ORIE 3120: Practical Tools for OR, DS, and ML

Lecture 15: Forecasting

Professor Frazier

2022

Outline

Forecasting: overview

Constant mean model

Simple exponential smoothing

Holt-Winters

- Holt's nonseasonal model
- Winters' seasonal methods

Forecasting time series

A time series, $x_1, x_2, x_3, \ldots$ is a data sequence observed over time, for example,

- demand for parts
- sales of a product
- unemployment rate

In this segment of the course we study special methods for forecasting time series.

- the idea is to develop an algorithm to track the time series and to extrapolate into the future

Outline

Forecasting: overview

Constant mean model

Simple exponential smoothing

Holt-Winters

- Holt's nonseasonal model
- Winters' seasonal methods

Constant mean model: introduction

Suppose demand for a product follows the (very) simple model

$$x_n = a + \epsilon_n$$

Here

- x_n = demand for time period n
- a is the expected demand – which is constant in this simple model
- $\epsilon_1, \epsilon_2, \ldots$ are independent with mean 0
- the best forecast of a future value of x_n is a
- we want to estimate a and update the estimate as each new x_n is observed

Constant mean model: forecasts

- Let $\widehat{x}_n(\ell)$ be the ℓ-step ahead forecast at time period n
- Stated differently, $\widehat{x}_n(\ell)$ is the forecast at time n of demand at time $n + \ell$
- Let

$$\widehat{a}_n = \frac{x_1 + \cdots + x_n}{n}$$

- Then, in this simple model, the best forecasts at time n are

$$\widehat{x}_n(\ell) = \widehat{a}_n, \text{ for all } \ell > 0$$

Constant mean model: updating $\widehat{a}_n$

- In this simple model, a does not change, but our estimate of a does
- Here is a simple way to update $\widehat{a}_n$ to $\widehat{a}_{n+1}$

$$\begin{aligned}
\widehat{a}_{n+1} &= \frac{(x_1 + \cdots + x_n) + x_{n+1}}{n+1} \\
&= \frac{n}{n+1}\widehat{a}_n + \frac{1}{n+1}x_{n+1} \\
&= \widehat{a}_n + \frac{1}{n+1}(x_{n+1} - \widehat{a}_n)
\end{aligned}$$

Advantages of the updating formula

The simple updating formula

$$\widehat{a}_{n+1} = \widehat{a}_n + \frac{1}{n+1}(x_{n+1} - \widehat{a}_n)$$

has several advantages:

- reduced storage
  - we only store $\widehat{a}_n$
- computational speed
  - the mean need not be recomputed each time
- suggests ways to handle a "wandering" mean
  - coming soon

Outline

Forecasting: overview

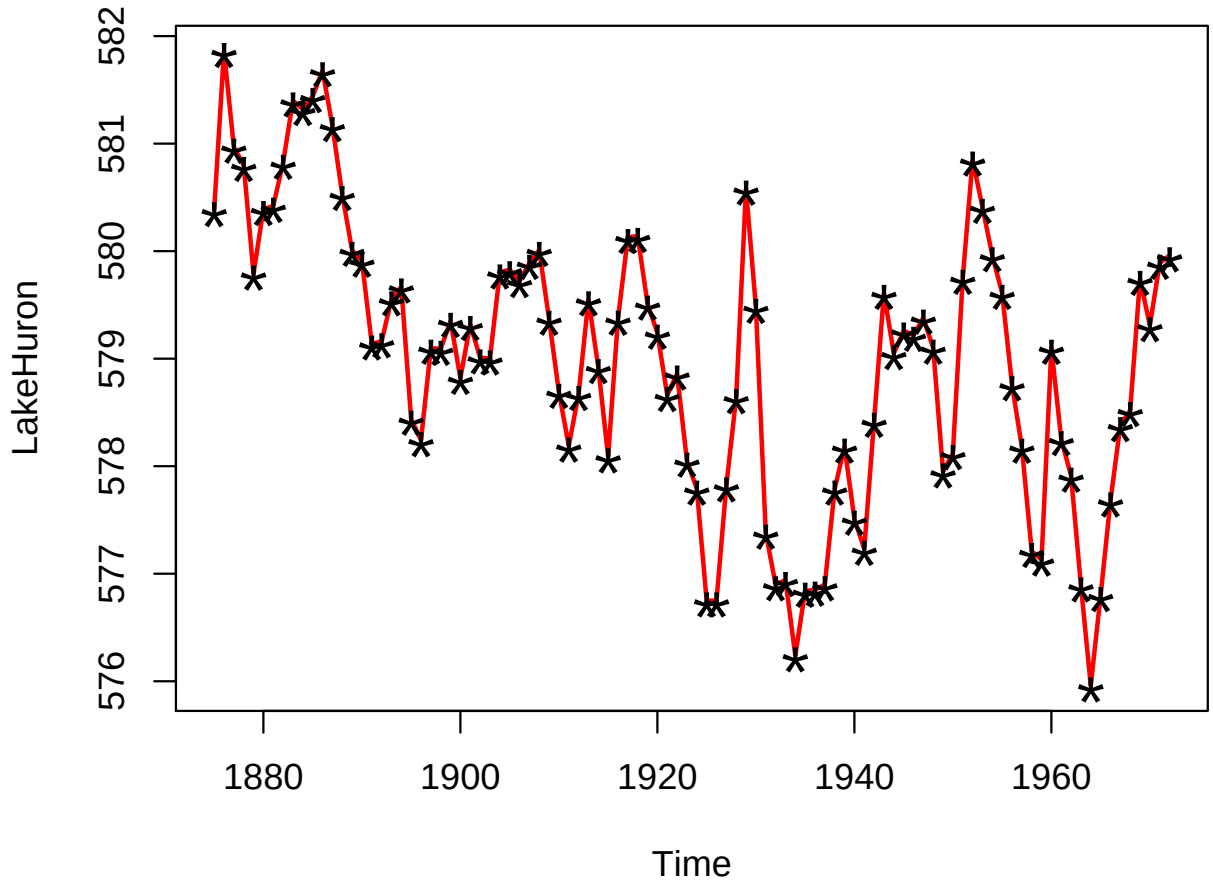
Constant mean model

Simple exponential smoothing

Holt-Winters

- Holt's nonseasonal model
- Winters' seasonal methods

Lake Huron level – example with a wandering mean



Wandering mean model: introduction

- Now suppose $x_n = a_n + \epsilon_n$ where a_n is "wandering"
- "Wandering" means $a_n = a_{n-1} + \delta_{n-1}$ where $E[\delta_{n-1}] = 0$
- The forecast is the same as for the constant mean model: $\widehat{x}_n(\ell) = \widehat{a}_n$, for all $\ell > 0$
- What changes is the way we update $\widehat{a}_n$ to track a_n

Wandering mean: updating

- For a constant mean, the update is

$$\widehat{a}_{n+1} = \widehat{a}_n + \frac{1}{n+1}(x_{n+1} - \widehat{a}_n)$$

- For a wandering mean, the update is

$$\widehat{a}_{n+1} = \widehat{a}_n + \alpha(x_{n+1} - \widehat{a}_n) = (1-\alpha)\widehat{a}_n + \alpha x_{n+1}$$

  for a constant α

- α is adjusted depending on how fast a_n is changing
  - $0 < \alpha < 1$
  - faster changes in a_n necessitate larger α
  - more noise in ϵ_n necessitates smaller α

Exponential smoothing

- We call this method "exponential smoothing" or "simple exponential smoothing"

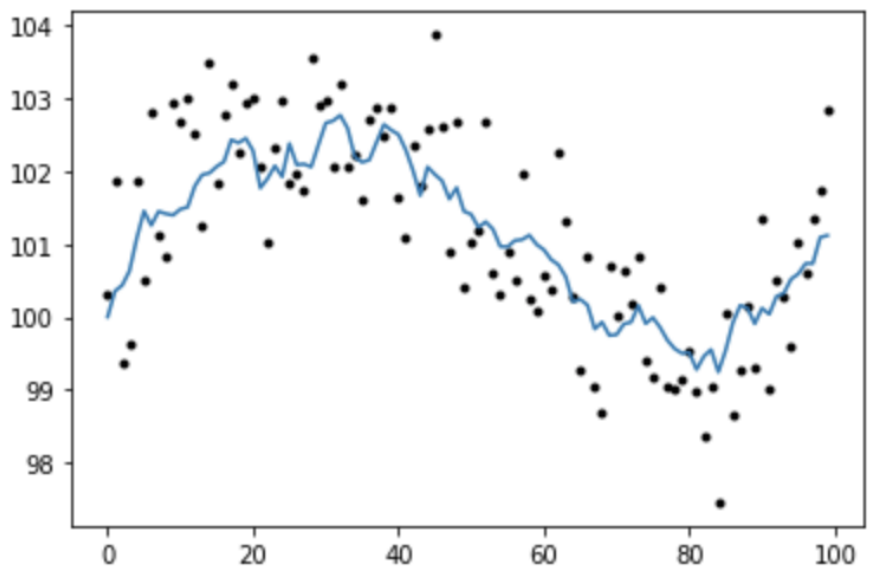
$$\begin{aligned}\widehat{a}_{n+1} &= \widehat{a}_n + \alpha(x_{n+1} - \widehat{a}_n) = (1-\alpha)\widehat{a}_n + \alpha x_{n+1} \\ \widehat{x}_n(\ell) &= \widehat{a}_n, \text{ for all } \ell > 0\end{aligned}$$

- α is chosen by the computer to minimize the sum of squared errors, as discussed later

Generating data with a wandering mean

:
```
# Generate synthetic data
np.random.seed(0)
n = 100
a  = np.zeros(n) # create an array of 0's of length n
s = .2             # controls how fast true mean wanders
a[0] = 100
for i in range(1,n):
    a[i] = a[i-1] + s*randn()
noise_std = .8
x = a + noise_std*randn(n)

plt.scatter(range(n), x, marker='.', color='black')
p = plt.plot(a)
```

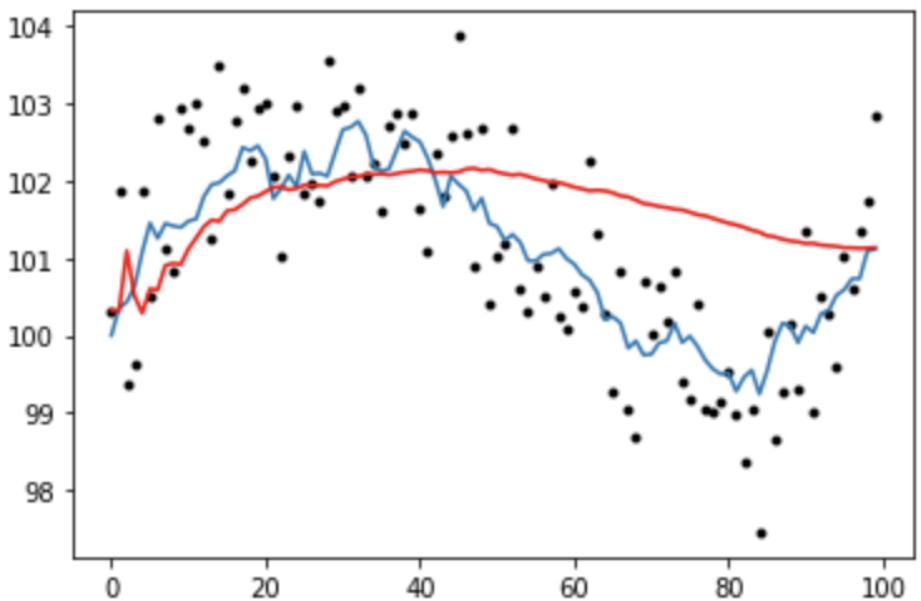


Fitting a constant mean model to this data

```
# fit a constant mean model
ahat = np.zeros(len(x))
ahat[0] = x[0]
for i in range(1,n):
    ahat[i] = ahat[i-1] + 1/i*(x[i-1] - ahat[i-1])

plt.scatter(range(n), x, marker='.', color='black')
p = plt.plot(a)
plt.plot(range(n), ahat, color='red')
```

```
[<matplotlib.lines.Line2D at 0x7fc7f0777490>]
```

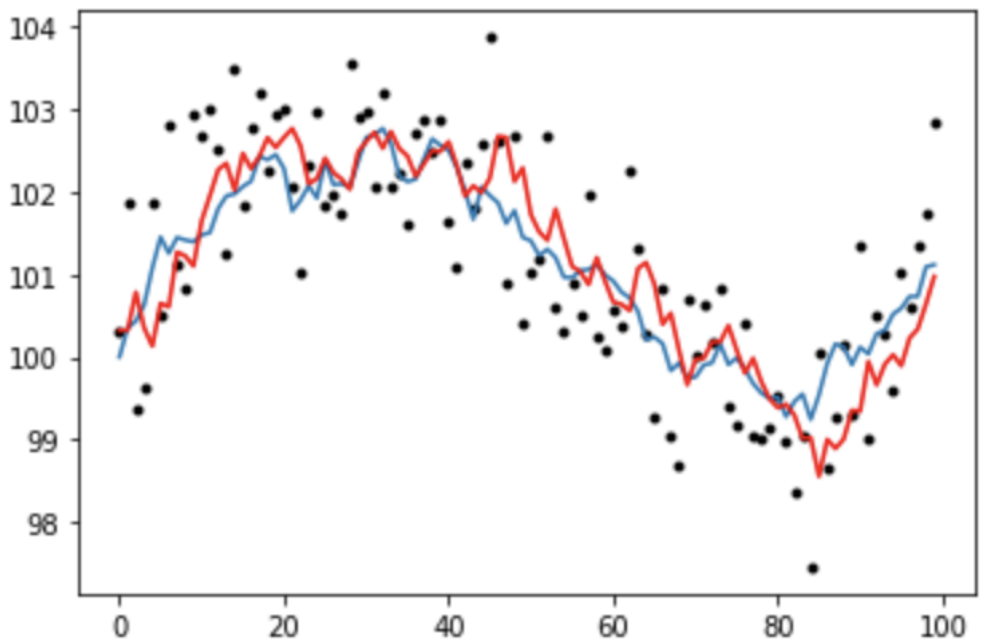


The red line is not very accurate

Fitting simple exponential smoothing to this data

```
fit = SimpleExpSmoothing(x).fit(smoothing_level=0.3, optimized=False)
xhat = fit.fittedvalues
plt.scatter(range(n), x, marker='.', color='black')
p = plt.plot(a)
plt.plot(range(n), xhat, color='red')
```

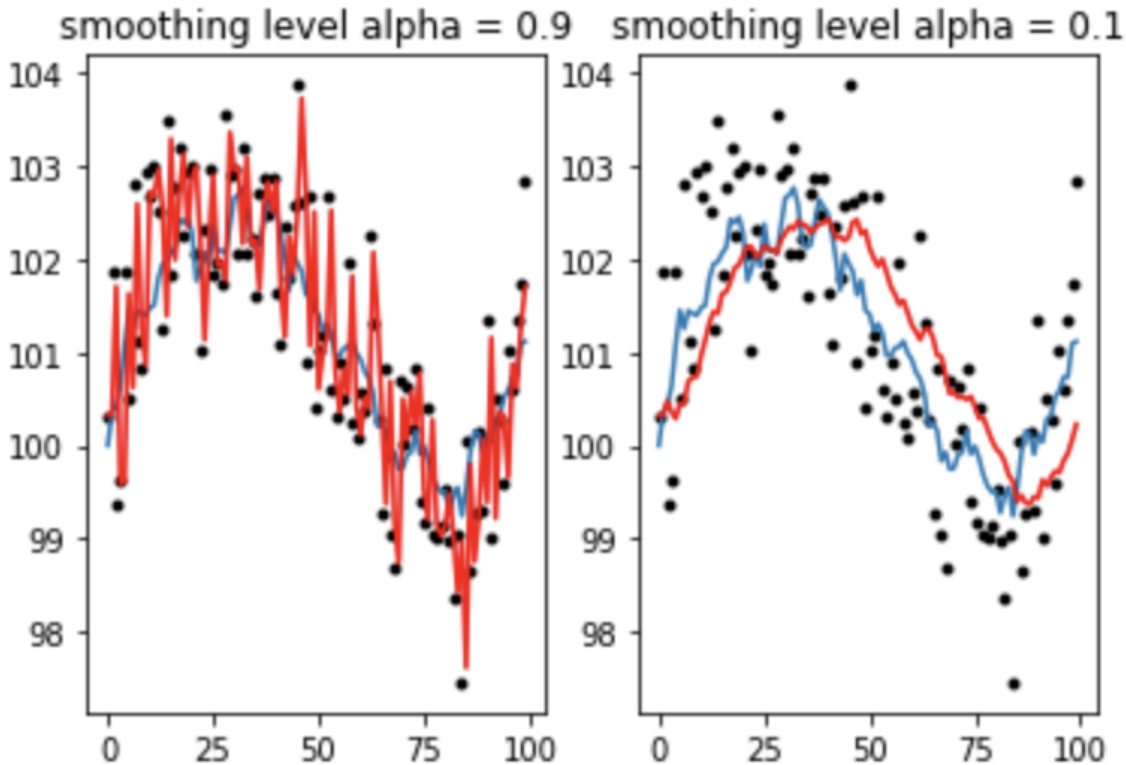
```
[<matplotlib.lines.Line2D at 0x7fc8206e4e50>]
```



The red line is much more accurate

The choice of α matters

- $\widehat{a}_{n+1} = \widehat{a}_n + \alpha(x_{n+1} - \widehat{a}_n)$.
- Bigger α reacts more quickly to the data. Follows trends more closely but is more prone to noise.
- Need to find the right value of α for each dataset



Python can choose α for you

```
fit = SimpleExpSmoothing(x).fit() # choose smoothing level to best fit data
fit.params # fit.params shows the chosen values for all parameters
```

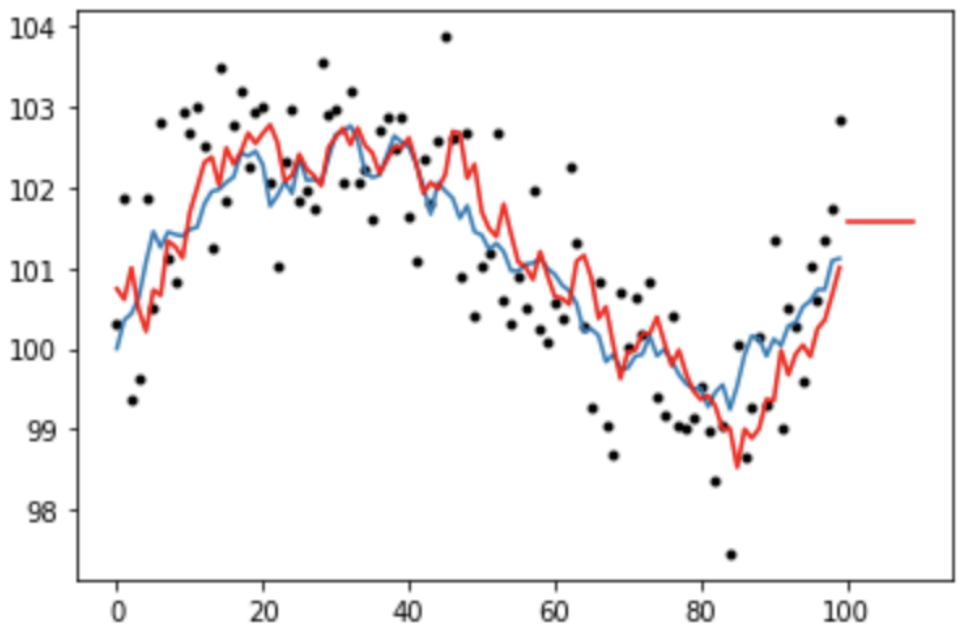
```
{'smoothing_level': 0.31305162196958597,
 'smoothing_slope': nan,
 'smoothing_seasonal': nan,
 'damping_slope': nan,
 'initial_level': 100.7417392654899,
 'initial_slope': nan,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

You can also forecast

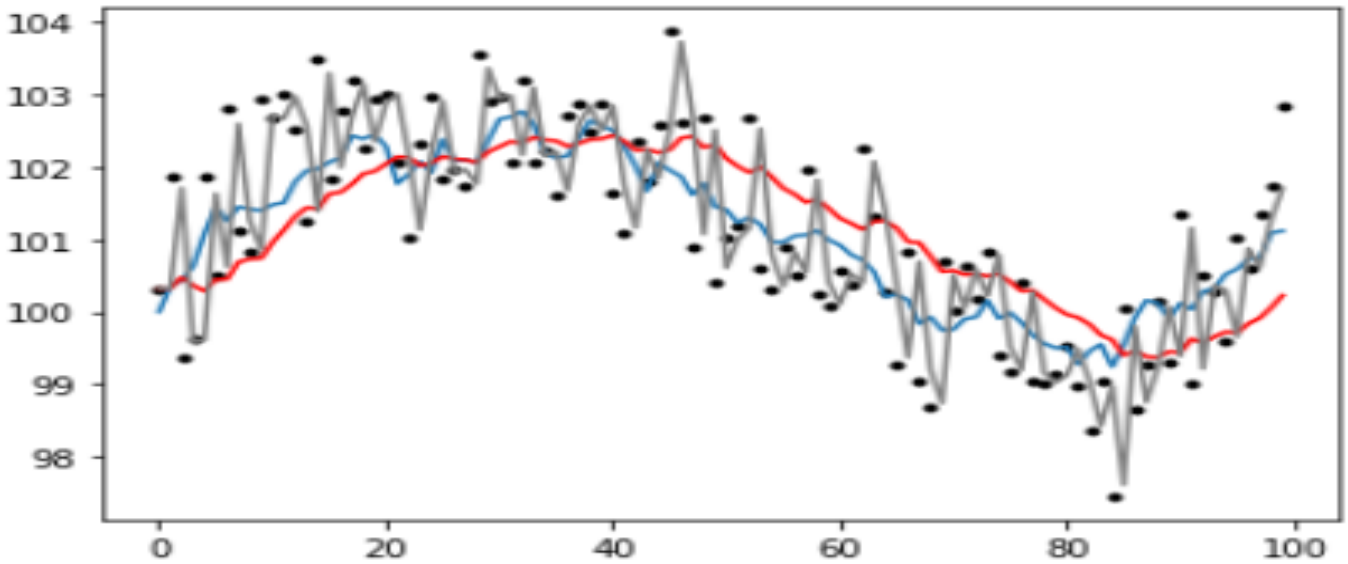
```
xhat = fit.fittedvalues
L = 10
forecast = fit.forecast(L)

# plot with forecast
plt.scatter(range(n), x, marker='.', color='black')
p = plt.plot(a)
plt.plot(range(n), xhat, color='red')
plt.plot(range(n,n+L), forecast, color='red')
```

```
[<matplotlib.lines.Line2D at 0x7fc8006d5210>]
```

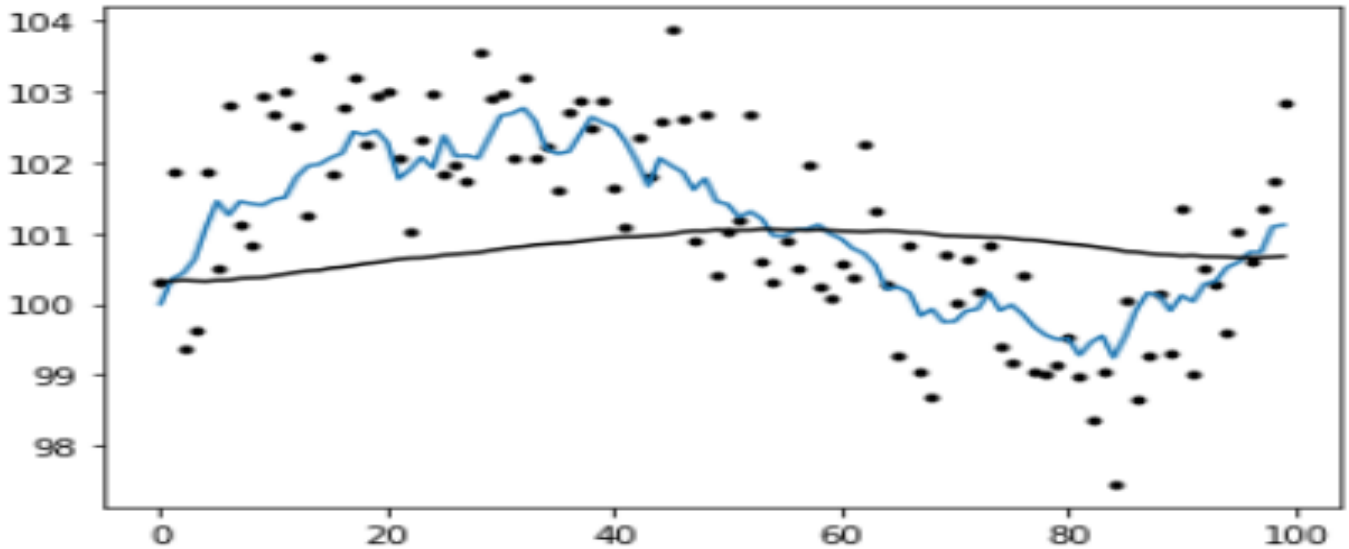


Question 1



- This plot was created using the code in previous slides. The blue line is a_n.
- Recall $\widehat{a}_{n+1} = \widehat{a}_n + \alpha(x_{n+1} - \widehat{a}_n) = (1-\alpha)\widehat{a}_n + \alpha x_{n+1}$.
- The two other lines (red and grey) were created using $\alpha = 0.1$ and $\alpha = 0.9$.
- Which line corresponds to which α?
  - A. red is $\alpha = 0.1$, grey is $\alpha = 0.9$
  - B. grey is $\alpha = 0.1$, red is $\alpha = 0.9$

Question 2



- This uses the same data and a_n as the previous question.
- Recall $\widehat{a}_{n+1} = \widehat{a}_n + \alpha(x_{n+1} - \widehat{a}_n) = (1-\alpha)\widehat{a}_n + \alpha x_{n+1}$.
- Which value of α created the black line?
  - A. $\alpha = 0$
  - B. $\alpha = 0.01$
  - C. $\alpha = 0.1$
  - D. $\alpha = 0.5$
  - E. $\alpha = 0.9$
  - F. $\alpha = 0.99$
  - G. $\alpha = 1$

Understanding exponential weights from exponential smoothing

Start with the updating equation and iterate backwards:

$$
\begin{aligned}
\widehat{a}_{n+1} &= (1-\alpha)\widehat{a}_n + \alpha x_{n+1} \\
&= (1-\alpha)\{\widehat{a}_{n-1}(1-\alpha) + \alpha x_n\} + \alpha x_{n+1} \\
&= (1-\alpha)^2\widehat{a}_{n-1} + (1-\alpha)\alpha x_n + \alpha x_{n+1} \\
&= (1-\alpha)^3\widehat{a}_{n-2} + (1-\alpha)^2\alpha x_{n-1} + (1-\alpha)\alpha x_n + \alpha x_{n+1} \\
&\approx \alpha\Big\{x_{n+1} + (1-\alpha)x_n + (1-\alpha)^2 x_{n-1} \\
&\qquad +(1-\alpha)^3 x_{n-2} + \cdots + (1-\alpha)^n x_1\Big\}
\end{aligned}
$$

Exponential weighted moving average

Use previous page + summation formula for geometric series (see next page):

$$\widehat{a}_{n+1} \approx$$

$$\alpha\left\{(1-\alpha)^0 x_{n+1} + (1-\alpha)^1 x_n + (1-\alpha)^2 x_{n-1} + \cdots + (1-\alpha)^n x_1\right\}$$

$$\approx \frac{\left\{(1-\alpha)^0 x_{n+1} + (1-\alpha)^1 x_n + (1-\alpha)^2 x_{n-1} + \cdots + (1-\alpha)^n x_1\right\}}{1 + (1-\alpha) + \cdots + (1-\alpha)^n}$$

Hence $\widehat{a}_{n+1}$ is an exponentially weighted moving average

Large values of α mean faster discounting of the past values.

Summing a geometric series

Assume $|\gamma| < 1$ so $\gamma^n \to 0$ as $n \to \infty$

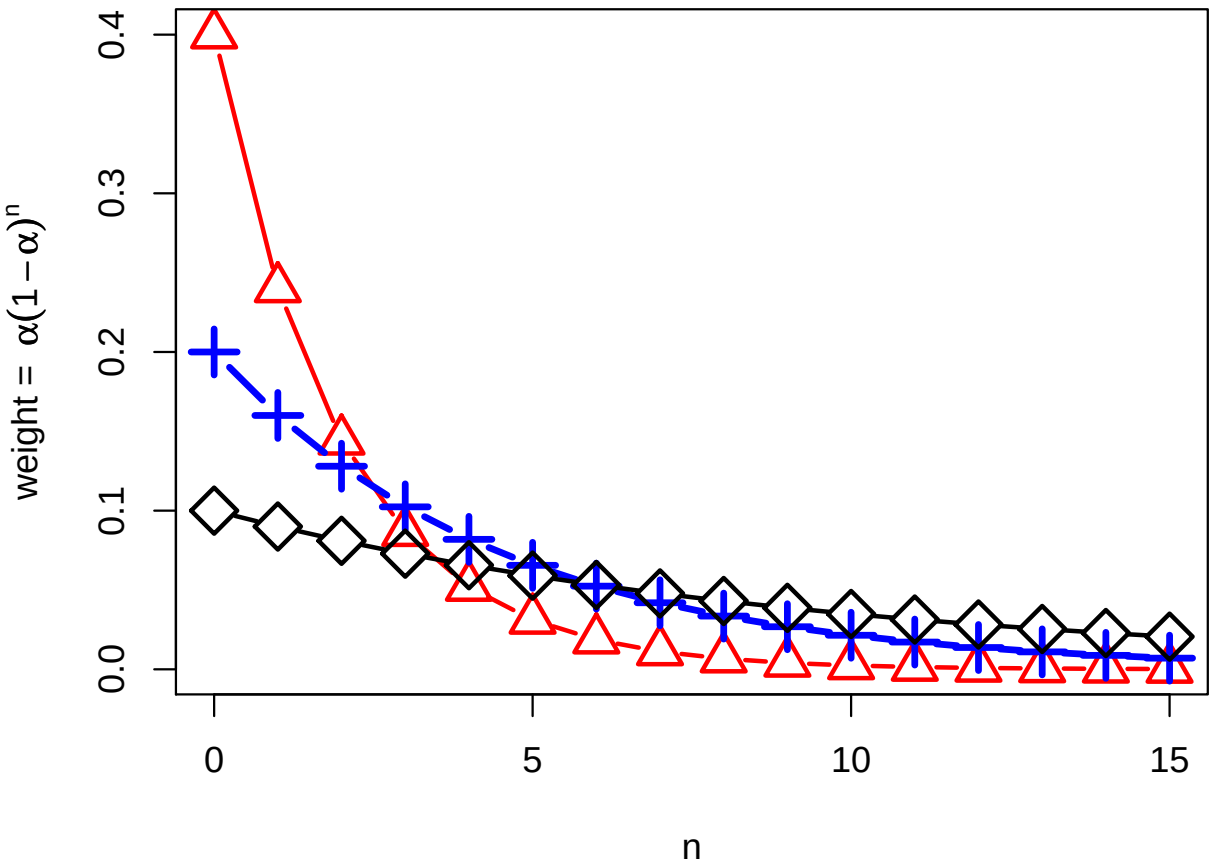
$$1 + \gamma + \gamma^2 + \cdots + \gamma^n = \frac{1 - \gamma^{n+1}}{1 - \gamma} \approx \frac{1}{1 - \gamma} \text{ (if } n \text{ is large enough)}$$

Now let $\gamma = 1 - \alpha$. Then

$$1 + (1 - \alpha) + \cdots + (1 - \alpha)^n \approx \frac{1}{\alpha}$$

since $1 - (1 - \alpha) = \alpha$.

Exponential weights: examples



Note: weights start at α when $n = 0$

Outline

Forecasting: overview

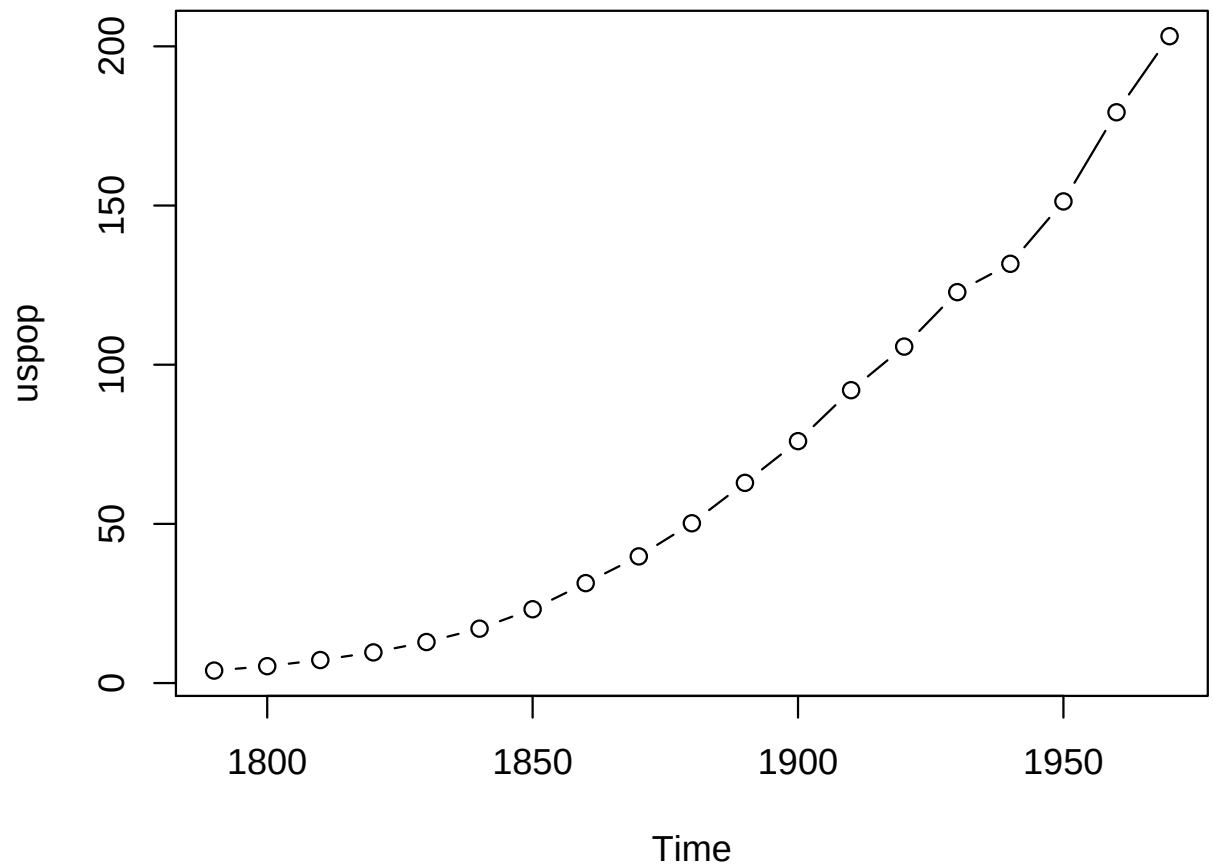
Constant mean model

Simple exponential smoothing

Holt-Winters

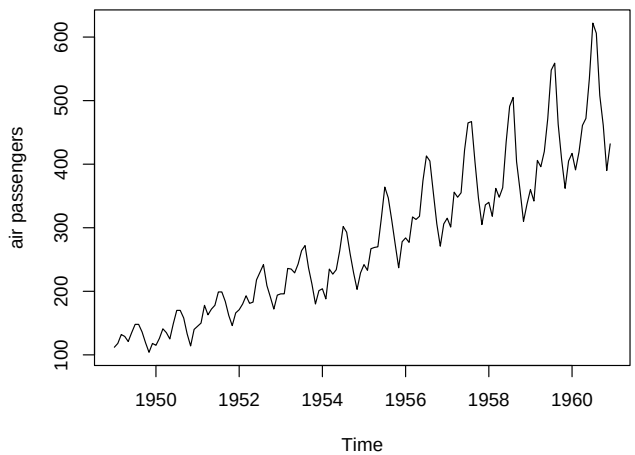
- Holt's nonseasonal model
- Winters' seasonal methods

Forecasting with trends: example



Census counts of US population

Forecasting with trends and seasonality: example



Airline passenger miles

Note the seasonal pattern and trend in this example

- these are typical of much business data

Holt method: forecasting with trend

For now, assume data has trend but no seasonality

Holt's forecasting method uses a linear trend

$$\text{estimate at time } n \text{ of } x_{n+\ell} := \widehat{x}_n(\ell) = \widehat{a}_n + \widehat{b}_n \ell$$

- n is "origin" – the point in time when forecasts are being made
- ℓ is the "lead" – how far ahead one is forecasting
- $\widehat{a}_n$ is called the level
- $\widehat{b}_n$ is called the slope

Both $\widehat{a}_n$ and $\widehat{b}_n$ are updated as we make more observations n

Holt method: Updating the level

In the Holt model, the level $\widehat{a}_n$ is updated by the equation:

$$\widehat{a}_{n+1} = (1-\alpha)(\widehat{a}_n + \widehat{b}_n) + \alpha x_{n+1}$$

or, equivalently,

$$\widehat{a}_{n+1} = \widehat{a}_n + (1-\alpha)\widehat{b}_n + \alpha(x_{n+1} - \widehat{a}_n)$$

- $\widehat{a}_n + \widehat{b}_n$ is predicted value at time $n+1$ = lead time 1
- α is for updating the level and β for the slope (next)

Compare with previous update equation (for no-trend model):

$$\widehat{a}_{n+1} = \widehat{a}_n + \alpha(x_{n+1} - \widehat{a}_n) = (1-\alpha)\widehat{a}_n + \alpha x_{n+1}$$

Holt model: updating the slope

In the Holt model, the slope $\widehat{b}_n$ is updated by the equation:

$$\widehat{b}_{n+1} = (1-\beta)\widehat{b}_n + \beta(\widehat{a}_{n+1} - \widehat{a}_n)$$

or, equivalently,

$$\widehat{b}_{n+1} = \widehat{b}_n + \beta\left\{(\widehat{a}_{n+1} - \widehat{a}_n) - \widehat{b}_n\right\}$$

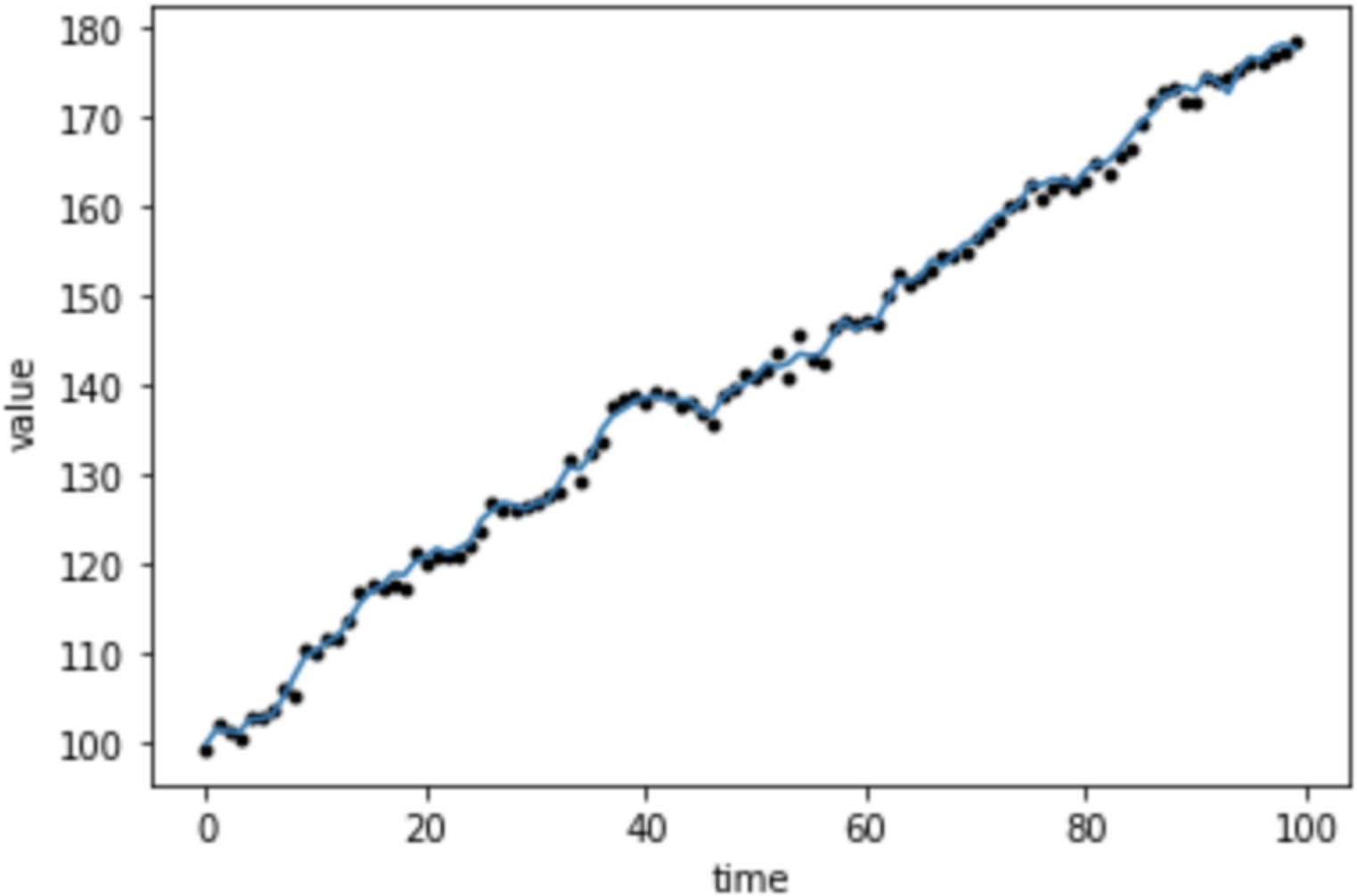
Demo: Holt's method

```
n = 100
a = np.zeros(n)
b = np.zeros(n)
a[0] = 100 # initial level
b[0] = 1 # initial trend

for i in range(1,n):
    # trend wanders
    b[i] = b[i-1] + 0.1*randn()
    # level moves with the trend + extra random movement
    a[i] = a[i-1] + b[i-1] + randn()

# We observe the level + extra noise
x = a + 0.8*randn(n)
```

Demo: Holt's method

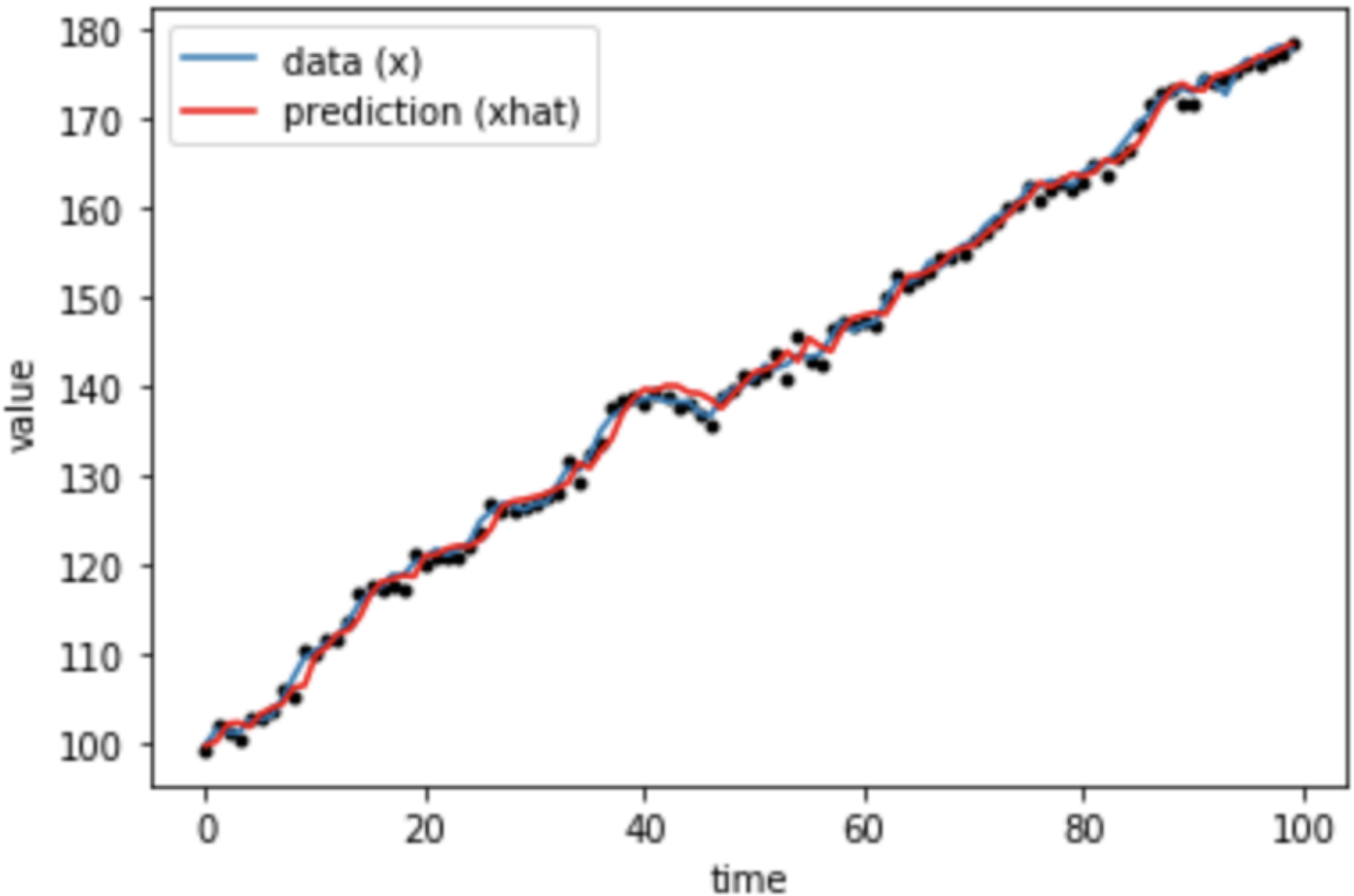


Demo: Holt's method

```
fit = Holt(x).fit()
xhat = fit.fittedvalues

plt.scatter(range(n), x, marker='.', color='black')
p = plt.plot(a)
plt.plot(range(n), xhat, color='red')
```

Demo: Holt's method



Demo: Holt's method

```
fit.params
```

```
{'smoothing_level': 0.6578170989208568,
 'smoothing_slope': 0.09349560964724064,
 'smoothing_seasonal': nan,
 'damping_slope': nan,
 'initial_level': 98.0036285349069,
 'initial_slope': 1.235132049511102,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

Lec15, Question 3

Suppose I change the code to set initial value of trend b[0] = -1. What will happen?

- A. The blue line will trend downward
- B. The blue line will trend upward more quickly
- C. The blue line will stay at 0
- D. The blue line will bounce around more
- E. The black dots will be further away from the blue line.

```
a[0] = 100 # initial level
b[0] = 1 # initial trend
for i in range(1,n):
    b[i] = b[i-1] + 0.1*randn()
    a[i] = a[i-1] + b[i-1] + randn()

x = a + 0.8*randn(n)
```

Lec15, Question 4

Suppose I change 0.8 to 5 in defining the observations x = a + 0.8*randn(n). What will happen?

- A. The blue line will trend downward
- B. The blue line will trend upward more quickly
- C. The blue line will stay at 0
- D. The blue line will bounce around more
- E. The black dots will be further away from the blue line.

```
a[0] = 100 # initial level
b[0] = 1 # initial trend
for i in range(1,n):
    b[i] = b[i-1] + 0.1*randn()
    a[i] = a[i-1] + b[i-1] + randn()

x = a + 0.8*randn(n)
```

Lec15, Question 5

Suppose I multiply randn() by 3 in defining the level a[i] = a[i-1] + b[i-1] + randn(). What will happen?

- A. The blue line will trend downward
- B. The blue line will trend upward more quickly
- C. The blue line will stay at 0
- D. The blue line will bounce around more
- E. The black dots will be further away from the blue line.

```
a[0] = 100 # initial level
b[0] = 1 # initial trend
for i in range(1,n):
    b[i] = b[i-1] + 0.1*randn()
    a[i] = a[i-1] + b[i-1] + randn()

x = a + 0.8*randn(n)
```

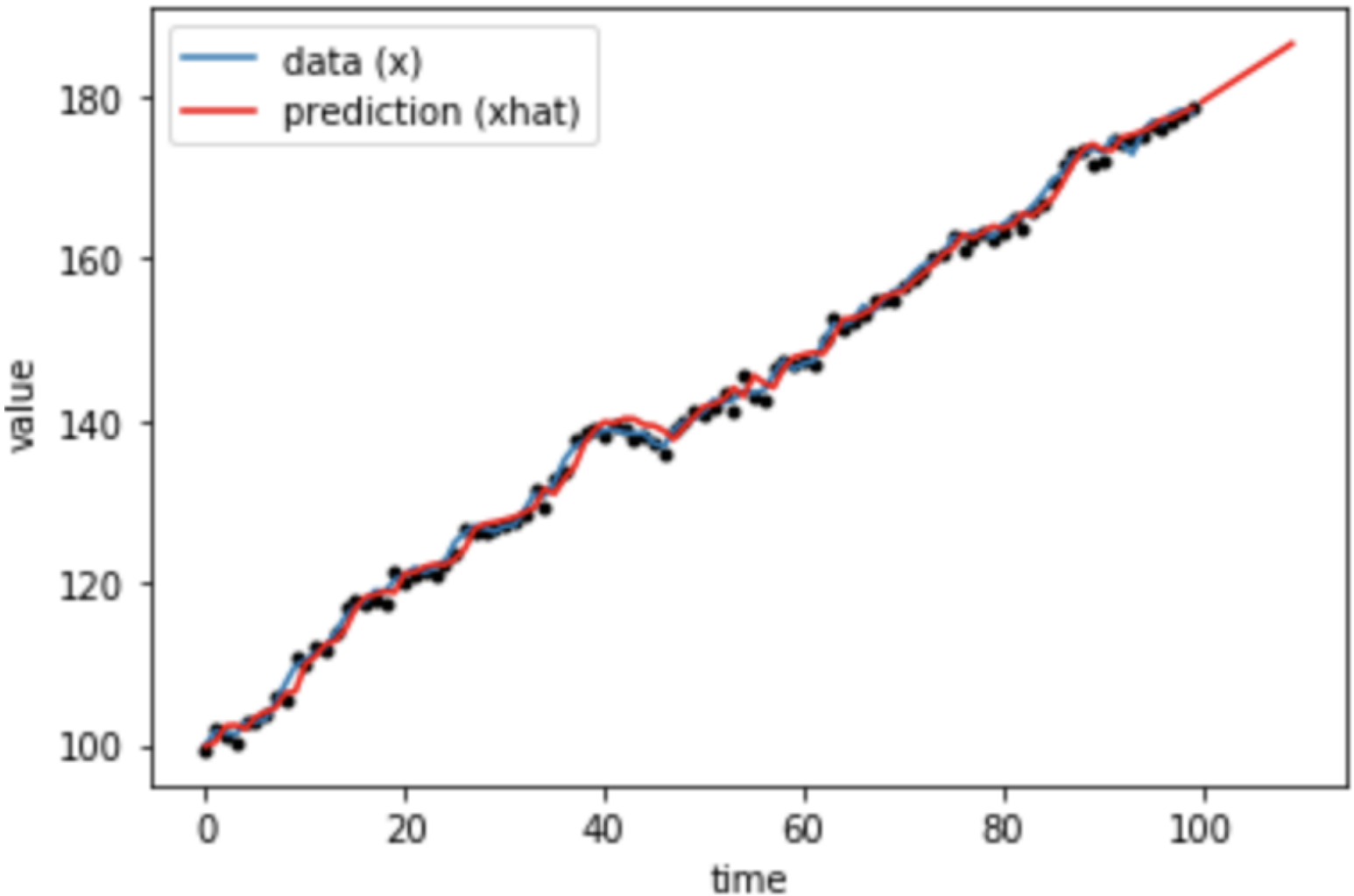
Plotting a Forecast

```
fit = Holt(x).fit()
xhat = fit.fittedvalues

xhat = fit.fittedvalues
l = 10 # how far out to forecast
forecast = fit.forecast(l)

# plot with forecast
plt.scatter(range(n), x, marker='.', color='black')
p = plt.plot(a)
plt.plot(range(n), xhat, color='red')
plt.plot(range(n,n+l), forecast, color='red')
```

Plotting a Forecast



Winters' additive seasonal method

Winters extended Holt's method to include seasonality. The method is usually called Holt-Winters forecasting

Let s be the period length:

- $s = 4$ for quarterly data
- $s = 12$ for monthly data
- $s = 52$ for weekly data
- $s = 13$ for data collected over 4-week periods
- $s = 24$ for hourly data

Holt-Winters updating

Holt-Winters forecasting can use either of two types of updating

- additive
- multiplicative

These refer to how the trend and seasonal components are put together

- the trend and seasonal components can be added or multiplied

Holt-Winters additive seasonal method

The forecasts are periodic

With the additive methods they are:

$$\begin{aligned} \widehat{x}_n(\ell) &= \widehat{a}_n + \widehat{b}_n\ell + \hat{S}_{n+\ell-s}, \text{ for } \ell = 1, 2, \ldots, s \\ &= \widehat{a}_n + \widehat{b}_n\ell + \hat{S}_{n+\ell-2s}, \text{ for } \ell = s+1, \ldots, 2s \end{aligned}$$

and so forth

Winters' additive seasonal model: updating

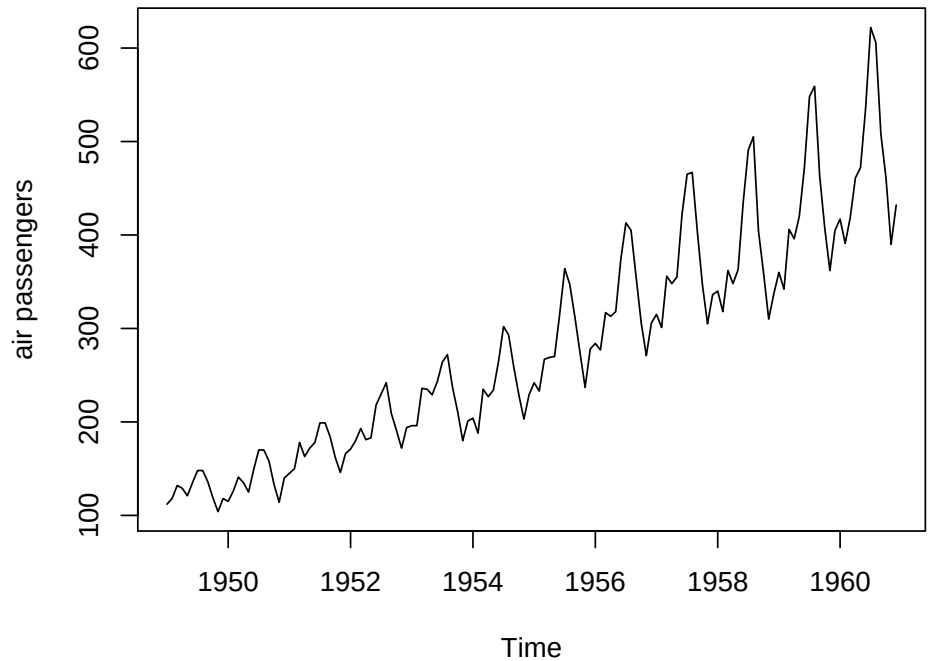
$$
\begin{aligned}
\widehat{a}_{n+1} &= \alpha(x_{n+1} - \hat{S}_{n+1-s}) + (1-\alpha)(\widehat{a}_n + \widehat{b}_n) \\
\widehat{b}_{n+1} &= \beta(\widehat{a}_{n+1} - \widehat{a}_n) + (1-\beta)\widehat{b}_n \\
\hat{S}_{n+1} &= \gamma(x_{n+1} - \widehat{a}_{n+1}) + (1-\gamma)\hat{S}_{n+1-s}
\end{aligned}
$$

α, β, and γ are "tuning parameters" that we need to adjust

Question 6

- x_n is the number of hours of sleep you got on day n
- You fit a Holt-Winters additive seasonal model
- What is the best value for s?
  - A. $s = 1$
  - B. $s = 2$
  - C. $s = 7$
  - D. $s = 14$
  - E. $s = 24$
  - F. $s = 30$
  - G. $s = 365$

Why we need multiplicative seasonal models



Notice the multiplicative behavior

- the seasonal fluctuations are larger where the trend is larger

Holt-Winters multiplicative seasonal method

$$\begin{aligned}\widehat{x}_n(\ell) &= (a_n + \widehat{b}_n\ell)\hat{S}_{n+\ell-s}, \text{ for } \ell = 1, 2, \ldots, s \\ &= (a_n + \widehat{b}_n\ell)\hat{S}_{n+\ell-2s}, \text{ for } \ell = s+1, \ldots, 2s\end{aligned}$$

and so forth

Winters' multiplicative seasonal model: updating

$$
\begin{aligned}
\widehat{a}_{n+1} &= \alpha \frac{x_{n+1}}{\hat{S}_{n+1-s}} + (1-\alpha)(\widehat{a}_n + \widehat{b}_n) \\
\widehat{b}_{n+1} &= \beta(\widehat{a}_{n+1} - \widehat{a}_n) + (1-\beta)\widehat{b}_n \\
\hat{S}_{n+1} &= \gamma \frac{x_{n+1}}{\widehat{a}_{n+1}} + (1-\gamma)\hat{S}_{n+1-s}
\end{aligned}
$$

Demo: Exponential smoothing in Python

Demo: forecasting.ipynb

Applications:

- Lake Huron
- US population
- CO2
- Airline passengers
- Sales

Outline

Selecting the tuning parameters

Residuals

For given values of α, β, and γ:

- $\widehat{a}_n, \widehat{b}_n, \hat{S}_n, \ldots, \hat{S}_{n-s}$ are the level, slope, and seasonalities at time n
- $\widehat{x}_{n+1} = x_n(1) = \widehat{a}_n + \widehat{b}_n + \hat{S}_{n+1-s}$ is the one-step ahead forecast at time n
- $\widehat{\epsilon}_{n+1} = x_{n+1} - \widehat{x}_{n+1}$ is the residual or one-step ahead forecast error

Choosing α, β, and γ

α, β, and γ are called "tuning parameters"

Suppose we have data $x_1, \ldots, x_N$:

- the usual way to select α, β, and γ is to minimize

$$\mathrm{SS}(\alpha, \beta, \gamma) = \sum_{n=N_1+1}^{N} \widehat{\epsilon}_n^2$$

  where the first N_1 residuals are discarded to let the forecasting method "burn-in"
- this technique is used by statsmodels, unless the user specifies the parameters explicitly in the `fit` call

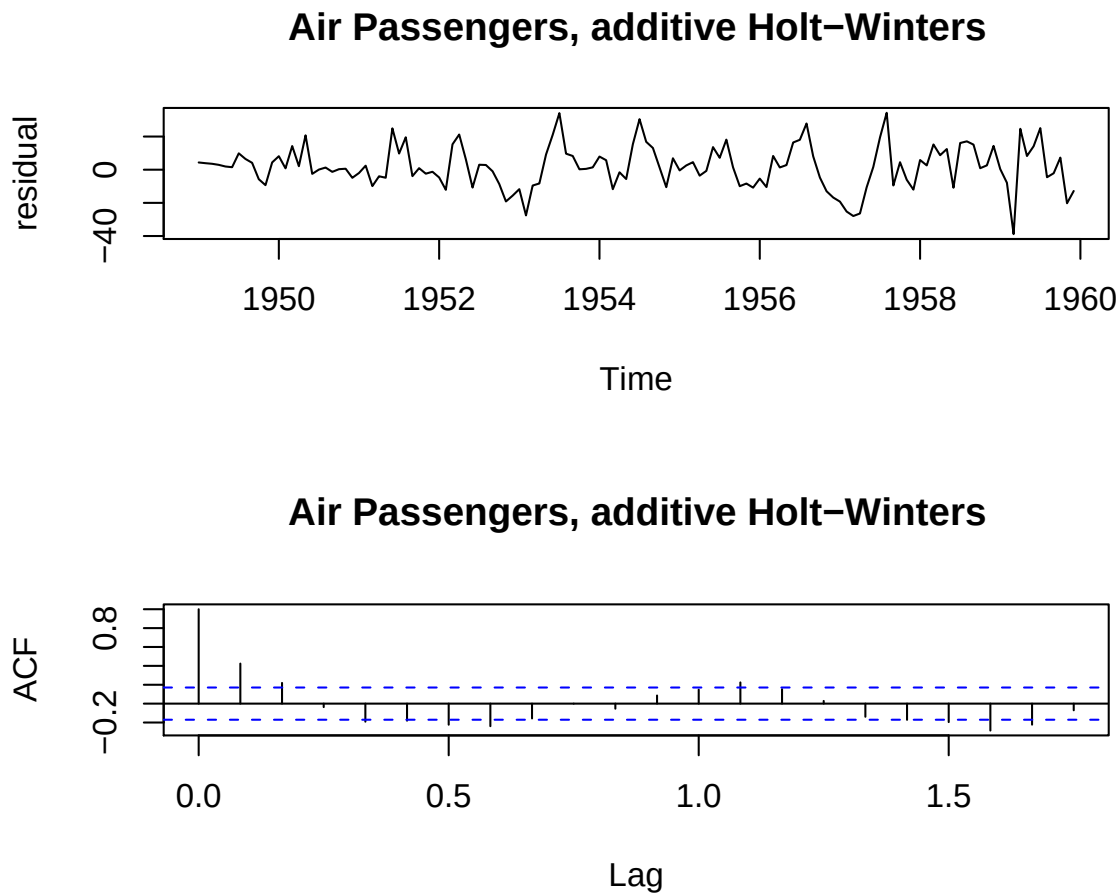
Comparing forecasting methods and diagnosing problem

- Two or more forecasting methods can be compared using

$$\min_{\alpha,\beta,\gamma} \text{SS}(\alpha, \beta, \gamma)$$

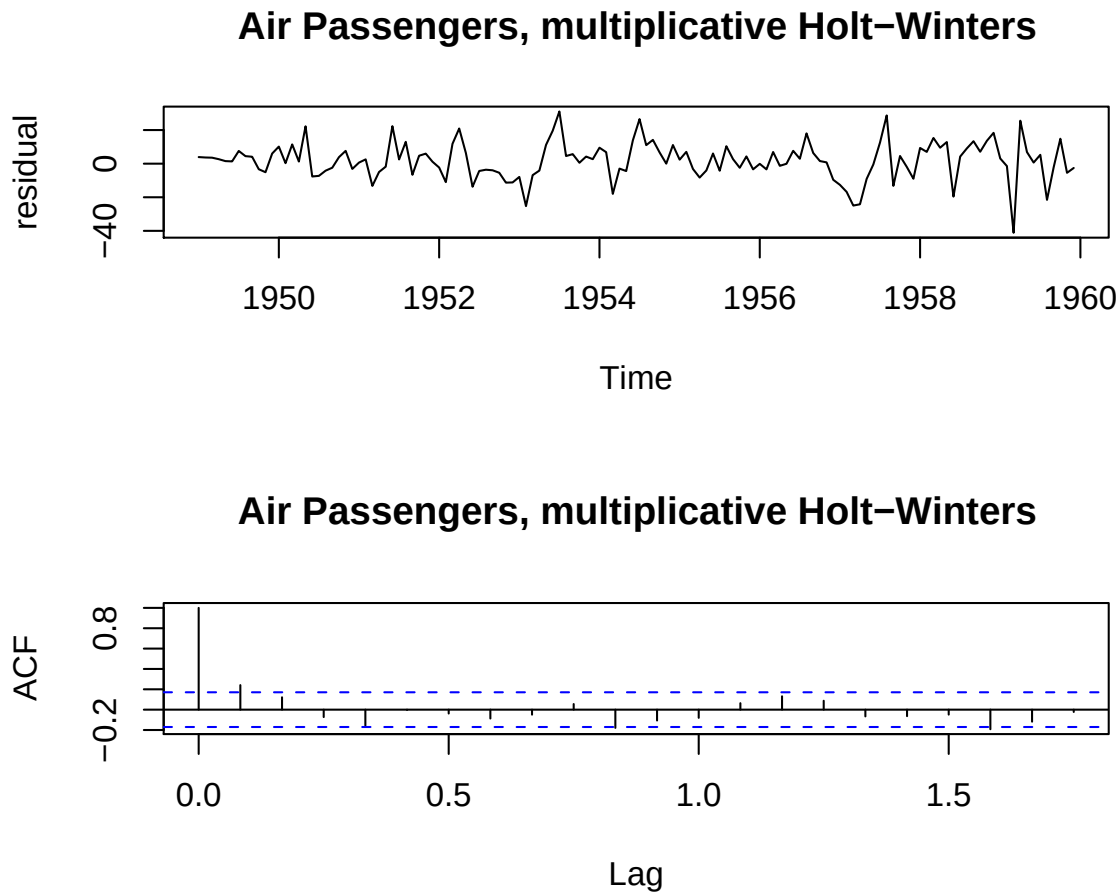
- If a forecasting method is working well, then the residuals should not exhibit autocorrelation

Air passengers: additive seasonal method



Notice some autocorrelation at small lags.

Air passengers: multiplicative seasonal method



Less autocorrelation than with additive model (good).