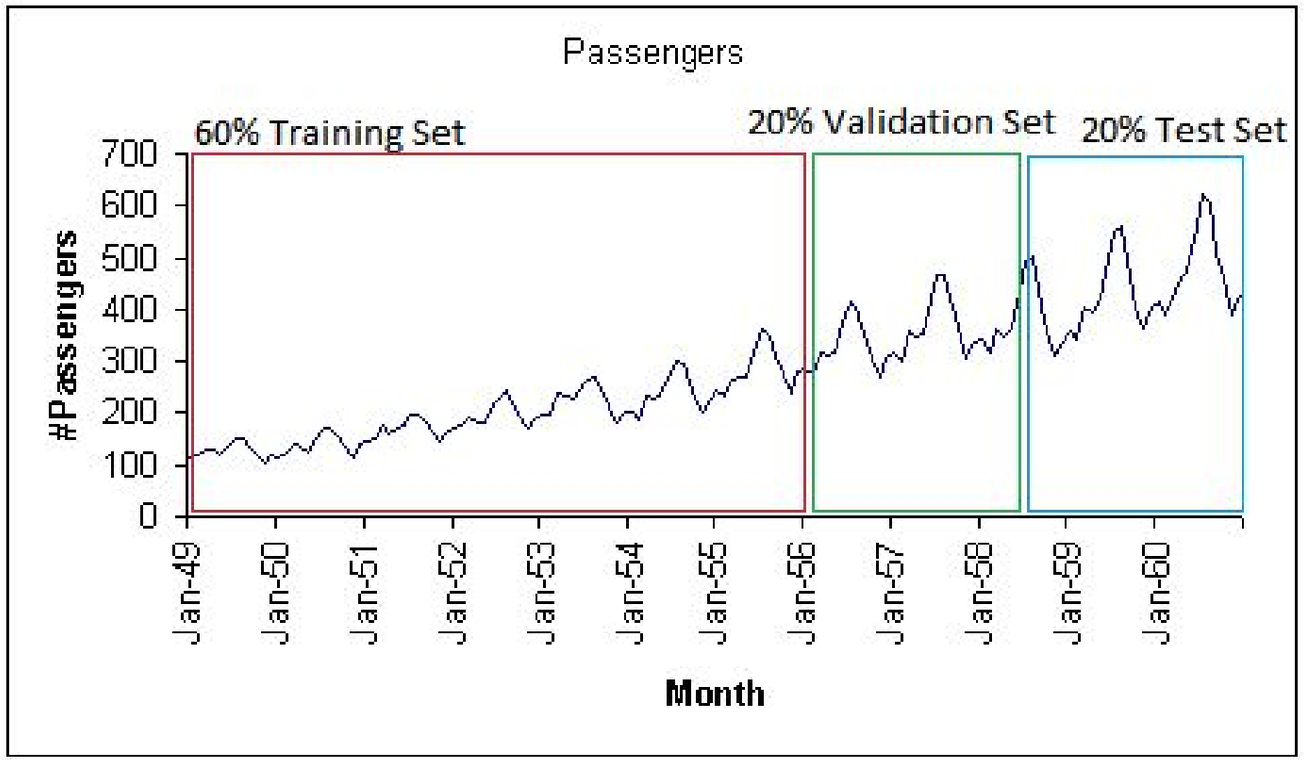
WHAT ARE TIME SERIES MODELS?

- 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.



WHAT ARE TIME SERIES MODELS?

- We will exclusively train on values earlier (in time) in our data and test our model on values at the end of the data period.



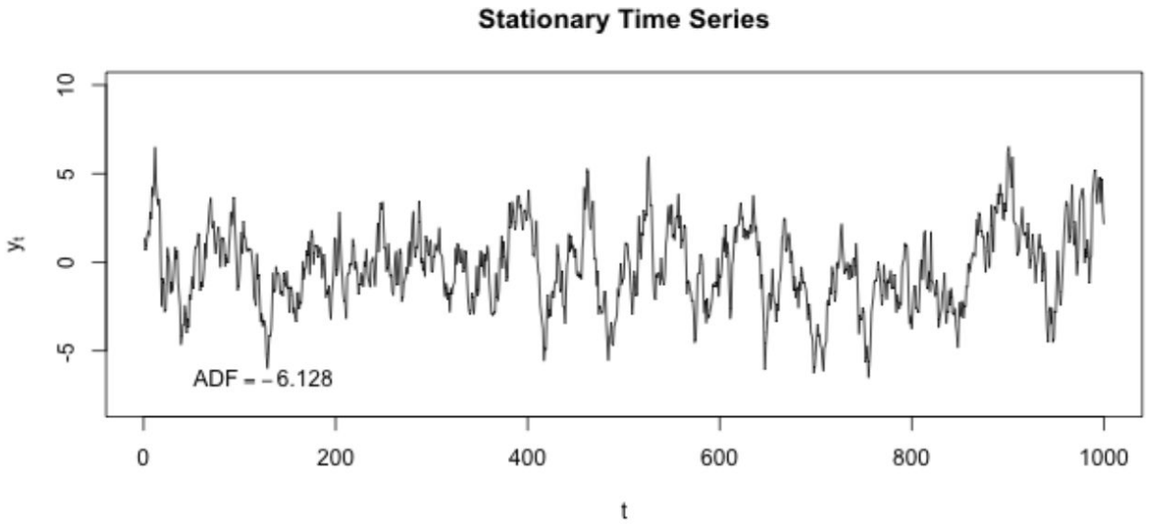
PROPERTIES FOR A TIME SERIES MODEL

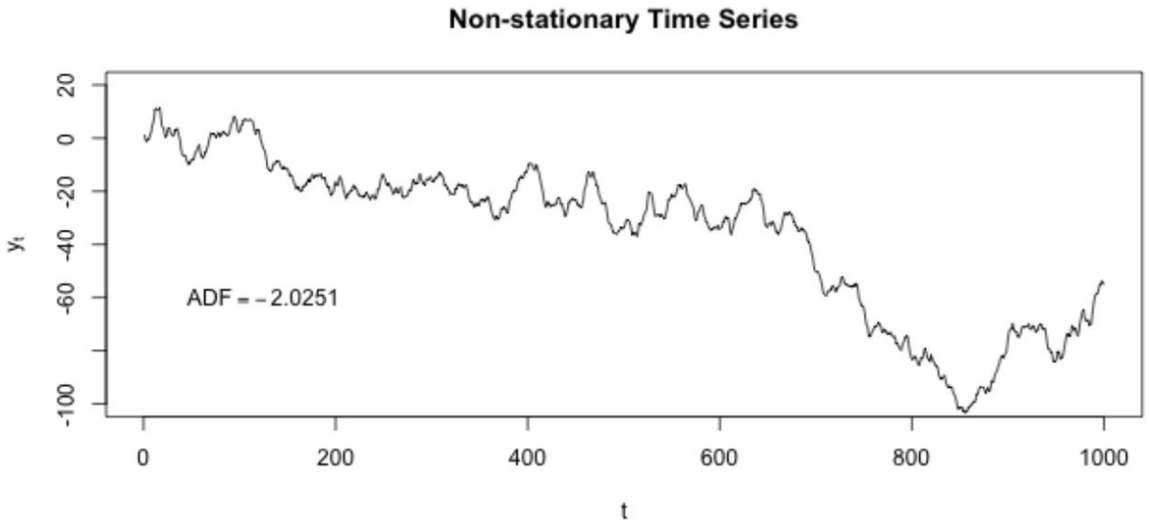
- We can use these values to assess how we plan to model our time series.
- Typically, for a high quality model, we require some autocorrelation in our data.
- We can compute autocorrelation at various lag values to determine how far back in time we need to go.

PROPERTIES FOR A TIME SERIES MODEL

- 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 A TIME SERIES MODEL

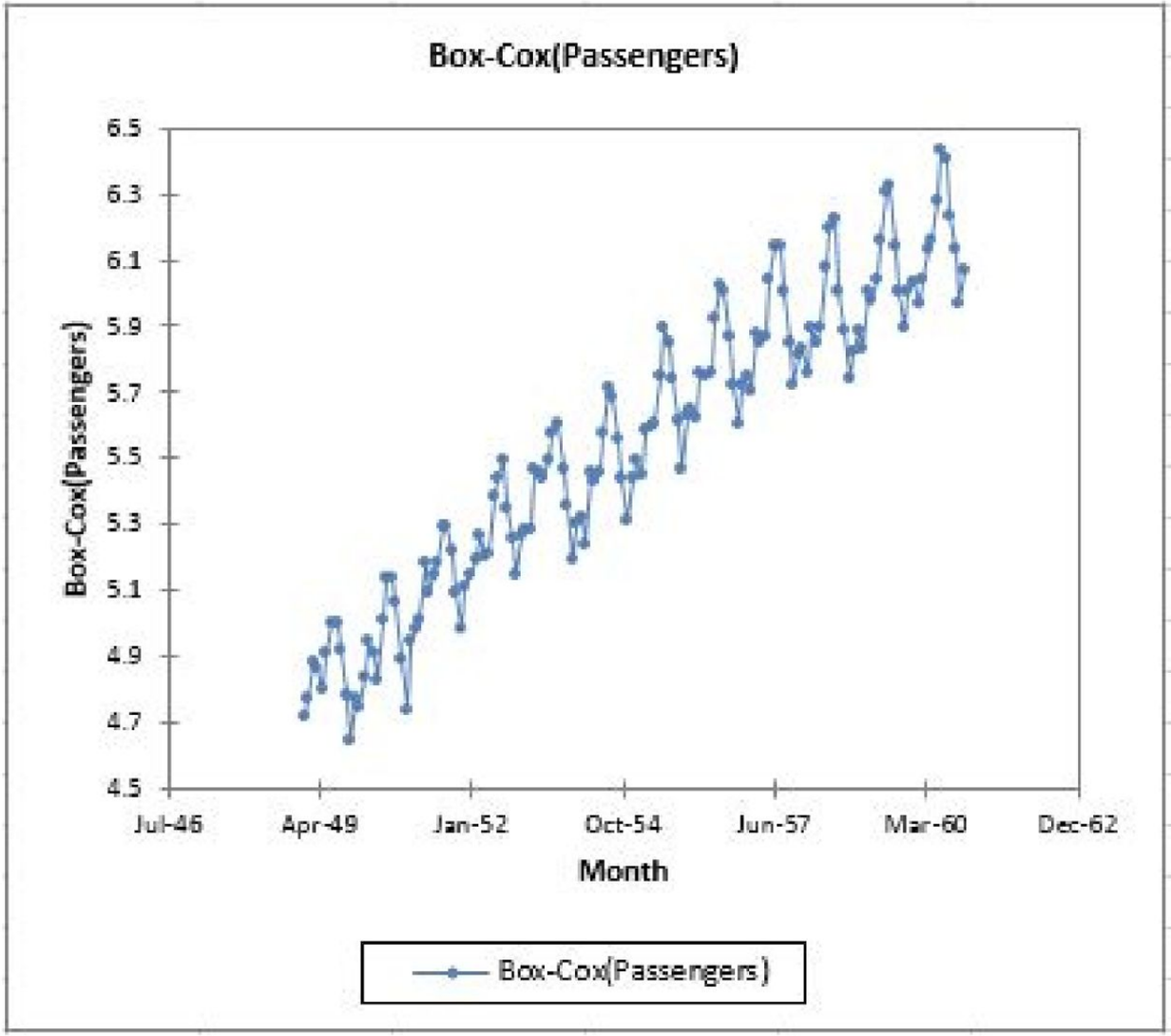


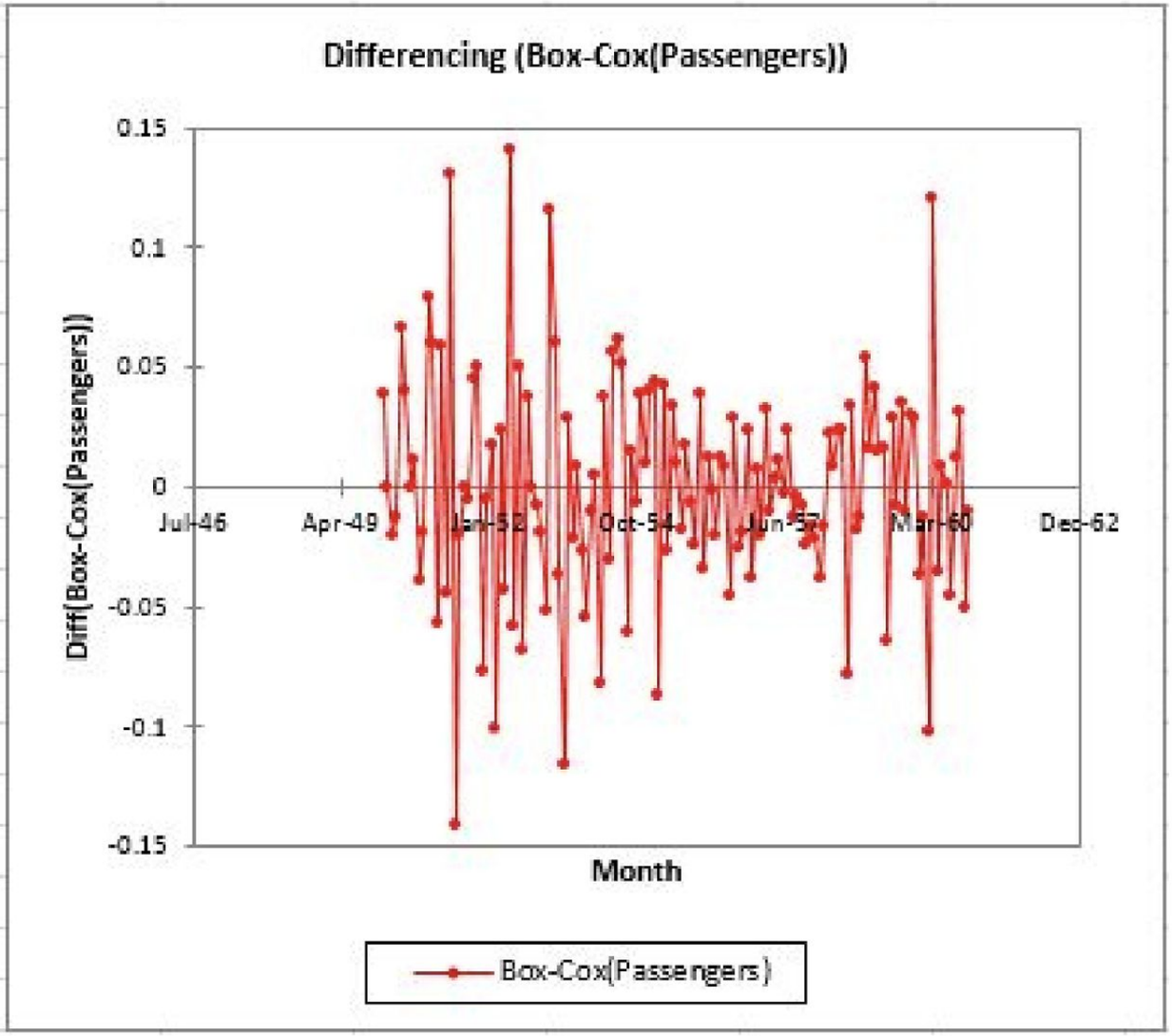


PROPERTIES FOR A TIME SERIES MODEL

- Often, if these assumptions don't hold, we can alter our data to make them true. Two common methods are detrending and differencing.
- The simplest method is differencing. This is very closely related to the diff function we saw earlier
- Instead of predicting the series (again our non-stationary series), we can predict the difference between two consecutive values.

PROPERTIES FOR A TIME SERIES MODEL





PROPERTIES FOR A TIME SERIES MODEL

- 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

DEEP DIVE

AUTOREGRESSIVE 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

- 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

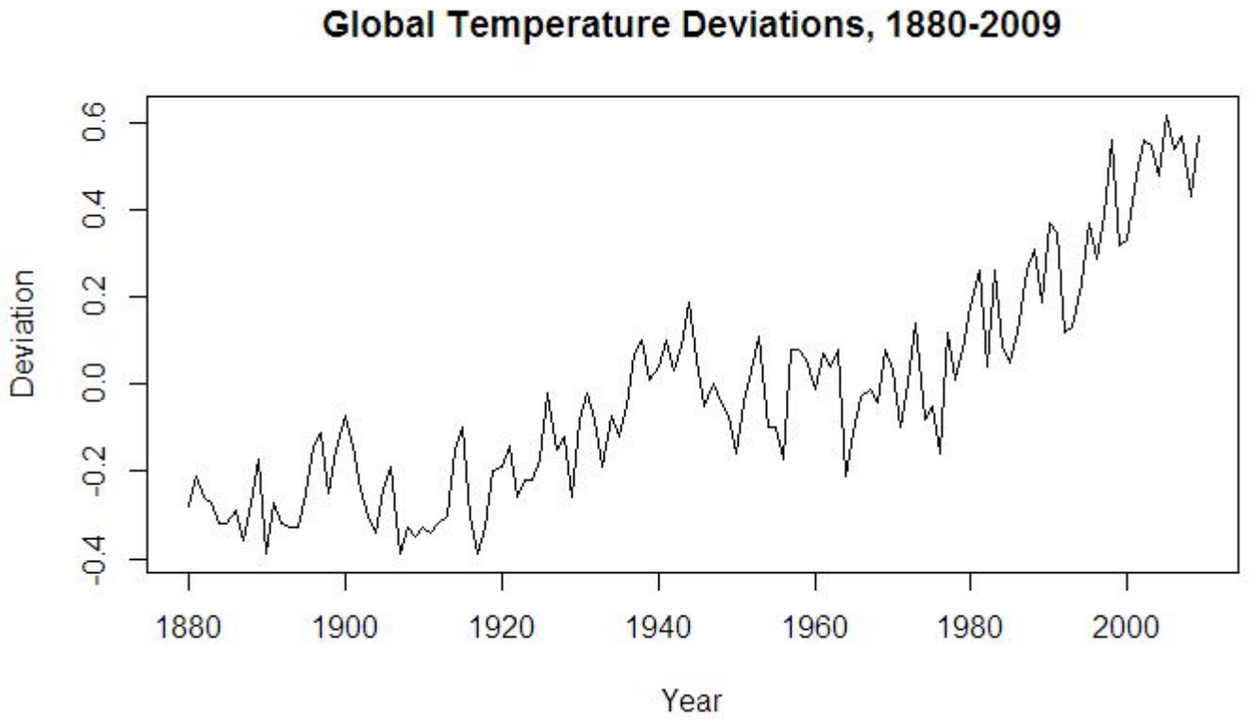
- For an AR(1) model, we will learn a single coefficient.

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

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

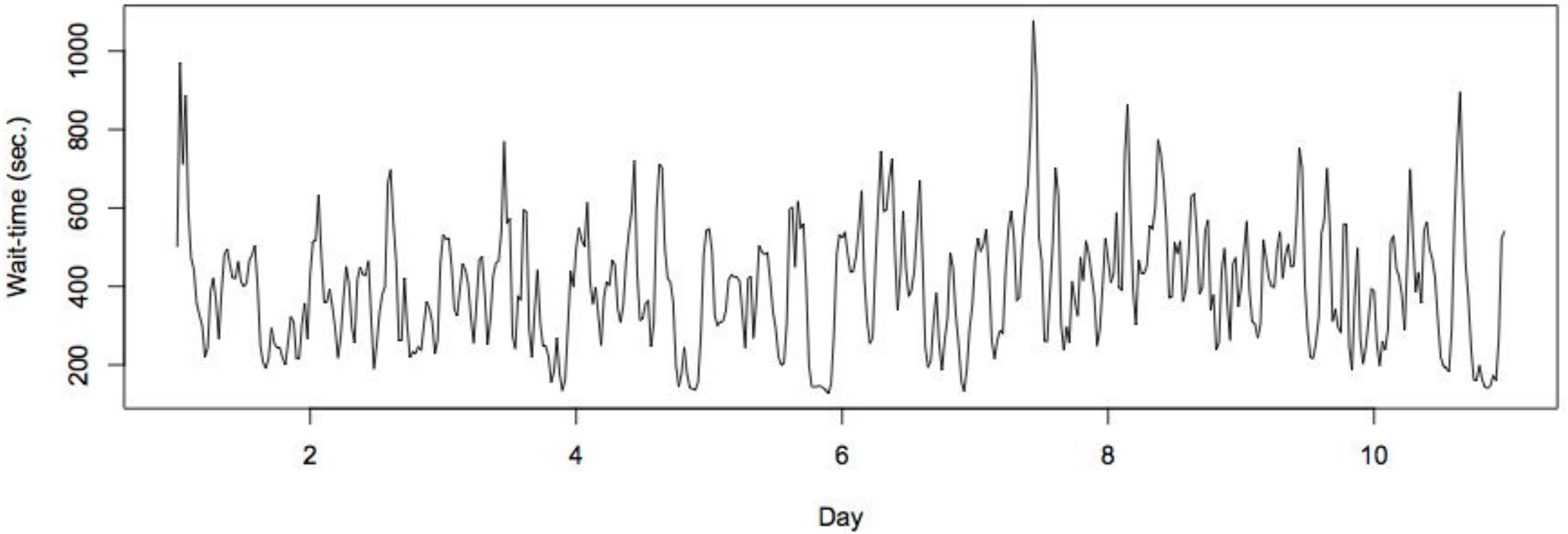
AR MODELS

- 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.

DEEP DIVE

MOVING AVERAGE MODELS

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.

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.

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

- 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.

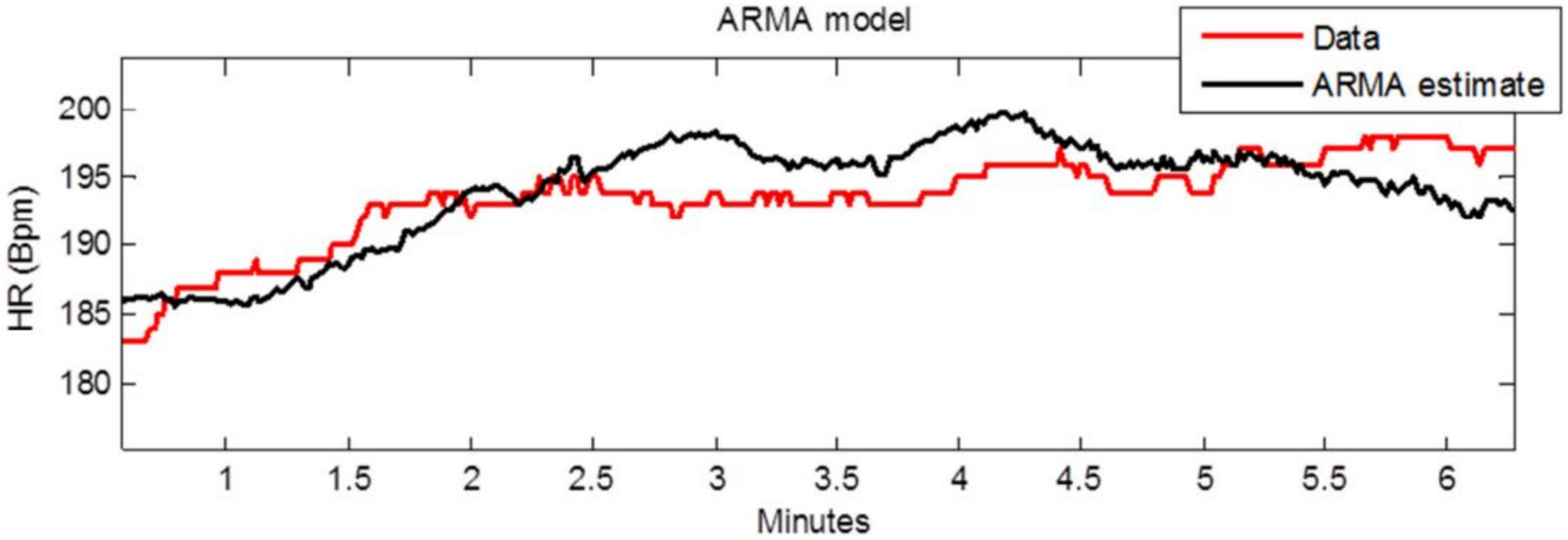
DEEP DIVE

AUTOREGRESSIVE MOVING AVERAGE MODELS

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.

ARIMA MODELS

- **ARIMA** (pronounced ‘uh-ri-mah’) is an **A**uto**R**egressive **I**ntegrated **M**oving **A**verage model.

- In this model, we learn an ARMA(p,q) model to predict *the difference* of the series (as opposed to the value of the series).

ARIMA MODELS

- Recall the pandas `diff` function. This computes the difference between two consecutive values.

- In an ARIMA model, we attempt to predict this difference instead of the actual values.

$$y_t - y_{t-1} = \text{ARIMA}(p,q)$$

- This handles the stationarity assumption we wanted for our data. Instead of detrending or differencing manually, the model does this.

ARIMA MODELS

- An ARIMA model has three parameters and is specified ARIMA(p, d, q).
  - *p* is the order of the autoregressive component
  - *q* is the order of the moving average component
  - *d* is the degree of differencing.
- *d* was 1 in our prior example. For d=2, our model would be

$$\text{diff}(\text{diff}(y)) = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) = \text{ARIMA}(p,q)$$

ARIMA MODELS

- Compared to an ARMA model, ARIMA models do **not** rely on the underlying series being stationary.
- The differencing operation can *convert* the series to one that is stationary.
- Instead of attempting to predict values over time, our new series is the difference in values over time.
- Since ARIMA models include differencing, they can be used on a broader set of data without the assumption of a constant mean.

DEMO

TIME SERIES MODELING IN STATSMODELS

TIME SERIES MODELING IN STATSMODELS

- To explore time series models, we will continue to 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.

ACTIVITY: KNOWLEDGE CHECK

ANSWER THE FOLLOWING QUESTIONS

1. Compute the autocorrelation of Sales in Store 1 for lag 1 and 2.
2. Will we be able to use a predictive model, particularly an autoregressive one?

DELIVERABLE

Answers to the above questions

TIME SERIES MODELING IN STATSMODELS

```
store1['Sales'].autocorr(lag=1) # -0.12
store1['Sales'].autocorr(lag=2) # -0.03
```

- We do see some minimal correlation in time, implying an AR model can be useful. An easier way to diagnose this may be to plot many autocorrelations at once.

TIME SERIES MODELING IN STATSMODELS

```
%matplotlib inline
from pandas.tools.plotting import autocorrelation_plot

autocorrelation_plot(store1['Sales'])
```

- This shows a typical pattern of an autocorrelation plot, that it should decrease to 0 as lag increases. However, it's hard to observe exactly what the values are.

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.

TIME SERIES MODELING IN STATSMODELS

- statsmodels also has a better autocorrelation plot that allows us to look at fixed number of lag values.

```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(store1['Sales'], lags=10)
```

- Here we observe autocorrelation at 10 lag values. 1 and 2 are what we saw before.

- This implies a small but limited impact based on the last few values. An autoregressive model might be useful.

TIME SERIES MODELING IN STATSMODELS

- We also see a larger spike at 7 (the seventh day in the week).
- If we observed a handful of random distributed spikes, a moving average model would be useful.

```
plot_acf(store1['Sales'], lags=25)
```

- We can expand the window to 25 days to see that the random spikes occur regularly at 7 days. What does this mean?

AR, MA, AND ARMA MODELS IN STATSMODELS

- To explore AR, MA, and ARMA models, we will use `sm.tsa.ARMA`.
- Remember, an ARMA model is a combination of autoregressive and moving average models.
- We can train an AR model by turning off the MA component (q=0).

```
from statsmodels.tsa.arima_model import ARMA

store1 = store1[['Sales']].astype(float)
model = ARMA(store1, (1, 0)).fit()
model.summary()
```

AR, MA, AND ARMA MODELS IN STATSMODELS

- By passing `(1, 0)` in the second argument, we are fitting an ARMA model with p=1, q=0. This is the same as an AR(1) model.
- In this AR(1) model, we learn an intercept (or base sales) value.
- Additionally, we learn a coefficient that tells us how to include the latest sales value.
- In this case, we add an intercept of ~4700 to 0.68 times the previous month's sales. Note that the coefficient is not equal to the lag 1 autocorrelation. This implies the data is **not** stationary.

AR, MA, AND ARMA MODELS IN STATSMODELS

- We can learn an AR(2) model, which regresses each sales value on the last two.

```
model = ARMA(store1, (2, 0)).fit()
model.summary()
```

- In this case, we learn two coefficients, which tell us the effect of the last two sales values on the current sales.

- While this model may perform better, it may be more difficult to interpret.

AR, MA, AND ARMA MODELS IN STATSMODELS

- Residuals are the errors of the model or how off our predictions are.
- Ideally, we want randomly distributed errors that are small.
- If the errors are large, our model does not perform well.
- If the errors have a pattern, particularly over time, we may have overlooked something in the model or have periods of time that are different than the rest of the dataset.

AR, MA, AND ARMA MODELS IN STATSMODELS

- We can use `statsmodels` to plot the residuals.

```
model.resid.plot()
```

- Here we see large spikes at the end of each year, indicating that our model does not account for the holiday spikes.

- Our model considers a short period of time, so it does not take into account the longer seasonal pattern.

AR, MA, AND ARMA MODELS IN STATSMODELS

- We can also plot the autocorrelations of the residuals. In an ideal world, these would all be near 0 and appear random.

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

- This plot shows a problem: the errors are increasing and decreasing every week in a clear pattern.

- We may need to expand our model.

AR, MA, AND ARMA MODELS IN STATSMODELS

- To expand this AR model to an ARMA model, we can include the moving average component as well.

```
model = ARMA(store1, (1, 1)).fit()
model.summary()
```

- Now we learn two coefficients, one for the AR(1) component and one for the MA(1) component.

AR, MA, AND ARMA MODELS IN STATSMODELS

- Remember that this is an AR(1) + MA(1) model. The AR coefficient represents dependency on the last value and the MA component represents any spikes independent of the last value.
- The coefficients here are 0.69 for the AR component and -0.03 for the MA component.
- The AR coefficient is the same as before (decreasing values).
- The MA component is fairly small (which we should have expected from the autocorrelation plots).

ARIMA MODELS IN STATSMODELS

- We can also use `statsmodels` to fit ARIMA models. Let's start by using ARIMA(1, 0, 1) to fit an ARMA(1, 1) model.

```
from statsmodels.tsa.arima_model import ARIMA

model = ARIMA(store1, (1, 0, 1)).fit()
model.summary()
```

- We can see that this model is the same as our previous ARMA model.

ARIMA MODELS IN STATSMODELS

- We can also fit a true ARIMA model to predict the difference of the series.

```
model = ARIMA(store1, (1, 1, 1)).fit()
model.summary()
```

- We can remove the MA component since it does not appear to be useful.

```
model = ARIMA(store1, (1, 1, 0)).fit()
model.summary()
```

- We now have an AR(1) model on the differenced series with a coefficient of -0.18.

ARIMA MODELS IN STATSMODELS

- With our models, we can also plot our predictions against the true series using the `plot_predict` function.

- We can compare the last 50 days of true values against our predictions.

```
model.plot_predict(0, 50)
```

- The function takes two arguments, the start and end index of the dataframe to plot. Here, we are plotting the last 50 values.

ARIMA MODELS IN STATSMODELS

- To plot earlier values with our predictions continuing where the true values stop, we can do the following.

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax = store1['2014'].plot(ax=ax)

fig = model.plot_predict(0, 200, ax=ax, plot_insample=False)
```

- This plots true values in 2014 and our predictions 200 days out from 2014.

ARIMA MODELS IN STATSMODELS

- The two previous problems remain:
  - Large errors around the holiday period
  - Errors with high autocorrelation

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, (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.

ARIMA MODELS IN STATSMODELS

- There are variants of ARIMA that will better handle the seasonal aspect of our data. This is referred to as Seasonal ARIMA.

- These models fit two ARIMA models, one on the current frequency (daily in our example) and another on the seasonal frequency (maybe monthly or yearly patterns).

- Additionally, issues with seasonality could be handled by preprocessing tricks such as detrending.

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

- ARIMA models train ARMA models on differenced data to account for non-stationary data.
- 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.