Time series modeling with ARIMA

Glenn Bruns
CSUMB

Learning outcomes

After this lecture you should be able to:

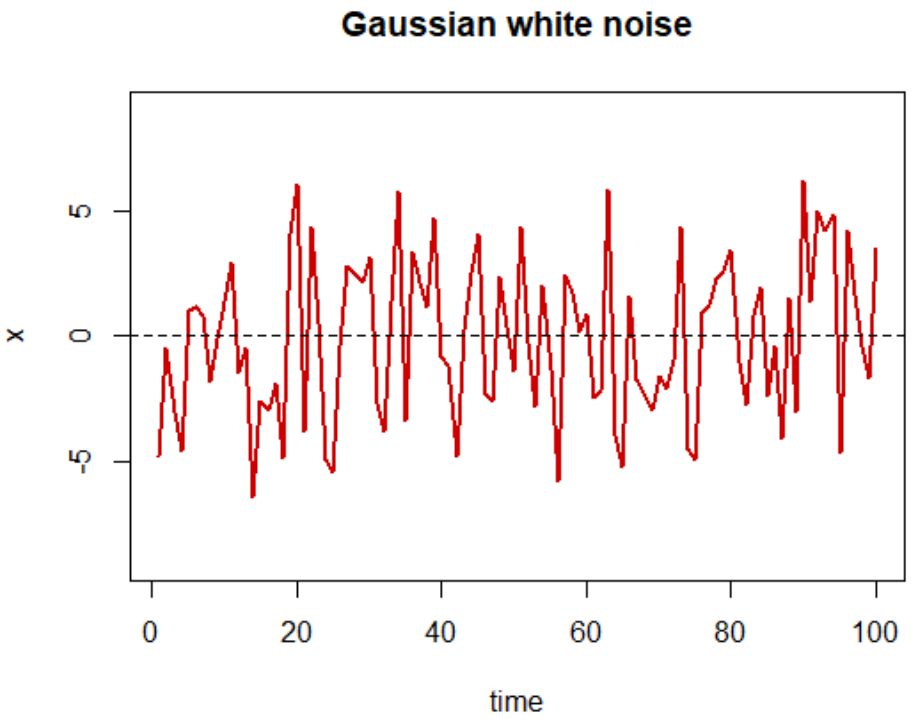
- define AR, MA, ARMA, and ARIMA models
- explain the idea behind the models
- simulate the models by hand
- use an ARIMA model for time-series forecasting

Time-series modeling

We've seen lots of models: linear models, trees, ensemble models, neural nets.

How to model time series data?

The most basic model is a sequence of random variables $x_1, x_2, x_3, \ldots$



In a white noise model, the random variables are all independent, and have mean 0.

The figure shows data from a Gaussian white noise model.

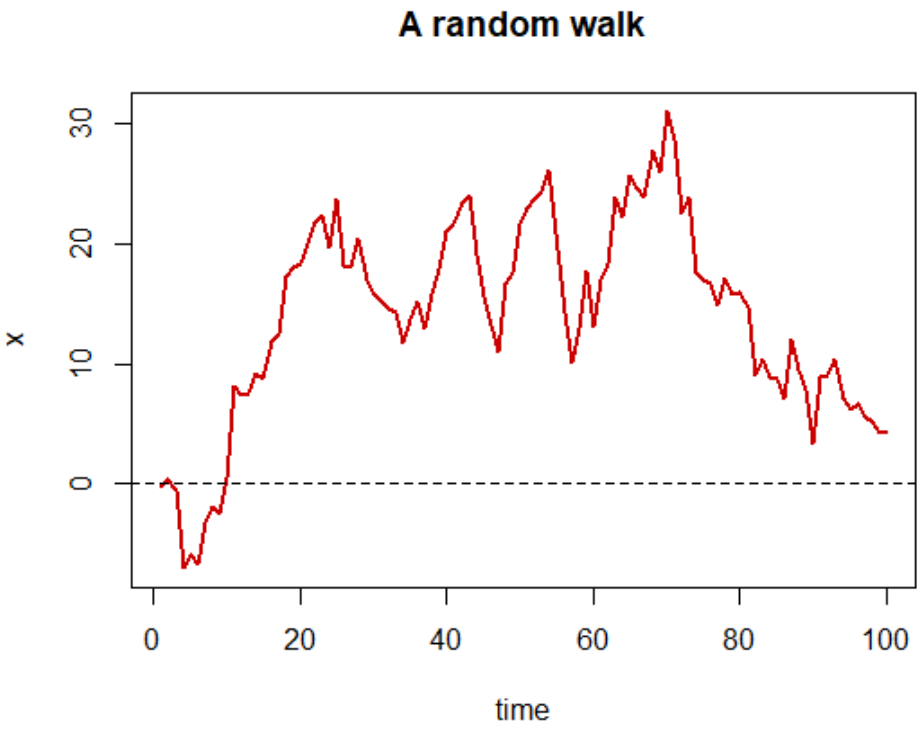
Random walk

Another basic model, like white noise.

In a random walk, the differences between values is white noise

$$x_1 = w_1$$
$$x_t = x_{t-1} + w_t$$

($w_1, w_2, w_3, \ldots$ is a white noise time series)



some example data:

w = 0.2, -0.5, -0.3, 0.4, …
x = 0.2, -0.3, -0.8, 0.4, …

More time series models

Modeling time series data with a sequence of random variables is very general.

The important thing is how all the variable are correlated.

- for example, if x_1 and x_2 are strongly positively correlated, then x_2 will be high if x_1 is high

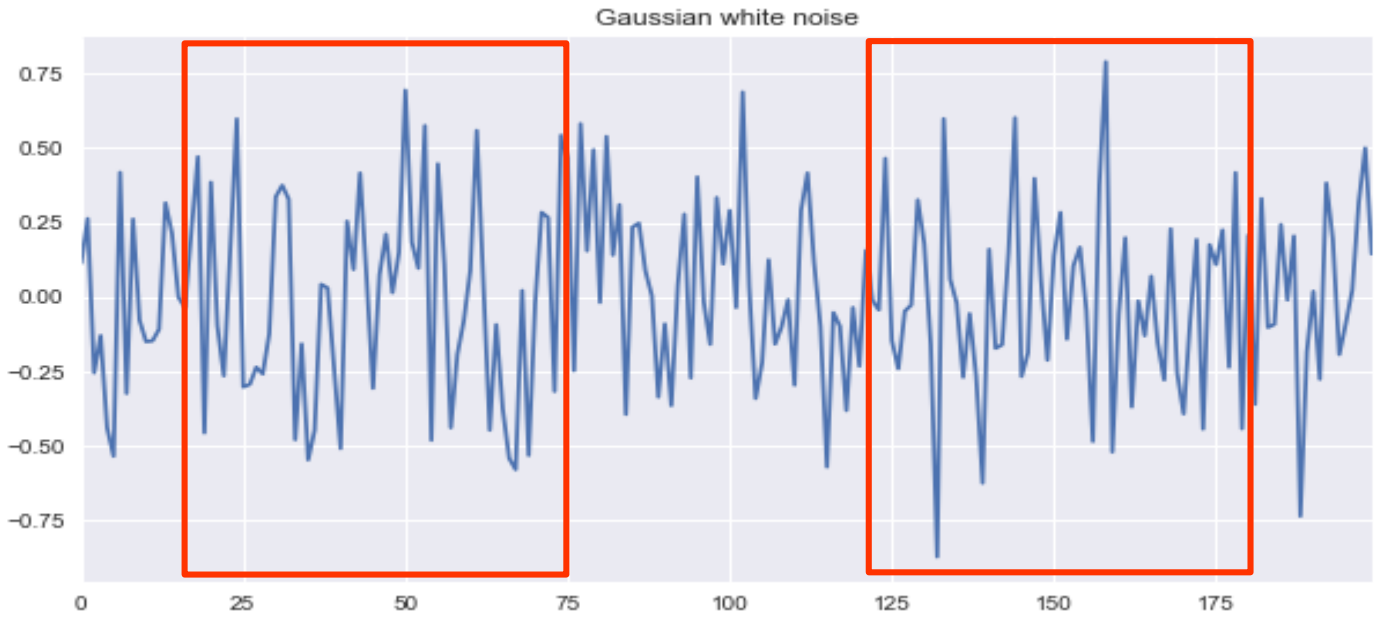
In practice, simpler models are used.

Most of the models assume the time series data is stationary!

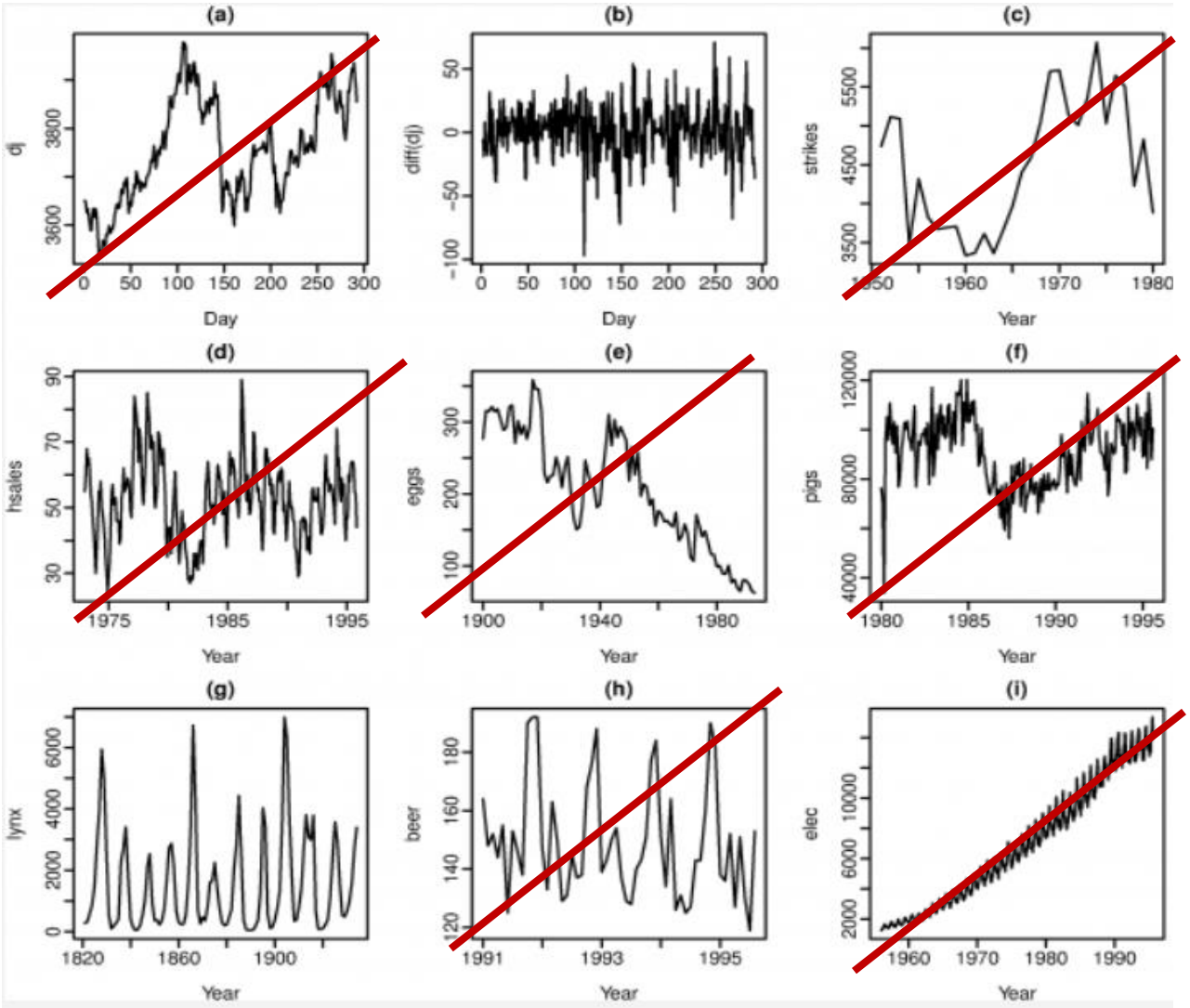
Stationarity

Roughly, this means the probabilistic behavior of a time series does not change over time.

For anything you want to know, you can use any window.



Which are stationary time series?



Not stationary if there is a trend or seasonality is present

Source: Forecasting: Principles and Practice, by Hyndman and Athanasopoulos

Autoregressive (AR) models

- a linear regression on past observations

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + w_t$$

$\phi_1, \phi_2, \ldots$ are constants, with $\phi_p \neq 0$
w_t is Gaussian white noise with mean 0

- past values influence the current value (e.g., if GDP is high this quarter, we expect it to be high next quarter)
- random walk is an example
- "normally" it is stationary

if p = 1, then we have an AR model of order 1, written AR(1)

if p = 2, AR(2), etc.

Simulate by hand

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + w_t$$

$\phi_1, \phi_2, \ldots$ are constants, with $\phi_p \neq 0$
w_t is Gaussian white noise with mean 0

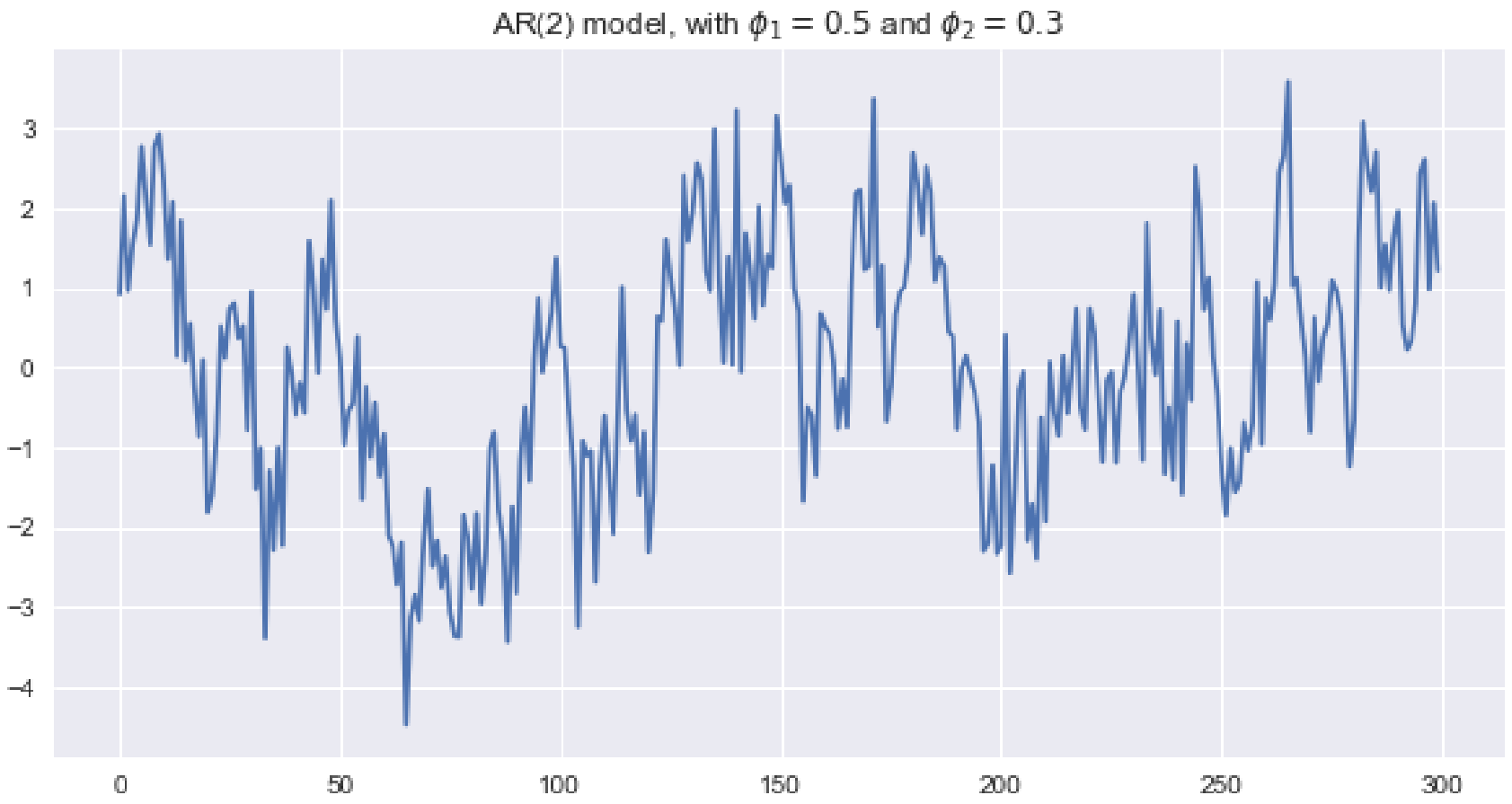
Let p = 1, and ϕ_1 be 1/2

$$x_t = 1/2\, x_{t-1} + w\_t$$

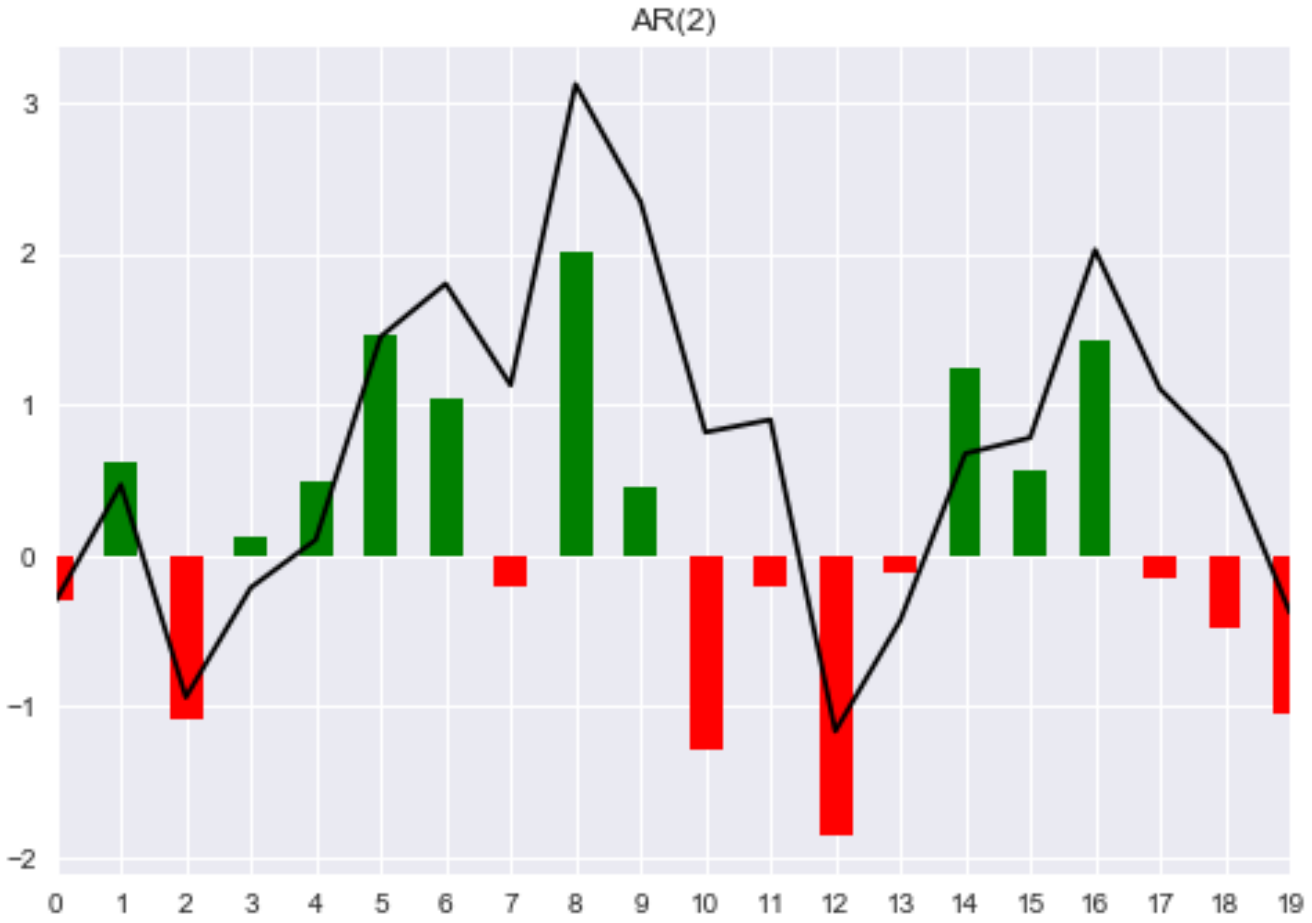
We start by generating random white noise.

t	w_t	x_t
1	-1.1	
2	0.6	
3	-0.6	
4	0.4	
5	0.8	
6	-0.3	
7	-0.9	

Simulated data from an AR model

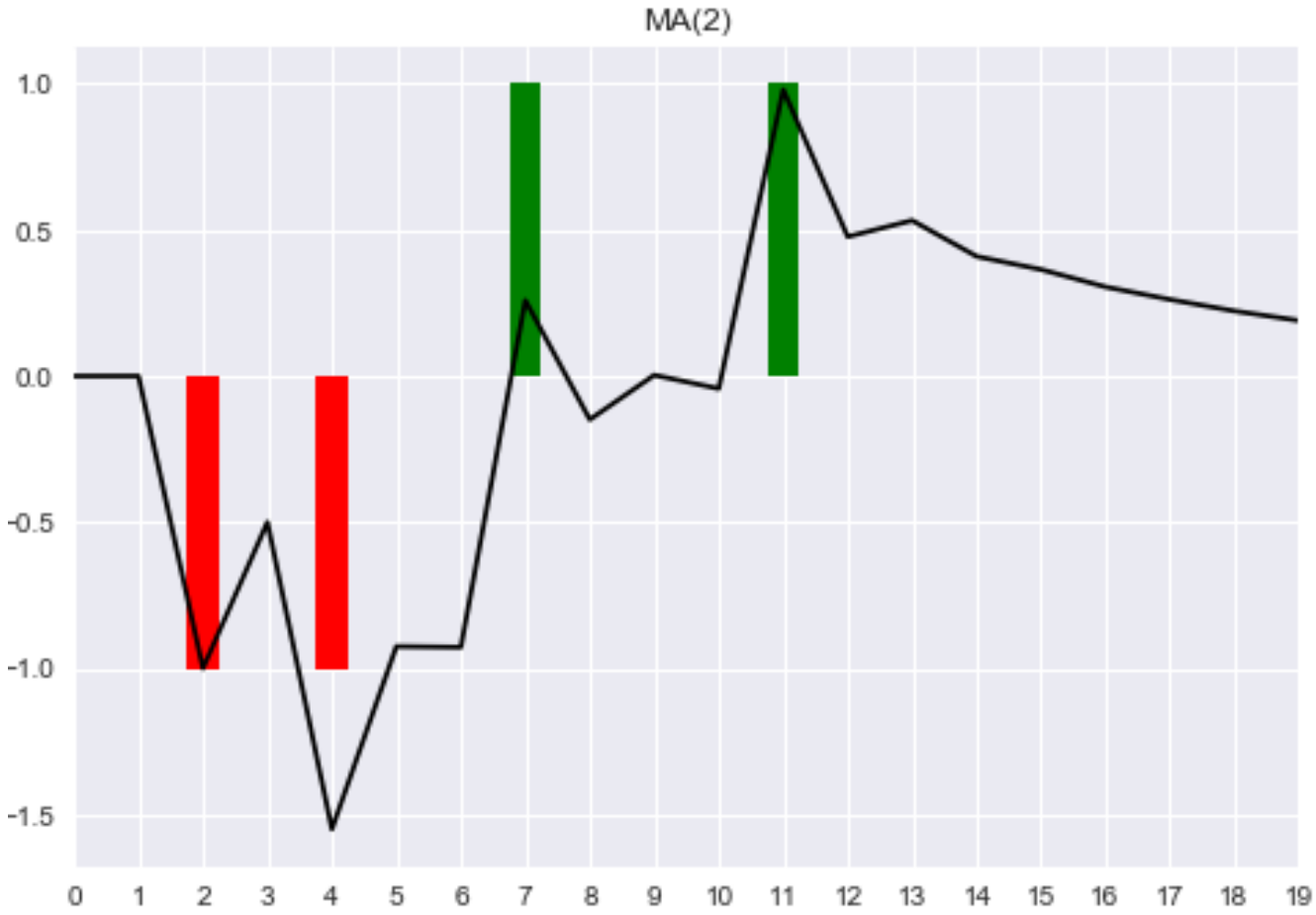


Visualizing an AR(2) model



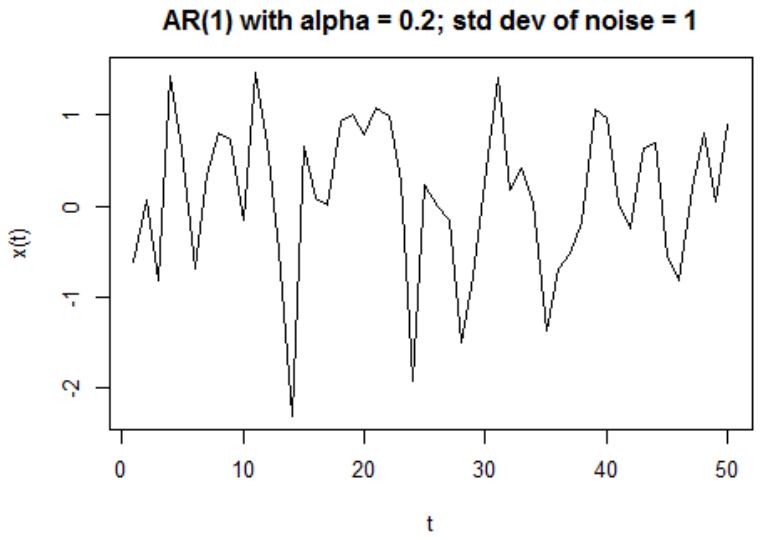
Two factors influence the value at a point in time: the random 'shock' at that time, and values at previous points in time.

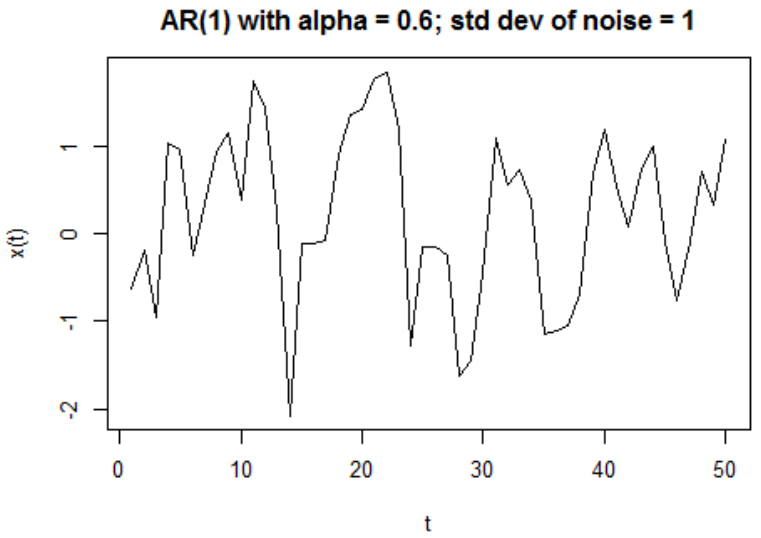
Visualizing an AR(2) model

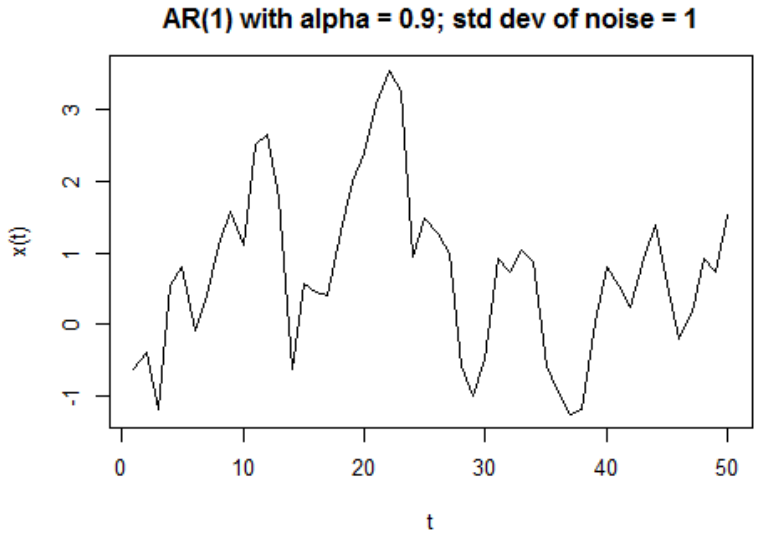


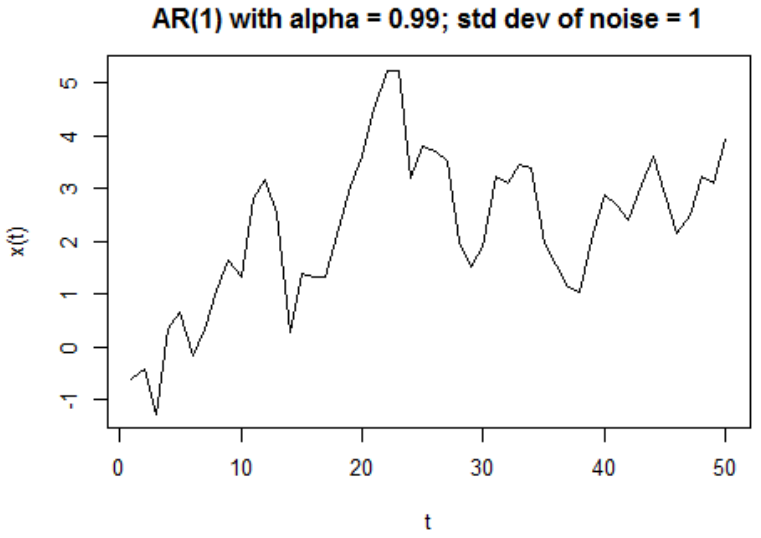
The degree of the model is 2, but you can see that the influence of a shock is much longer than 2 time steps.

What happens as ϕ_1 gets larger?









Moving average (MA) models

- a linear regression of past white noise values

$$x_t = w_t + \beta_1 w_{t-1} + \beta_2 w_{t-2} + \cdots + \beta_q w_{t-q}$$

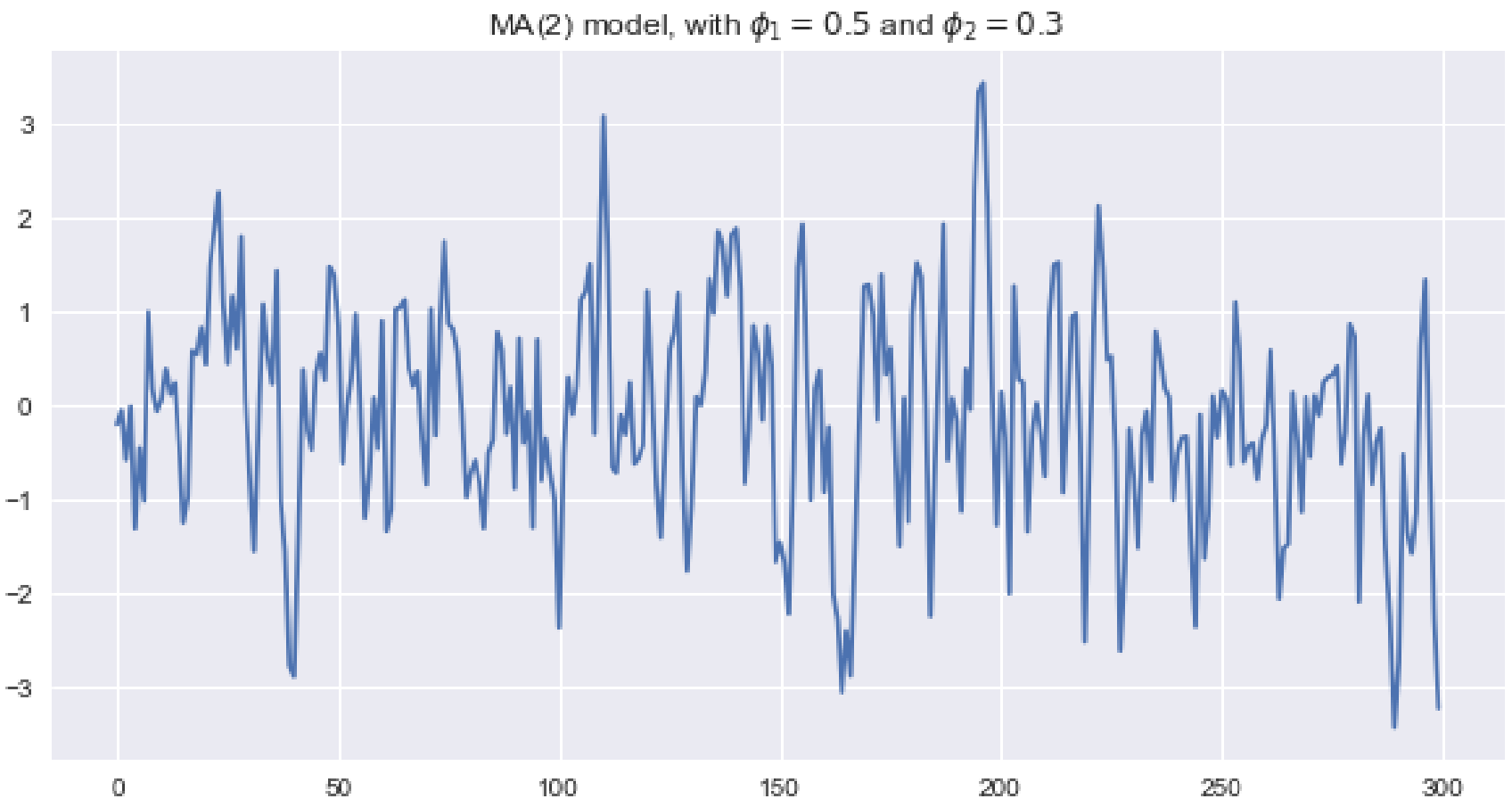
w_t is Gaussian white noise with mean 0

- intuitively, the process is responding to random "shocks" (e.g., a severe earthquake affects the economy)
- the influence of a shock can persist (e.g. the earthquake affects the economy not only now but in the near future)
- an MA process is always stationary

if q = 1, then we have an MA model of order 1, written MA(1)

if q = 2, MA(2), etc.

Simulated MA(2) model



Simulate by hand

$$x_t = w_t + \beta_1 w_{t-1} + \beta_2 w_{t-2} + \cdots + \beta_q w_{t-q}$$

$\beta_1, \beta_2, \ldots$ are constants, with $\beta_p \neq 0$
w_t is Gaussian white noise with mean 0

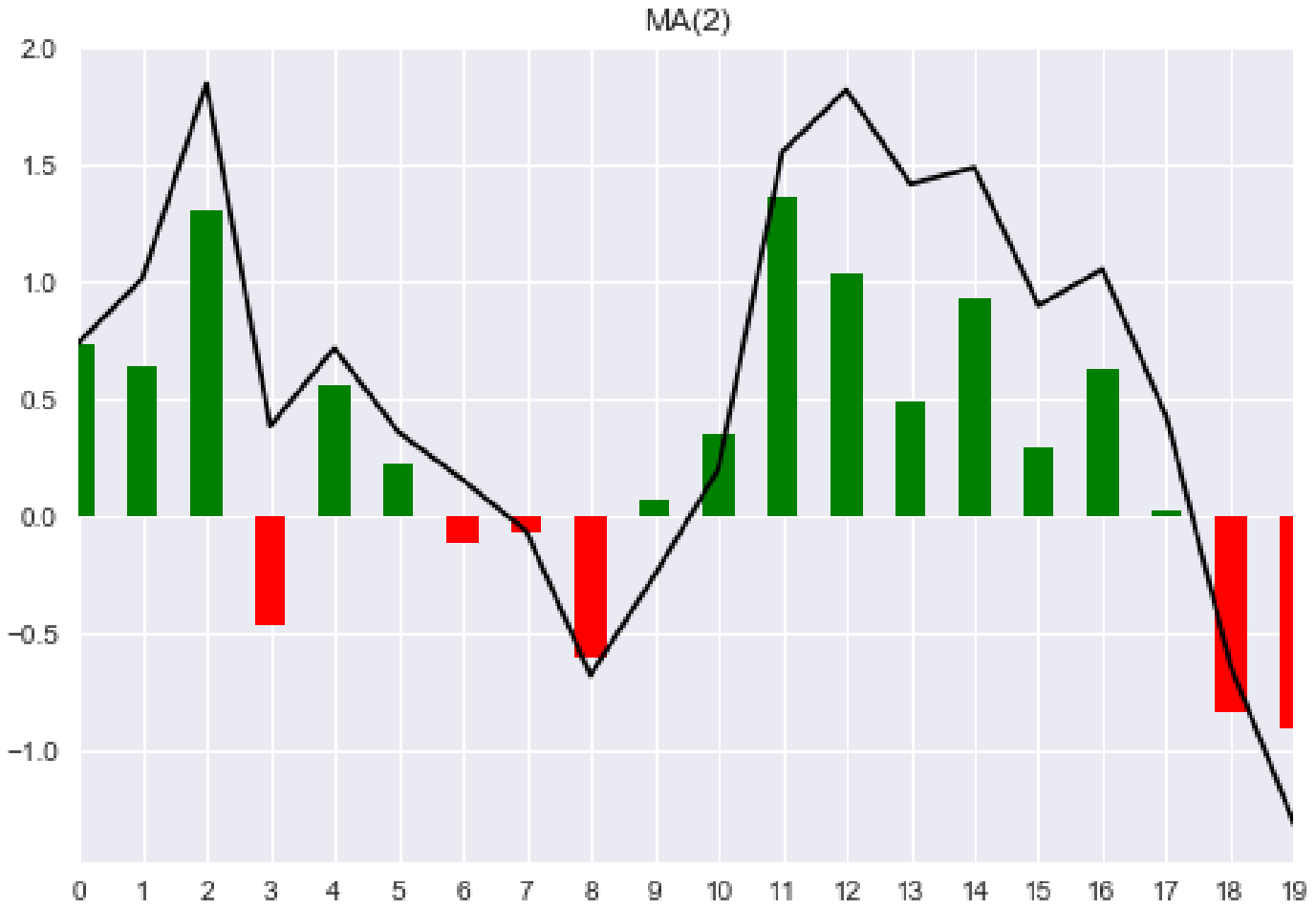
Let p = 1, and β_1 be 1/2

$$x_t = w_t + 1/2\, w_{t-1}$$

We start by generating random white noise.

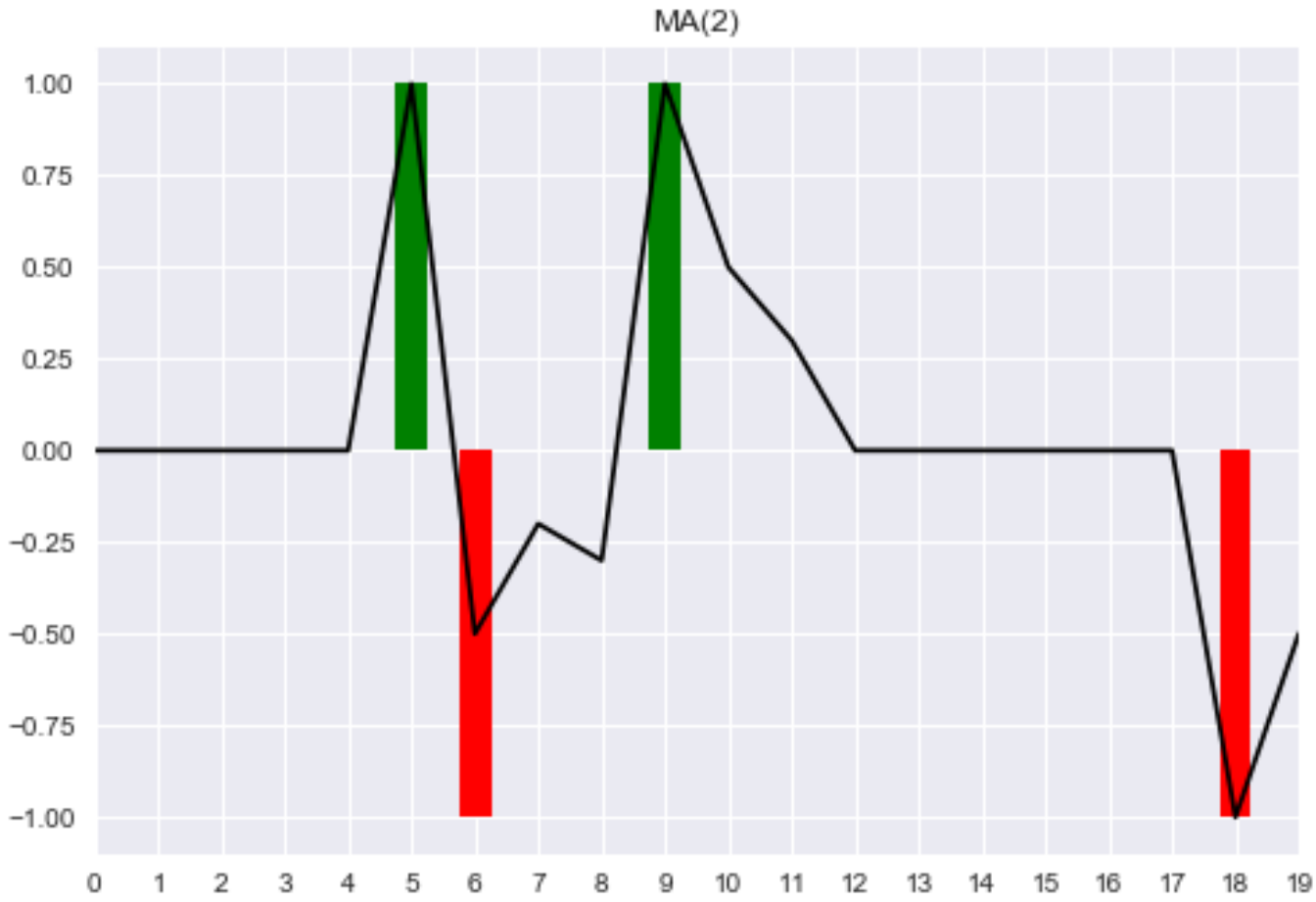
t	w_t	x_t
1	-0.2	
2	0.7	
3	-0.6	
4	0.4	
5	0.8	
6	-0.3	
7	-0.9	

Visualizing an MA(2) model



The process's memory of random 'shocks' is two time units long.

Visualizing an MA(2) model



In this random process, most shocks have value 0, some 1, some -1.

Autoregressive moving average models

ARMA: a mixed model, built from AR and MA models

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + w_t + \beta_1 w_{t-1} + \beta_2 w_{t-2} + \cdots + \beta_q w_{t-q}$$

This is a ARMA(p,q) process (p=AR order, q=MA order)

AR and MA models are special cases

Key thing: ARMA is for modeling stationary time series

ARIMA

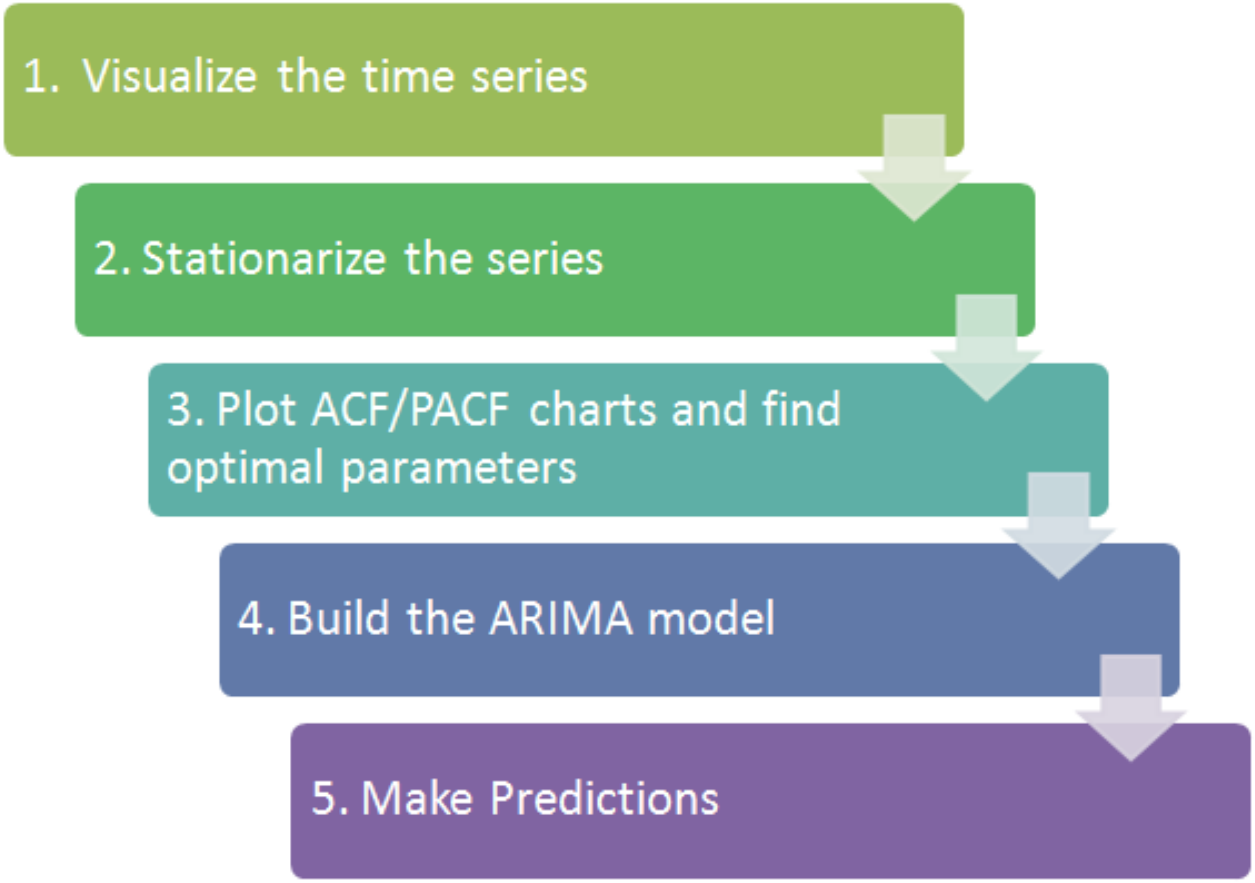
- ARIMA: Autoregressive Integrated Moving Average Models
- ARIMA = ARMA, plus differencing to handle non-stationarity
- A time series follows an ARIMA(p,d,q) process if the d^{th} differences of the series are an ARMA(p,q) process.

Key thing: ARIMA can handle non-stationary time series, as long as stationarity can be achieved by simple differencing

Seasonality can't usually be removed with simple differencing

ARIMA modeling process

Very high level view:



source: seanabu.com/2016/03/22/time-series-seasonal-ARIMA-model-in-python/

More detail on the process

Visualize the time series

Stationarize the series

Find optimal values for hyperparameters p,d,q:

- manual approach
  - difference data manually to find d
  - use ACF plots, PACF plots to find p,q
- automatic approach
  - use grid search to find p,d,q

Fit the model to your data

- this sets the coefficients of your model

Predict

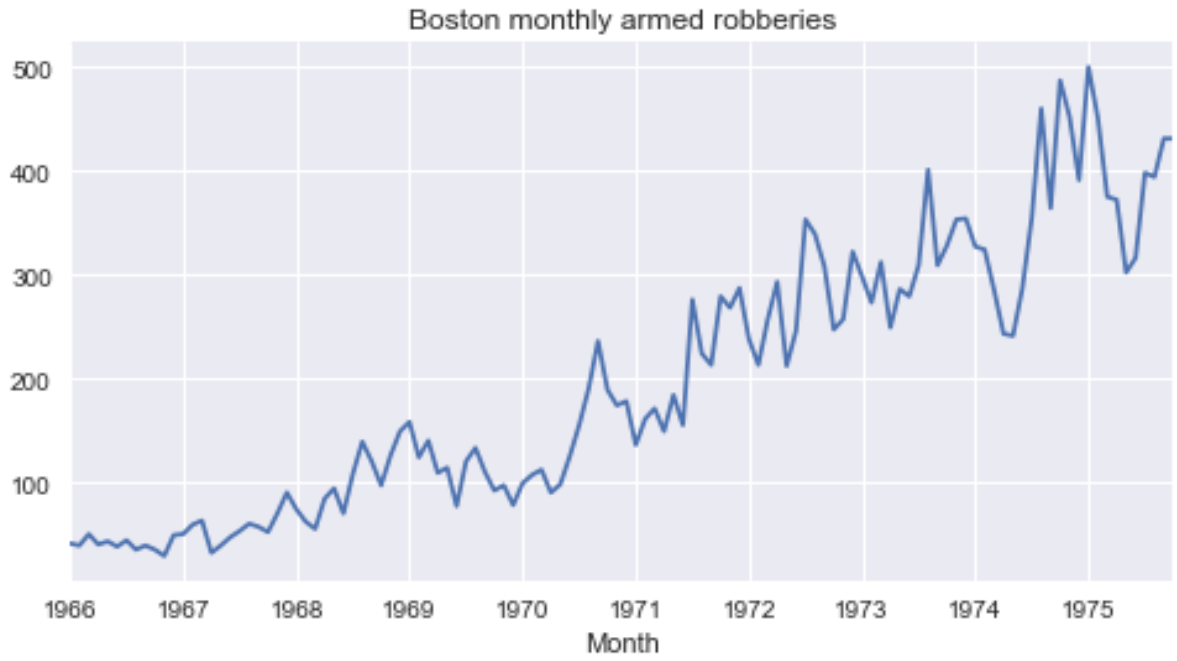
Evaluate predictions

- plot residuals
- compute RMSE or other error metric

Example: Boston armed robberies

```
series =
Series.from_csv("https://raw.githubusercontent.com/grbruns/cst495/master/monthly-boston-armed-robberies-j.csv", header=0)

series.plot()
plt.title("Boston monthly armed robberies")
```

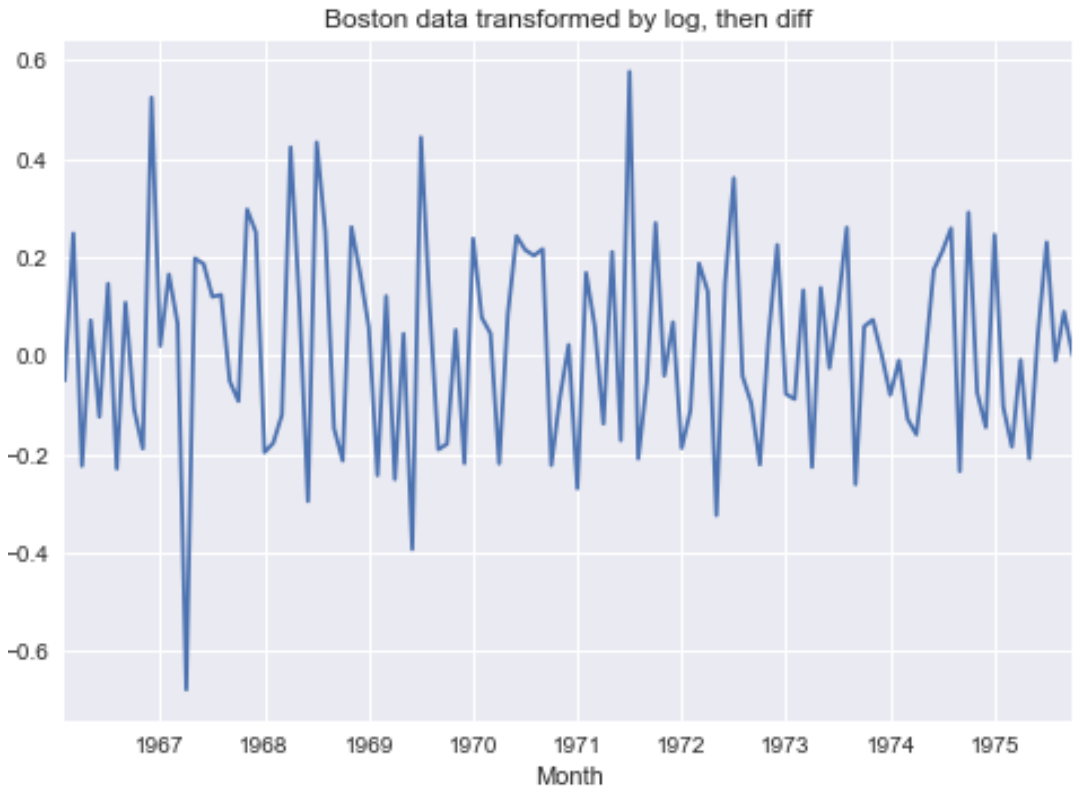


Does this look stationary?

What would you use as a simple prediction method?

Stationarize the data

```
logdiff = series.apply(np.log).diff()[1:]
logdiff.plot()
plt.title("Boston data transformed by log, then diff")
```



"Dickey-Fuller" test for stationarity:

ADF Statistic: -7.601792
p-value: 0.000000
Critical Values:
1%: -3.490
5%: -2.887
10%: -2.581

Detail on stationarity test

```
# Perform adjusted Dickey-Fuller test on differenced data
diff = series.diff()[1:]
X = diff.values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

output:

```
ADF Statistic: -7.428564
p-value: 0.000000
Critical Values:
        1%: -3.494
        5%: -2.889
        10%: -2.582
```

A value less than the 5% value means we have high confidence is stationary

Fitting an ARIMA model

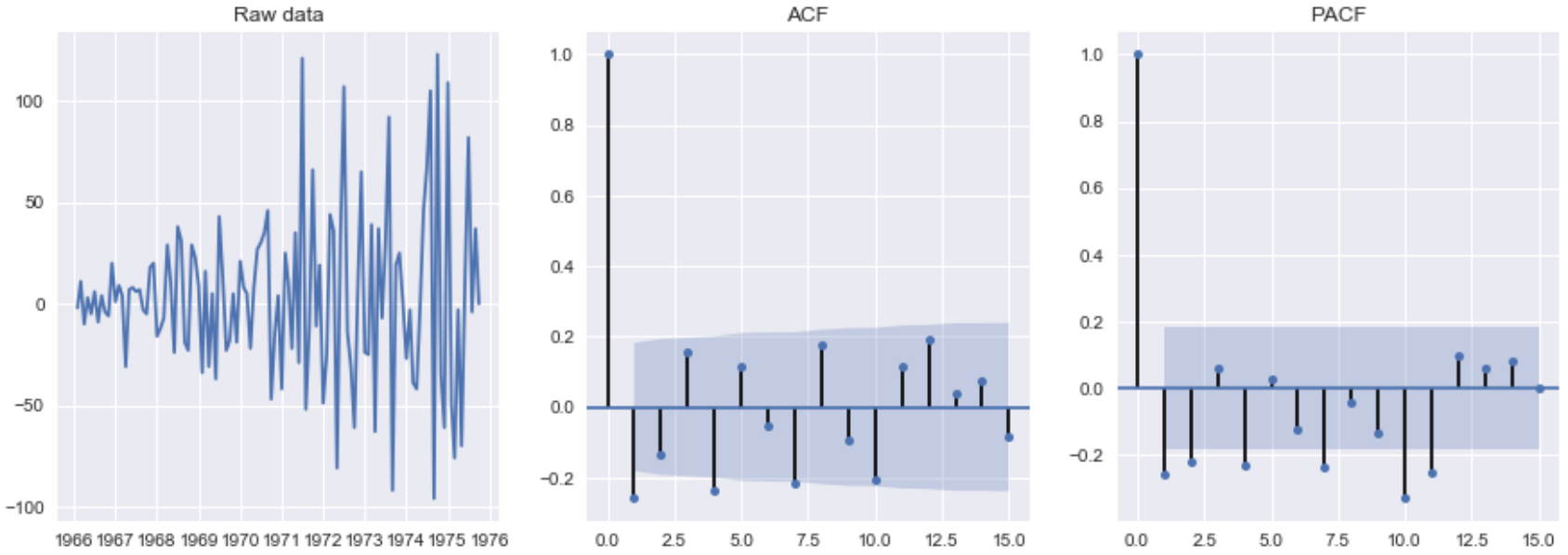
1. Determine order of the model (using ACF and PACF plots)
   - We used p=2 (AR), d=1, and q=1 (MA).
2. Use a model fitting algorithm on training data to determine the best values for the model coefficients

```
# put data into training and test sets
X = series.astype(dtype='float').values
forecast_length = 12
train_size = len(X) - forecast_length
train, test = X[0:train_size], X[train_size:]

# fit the model
model = ARIMA(train, order=(2,1,1))
model_fit = model.fit()
```

The test set contains the latest observations.
In this case, the last 12 months of the data.

ACF,PACF to determine p,q params



Diagnosis of these plots gets complicated, especially when there is not much data

An alternative is to use grid search.

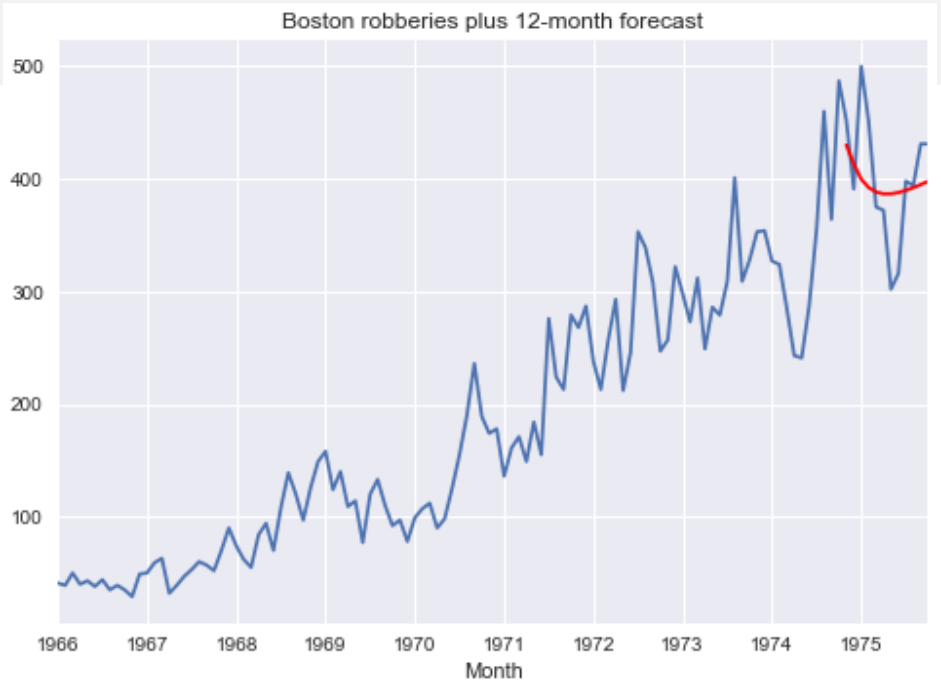
Make a 12-month forecast

```
# forecast one year into the future
predict = model_fit.forecast(steps=forecast_length)[0]

# compute the error
rmse = sqrt(mean_squared_error(test, predict))

# plot the actual data plus forecast
predicted = pd.Series(predict, index=series.index[train_size:])
series.plot()
predicted.plot(color="red")
```

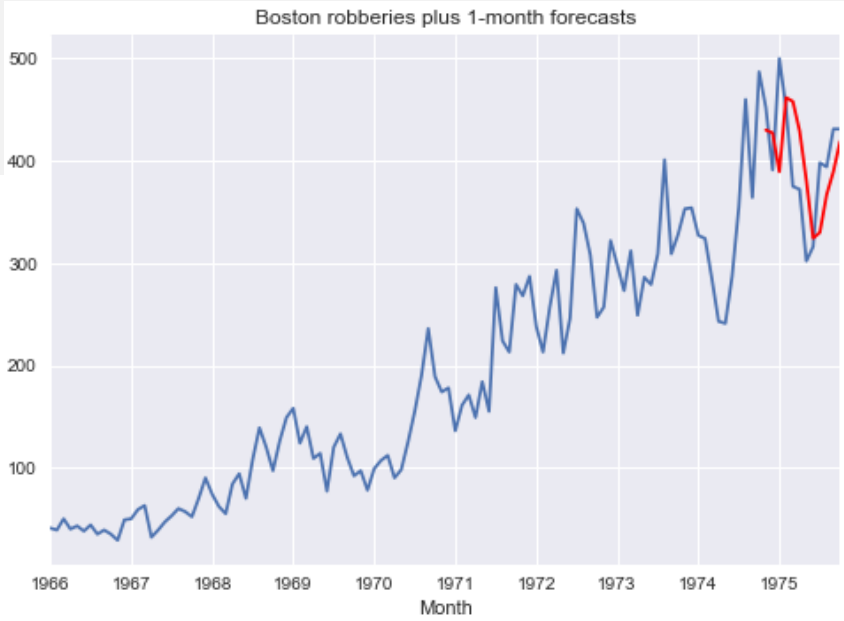
How would this compare to predictions based on a linear regression?



Make 12 single-month forecasts

```
history = [x for x in train]
predictions = list()
for i in range(len(test)):
        # 1-month prediction
        model = ARIMA(history, order=(2,1,1))
        model_fit = model.fit(disp=0)
        yhat = model_fit.forecast()[0]
        predictions.append(yhat[0])
        # actual
        obs = test[i]
        history.append(obs)
```

"walk-forward validation"



source:
machinelearningmastery.com/time-series-forecast-case-study-python-monthly-armed-robberies-boston/

Alternatives to ARIMA

- Transform data so that standard machine learning algorithms can be applied
- Neural nets (especially RNN)
- Then there are the fancier time series models, such as ARCH and GARCH

Summary

- Time series forecasting with ARIMA has been around 40+ years
- It's still used, and competitive with newer methods
- We got a basic understanding of how ARIMA works
- …and saw a process for forecasting with ARIMA

References

- Introductory Time Series with R, Cowpertwait and Metcalfe
- Time Series Analysis and Applications, Shumway and Stoffer (stat.pitt.edu/stoffer/tsa4)
- Forecasting: Principles and Practice, Hyndman and Athanasopoulos (OTexts.org/fpp2)
- How to Create an ARIMA Model for Time Series Forecasting with Python, Jason Brownlee (machinelearningmastery.com/arima-for-time-series-forecasting-with-python)