Exponential smoothing models

and the forecast package

Eli Holmes

7 February 2017

Load some data to play with

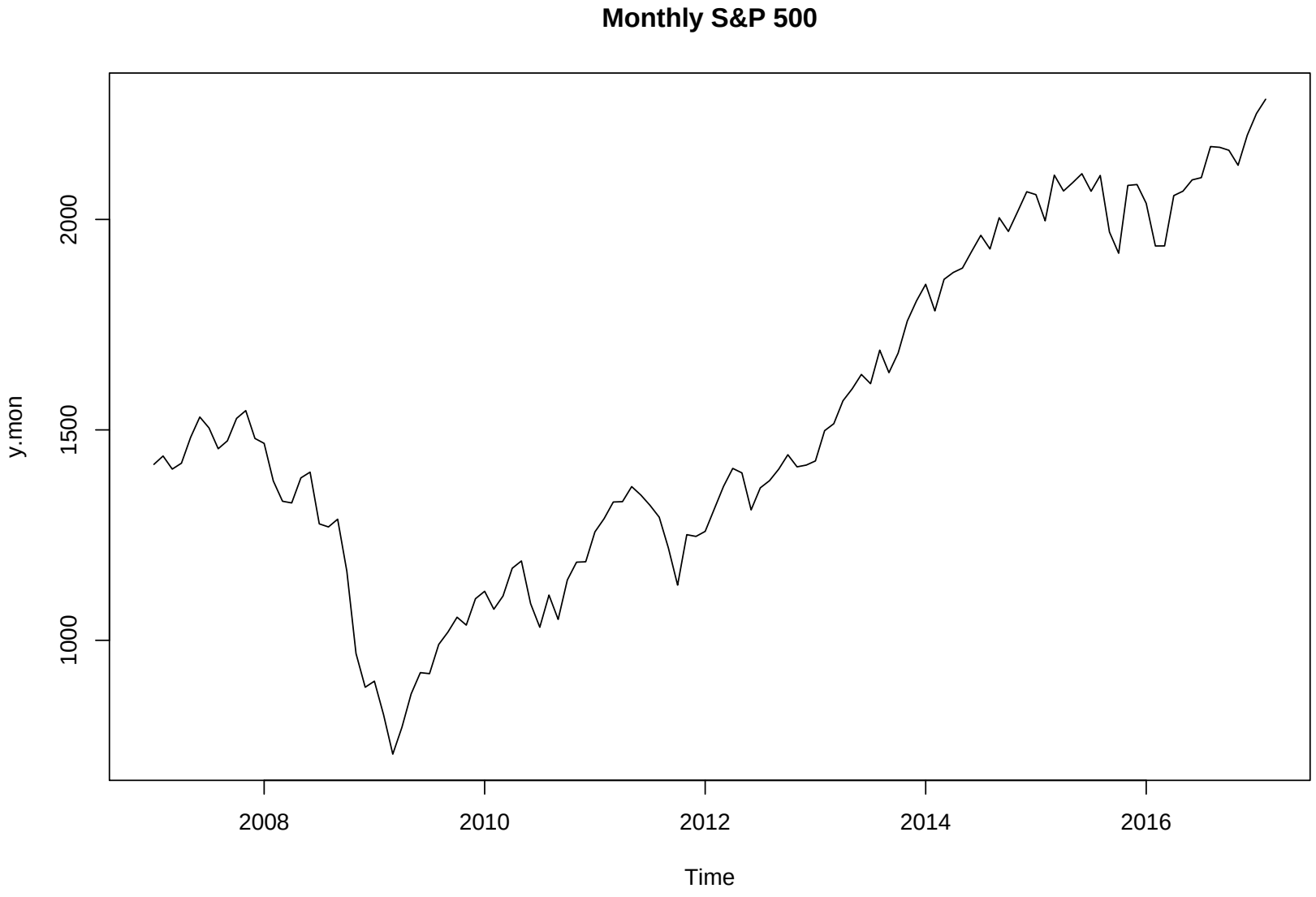
Let's use the S&P 500 monthly averages. Using the quanmod R package to load it up.

```
library(quantmod)
# load the S&P 500 data
getSymbols("^GSPC")
```

```
## [1] "GSPC"
```

```
# convert to monthly
y = to.monthly(GSPC)$GSPC.Open
# convert to ts class from xts
y.mon = as.ts(y, start=c(2007,1))
y = ts(y.mon, frequency = 1)
n = length(y)
```

Load some data to play with



Simple Forecasts: Average

- Average

$$\hat{y}_{t+1|t} = (y_1 + y_2 + \ldots y_t)/t$$

  The mean of the data. Not bad if your data fluctuate around a mean value.

- Fit $y_t = \mu + e_t$ where $e_t \sim N(0, \sigma^2)$ So the errors are i.i.d. (independent and identically distributed). This is white noise.

Simple Forecasts: Average

Use forecast to create forecasts from average with CIs:

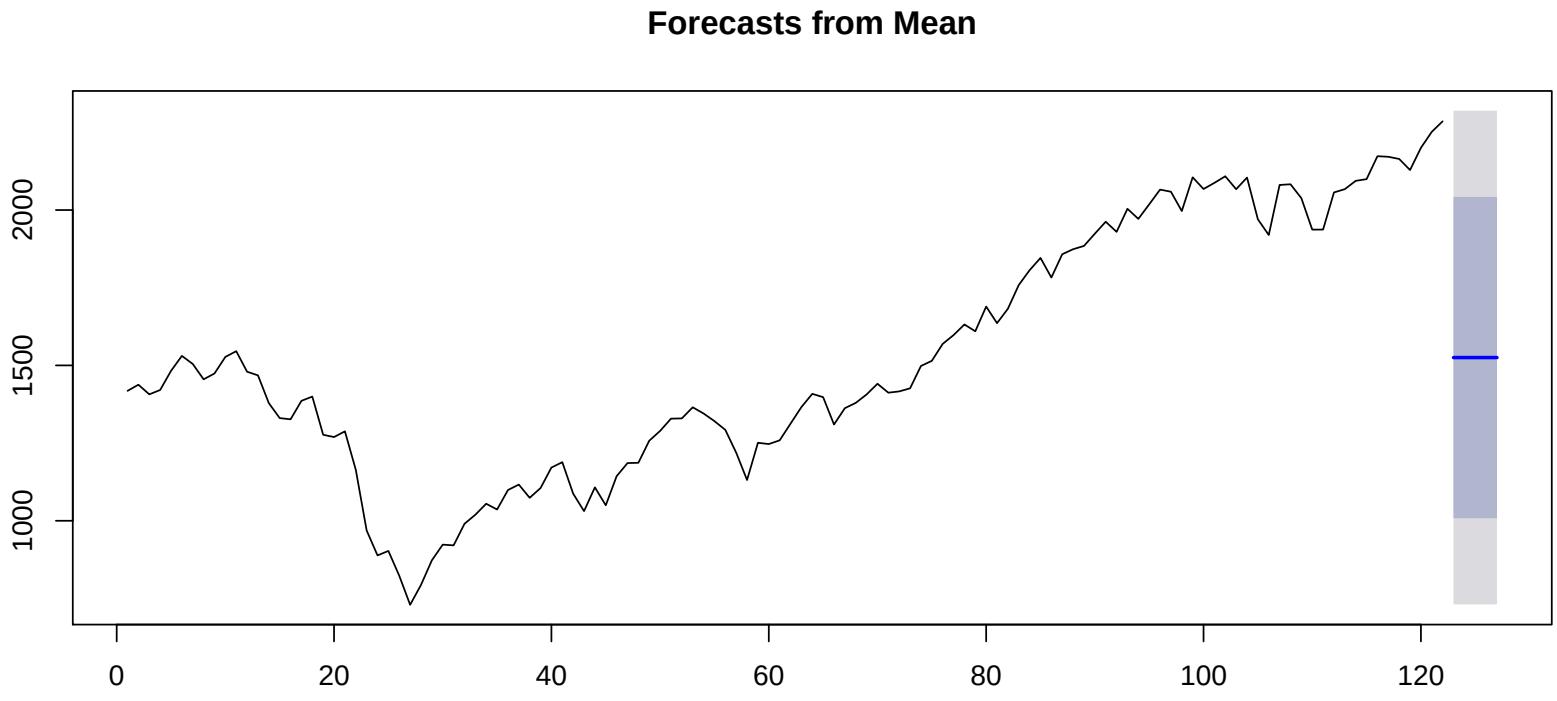
```
meanf(y, 5, level=95)
```

```
##     Point Forecast    Lo 95    Hi 95
## 123       1525.261 730.9762 2319.546
## 124       1525.261 730.9762 2319.546
## 125       1525.261 730.9762 2319.546
## 126       1525.261 730.9762 2319.546
## 127       1525.261 730.9762 2319.546
```

Simple Forecasts: Average

forecast makes it easy to plot your forecasts:

```
plot(meanf(y, 5), type="l")
```



Simple Forecasts: Last observed value

$$\hat{y}_{t+1|t} = y_t$$

This is a surprisingly hard forecast to beat in many situations.
Making the forecast is easy, but how do you come up with the CIs?

- Let's walk through exactly what our forecast is:

$$y_t = y_{t-1} + e_t, \text{where } e_t \sim N(0, \sigma^2)$$

y_t (your forecast) is y_{t-1} plus e_t (error).

- That is ARIMA(0,1,0), aka a random walk without drift.

$$y_t - y_{t-1} = e_t$$

Simple Forecasts: Last observed value

To get the prediction interval, we need the prediction interval for e_t. Let's say that we know the variance of e_t. In that case, our prediction is distributed as follows

$$y_{t+1} \sim N(y_t, \sigma^2)$$

and the 95% distribution of that is $z_{0.05/2}\sigma$. So our forecast is

$$y_t \pm z_{0.05/2}\sigma$$

The errors are the differences: $e_t = y_t - y_{t-1}$. There are $n-1$. The estimated variance of e_t is

$$\frac{1}{n-1}\sum_{2}^{n}(\hat{e}_t - \mu)^2 = \frac{1}{n-1}\sum_{1}^{n}\hat{e}_t^2$$

since $\mu = 0$. This is the mean squared error.

```
mse=mean(diff(y)^2)
```

Simple Forecasts: Last observed value

The variance of a random walk increases with time:

$$y_{t-k} - y_t \sim N(0, k\sigma^2)$$

so for a forecast k steps in the future our forecast is:

$$y_t \pm z_{0.05/2}\sqrt{k}\sigma$$

So in R, the 95% Prediction intervals 1-5 steps ahead are (treating the estimated variance as true):

```
zs = qnorm(0.975)*sqrt(mse)
cbind(y[n]-zs*sqrt(1:5), y[n]+zs*sqrt(1:5))
```

```
##          [,1]     [,2]
## [1,] 2170.657 2400.523
## [2,] 2123.051 2448.130
## [3,] 2086.521 2484.659
## [4,] 2055.725 2515.456
## [5,] 2028.593 2542.587
```

Simple Forecasts: Last observed value

forecast computes these for you with rwf():

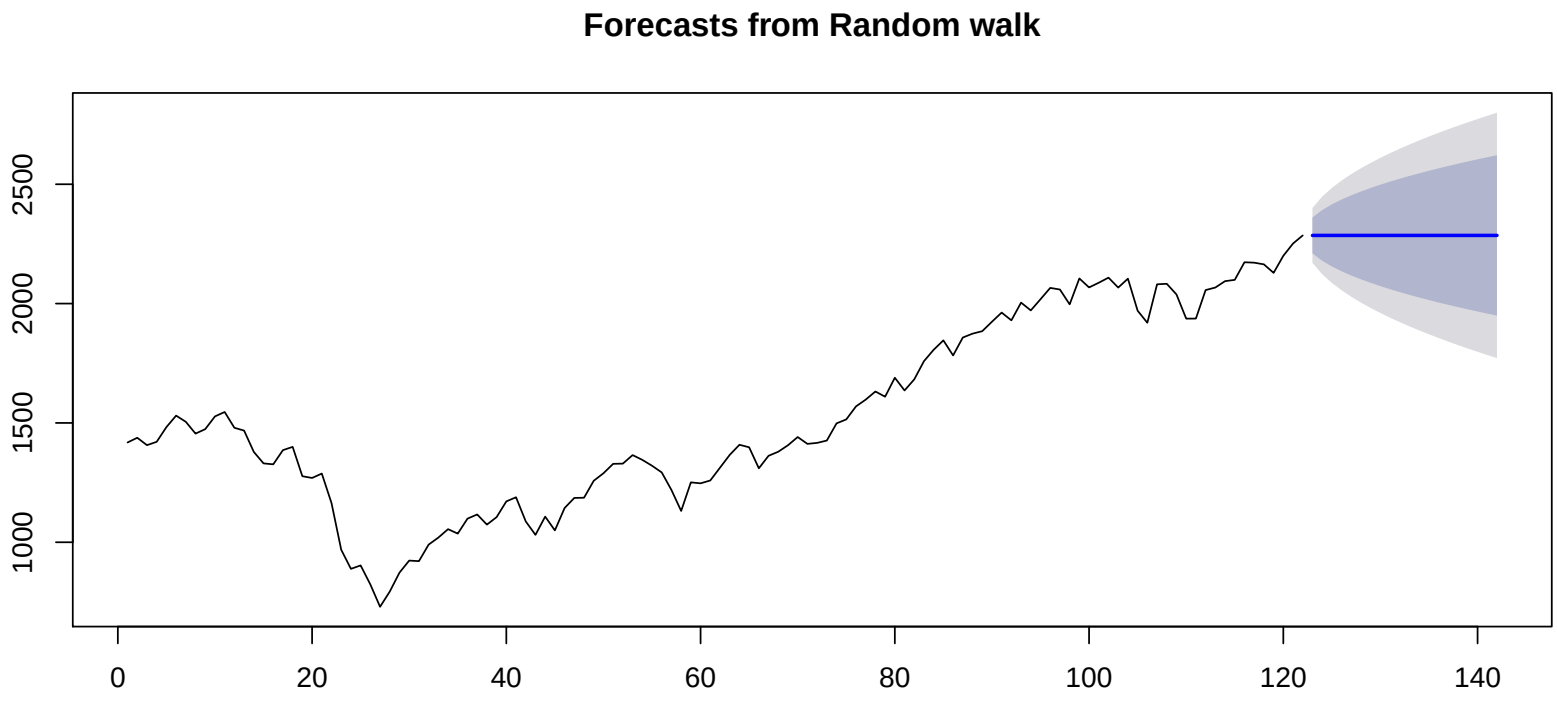
```
rwf(y, 5, level=95)
```

```
##     Point Forecast    Lo 95    Hi 95
## 123        2285.59 2170.657 2400.523
## 124        2285.59 2123.051 2448.130
## 125        2285.59 2086.521 2484.659
## 126        2285.59 2055.725 2515.456
## 127        2285.59 2028.593 2542.587
```

Simple Forecasts: Last observed value

plot your forecasts:

```
plot(rwf(y, 20), type="l")
```



Simple Forecasts: Last observed value

Note, forecast treats the variance as known and equal to the estimated value. Obviously it is an estimate and unknown. The correct prediction interval includes the uncertainty in your σ^2. For this particular problem, it is the same as the prediction error for known mean and unknown variance for a normal distribution. You use a t-distribution with $n-2$ degrees of freedom instead of a standard Normal (meaning mean of 0, var of 1):

$$y_t \pm t^{n-2}_{0.05/2}\sqrt{k}\sigma$$

```
zt = qt(0.975, df=n-1)*sqrt(mse)
cbind(qnorml=y[n]-zs*sqrt(1:5), qnormu=y[n]+zs*sqrt(1:5),
      qtl=y[n]-zt*sqrt(1:5), qtu= y[n]+zt*sqrt(1:5))
```

```
##         qnorml   qnormu      qtl      qtu
## [1,] 2170.657 2400.523 2169.496 2401.684
## [2,] 2123.051 2448.130 2121.409 2449.772
## [3,] 2086.521 2484.659 2084.510 2486.670
```

Simple Forecasts: Last observed value WITH drift

Now our forecast is:

$$y_t = y_{t-1} + e_t, \text{where } e_t \sim N(\mu, \sigma^2)$$

The logic behind the calculation of the prediction intervals is the same except - we estimate the mean μ - the estimate of the variance of e_t is different because we are estimating the mean, so the variance estimate is $\frac{1}{n-1}\sum_2^n(\hat{e}_t - \bar{e}_t)^2$. That's just the variance of the differences.

forecast computes the forecasts and prediction intervals, treating μ (drift) as unknown and σ^2 as known (and equal to estimate).

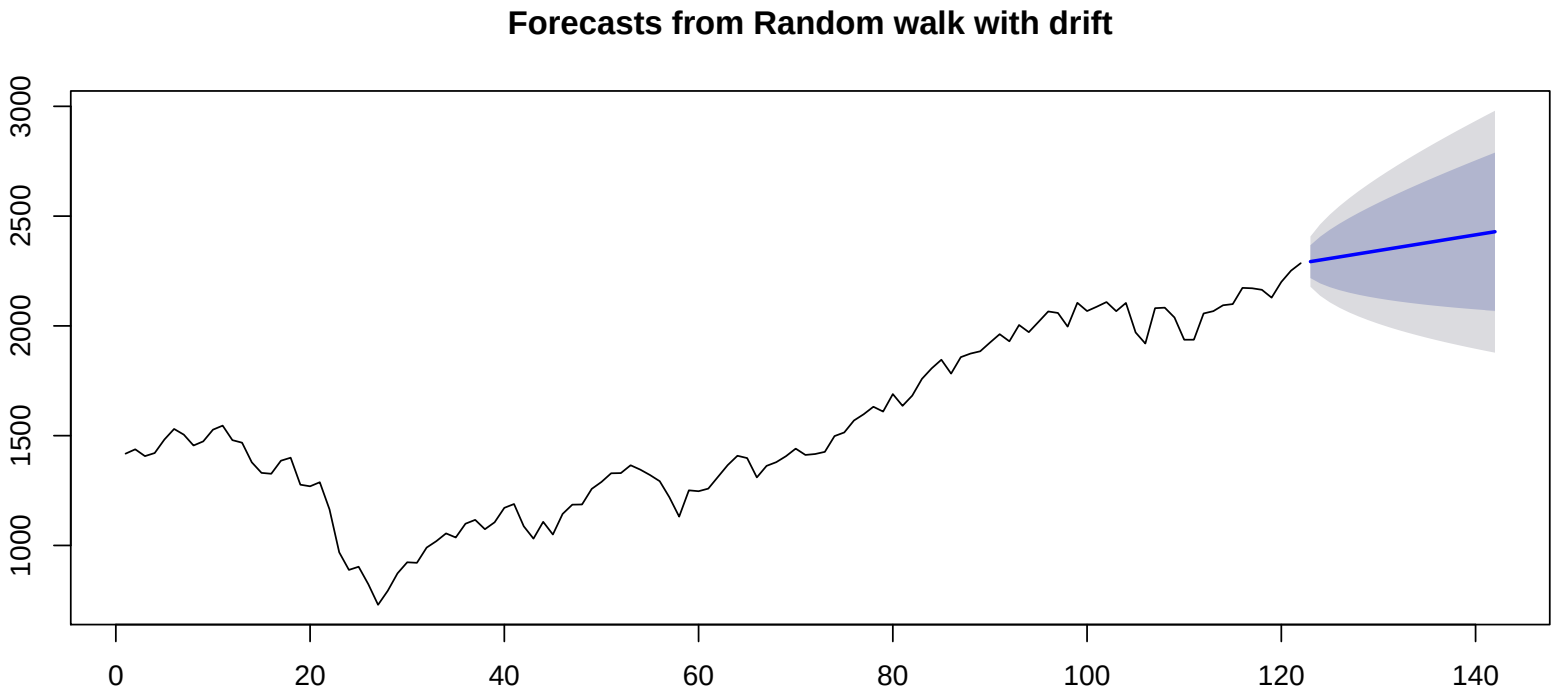
```
rwf(y, 2, level=95, drift=TRUE)
```

```
##     Point Forecast    Lo 95    Hi 95
## 123        2292.76 2178.215 2407.305
## 124        2299.93 2137.271 2462.589
```

Simple Forecasts: Last observed value WITH drift

plot your forecasts:

```
plot(rwf(y, 20, drift=TRUE), type="l")
```



Simple Forecasts: Last observed value in season

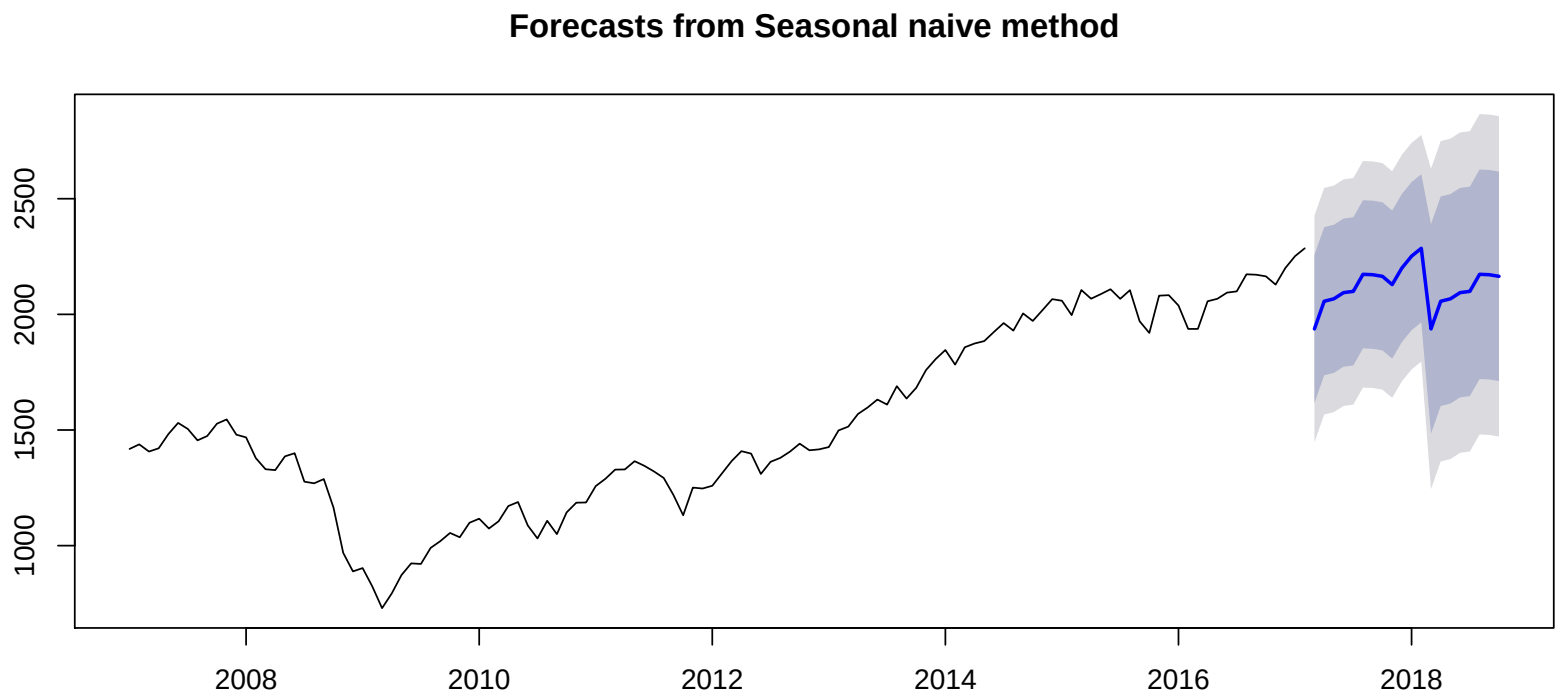
Forecast is the last value in the same season. Say data are monthly, the next Jan forecast is the last Jan observed value. If m is the frequency (12 for monthly), then the forecast is

$$y_t = y_{t-m} + e_t, \text{where } e_t \ N(0, \sigma^2)$$

This is not so useful since it doesn't allow you to include drift (trend). We'll see more useful season models when we use forecast's exponential smoothing models.

Simple Forecasts: Last observed value in season

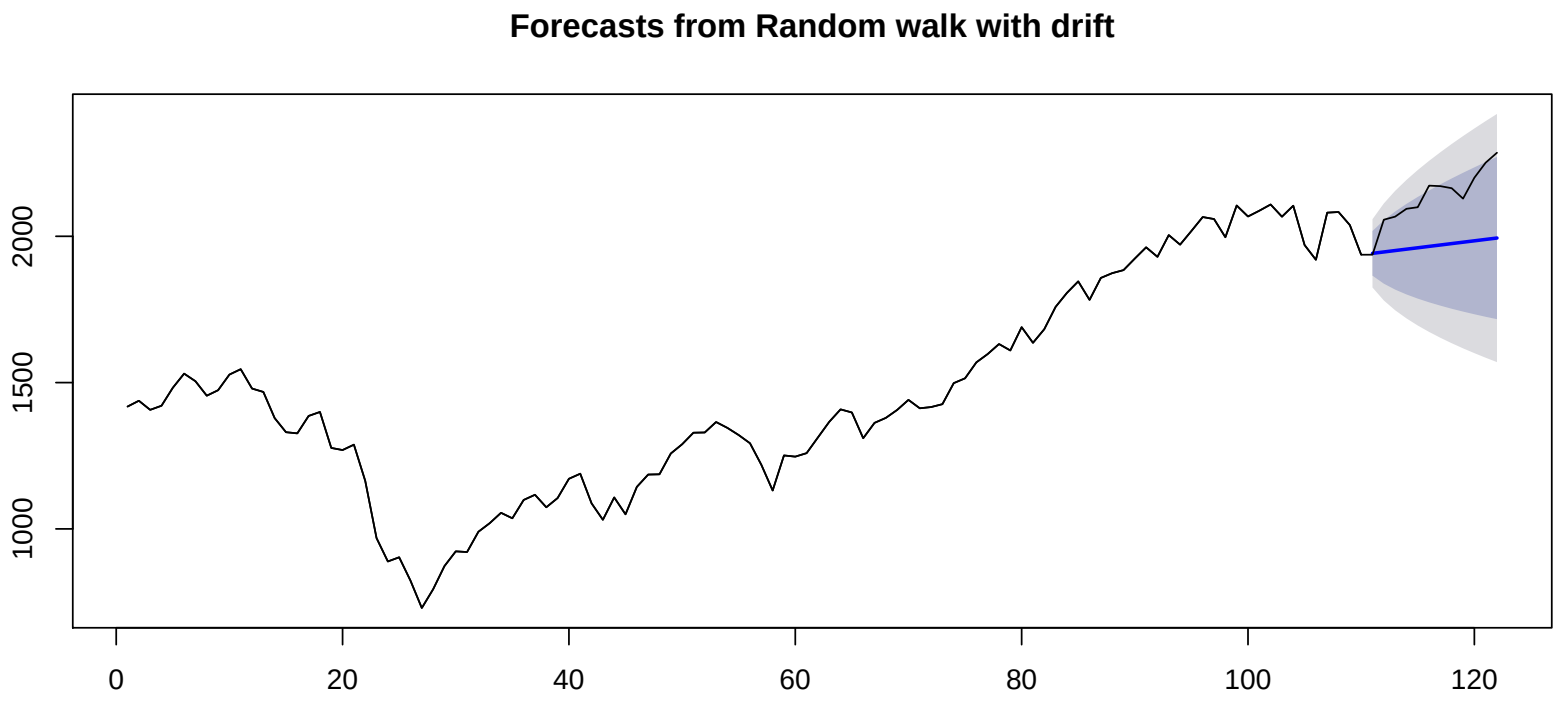
```
plot(snaive(y.mon, 20), type="l")
```



Assessing forecast error

forecast has the accuracy() function which will compute a variety of standard metrics using predictions and test data.

```
y2 <- window(y,start=1,end=n-12)
plot(rwf(y2, 12,drift=TRUE), type="l")
lines(y)
```



Assessing forecast error: common metrics

Here are common ones. The accuracy() function has a few others.

- RMSE: root mean square error
- MAE mean absolute error
- MAPE mean absolute percentage error
- MASE mean absolute scaled error (useful for meta analyses)

Assessing forecast error: common metrics

```
y2 <- window(y,start=1,end=n-12)
fit1 <- meanf(y2,h=12)
fit2 <- rwf(y2,h=12)
fit3 <- rwf(y2,h=12,drift=TRUE)
```

```
y3 <- window(y, start=n-11)
vals=c("RMSE", "MAE", "MAPE", "MASE")
rbind(
  meanf=accuracy(fit1, y3)[2,vals],
  rwf = accuracy(fit2, y3)[2,vals],
  rwf.drift=accuracy(fit3, y3)[2,vals])
```

```
##                  RMSE      MAE       MAPE       MASE
## meanf      683.0320 677.0857 31.578710 14.108030
## rwf        218.2028 198.8084  9.144221  4.142452
## rwf.drift 183.8597 168.6327  7.762764  3.513698
```

Simple exponential smoothing: no trend

Exponential smoothing is similar to the random walk forecast but with adjustable weight on the observations and allows time-varying trend and season. The simple exponential forecast is:

$$\hat{y}_{t+1|t} = l_t$$

The model is a type of state-space model and DLM, but with a single error term shared across both the process and observation. We've been working with multi-error models so far.

$$y_t = l_{t-1} + e_t$$

$$l_t = l_{t-1} + \alpha e_t$$

If $\alpha = 1$ you revert to the random walk without drift model. α allows you to adjust how much weight is given to the observations in the past. If α is close to 0 then a lot of weight is given to the past and the forecast is smoother. The weight falls off exponentially, thus the name exponential smoothing.

Simple exponential smoothing: no trend

forecast has the ets() function to fit these models. The model argument specifies the form of the exponential smoothing model.

Form is (Error, Trend, Seasonal).

- "ANN" = additive error, no trend, no season This is the simple exponential smoothing model

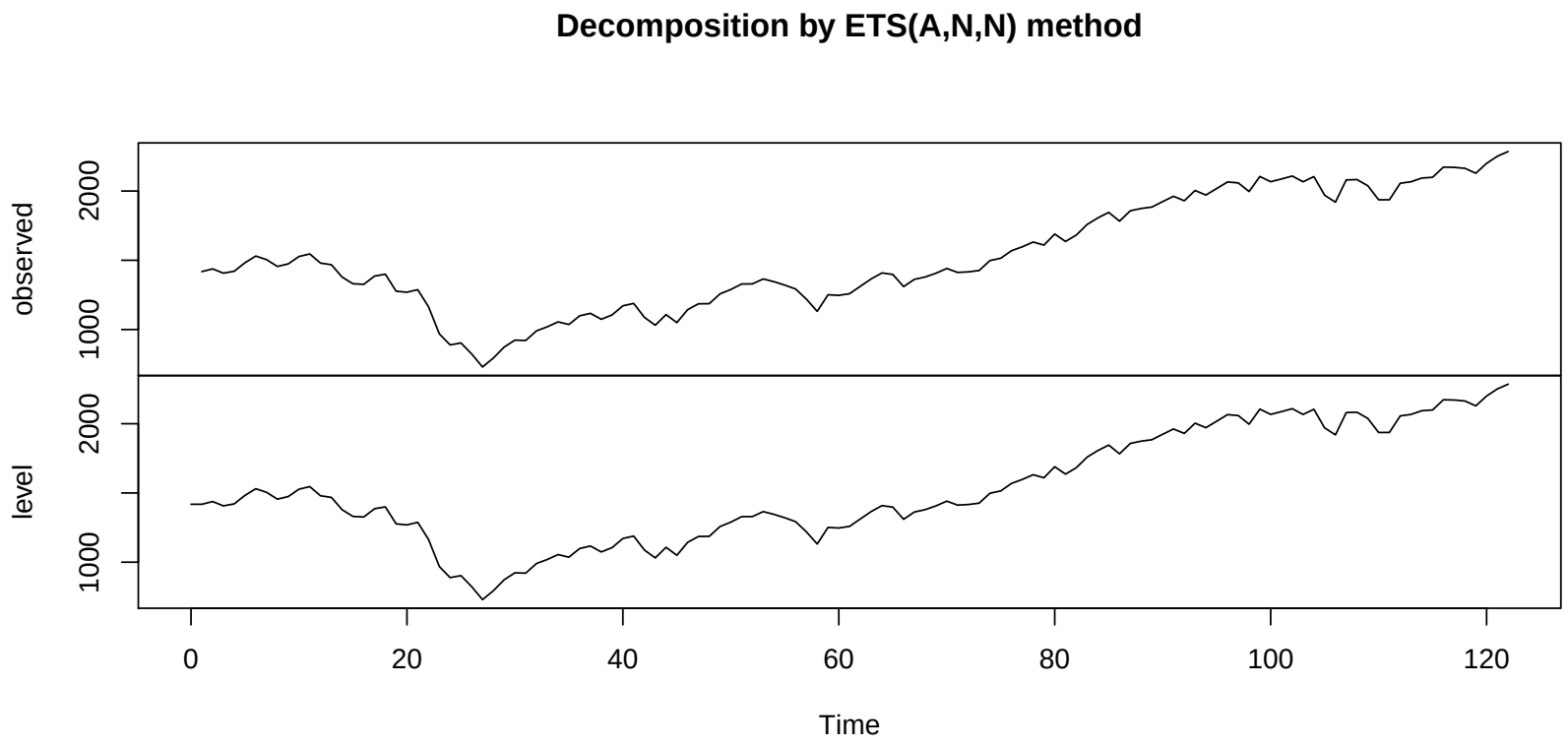
```
ets.simple = ets(y, model="ANN")
ets.simple
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.9999
##
```

Simple exponential smoothing: no trend

The estimate of α is close to 1, so we have the random walk model and the estimated level is just the data. No smoothing.

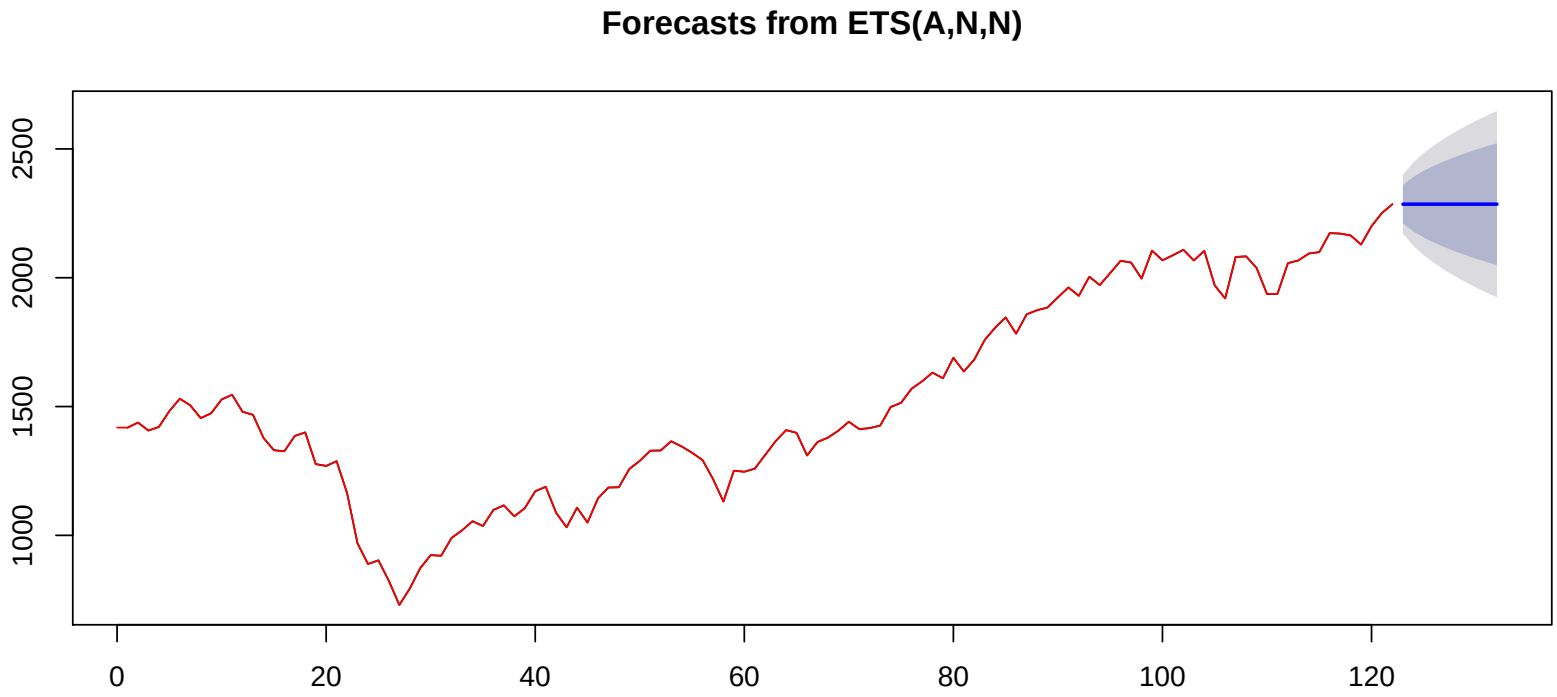
```
plot(ets.simple)
```



Simple exponential smoothing: no trend

We can easily forecast with this fitted model. Note I plotted the states (aka level, aka l_t) over the data. You don't see the data line (black) since $\alpha = 1$.

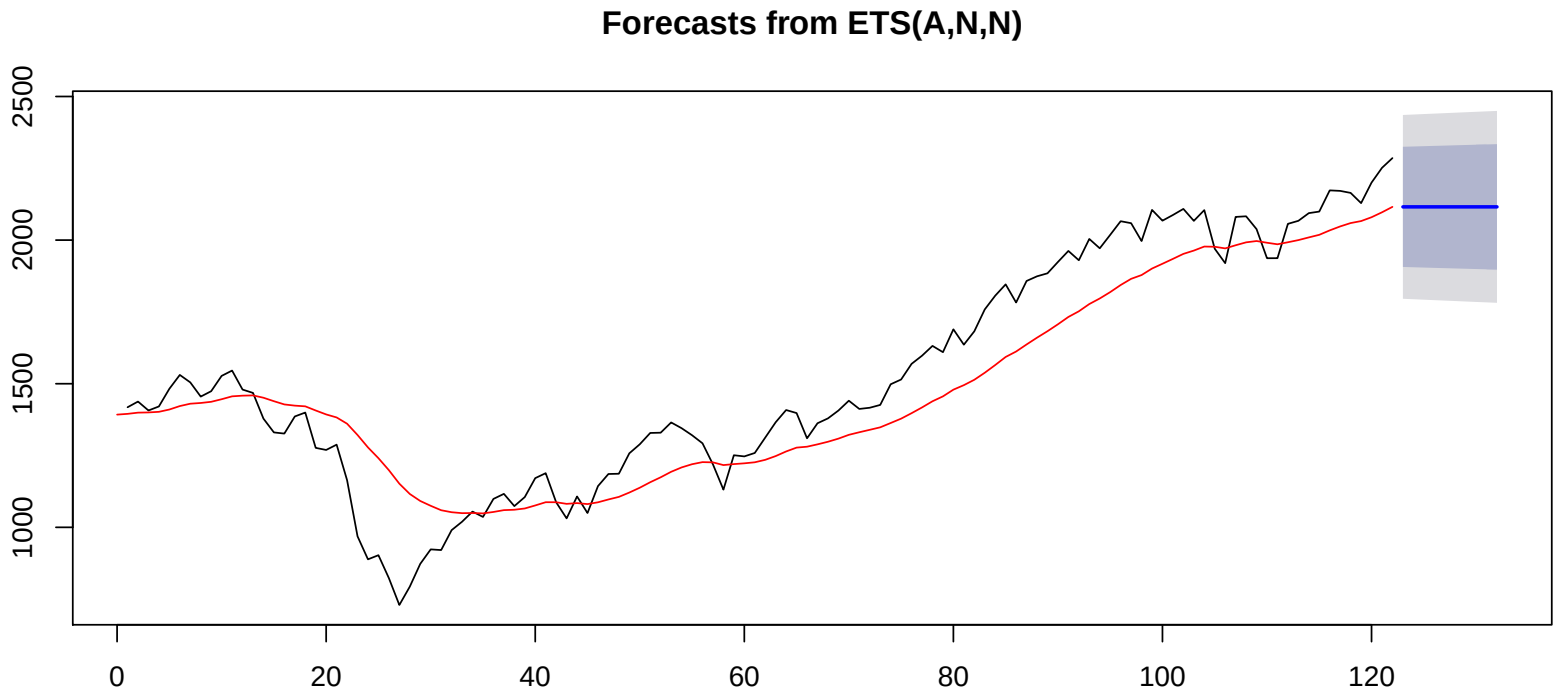
```
plot(forecast(ets.simple))
lines(ets.simple$states, col="red")
```



Simple exponential smoothing: no trend

We can pass in α for control the smoothing. I plotted the states over the data, and now you see that the estimated level is smooth.

```
ets.simple2 = ets(y, model="ANN", alpha=0.1)
plot(forecast(ets.simple2))
lines(ets.simple2$states, col="red")
```



Exponential smoothing with trend

The forecast is a combination of two estimated states, the level term (l_t) and a trend term (b_t).

$$\hat{y}_{t+1|t} = l_t + b_t$$

$$y_t = l_{t-1} + b_{t-1} + e_t \tag{1}$$
$$l_t = l_{t-1} + b_{t-1} + \alpha e_t \tag{2}$$
$$b_t = b_{t-1} + \beta e_t \tag{3}$$

$$y_t = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} l \\ b \end{bmatrix}_{t-1} + e_t$$

$$\begin{bmatrix} l \\ b \end{bmatrix}_t = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} l \\ b \end{bmatrix}_{t-1} + \begin{bmatrix} \alpha \\ \beta \end{bmatrix} e_t$$

Exponential smoothing with trend

We fit this model using "AAN" to specify additive trend model.

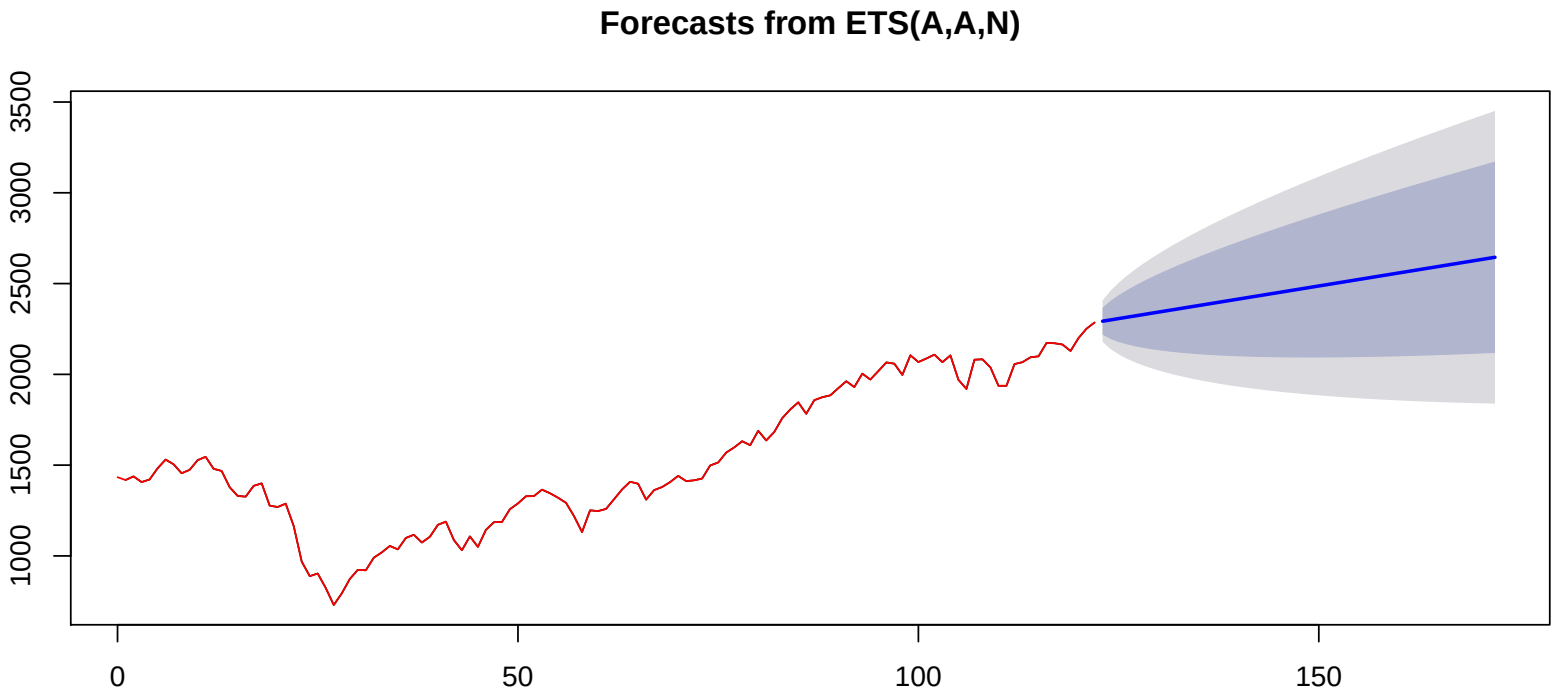
```
ets.trend = ets(y, model="AAN")
ets.trend
```

```
## ETS(A,A,N)
##
## Call:
##  ets(y = y, model = "AAN")
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 1e-04
##
##   Initial states:
##     l = 1433.4643
##     b = 7.1875
##
##   sigma:  58.0012
```

Exponential smoothing with trend

When we forecast with this model, we see that a trend is estimated. However α is estimated to be 1 and β is about 0, so it's just fitting a random with with fixed trend.

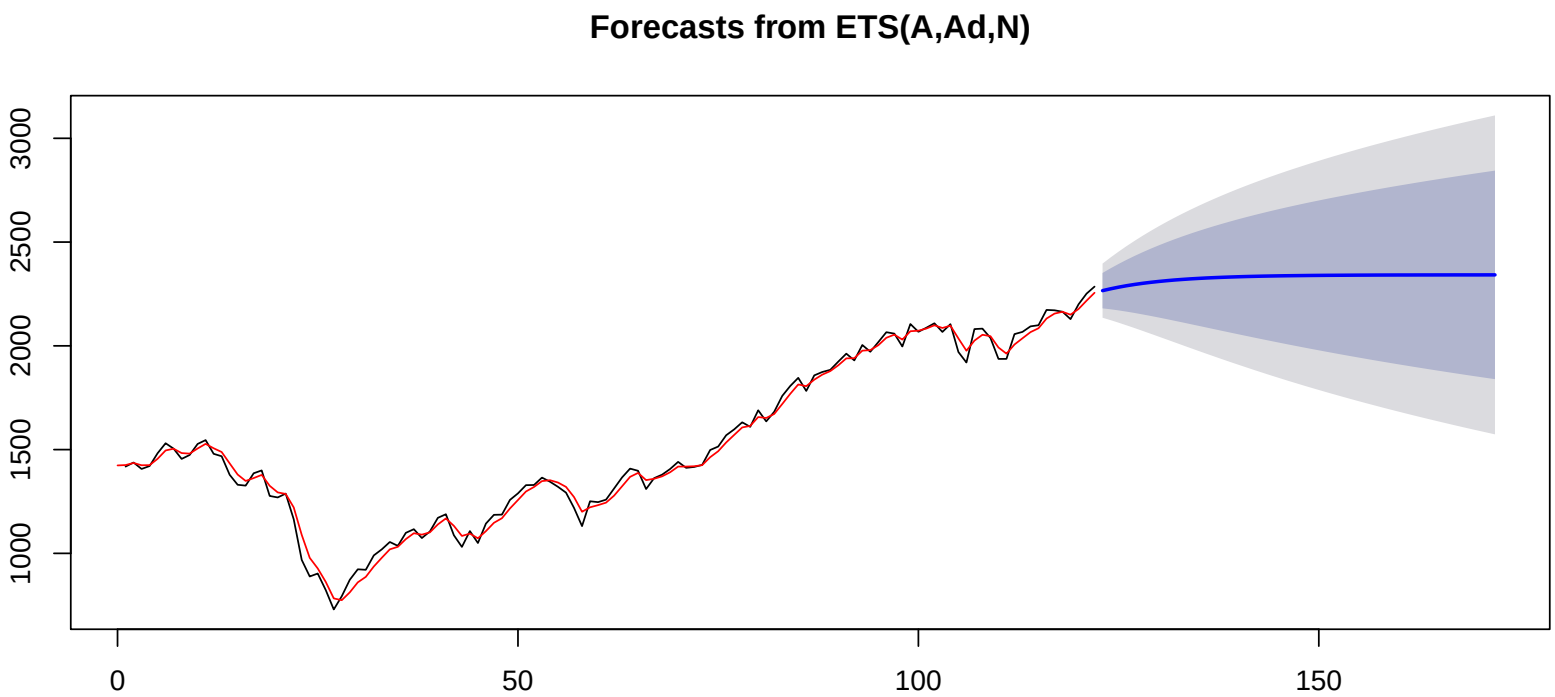
```
plot(forecast(ets.trend,50))
lines(ets.trend$states[,1], col="red")
```



Exponential smoothing with trend

We can adjust the amount of smoothing for our level and trend.

```
ets.trend2 = ets(y, model="AAN", alpha=.5, beta=.05)
plot(forecast(ets.trend2,50))
lines(ets.trend2$states[,1], col="red")
```



Exponential smoothing with season and trend

The forecast is a combination of three estimated states, the level term (l_t) and a trend term (b_t).

$$\hat{y}_{t+1|t} = l_t + b_t + s_{t-m}$$

$$y_t = l_{t-1} + b_{t-1} + s_{t-m} + e_t \tag{4}$$

$$l_t = l_{t-1} + b_{t-1} + \alpha e_t \tag{5}$$

$$b_t = b_{t-1} + \beta e_t s_t = s_{t-m} + \gamma e_t \tag{6}$$

This is a very flexible model. You might want to fix β or γ to constrain the amount of time-variation or smoothing you allow.

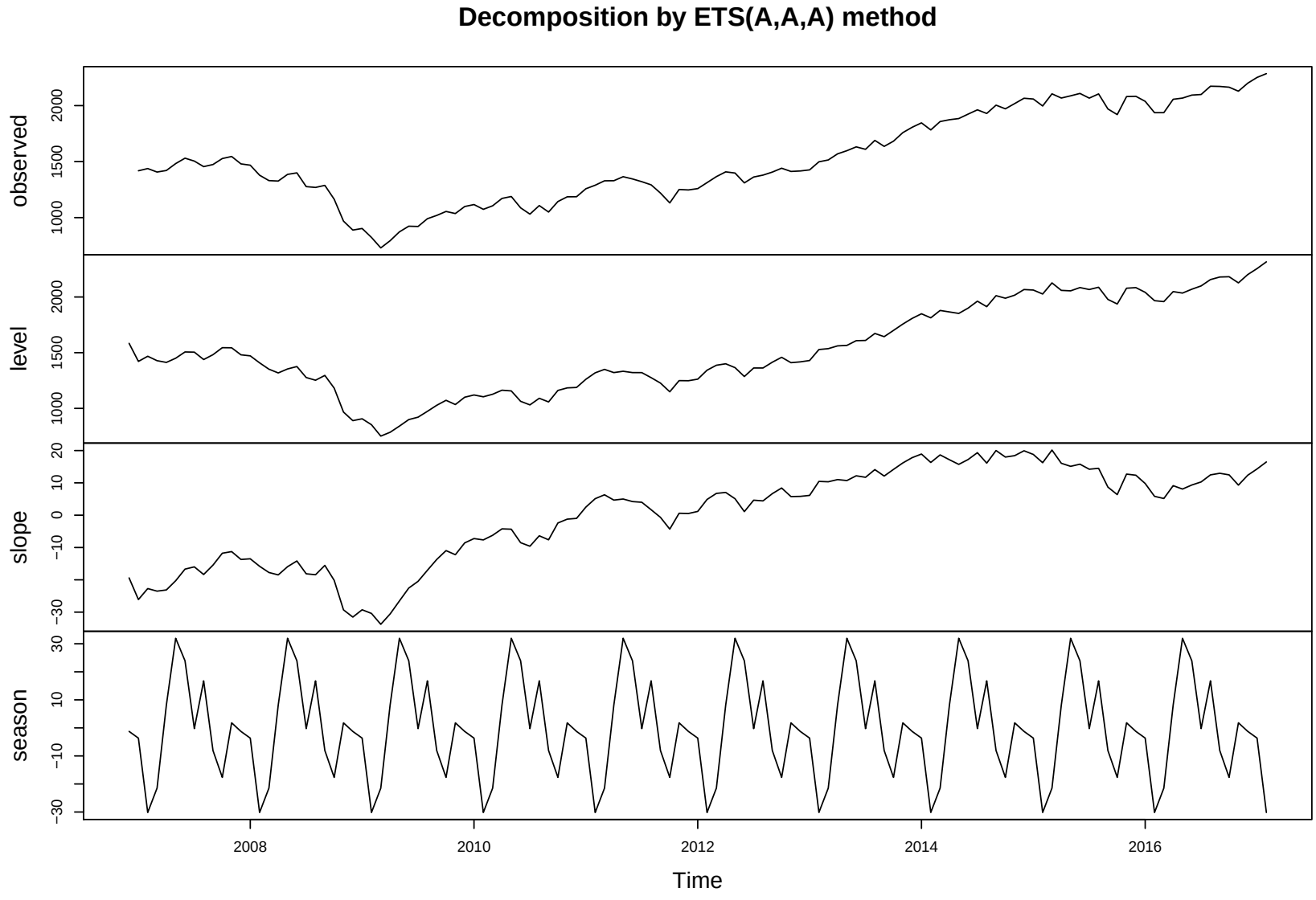
Exponential smoothing with season and trend

Fit this model using "AAA" to specify additive trend model with additive season model. You need to use the ts object with the monthly season.

```
ets.season = ets(y.mon, model="AAA")
ets.season
```

```
## ETS(A,A,A)
##
## Call:
##  ets(y = y.mon, model = "AAA")
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.0469
##     gamma = 1e-04
##
##   Initial states:
##     l = 1584.0377
```

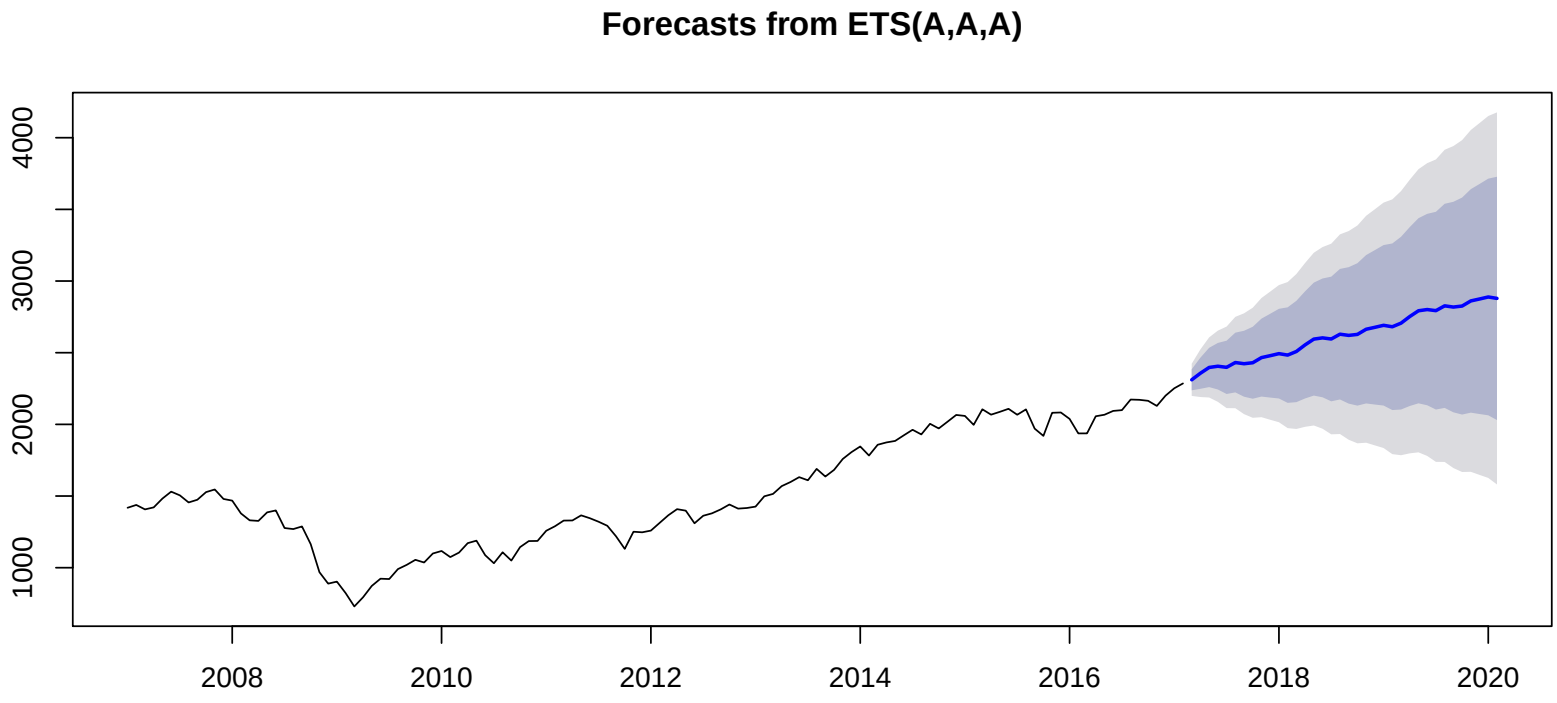
Exponential smoothing with season and trend



Exponential smoothing with season and trend

When we forecast with this model, we see that the forecast is wavy. That's the seasonal effect.

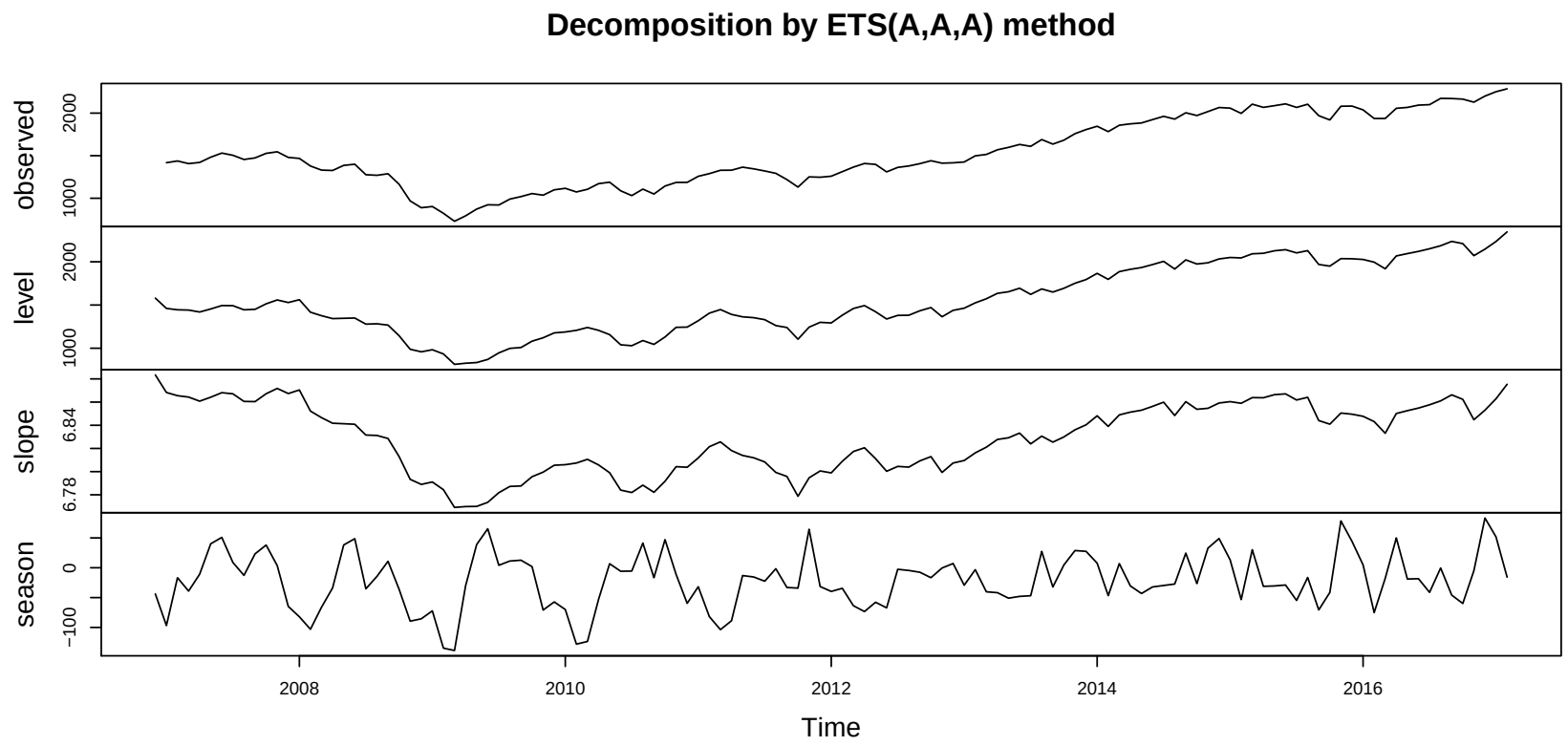
```
plot(forecast(ets.season,3*12))
```



Exponential smoothing with season and trend

We can adjust the amount of smoothing for our season effect. Now the season effect varies in time.

```
ets.season2 = ets(y.mon, model="AAA", gamma=.5)
plot(ets.season2)
```



Summary

forecast has many other useful functions for forecasting and the ets() function has many other options, such as multiplicative error models.

- The Hyndman & Athanasopoulos's open-access text is very accessible and shows many code examples: `https://www.otexts.org/fpp`
- The package treats the variance as true instead of estimated for the prediction intervals (at least for the random walk). If your time series is long, that's probably fine. If your time series is short, that can cause your prediction intervals to be too narrow.
- If your time series is short, you need to estimate the initial state most likely.
- Always test your statistical approach on simulated data.
- The section on Advanced forecasting methods talks about the types of models we've been using Fish 507: dynamic regression models, vector autoregressions (MAR), and hierarchical time-series models.