Time Series Modeling

Shouvik Mani

April 5, 2018

Goals

After this lecture, you will be able to:

- Explain key properties of time series data
- Describe, measure, and remove **trend** and **seasonality** from a time series
- Understand the concept of **stationarity**
- Create and interpret **autocorrelation function** (acf) plots
- Understand **ARIMA** models for forecasting
- Create your own time series forecast

Outline

Properties of time series data

Applications and examples

Descriptive methods for understanding a time series

Forecasting

Outline

Properties of time series data

Applications and examples

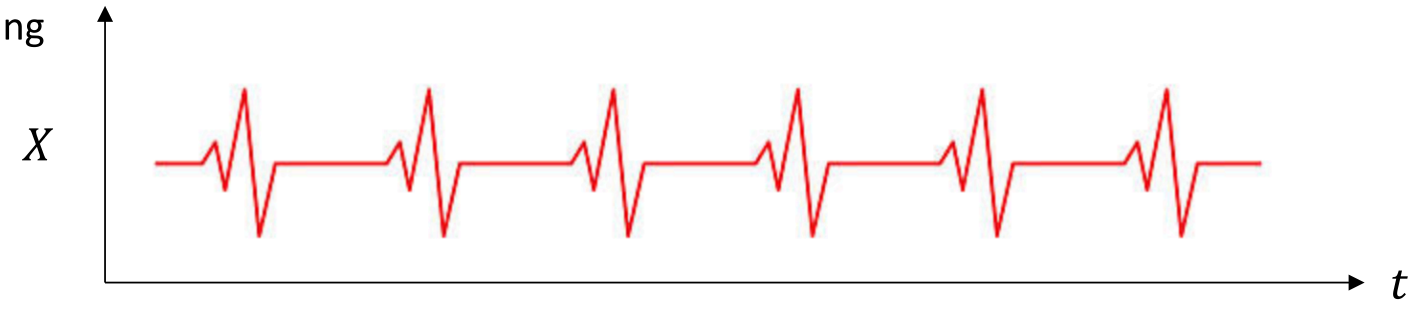
Descriptive methods for understanding a time series

Forecasting

What is a time series?

A time series is a sequence of observations over time.

ECG graph measuring heart activity

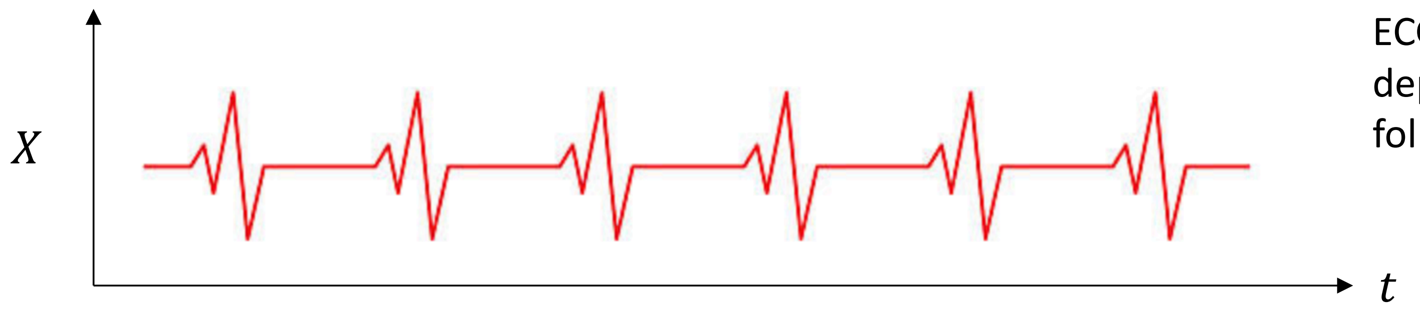


Notation: We have observations $X_1, \ldots, X_n$, where X_t denotes the observation at time t

In this lecture, we will consider time series with observations at equally-spaced times (not always the case, e.g. point processes).

Dependent Observations

Each observation in a time series is **dependent** on all other observations.



ECG graph has clear dependence: peaks followed by valleys

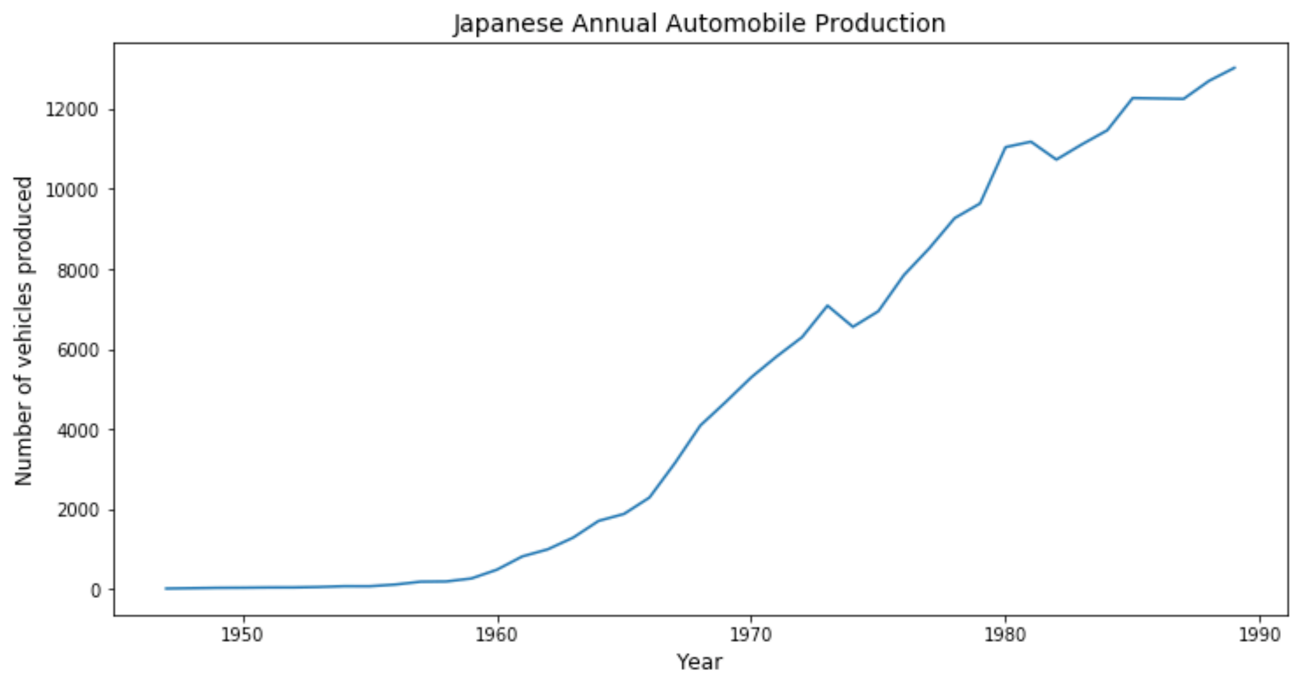
Why is this important? Most statistical models assume that individual observations are independent. But this assumption does not hold for time series data.

Analysis of time series data must take into account the *time order* of the data.

Trend and Seasonality

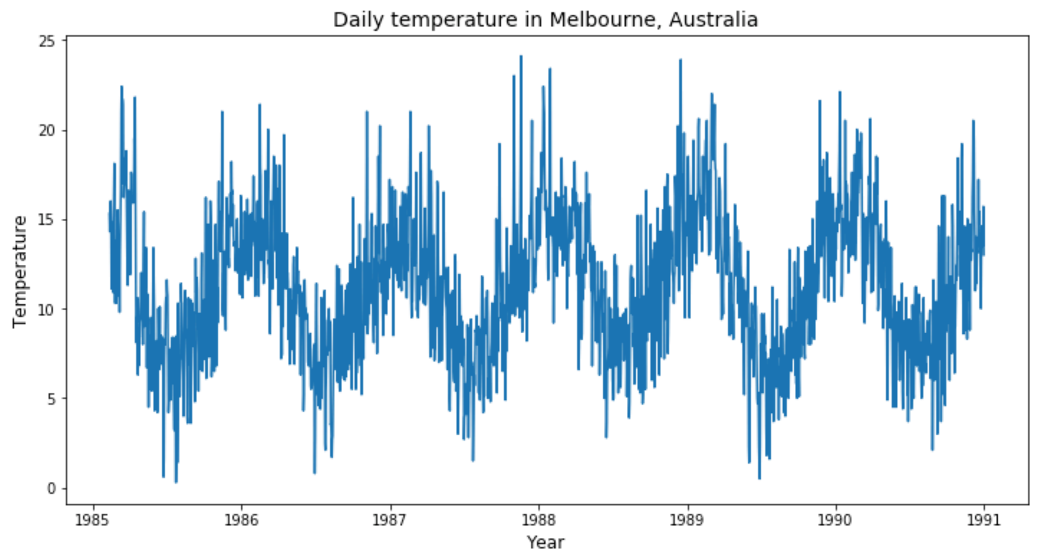
Many time series display trends and seasonal effects.

A **trend** is a change in the long term mean of the series.



Trend and Seasonality

A **seasonal effect** is a cyclic pattern of a *fixed period* present in the series.

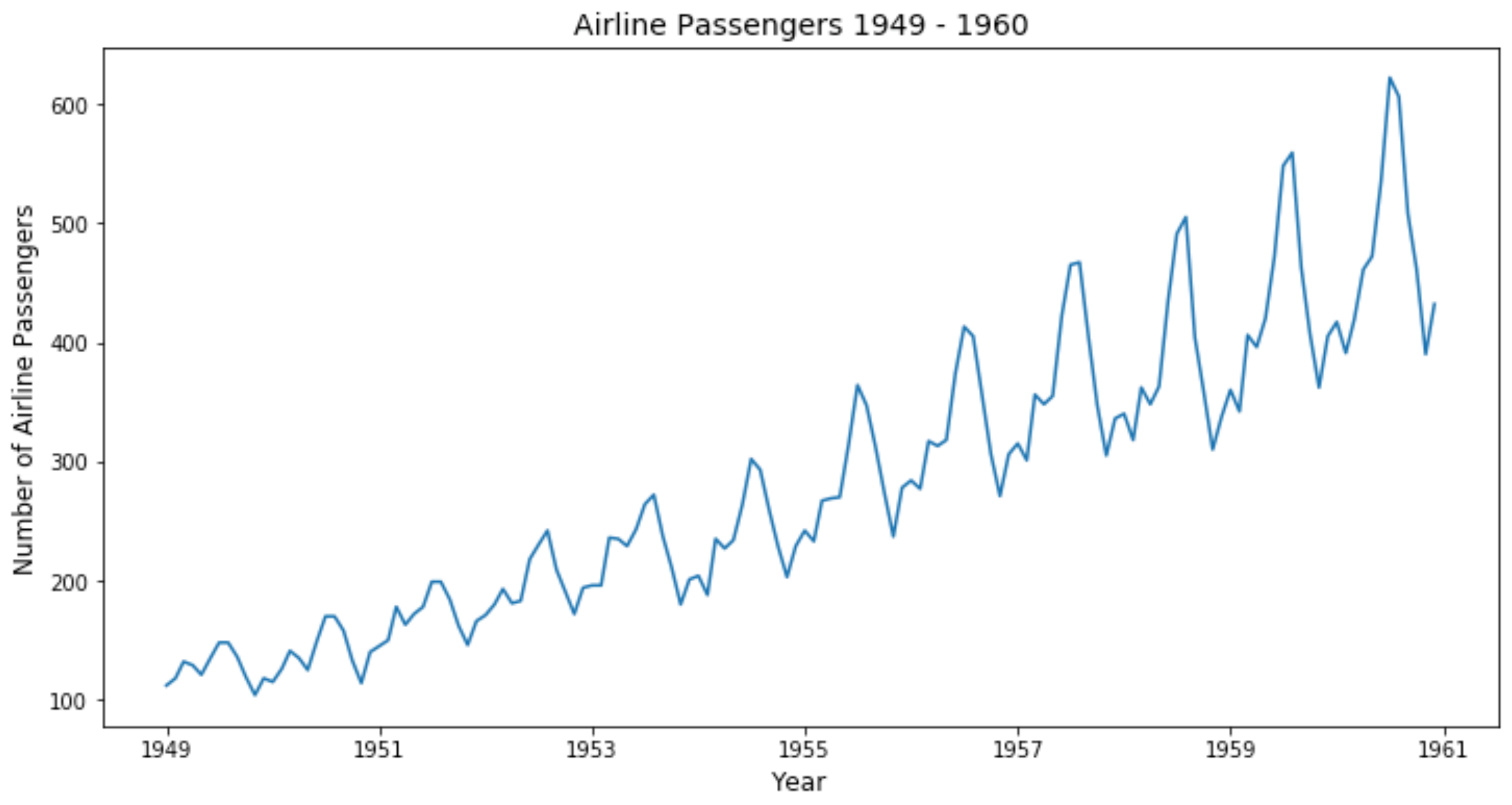


The season (or period) is the length of the cycle (e.g. an annual season).

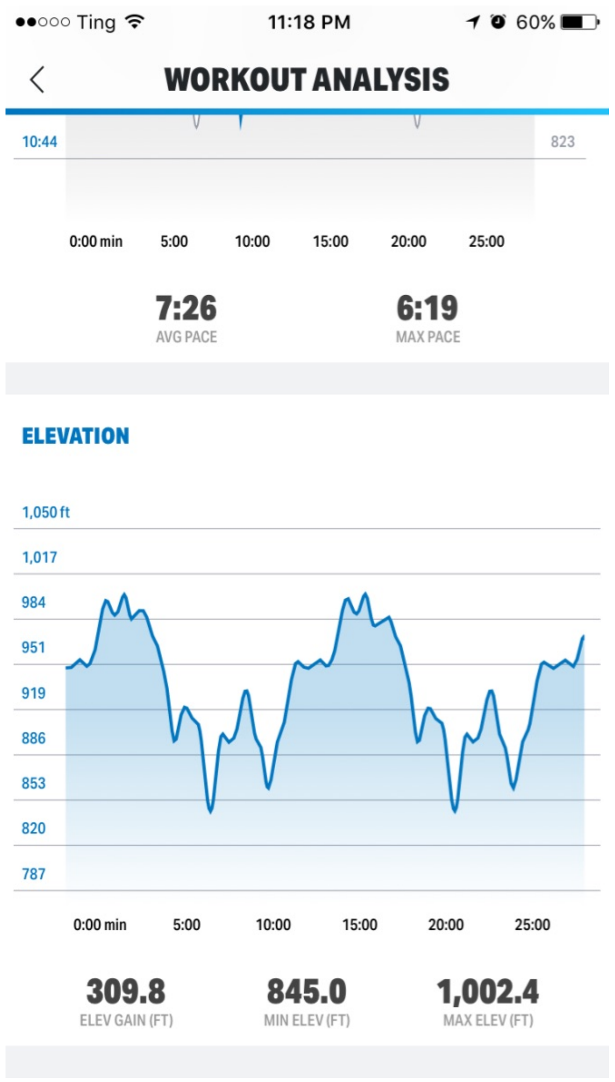
Seasonal effect can be additive (constant over time) or multiplicative (increasing over time).

Trend and Seasonality

A series can have both a trend and a seasonal effect.



Trend and Seasonality



7:26 AVG PACE

6:19 MAX PACE

309.8 ELEV GAIN (FT)

845.0 MIN ELEV (FT)

1,002.4 MAX ELEV (FT)

A fun example: seasonal patterns are quite common.

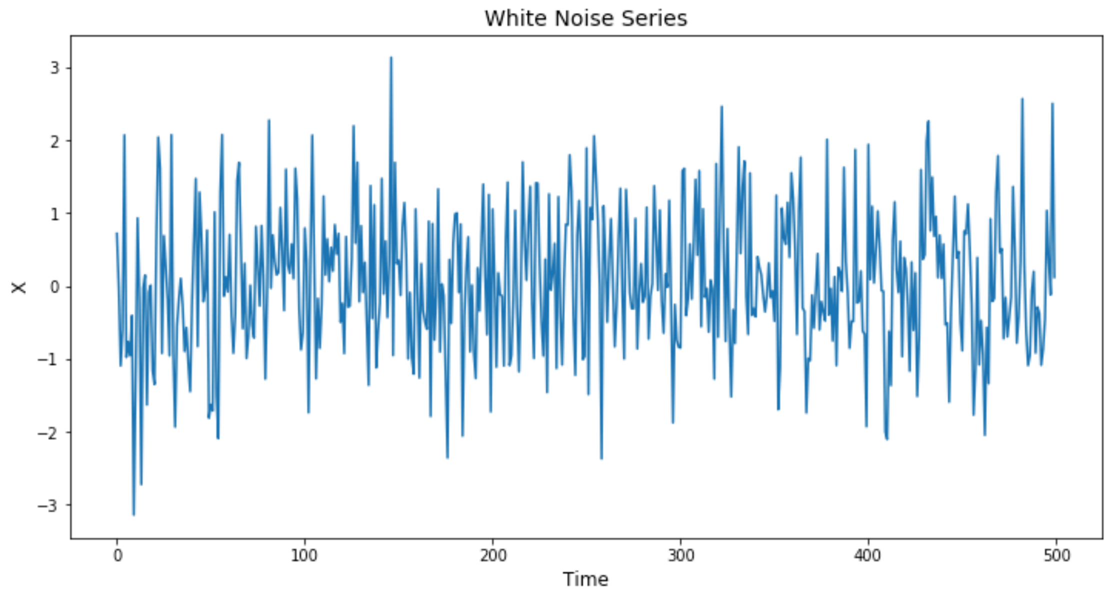
My elevation while running around Schenley Park seems to have a seasonal effect!

(Makes sense because running the same loop repeatedly).

Stationarity

A time series is called **stationary** if one section of the data looks like any other section of the data, in terms of its distribution.

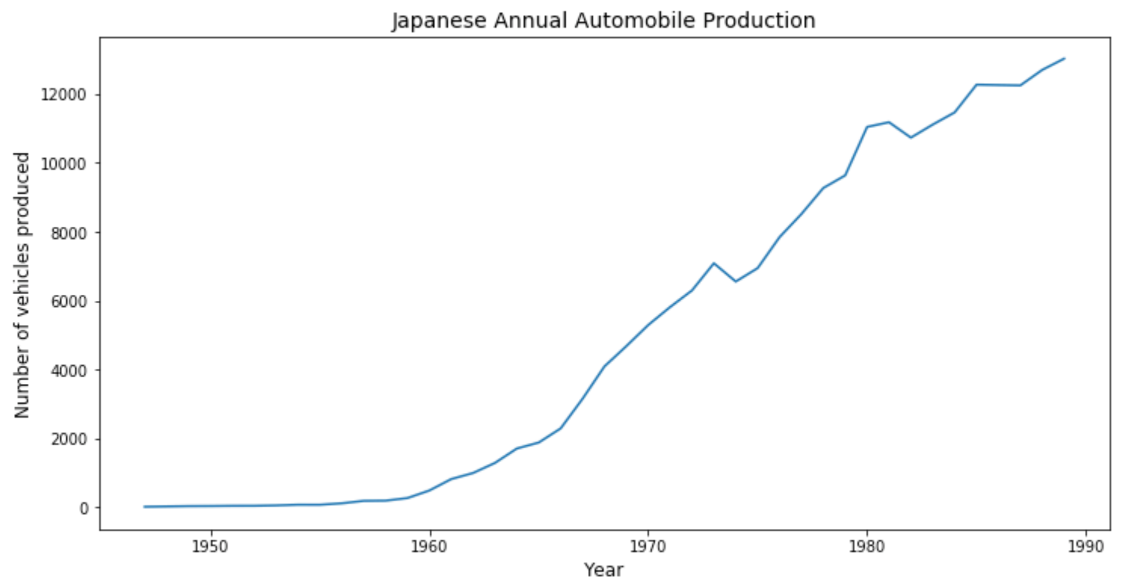
A white noise series (sequence of random numbers) is stationary.



More formally, a time series is stationary if $X_{1:k}$ and X_{t+k-1} have the same distribution, for all k and t. (Every section of length k has the same distribution of values).

Stationarity

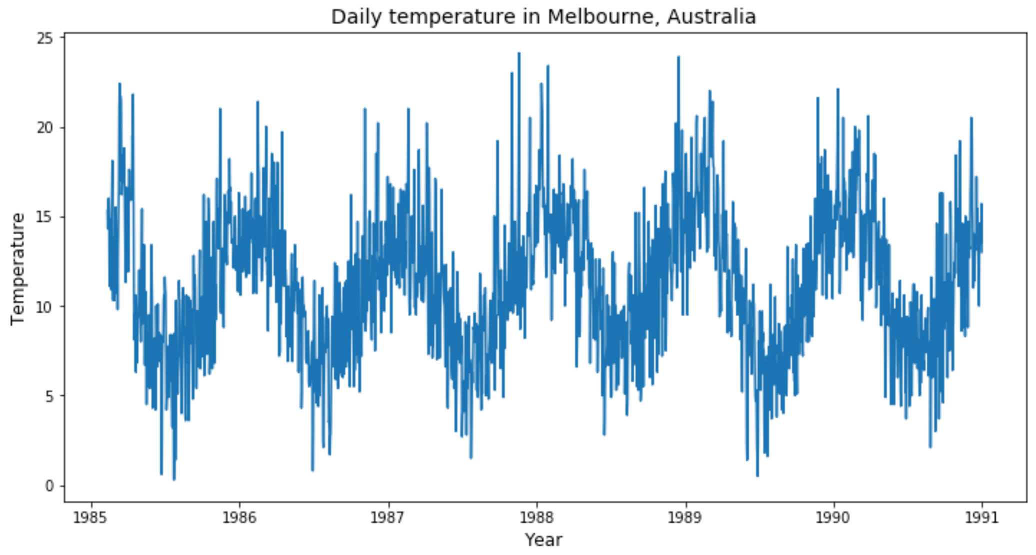
Is this time series stationary?



No, a series with a trend is non-stationary.

Stationarity

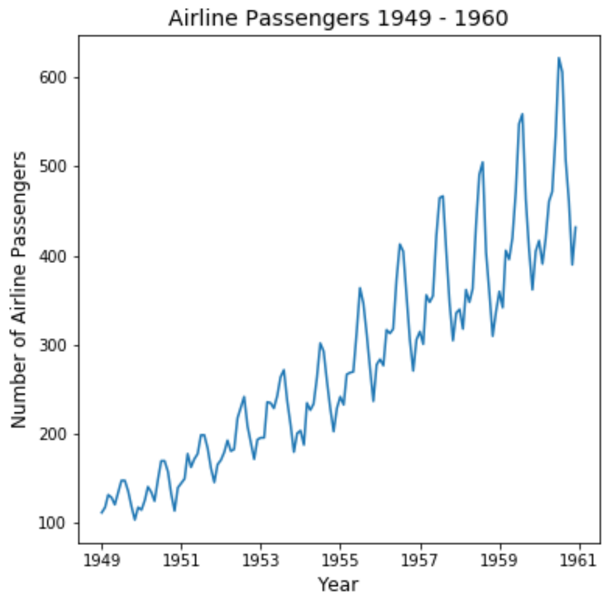
Is this time series stationary?



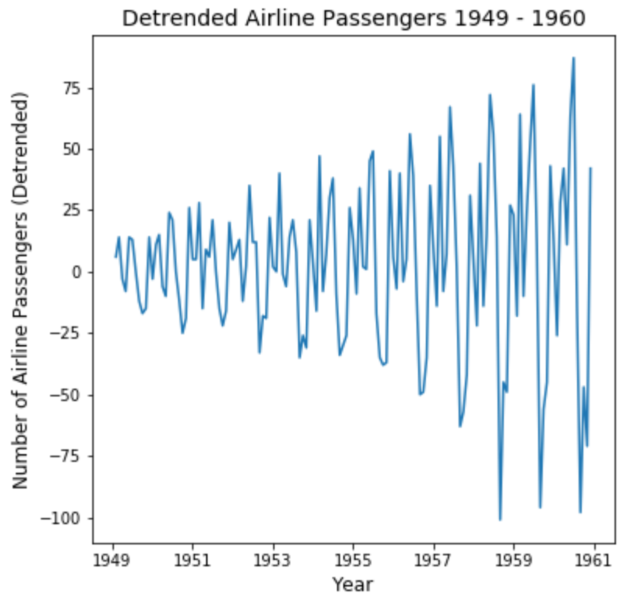
No, a series with seasonality is non-stationary.

Stationarity

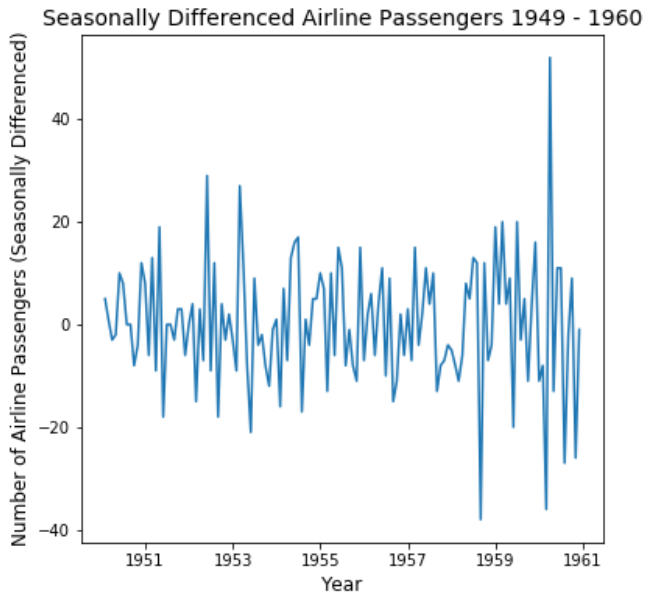
It's often useful to transform a non-stationary series into a stationary series for modeling.



Original series



Removing trend
(First-order differencing)



Removing seasonality
(Seasonal differencing)

This is stationary

Outline

Properties of time series data

Applications and examples

Descriptive methods for understanding a time series

Forecasting

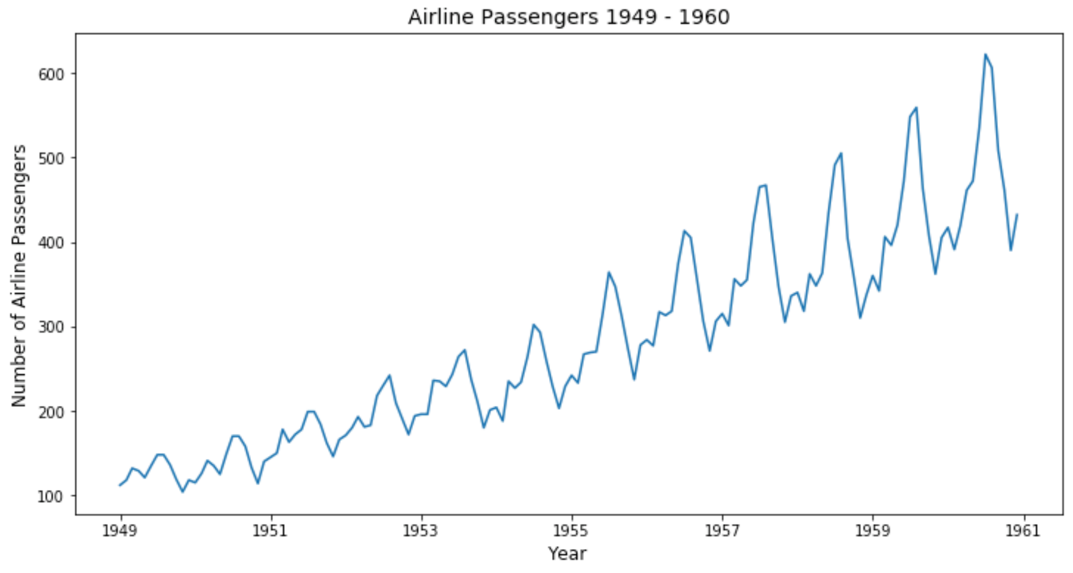
Applications of Time Series

A few applications of time series data:

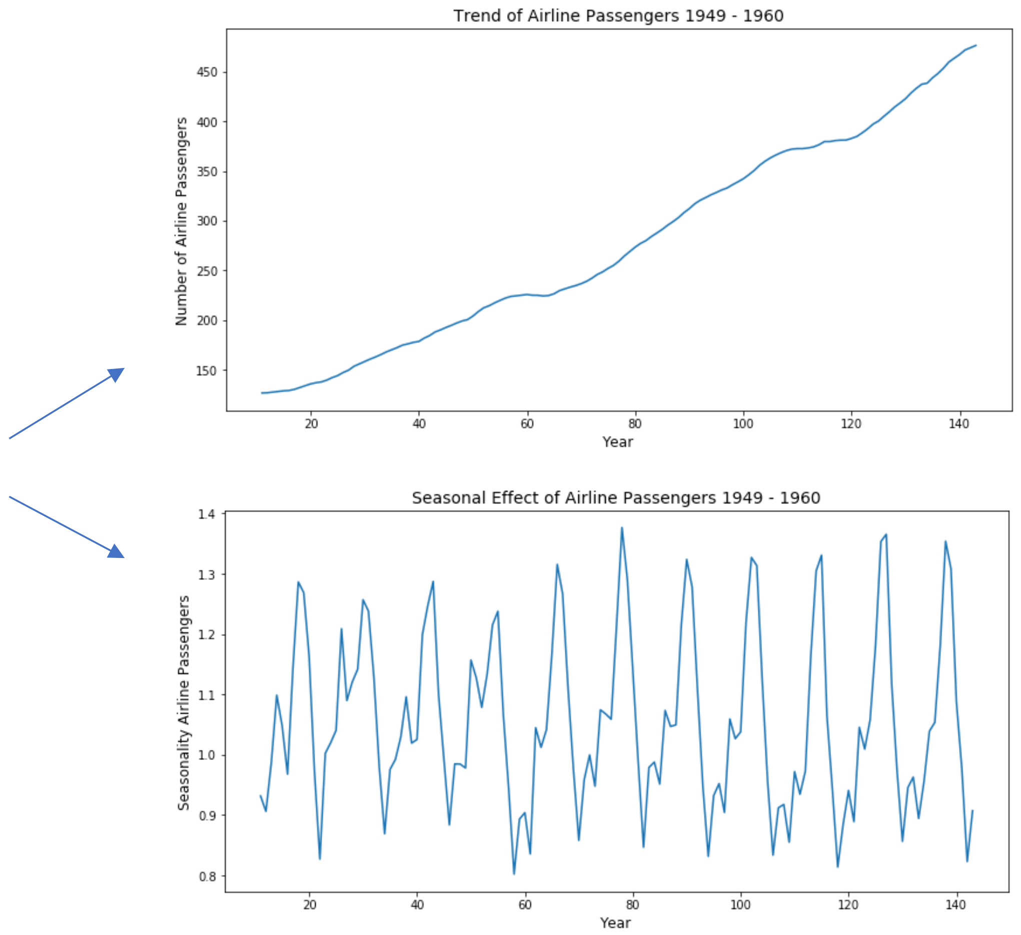
- Description
- Explanation
- Control
- Forecasting

Application: description

Can we identify and measure the trends, seasonal effects, and outliers in the series?



Original Series

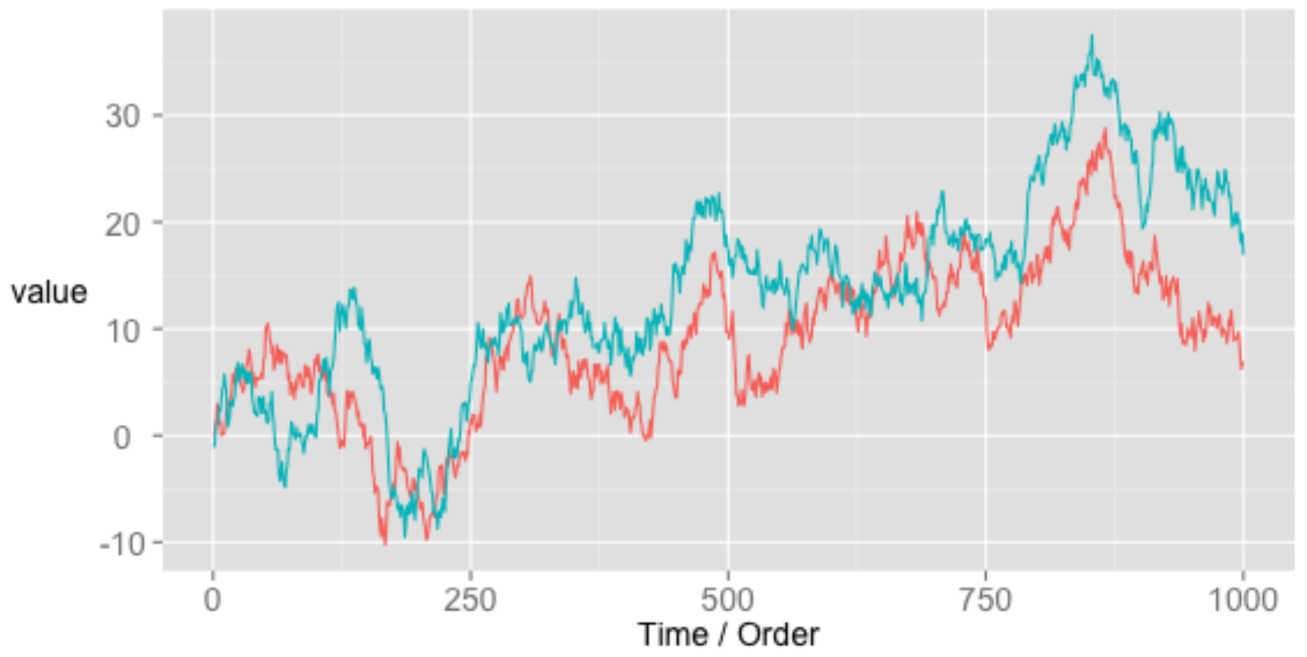


Trend component

Seasonal component

Application: explanation

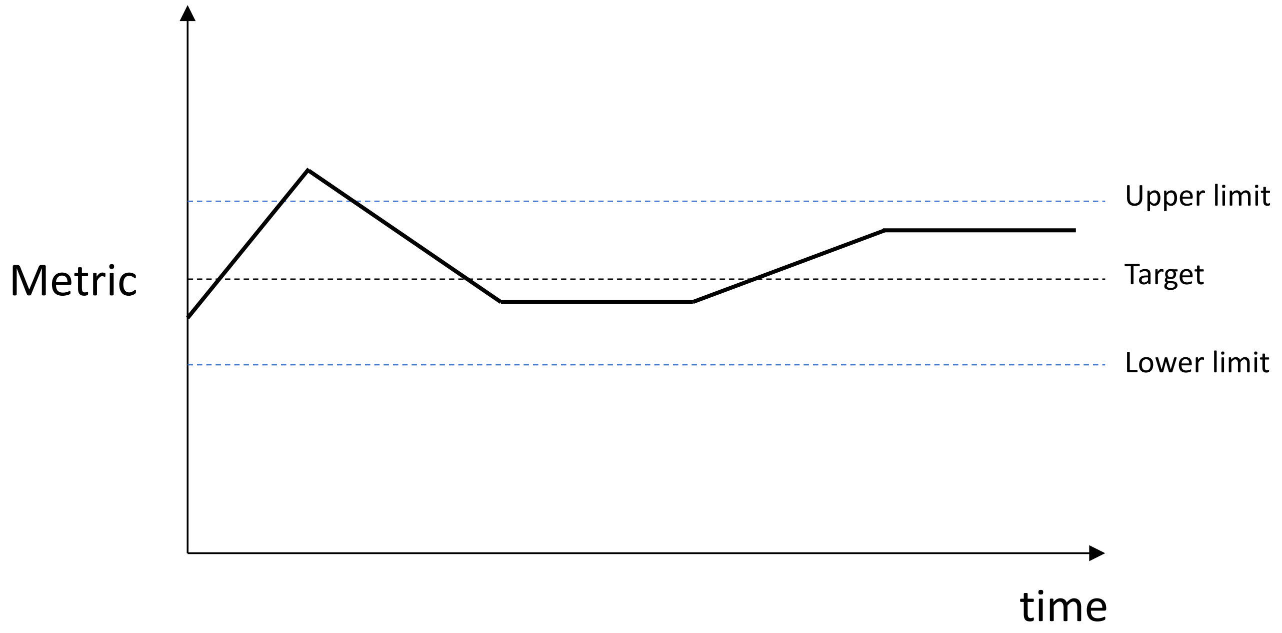
Can we use one time series to explain/predict values in another series?



Model using linear systems: convert one series to another using linear operations.

Application: control

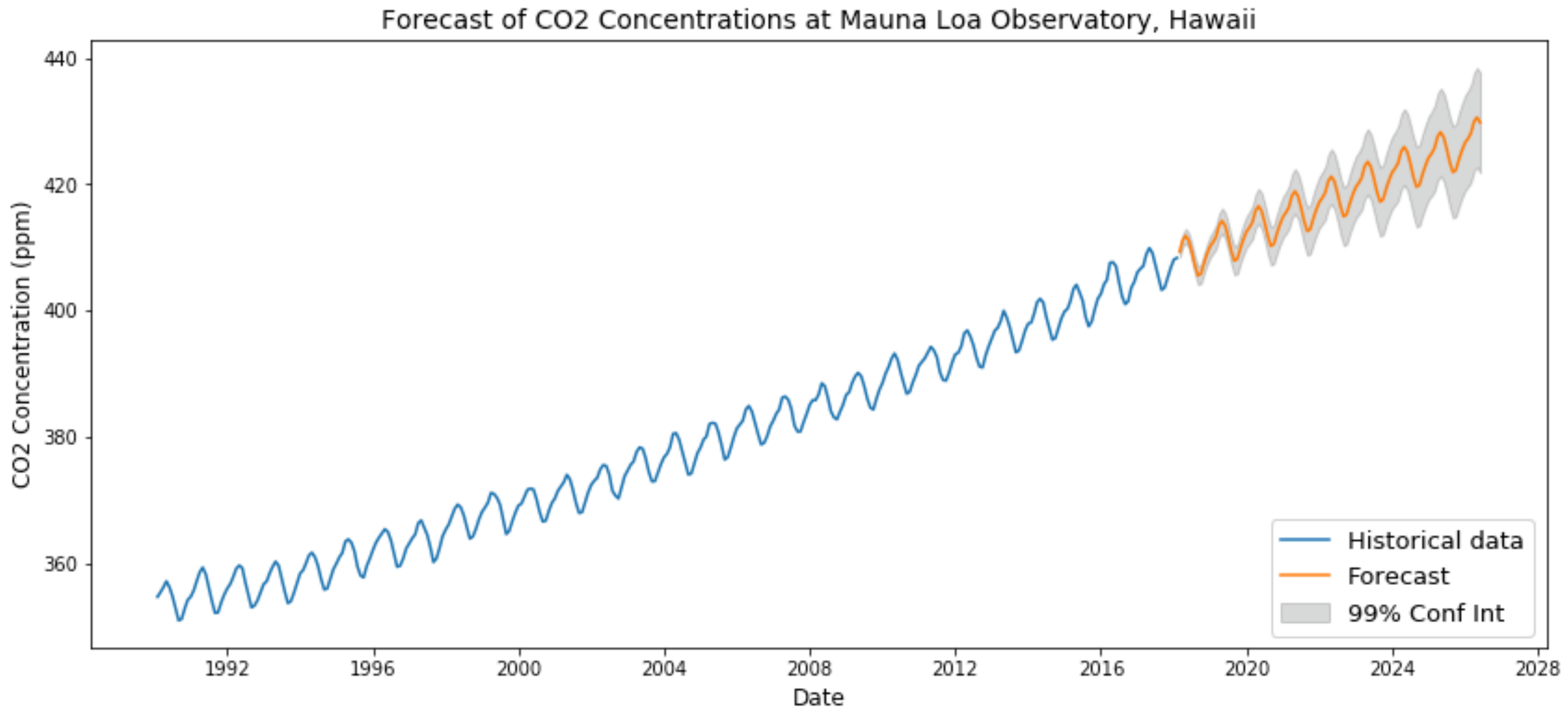
Can we identify when a time series is deviating away from a target?



Example: Manufacturing quality control

Application: forecasting

Using observed values, can we predict future values of the series?



Applications of Time Series

In this lecture:

- Description

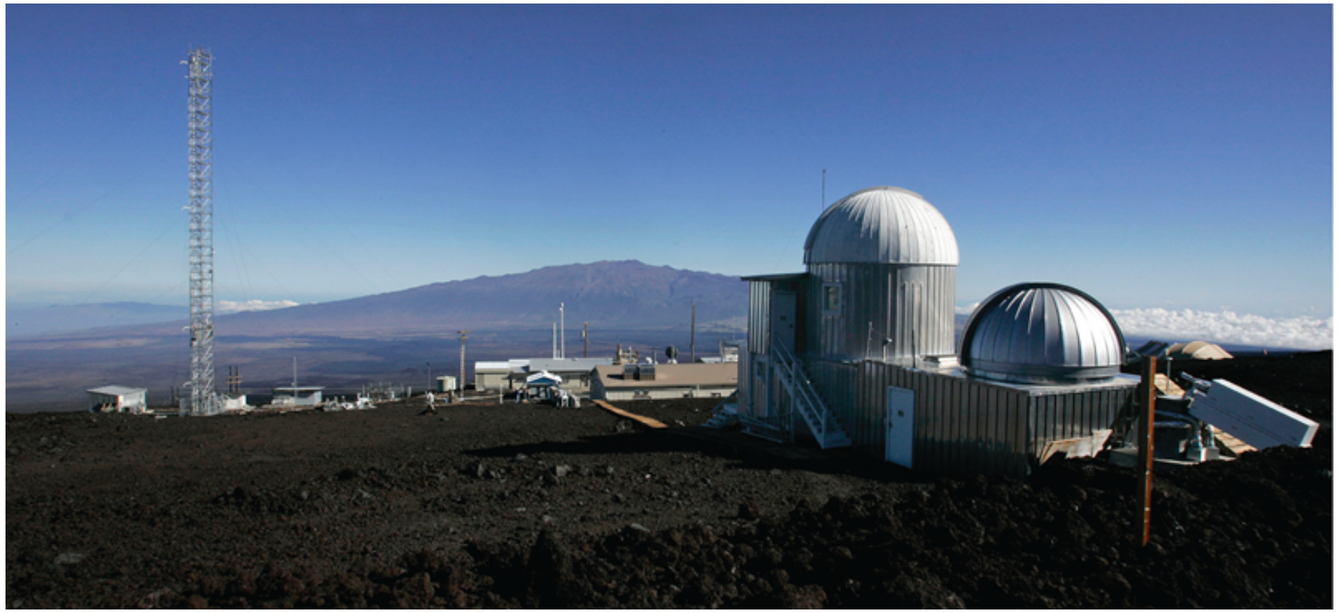
  Can we identify and measure the trends, seasonal effects, and outliers in the series?

- Explanation
- Control
- Forecasting

  Using observed values, can we predict future values of the series?

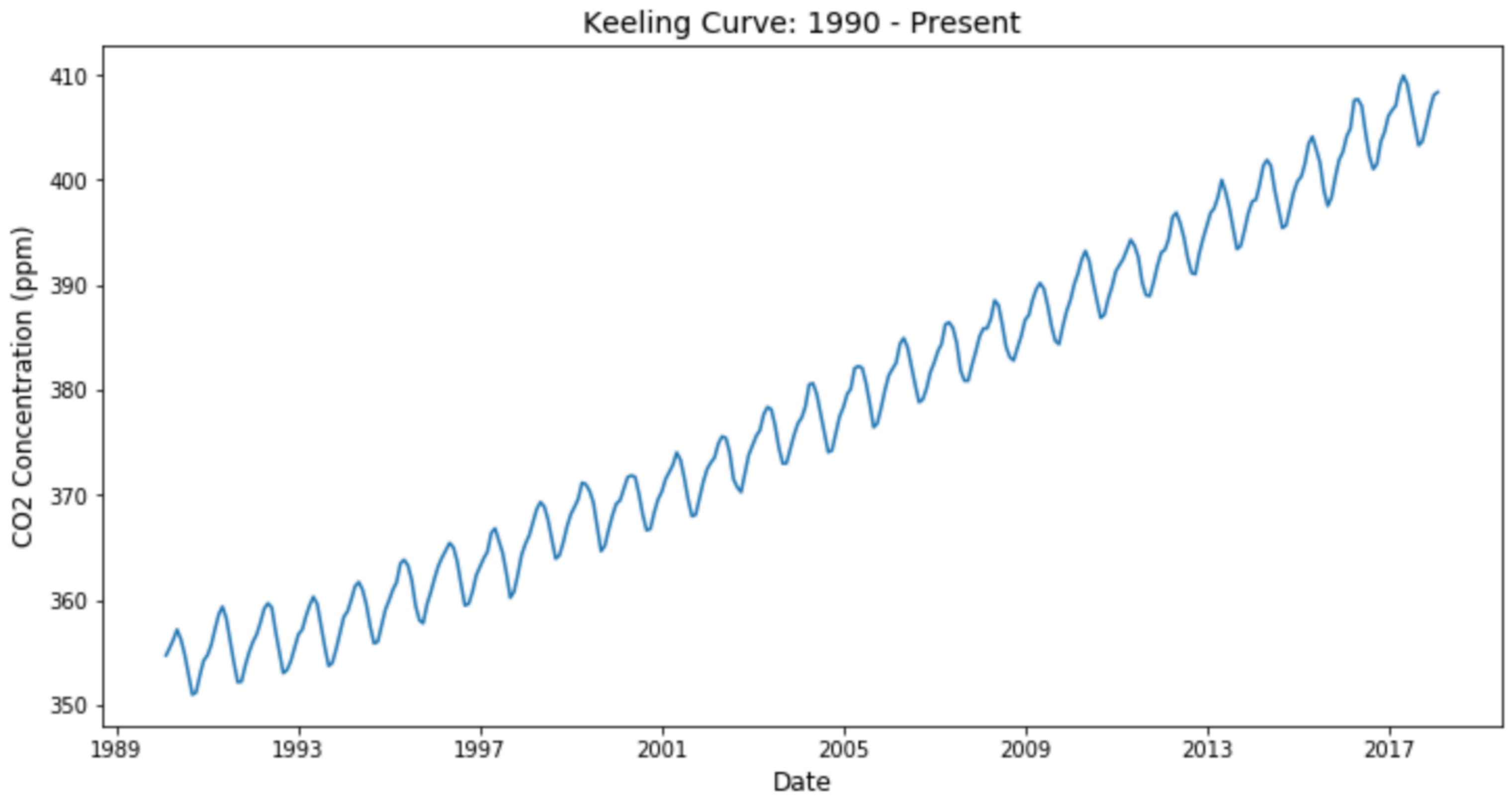
Example: Keeling Curve

The Keeling Curve is the foundation of modern climate change research.



Daily observations of atmospheric CO_2 concentrations since 1958 at the Mauna Loa Observatory in Hawaii.

Example: Keeling Curve



Why is there an annual season? Plants grow in spring, die in fall

Why is there a trend? Climate change

Outline

Properties of time series data

Applications and examples

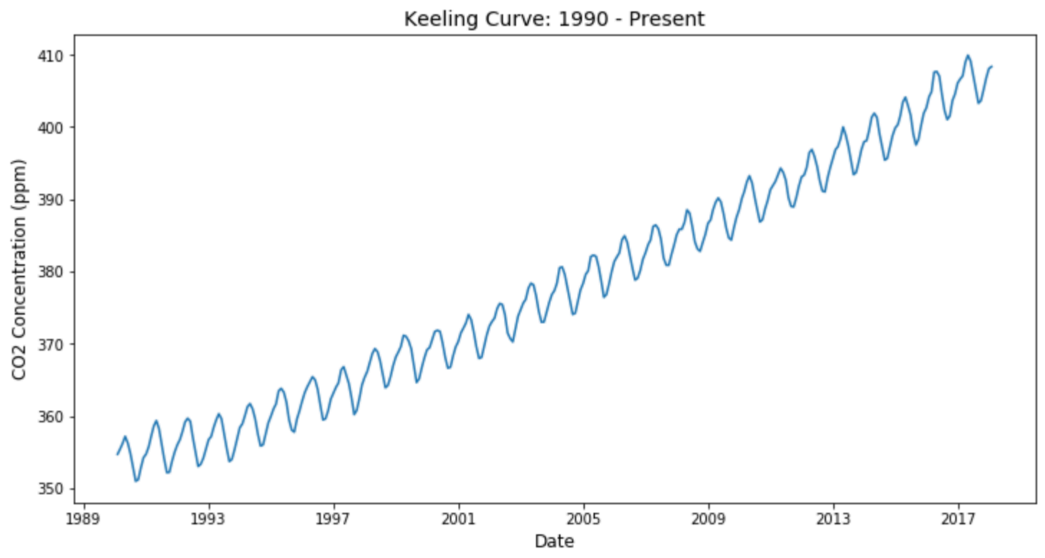
Descriptive methods for understanding a time series

Forecasting

Time plot

The first thing you should do in any time series analysis is plot the data.

```
plt.plot(df['date'], df['CO2'])
plt.xlabel('Date', fontsize=12)
plt.ylabel('CO2 Concentration (ppm)', fontsize=12)
plt.title('Keeling Curve: 1990 - Present', fontsize=14)
```



Plotting helps us identify salient properties of the series:

- Trend
- Seasonality
- Outliers
- Missing data

Measuring the trend

Next, we can take a more systematic approach in measuring the trend of the series.

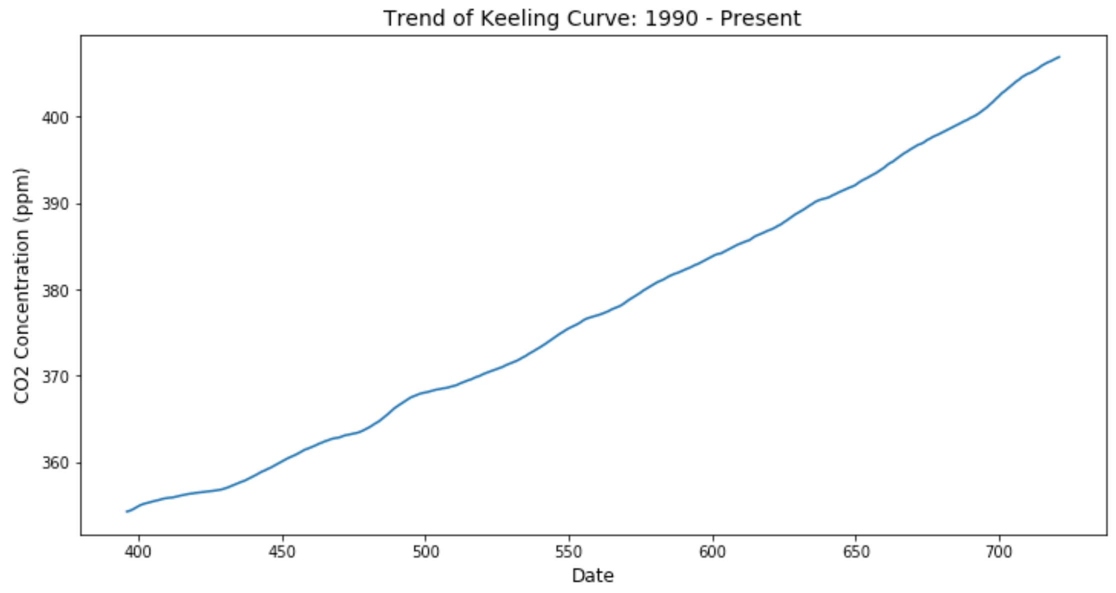
We can estimate a trend by using a moving average.

$$X_t = \frac{1}{2k} \sum_{i=-k}^{k} X_{t+i}$$

Measuring the trend

Implementing the moving average is easy.

```
moving_avg = df['CO2'].rolling(12).mean()
fig = plt.figure(figsize=(12,6))
plt.plot(moving_avg.index, moving_avg)
plt.xlabel('Date', fontsize=12)
plt.ylabel('CO2 Concentration (ppm)', fontsize=12)
plt.title('Trend of Keeling Curve: 1990 - Present', fontsize=14)
```



Removing the trend

We can also remove the trend by first-order differencing.

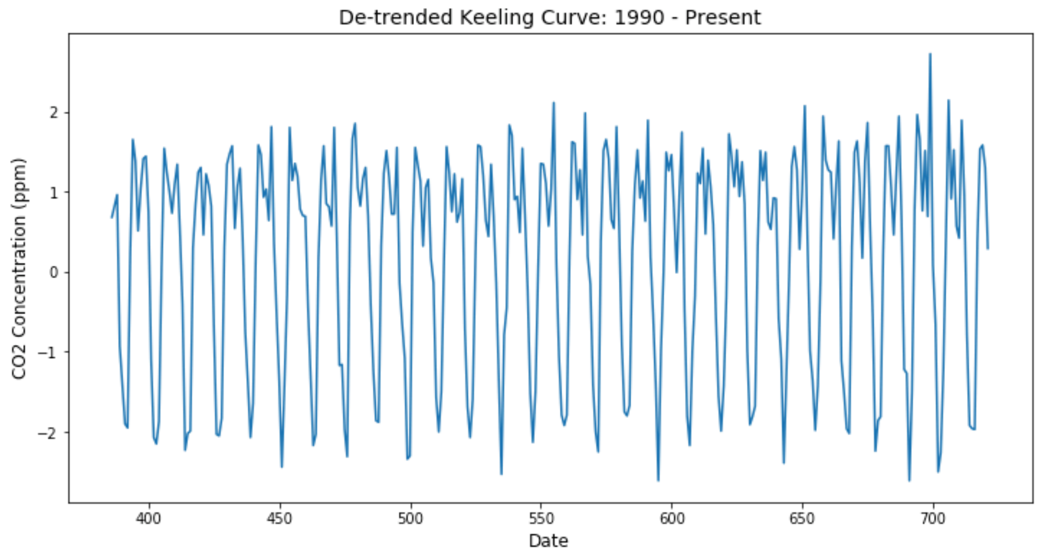
$$X'_t = \mathrm{X_t} - \mathrm{X_{t-1}}$$

X'_t will be a de-trended series.

Removing the trend

Implementing first-order differencing.

```
detrended = df['CO2'].diff()
fig = plt.figure(figsize=(12,6))
plt.plot(detrended.index, detrended)
plt.xlabel('Date', fontsize=12)
plt.ylabel('CO2 Concentration (ppm)', fontsize=12)
plt.title('De-trended Keeling Curve: 1990 - Present', fontsize=14)
```



Removing seasonality

We can also remove the seasonality through seasonal differencing.

$$X'_t = \mathrm{X_t} - \mathrm{X_{t-m}}$$

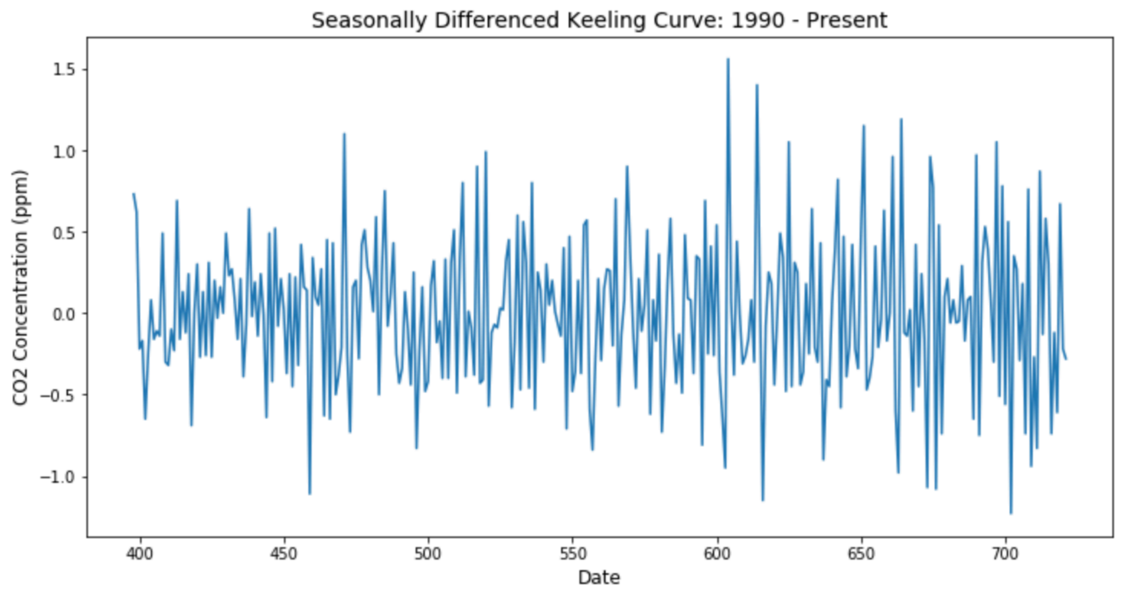
where m is the length of the season

X'_t will be a de-seasonalized series

Removing seasonality

Implementing seasonal differencing.

```
seasonal_diff = detrended.diff(12)
fig = plt.figure(figsize=(12,6))
plt.plot(seasonal_diff.index, seasonal_diff)
plt.xlabel('Date', fontsize=12)
plt.ylabel('CO2 Concentration (ppm)', fontsize=12)
plt.title('Seasonally Differenced Keeling Curve: 1990 - Present', fontsize=14)
```



Outline

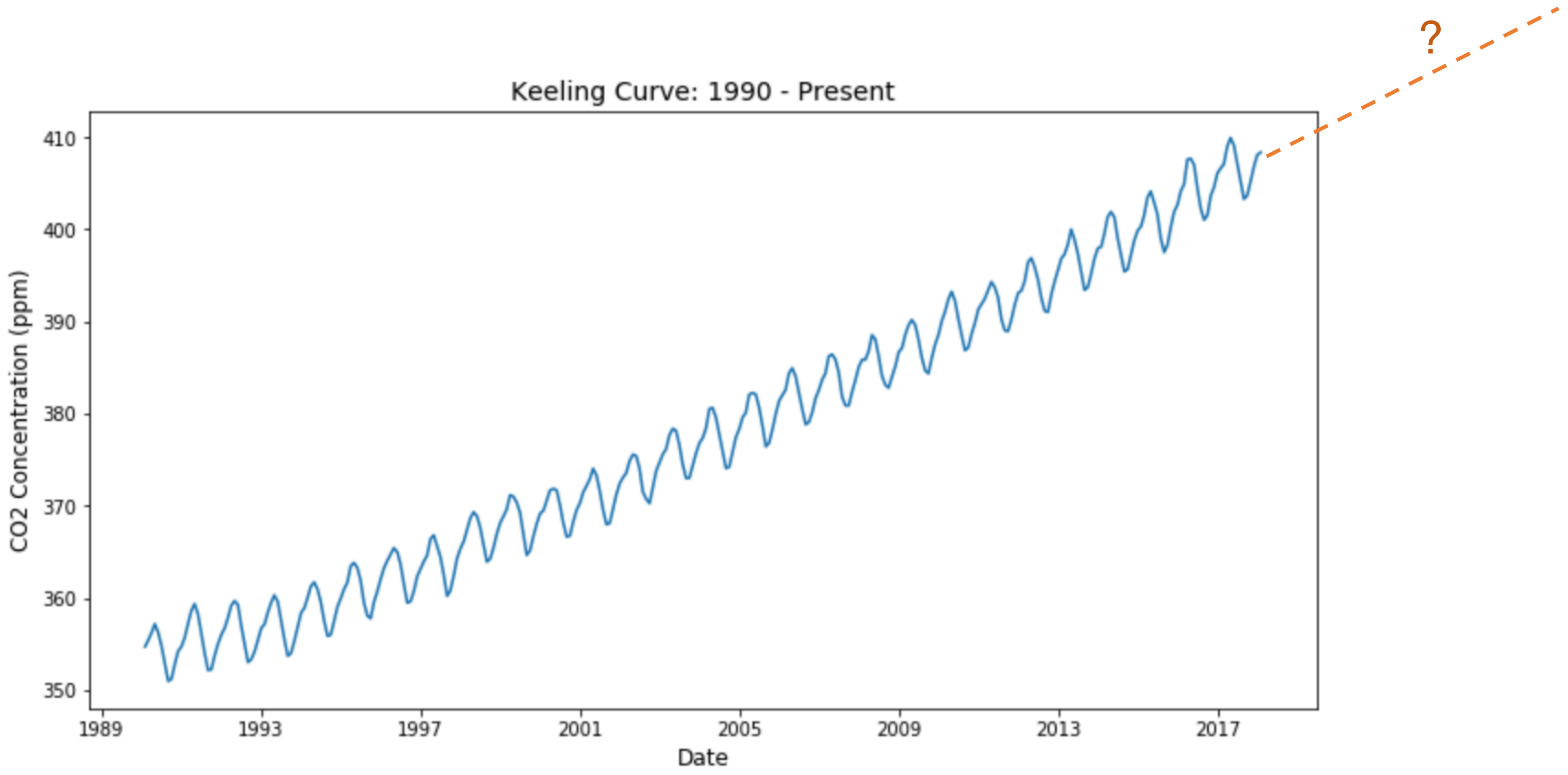
Properties of time series data

Applications and examples

Descriptive methods for understanding a time series

Forecasting

Forecasting

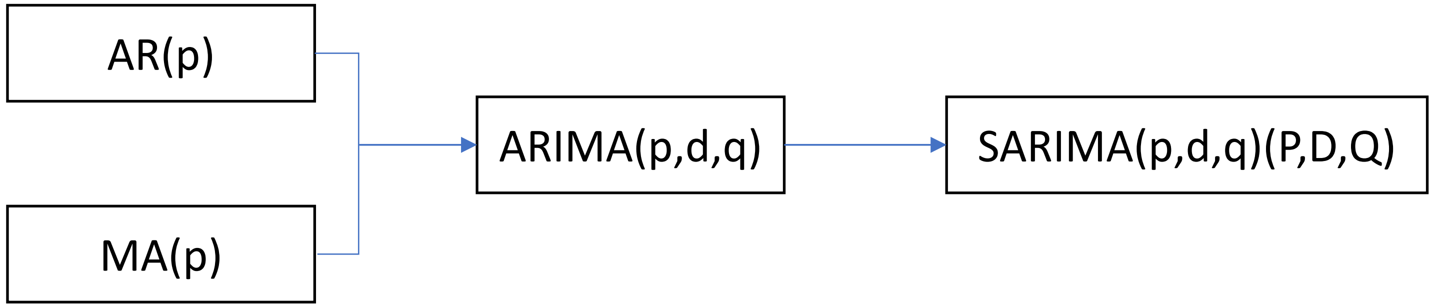


Can we predict future values of the Keeling curve using observed values?

Forecasting

Now, we will introduce a class of linear models called the ARIMA models, which can be used for time series forecasting.

There are several variants of ARIMA models, and they build on each other.



ARIMA models work by modeling the autocorrelations (correlations between successive observations) in the data.

Autoregressive Model: AR

An autoregressive model predicts the response X_t using a linear combination of past values of the variable. Parameterized by p, (the number of past values to include).

$$X_t = \theta_0 + \theta_1 X_{t-1} + \theta_2 X_{t-2} + \ \dots + \theta_p X_{t-p}$$

This is the same as doing linear regression with lagged features. For example, this is how you would set up your dataset to fit an autoregressive model with $p = 2$:

t	X_t
1	400
2	500
3	300
4	100
5	200

$\longrightarrow$

X_{t-2}	X_{t-1}	X_t
400	500	300
500	300	100
300	100	200

Moving Average Model: MA

A moving average model predicts the response X_t using a linear combination of past forecast errors.

$$X_t = \beta_0 + \beta_1 \epsilon_{t-1} + \beta_2 \epsilon_{t-2} + \ \dots + \beta_q \epsilon_{t-q}$$

where ϵ_i is normally distributed white noise (mean zero, variance one)

Parameterized by q, the number of past errors to include. The predictions X_t can be the weighted moving average of past forecast errors.

AutoRegressive Integrated Moving Average Model: ARIMA

Combining a autoregressive (AR) and moving average (MA) model, we get the ARIMA model.

$$X'_t = \theta_0 + \theta_1 X_{t-1} + \theta_2 X_{t-2} + \ \dots + \theta_p X_{t-p}$$

$$+ \beta_0 + \beta_1 \epsilon_{t-1} + \beta_2 \epsilon_{t-2} + \ \dots + \beta_q \epsilon_{t-q}$$

Note that now we are regressing on X'_t, which is the differenced series X_t. The order of difference is determined by the the parameter d. For example, if $d = 1$:

$$X'_t = \mathrm{X_t} - \mathrm{X_{t-1}} \ \text{ for t} = 2, 3, \dots, \mathrm{N}$$

So the ARIMA model is parameterized by: p (order of the AR part), q (order of the MA part), and d (degree of differencing).

Seasonal ARIMA: SARIMA

Extension of ARIMA to model seasonal data.

Includes a non-seasonal part (same as ARIMA) and a seasonal part. The seasonal part is similar to ARIMA, but involves backshifts of the seasonal period.

In total, 6 parameters:

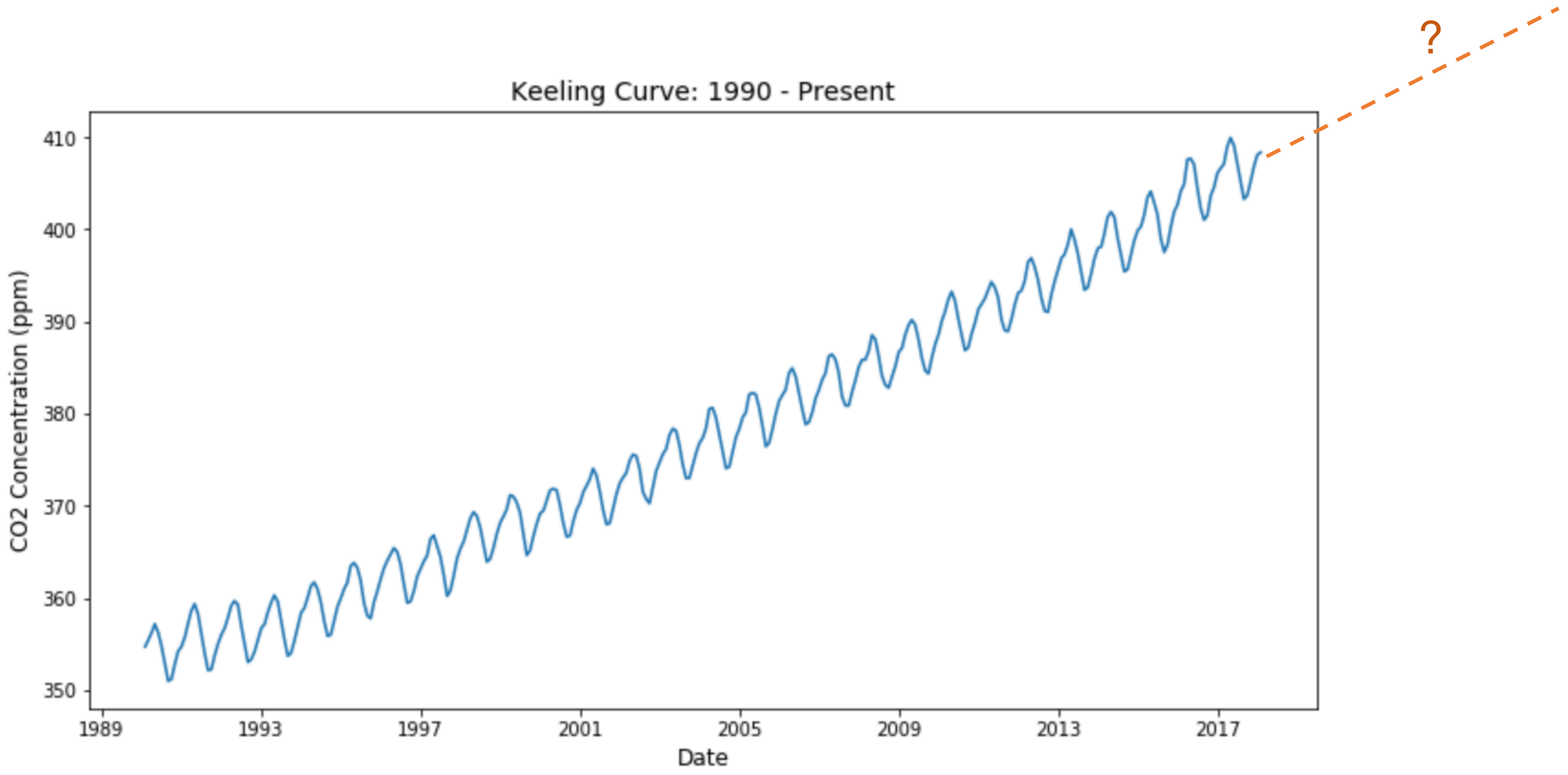
- (p, d, q) for non-seasonal part
- $(P, D, Q)_s$ for seasonal part, where s is the length of season

Implementing an ARIMA model

How to find the parameters (p, d, q) and $(P, D, Q)_m$ that best fit the data?

- m is known: just visualize the data to know season length
- d and D are easy to determine:
  - Does you data need de-trending? If so, d = 1 or 2. If not, d = 0.
  - Does you data need seasonal differencing? If so, D = 1 or 2. If not, D = 0.
- p, P, q, and Q can be estimated by looking the autocorrelation and partial autocorrelation
- In practice, just do grid search over the (p, q) and (P, Q) values to find the parameters that optimize performance (usually minimize AIC).

Implementing an ARIMA model

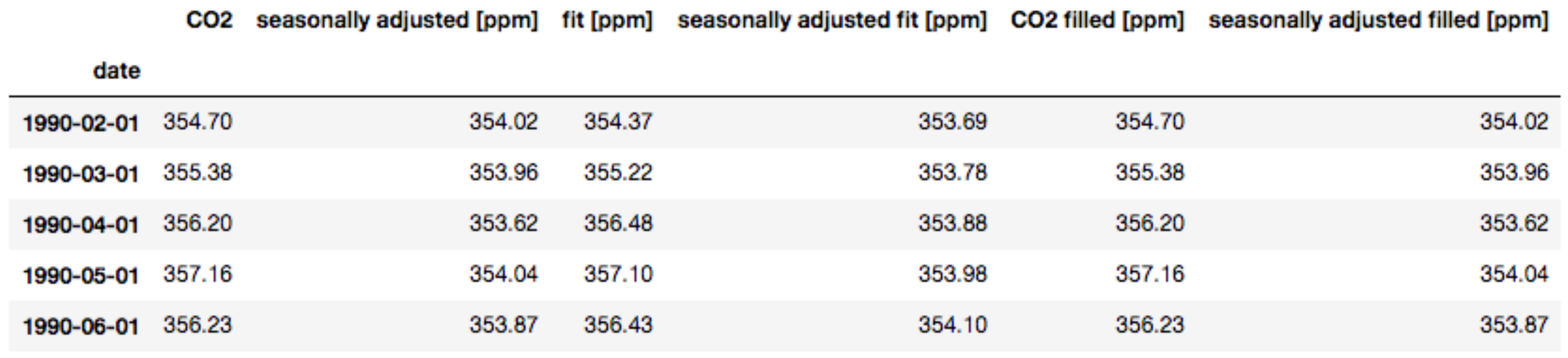


Lets fit an SARIMA model to the Keeling curve to forecast future values.

Implementing an ARIMA model

Dataframe contains variable CO2, which we want to predict

```
df.head()
```



date	CO2	seasonally adjusted [ppm]	fit [ppm]	seasonally adjusted fit [ppm]	CO2 filled [ppm]	seasonally adjusted filled [ppm]
1990-02-01	354.70	354.02	354.37	353.69	354.70	354.02
1990-03-01	355.38	353.96	355.22	353.78	355.38	353.96
1990-04-01	356.20	353.62	356.48	353.88	356.20	353.62
1990-05-01	357.16	354.04	357.10	353.98	357.16	354.04
1990-06-01	356.23	353.87	356.43	354.10	356.23	353.87

Implementing an ARIMA model

Fit SARIMA model using StatsModels library.

(p, d, q)

(P, D, Q, m)

```
from statsmodels.tsa.statespace.sarimax import SARIMAX


model = SARIMAX(df['CO2'],
                order=(1, 1, 1),
                seasonal_order=(1, 1, 1, 12))


result = model.fit()
print(result.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.1748      0.114      1.534      0.125      -0.049       0.398
ma.L1         -0.5632      0.096     -5.879      0.000      -0.751      -0.375
ar.S.L12      -0.0157      0.073     -0.215      0.830      -0.159       0.128
ma.S.L12      -0.8441      0.050    -16.943      0.000      -0.942      -0.746
sigma2         0.1094      0.008     14.138      0.000       0.094       0.125
==============================================================================
```

Implementing an ARIMA model

Generating point forecasts and confidence intervals 100 time steps into the future.

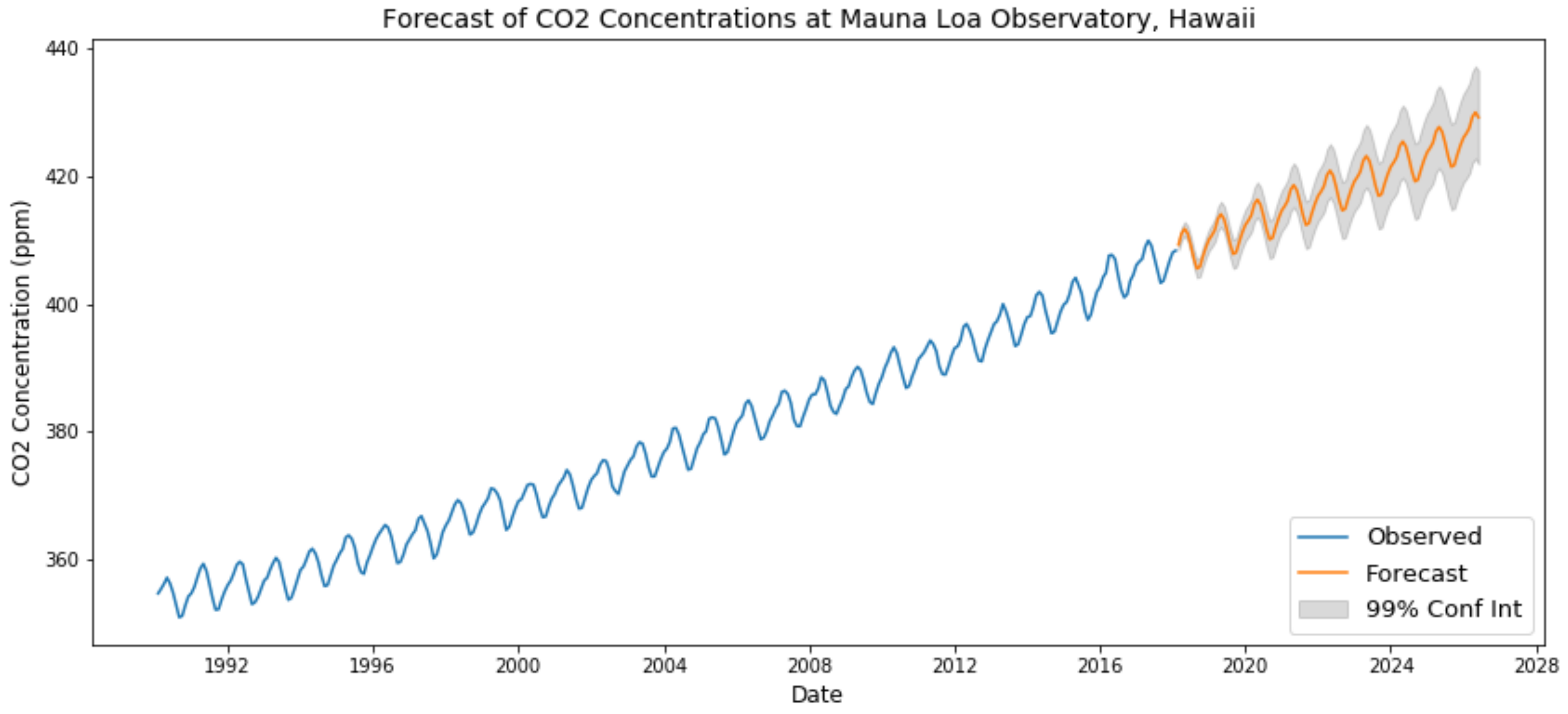
```
pred = result.get_forecast(steps=100)
pred_point = pred.predicted_mean
pred_ci = pred.conf_int(alpha=0.01)
```

Plot the forecast!

```
fig = plt.figure(figsize=(14,6))
plt.plot(df['CO2'], label='Observed')
plt.plot(pred_point, label='Forecast')
plt.fill_between(pred_ci.index, pred_ci.iloc[:, 0], pred_ci.iloc[:, 1],
                 color='k', alpha=.15, label='99% Conf Int')
plt.xlabel('Date', fontsize=12)
plt.ylabel('CO2 Concentration (ppm)', fontsize=12)
plt.title("Forecast of CO2 Concentrations at Mauna Loa Observatory, Hawaii",
          fontsize=14)
plt.legend(loc='lower right', fontsize=13)
```

Implementing an ARIMA model

Result of the forecast



Goals

After this lecture, you will be able to:

- Explain key properties of time series data
- Describe, measure, and remove **trend** and **seasonality** from a time series
- Understand the concept of **stationarity**
- Create and interpret **autocorrelation function** (acf) plots
- Understand **ARIMA** models for forecasting
- Create your own time series forecast

References

Books (good for learning the theory)

- Forecasting: Principles and Practice by Hyndman, Athanasopoulos
- The Analysis of Time Series by Chris Chatfield
- Time Series Analysis and it’s Applications by Shumway, Stoffer

Articles (good for seeing examples in Python)

- A Guide to Time Series Forecasting with ARIMA in Python:

  www.digitalocean.com/community/tutorials/a-guide-to-time-series-forecasting-with-arima-in-python-3
- Kaggle Time Series Notebook:

  https://www.kaggle.com/berhag/co2-emission-forecast-with-python-seasonal-arima