

# Advanced Forecasting Methods

## Session 4

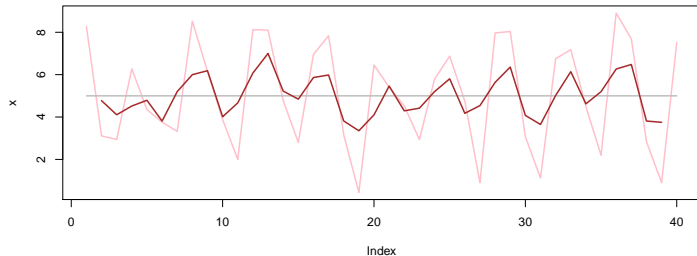
Jyotirmoy Bhattacharya

IIM Kozhikode

Winter 2009

# A seasonal series and 3 MA

```
> N <- 40
> seas <- c(2, -1, -3, 2)
> trend <- rep(5, N)
> e <- rnorm(N)
> x <- trend + seas + e
> plot(x, type = "l", col = "pink", lwd = 2)
> lines(trend, col = "gray", lwd = 2)
> lines(filter(x, rep(1/3, 3)), col = "brown",
+       lwd = 2)
```



## Simple moving average of even order

- Suppose we want to calculate simple moving average of order 4. Do we use,

$$G_t = (Y_{t-1} + Y_t + Y_{t+1} + Y_{t+2})/4$$

or

$$H_t = (Y_{t-2} + Y_{t-1} + Y_t + Y_{t+1})/4$$

neither of which are symmetric.

- Solution: take moving average of moving average

$$\begin{aligned} F_t &= (G_{t-1} + G_t)/2 \\ &= \frac{1}{8}Y_{t-2} + \frac{1}{4}Y_{t-1} + \frac{1}{4}Y_t + \frac{1}{4}Y_{t+1} + \frac{1}{8}Y_{t+2} \end{aligned}$$

- $F_t$  is known as the  $2 \times 4$  MA of  $Y_t$

# An MA as the same order as frequency removes additive seasonality

For a quarterly example, take:

$$Y_t = a + s_t + e_t$$

where

$$s_1 + s_2 + s_3 + s_4 = 0 \quad \text{and} \quad s_{t+4} = s_t$$

If we take the  $2 \times 4$  moving average

$$F_t = \frac{1}{8}Y_{t-2} + \frac{1}{4}Y_{t-1} + \frac{1}{4}Y_t + \frac{1}{4}Y_{t+1} + \frac{1}{8}Y_{t+2}$$

we have,

$$F_t = a + (0.5s_{t-2} + s_{t-1} + s_t + s_{t+1} + 0.5s_{t+2})/4 + (0.5e_{t-2} + e_{t-1} + e_t + e_{t+1} + 0.5e_{t+2})/4$$

$$\text{but } s_{t-2} = s_{t-2+4} = s_{t+2}$$

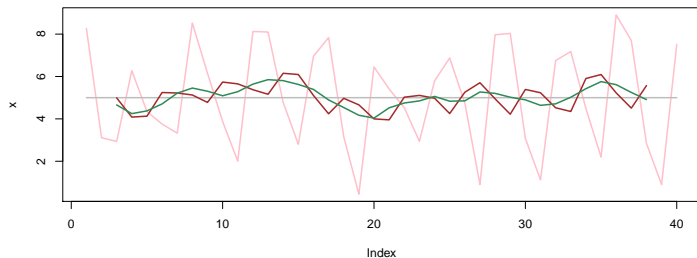
$$= a + (s_{t-1} + s_t + s_{t+1} + s_{t+2})/4 + (0.5e_{t-2} + e_{t-1} + e_t + e_{t+1} + 0.5e_{t+2})/4$$

and since the sum of four consecutive seasonal terms is 0

$$= a + (0.5e_{t-2} + e_{t-1} + e_t + e_{t+1} + 0.5e_{t+2})/4$$

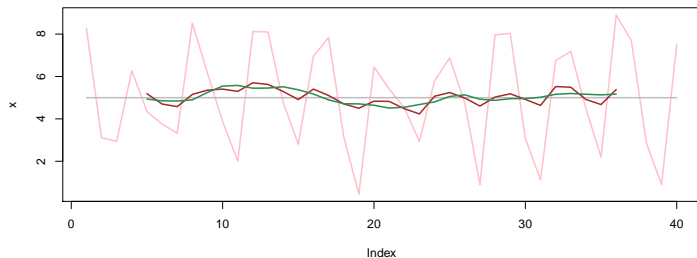
# A seasonal series, 5 and $2 \times 4$ MA

```
> plot(x, type = "l", col = "pink", lwd = 2)
> lines(trend, col = "gray", lwd = 2)
> lines(filter(x, rep(1/5, 5)), col = "brown",
+       lwd = 2)
> lines(filter(x, c(1/8, 1/4, 1/4, 1/4, 1/8)),
+       col = "seagreen", lwd = 2)
```



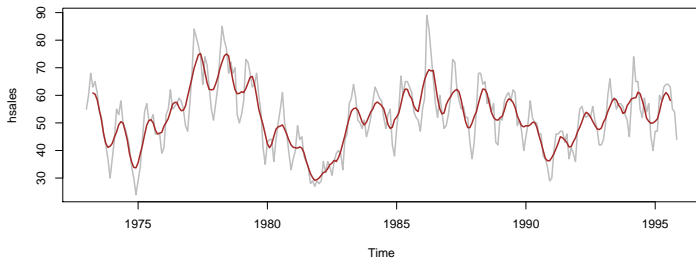
# A seasonal series, 9 and $2 \times 8$ MA

```
> plot(x, type = "l", col = "pink", lwd = 2)
> lines(trend, col = "gray", lwd = 2)
> lines(filter(x, rep(1/9, 9)), col = "brown",
+       lwd = 2)
> lines(filter(x, c(1/16, rep(1/8, 7), 1/16)),
+       col = "seagreen", lwd = 2)
```



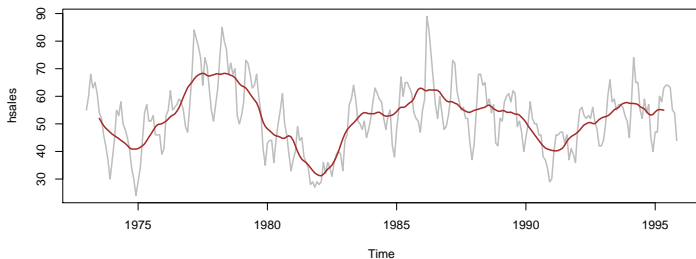
## Real seasonal data, 7 MA

```
> plot(hsales, col = "gray", lwd = 2)
> lines(filter(hsales, rep(1/7, 7)), col = "brown",
+       lwd = 2)
```



## Real seasonal data, $2 \times 12$ MA

```
> plot(hsales, col = "gray", lwd = 2)
> lines(filter(hsales, c(1/24, rep(1/12, 11),
+      1/24))), col = "brown", lwd = 2)
```



## Classical additive decomposition (trend)

$$Y_t = T_t + S_t + E_t$$

- Compute  $T_t$  by filtering  $Y_t$  with a  $2 \times 12$  MA filter.  

```
> hsales.trend <- filter(hsales, c(1/24, rep(1/12,  
+ 11), 1/24))
```
- Detrend the series by subtracting  $T_t$  from  $Y_t$   

```
> hsales.detrend <- hsales - hsales.trend
```

## Classical additive decomposition (seasonal component)

- Compute seasonal component by taking seasonal means

```
> hsales.mmean <- tapply(hsales.detrend, cycle(hsales.detrend),  
+   mean, na.rm = TRUE)  
> hsales.seas <- unsplit(hsales.mmean, cycle(hsales.detrend))
```

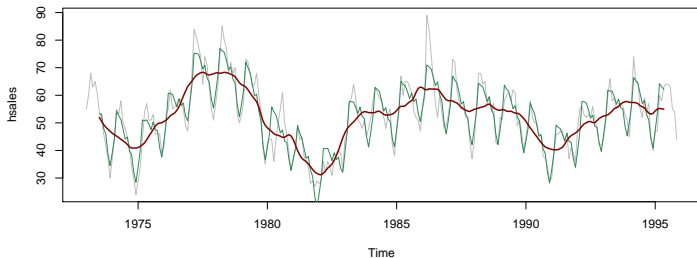
## Classical additive decomposition (the erratic component)

- What remains is the erratic component

```
> hsales.erratic <- hsales.detrend - hsales.seas
```

# Classical additive decomposition (putting it all together)

```
> plot(hsales, col = "gray")  
> lines(hsales.trend + hsales.seas, col = "seagreen")  
> lines(hsales.trend, col = "darkred", lwd = 2)
```



## Classical multiplicative decomposition (trend)

$$Y_t = T_t \times S_t \times E_t$$

- Compute  $T_t$  by filtering  $Y_t$  with a  $2 \times 12$  MA filter.

```
> hsales.xtrend <- filter(hsales, c(1/24,  
+   rep(1/12, 11), 1/24))
```

- Detrend the series by dividing  $Y_t$  by  $T_t$

```
> hsales.xdetrend <- hsales/hsales.xtrend
```

# Classical multiplicative decomposition (seasonal component)

- Compute seasonal component by taking seasonal means

```
> hsales.xmmean <- tapply(hsales.xdetrend,  
+   cycle(hsales.xdetrend), mean, na.rm = TRUE)  
> hsales.xseas <- unsplit(hsales.xmmean, cycle(hsales.xdetrend))
```

# Classical multiplicative decomposition (the erratic component)

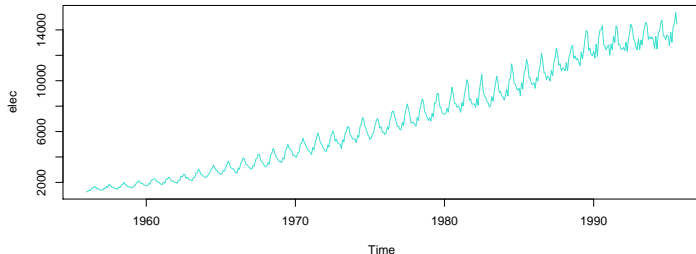
- What remains is the erratic component

```
> hsales.erratic <- hsales.xdetrend/hsales.seas
```

## Multiplicative or additive

- Use multiplicative when seasonal and irregular components are proportionate to the underlying trend, additive when their magnitude remains constant.
- An example when the multiplicative model may be appropriate, electricity production data:

```
> plot(elec, col = "turquoise")
```

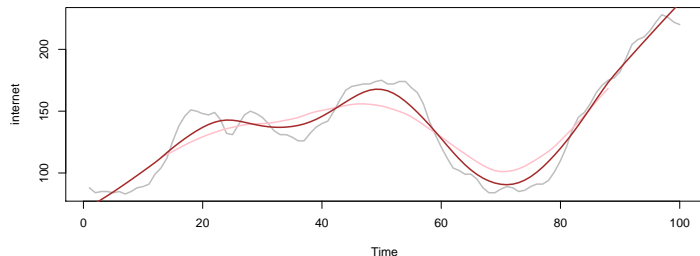


# Local regression

- For each observation take a subset of data centered around that observation (like moving averages).
- Give high weights to nearby observations and low weights to distant observation (like weighted moving average).
- Fit a weighted regression on this local subset.
- Take the predicted value from this regression as the trend for this observation.
- Repeat

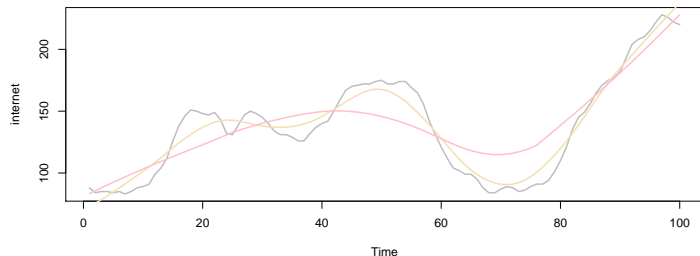
# Local regression smoothing

```
> plot(internet, col = "gray", lwd = 2)
> lines(filter(internet, rep(1/25, 25)), col = "pink",
+       lwd = 2)
> lines(lowess(internet, f = 0.25), col = "brown",
+       lwd = 2)
```



## Degrees of smoothing

```
> plot(internet, col = "gray", lwd = 2)
> lines(lowess(internet, f = 0.25), col = "wheat",
+       lwd = 2)
> lines(lowess(internet, f = 0.5), col = "pink",
+       lwd = 2)
```

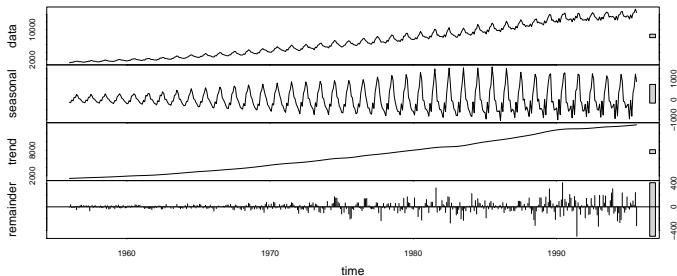


# The STL decomposition method

- 1 Use loess on the detrended seasonal subseries to compute the seasonal component.
- 2 Use loess on the deseasonalised series to compute the trend component.
- 3 Iterate.
- 4 The actual process is more complicated. See textbook.

# STL decomposition of electricity production

```
> plot(stl(elec, s.window = 4))
```



# Forecasting using simple exponential smoothing



$$\begin{aligned}F_{t+1} &= F_t + \alpha(Y_t - F_t) \\ &= \alpha Y_t + (1 - \alpha)F_t\end{aligned}$$

- The next period's forecast is a weighted average of last period's actual and last period's forecast. A low  $\alpha$  gives smoother forecasts, a high  $\alpha$  follows patterns better.

## Why is it called “exponential smoothing”?

$$\begin{aligned}F_{t+1} &= F_t + \alpha(Y_t - F_t) \\&= \alpha Y_t + (1 - \alpha)F_t \\&= \alpha Y_t + (1 - \alpha)[\alpha Y_{t-1} + (1 - \alpha)F_{t-1}] \\&= \alpha \sum_{i=0}^n (1 - \alpha)^i Y_{t-i} + (1 - \alpha)^{n+1} F_{t-n}\end{aligned}$$

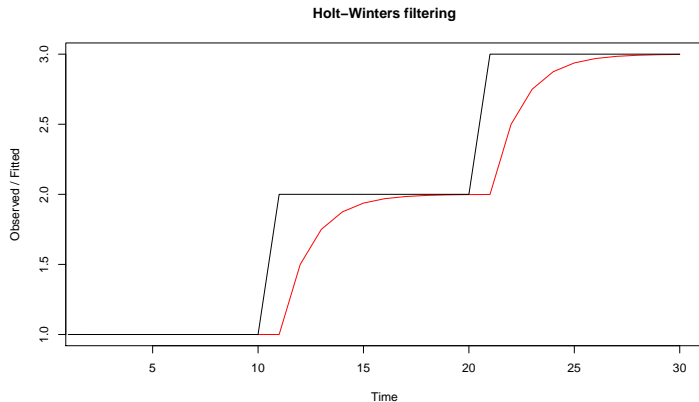
So the current forecast is a weighted average of old actuals, with the weights decreasing exponentially.

# Forecasting with simple exponential smoothing

- How do we get  $F_1$ ?
  - Use  $F_1 = Y_1$ .
  - Use results from decomposition methods.
- How do we get  $\alpha$ ?
  - Choose a value based on intuition.
  - Choose a value to achieve a good fit on test data.

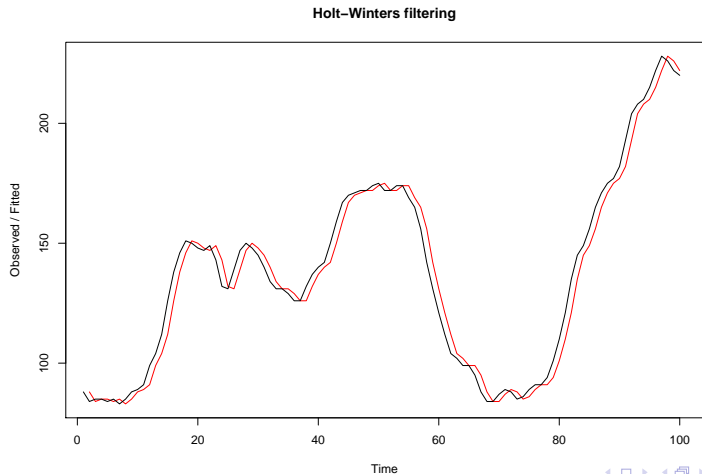
## SES lags behind jumps

```
> x <- c(rep(1, 10), rep(2, 10), rep(3, 10))  
> plot(HoltWinters(x, alpha = 0.5, beta = FALSE,  
+       gamma = FALSE))
```



## SES lags behind trends

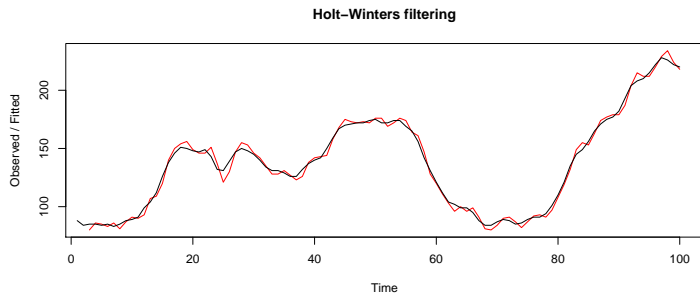
```
> plot(HoltWinters(internet, beta = FALSE,  
+       gamma = FALSE))
```



## Following trends: Holt's linear method

$$L_t = \alpha Y_t + (1 - \alpha)(L_{t-1} + b_{t-1})$$
$$b_t = \beta(L_t - L_{t-1}) + (1 - \beta)b_{t-1}$$
$$F_{t+m} = L_t + b_t m$$

```
> plot(HoltWinters(internet, gamma = FALSE))
```



## Seasonality: the multiplicative Holt-Winters model

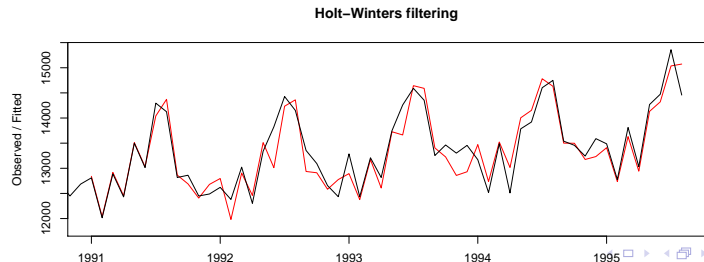
$$L_t = \alpha(Y_t/S_{t-s}) + (1 - \alpha)(L_{t-1} + b_{t-1})$$

$$b_t = \beta(L_t - L_{t-1}) + (1 - \beta)b_{t-1}$$

$$S_t = \gamma(Y_t/L_t) + (1 - \gamma)S_{t-s}$$

$$F_{t+m} = (L_t + b_t m)S_{t-s+m}$$

```
> elec.part = window(elec, start = 1990)
> plot(HoltWinters(elec.part, seasonal = "mult"))
```



## Seasonality: the additive Holt-Winters model

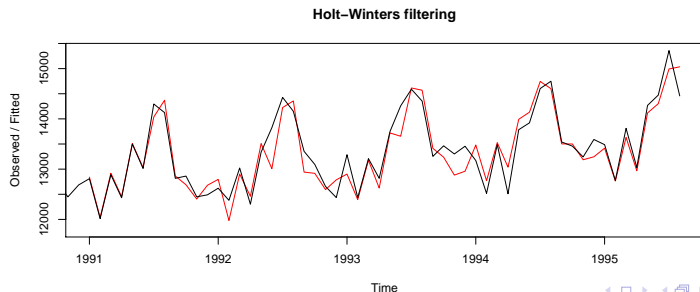
$$L_t = \alpha(Y_t - S_{t-s}) + (1 - \alpha)(L_{t-1} + b_{t-1})$$

$$b_t = \beta(L_t - L_{t-1}) + (1 - \beta)b_{t-1}$$

$$S_t = \gamma(Y_t - L_t) + (1 - \gamma)S_{t-s}$$

$$F_{t+m} = L_t + b_t m + S_{t-s+m}$$

```
> plot(HoltWinters(elec.part, seasonal = "add"))
```



## Additive or Multiplicative?

```
> accuracy(HoltWinters(beer, seasonal = "mult"))
```

ME	RMSE	MAE	MPE
73.310695	104.928019	78.938160	49.506705
MAPE	MASE		
53.721903	4.883688		

```
> accuracy(HoltWinters(beer, seasonal = "add"))
```

ME	RMSE	MAE	MPE
74.118042	105.509976	79.581294	50.386547
MAPE	MASE		
54.471063	4.923477		

# Out of sample point forecasts

```
> beer.t <- window(beer, end = c(1994, 12))
> beer.h <- window(beer, start = c(1995, 1))
> beer.HW <- HoltWinters(beer.t)
> beer.f <- predict(beer.HW, 8)
> beer.f
```

```
      Jan      Feb      Mar      Apr      May
1995 144.4677 131.7392 155.7921 130.4854 127.7896
      Jun      Jul      Aug
1995 121.8538 124.6986 136.7367
```

```
> beer.h
```

```
      Jan Feb Mar Apr May Jun Jul Aug
1995 138 136 152 127 151 130 119 153
```

```
> accuracy(beer.f, beer.h)
```

```
      ME      RMSE      MAE      MPE
4.0546074 11.1162192  8.9155565  2.5856592
      MAPE      ACF1  Theil's U
6.2643304 -0.1979585  0.5656064
```

# Prediction intervals

- A 95% prediction interval is an interval constructed using a method that is guaranteed to produce an interval containing the actual future value in 95% of samples.
- The function `hw()` from the `forecast` package is a easy way to construct prediction intervals for Holt-Winters forecasting. Similar functions `holt()` and `ses()` are also provided.

```
> beer.f2 <- hw(beer.t, 8)
> beer.f2
```

	Point Forecast	Lo 80	Hi 80	Lo 95
Jan 1995	141.0515	131.0711	151.0319	125.7878
Feb 1995	134.7984	124.8036	144.7931	119.5127
Mar 1995	156.1973	146.1771	166.2174	140.8728
Apr 1995	142.6453	132.5856	152.7050	127.2603
May 1995	132.6270	122.5106	142.7434	117.1553
Jun 1995	127.0027	116.8097	137.1958	111.4138
Jul 1995	138.4441	128.1518	148.7364	122.7034
Aug 1995	138.5173	128.1007	148.9338	122.5865

Hi 95

Jan 1995	156.3152
Feb 1995	150.0840
Mar 1995	171.5217
Apr 1995	158.0302
May 1995	148.0987
Jun 1995	142.5916
Jul 1995	154.1848
Aug 1995	154.4480

# Visualising prediction intervals

```
> plot(beer.f2, include = 12, shadecols = hsv(h = 0.9,  
+      s = c(0.3, 0.4)))
```

