Exponential smoothing models

and the forecast package

Eli Holmes

7 February 2017

Load some data to play with

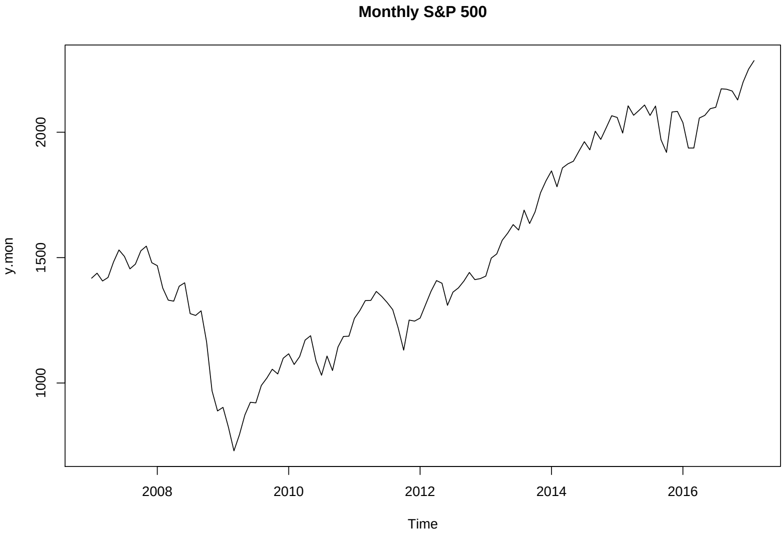
Let's use the S&P 500 monthly averages. Using the quanmod R package to load it up.

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

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

```r
# 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
$$\hat{y}_{n+1|1:n} = (y_1 + y_2 + \dots y_n)/n$$
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:

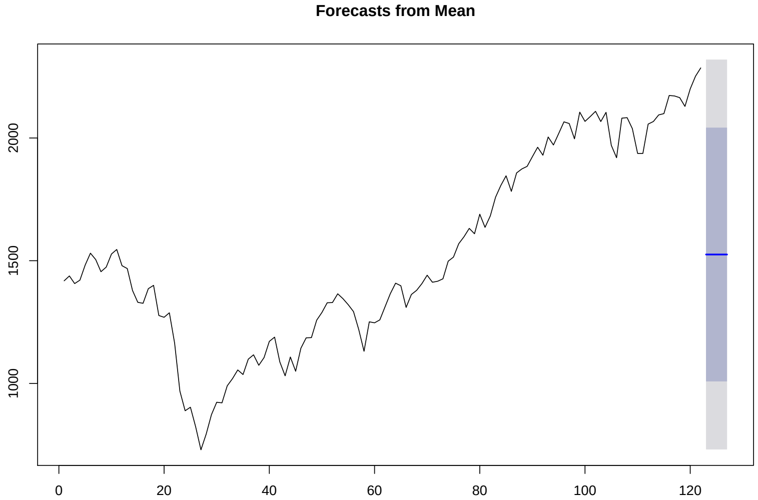
```r
meanf(y, 5, level=95)
```

```
##     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

$$\hat{y}_{n+1|1:n} = y_n$$

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 estimated variance of e_t is

$$\hat{e}_t = y_t - y_{t-1}$$

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

since $\mu = 0$. aka 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():

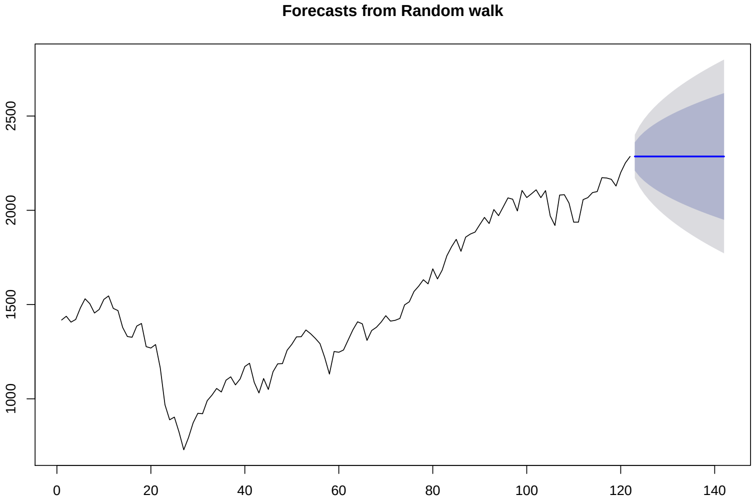
```r
rwf(y, 5, level=95)
```

```
##     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 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).

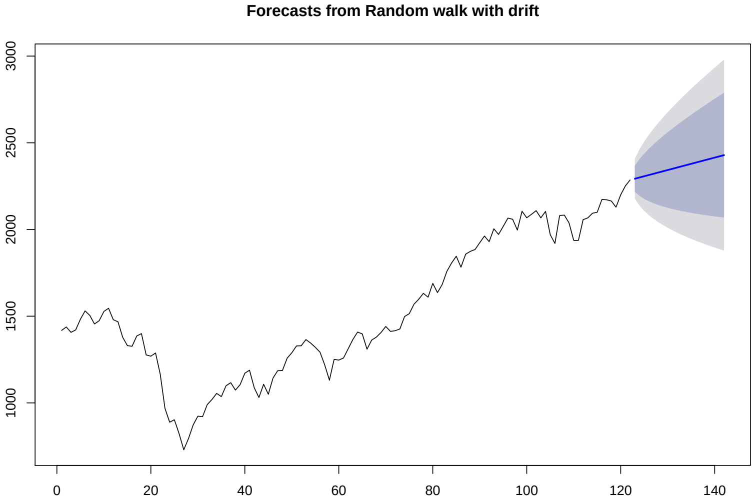
```
rwf(y, 5, level=95, drift=TRUE)
```

```
##     Point Forecast    Lo 95    Hi 95
## 123        2292.76 2178.215 2407.305
## 124        2299.93 2137.271 2462.589
## 125        2307.10 2107.070 2507.130
## 126        2314.27 2082.358 2546.182
```

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

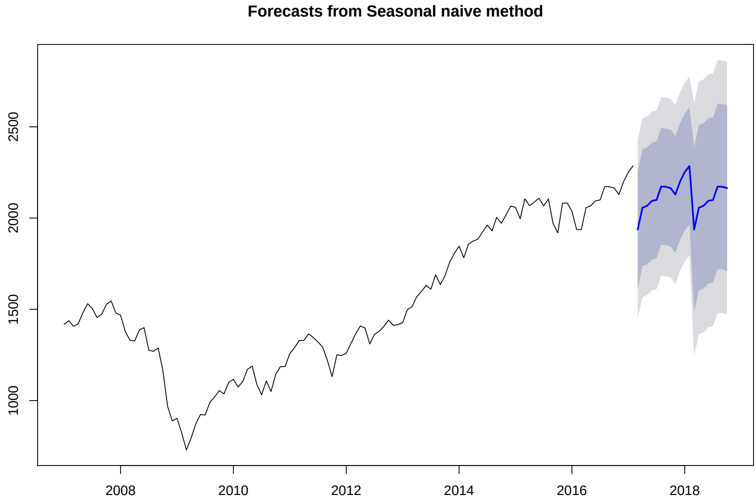
Forecast is the last value in the same season. Say data are monthly, the next Jan forecast is the last Jan observed value. If m is the frequency (12 for monthly), then the forecast is

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

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

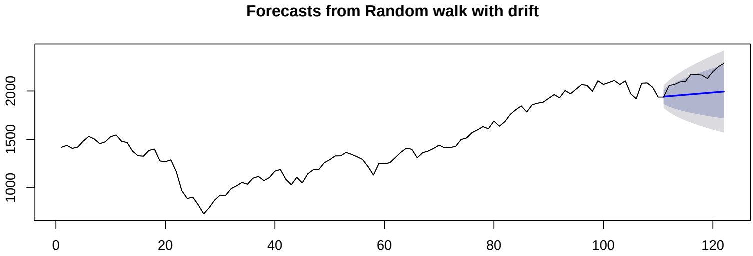
```
plot(snaive(y.mon, 20), type="l")
```



Tools for 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)
```



Tools for assessing forecast error

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

Tools for assessing forecast error

```
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)
rbind(
  meanf=accuracy(fit1, y3)[2,c("RMSE", "MAE", "MAPE", "MASE
  rwf = accuracy(fit2, y3)[2,c("RMSE", "MAE", "MAPE", "MASE
  rwf.drift=accuracy(fit3, y3)[2,c("RMSE", "MAE", "MAPE", "
```

```
##                 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
```

Exponential smoothing: similar idea but with observation error