Forecasting time series using R

Professor Rob J Hyndman

27 October 2011

MONASH University

Outline

Australian GDP

```
ausgdp <- ts(scan("gdp.dat"),frequency=4,
                             start=1971+2/4)
```

Australian GDP

```
ausgdp <- ts(scan("gdp.dat"),frequency=4,
                              start=1971+2/4)
```

- Class: `ts`
- Print and plotting methods available.

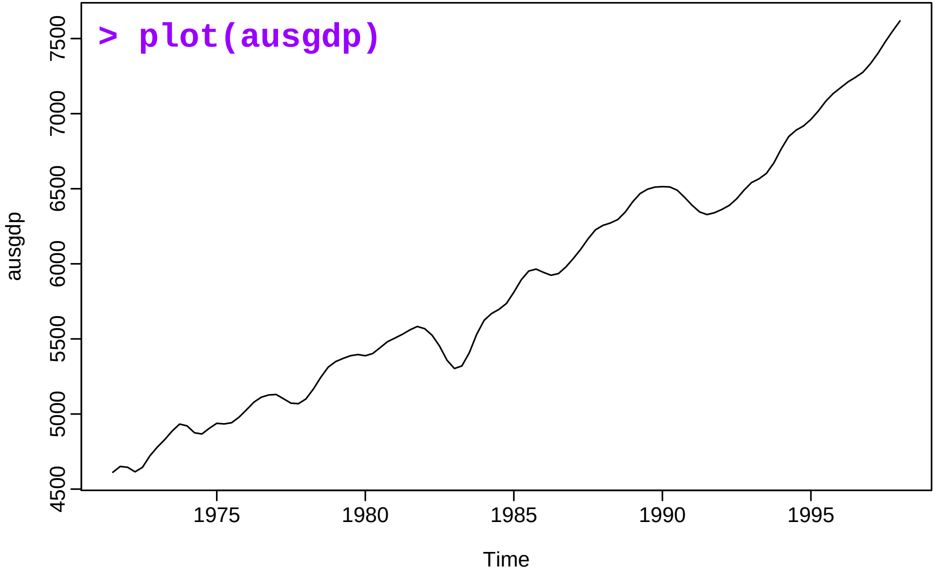
Australian GDP

```
ausgdp <- ts(scan("gdp.dat"),frequency=4,
                              start=1971+2/4)
```

- Class: `ts`
- Print and plotting methods available.

```
> ausgdp
     Qtr1 Qtr2 Qtr3 Qtr4
1971           4612 4651
1972 4645 4615 4645 4722
1973 4780 4830 4887 4933
1974 4921 4875 4867 4905
1975 4938 4934 4942 4979
1976 5028 5079 5112 5127
1977 5130 5101 5072 5069
```

Australian GDP



Residential electricity sales

```
> elecsales
Time Series:
Start = 1989
End = 2008
Frequency = 1
 [1] 2354.34 2379.71 2318.52 2468.99 2386.09 2569.47
 [7] 2575.72 2762.72 2844.50 3000.70 3108.10 3357.50
[13] 3075.70 3180.60 3221.60 3176.20 3430.60 3527.48
[19] 3637.89 3655.00
```

Useful packages

Time series task view: `http://cran.r-project.org/web/views/TimeSeries.html`

Useful packages

Time series task view: `http://cran.r-project.org/web/views/TimeSeries.html`

forecast	for forecasting functions
tseries	for unit root tests and GARCH models
Mcomp	for the M-competition and M3-competition data
fma	for data from Makridakis, Wheelwright & Hyndman (1998)
expsmooth	for data from Hyndman et al. (2008)
fpp	for data from Hyndman & Athanasopoulos (forthcoming).

Outline

Some simple forecasting methods

Mean method

- Forecast of all future values is equal to mean of historical data $\{y_1, \ldots, y_n\}$.

Some simple forecasting methods

Mean method

- Forecast of all future values is equal to mean of historical data $\{y_1, \dots, y_n\}$.
- Forecasts: $\hat{y}_{n+h|n} = \bar{y} = (y_1 + \cdots + y_n)/n$

Some simple forecasting methods

Naïve method

- Forecasts equal to last observed value.

Some simple forecasting methods

Naïve method

- Forecasts equal to last observed value.
- Forecasts: $\hat{y}_{n+h|n} = y_n$.

Some simple forecasting methods

Naïve method

- Forecasts equal to last observed value.
- Forecasts: $\hat{y}_{n+h|n} = y_n$.
- Optimal for efficient stock markets.

Some simple forecasting methods

Seasonal naïve method

- Forecasts equal to last value from same season.

Some simple forecasting methods

Seasonal naïve method

- Forecasts equal to last value from same season.
- Forecasts: $\hat{y}_{n+h|n} = y_{n-m}$ where $m =$ seasonal period and $k = \lfloor (h-1)/m \rfloor + 1$.

Some simple forecasting methods

Drift method

- Forecasts equal to last value plus average change.

Some simple forecasting methods

Drift method

- Forecasts equal to last value plus average change.
- Forecasts:

$$\begin{aligned}\hat{y}_{n+h|n} &= y_n + \frac{h}{n-1}\sum_{t=2}^{n}(y_t - y_{t-1})\\ &= y_n + \frac{h}{n-1}(y_n - y_1).\end{aligned}$$

Some simple forecasting methods

Drift method

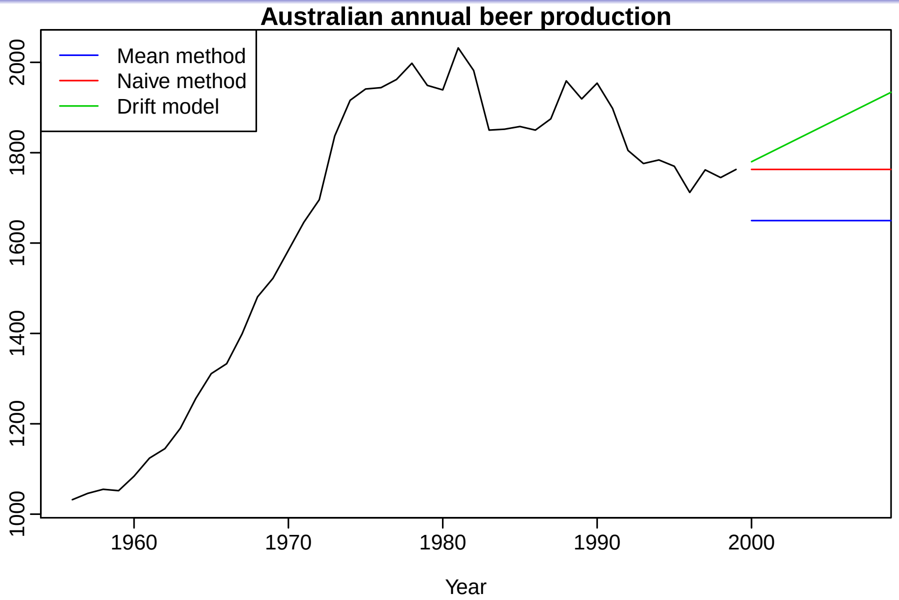
- Forecasts equal to last value plus average change.
- Forecasts:

$$\hat{y}_{n+h|n} = y_n + \frac{h}{n-1}\sum_{t=2}^{n}(y_t - y_{t-1})$$
$$= y_n + \frac{h}{n-1}(y_n - y_1).$$

- Equivalent to extrapolating a line drawn between first and last observations.

Some simple forecasting methods

Some simple forecasting methods



Some simple forecasting methods

- Mean: `meanf(x,h=20)`

Some simple forecasting methods

- Mean: `meanf(x,h=20)`
- Naive: `naive(x,h=20)` or `rwf(x,h=20)`

Some simple forecasting methods

- Mean: `meanf(x,h=20)`
- Naive: `naive(x,h=20)` or `rwf(x,h=20)`
- Seasonal naive: `snaive(x,h=20)`

Some simple forecasting methods

- Mean: `meanf(x,h=20)`
- Naive: `naive(x,h=20)` or `rwf(x,h=20)`
- Seasonal naive: `snaive(x,h=20)`
- Drift: `rwf(x,drift=TRUE,h=20)`

forecast objects in R

Functions that output a `forecast` object:

- `meanf()`
- `naive()`, `snaive()`
- `rwf()`
- `croston()`
- `stlf()`
- `ses()`
- `holt()`, `hw()`
- `splinef`
- `thetaf`
- `forecast()`

forecast objects in R

Functions that output a `forecast` object:

- `meanf()`
- `naive()`, `snaive()`
- `rwf()`
- `croston()`
- `stlf()`
- `ses()`
- `holt()`, `hw()`
- `splinef`
- `thetaf`
- `forecast()`

forecast class contains

- Original series

forecast objects in R

Functions that output a `forecast` object:

- `meanf()`
- `naive()`, `snaive()`
- `rwf()`
- `croston()`
- `stlf()`
- `ses()`
- `holt()`, `hw()`
- `splinef`
- `thetaf`
- `forecast()`

forecast class contains

- Original series
- Point forecasts

forecast objects in R

Functions that output a `forecast` object:

- `meanf()`
- `naive()`, `snaive()`
- `rwf()`
- `croston()`
- `stlf()`
- `ses()`
- `holt()`, `hw()`
- `splinef`
- `thetaf`
- `forecast()`

forecast class contains

- Original series
- Point forecasts
- Prediction interval

forecast objects in R

Functions that output a `forecast` object:

- `meanf()`
- `naive()`, `snaive()`
- `rwf()`
- `croston()`
- `stlf()`
- `ses()`
- `holt()`, `hw()`
- `splinef`
- `thetaf`
- `forecast()`

forecast class contains

- Original series
- Point forecasts
- Prediction interval
- Forecasting method used

forecast objects in R

Functions that output a `forecast` object:

- `meanf()`
- `naive()`, `snaive()`
- `rwf()`
- `croston()`
- `stlf()`
- `ses()`
- `holt()`, `hw()`
- `splinef`
- `thetaf`
- `forecast()`

forecast class contains

- Original series
- Point forecasts
- Prediction interval
- Forecasting method used
- Residuals and in-sample one-step forecasts

Outline

Measures of forecast accuracy

Let y_t denote the tth observation and f_t denote its forecast, where $t = 1, \dots, n$. Then the following measures are useful.

$$\text{MAE} = n^{-1} \sum_{t=1}^{n} |y_t - f_t|$$

$$\text{MSE} = n^{-1} \sum_{t=1}^{n} (y_t - f_t)^2 \qquad \text{RMSE} = \sqrt{n^{-1} \sum_{t=1}^{n} (y_t - f_t)^2}$$

$$\text{MAPE} = 100 n^{-1} \sum_{t=1}^{n} |y_t - f_t| / |y_t|$$

Measures of forecast accuracy

Let y_t denote the tth observation and f_t denote its forecast, where $t = 1, \dots, n$. Then the following measures are useful.

$$\text{MAE} = n^{-1} \sum_{t=1}^{n} |y_t - f_t|$$

$$\text{MSE} = n^{-1} \sum_{t=1}^{n} (y_t - f_t)^2 \qquad \text{RMSE} = \sqrt{n^{-1} \sum_{t=1}^{n} (y_t - f_t)^2}$$

$$\text{MAPE} = 100 n^{-1} \sum_{t=1}^{n} |y_t - f_t| / |y_t|$$

- MAE, MSE, RMSE are all scale dependent.

Measures of forecast accuracy

Let y_t denote the tth observation and f_t denote its forecast, where $t = 1, \ldots, n$. Then the following measures are useful.

$$\text{MAE} = n^{-1}\sum_{t=1}^{n}|y_t - f_t|$$

$$\text{MSE} = n^{-1}\sum_{t=1}^{n}(y_t - f_t)^2 \qquad \text{RMSE} = \sqrt{n^{-1}\sum_{t=1}^{n}(y_t - f_t)^2}$$

$$\text{MAPE} = 100n^{-1}\sum_{t=1}^{n}|y_t - f_t|/|y_t|$$

- MAE, MSE, RMSE are all scale dependent.
- MAPE is scale independent but is only sensible if $y_t \gg 0$ for all i, and y has a natural zero.

Measures of forecast accuracy

Mean Absolute Scaled Error

$$\text{MASE} = n^{-1} \sum_{t=1}^{n} |y_t - f_t| / q$$

where q is a stable measure of the scale of the time series $\{y_t\}$.

Measures of forecast accuracy

Mean Absolute Scaled Error

$$\text{MASE} = n^{-1} \sum_{t=1}^{n} |y_t - f_t|/q$$

where q is a stable measure of the scale of the time series $\{y_t\}$.

Proposed by Hyndman and Koehler (IJF, 2006)

Measures of forecast accuracy

Mean Absolute Scaled Error

$$\text{MASE} = n^{-1}\sum_{t=1}^{n}|y_t - f_t|/q$$

where q is a stable measure of the scale of the time series $\{y_t\}$.

For non-seasonal time series,

$$q = (n-1)^{-1}\sum_{t=2}^{n}|y_t - y_{t-1}|$$

works well. Then MASE is equivalent to MAE relative to a naive method.

Measures of forecast accuracy

Mean Absolute Scaled Error

$$\text{MASE} = n^{-1} \sum_{t=1}^{n} |y_t - f_t|/q$$

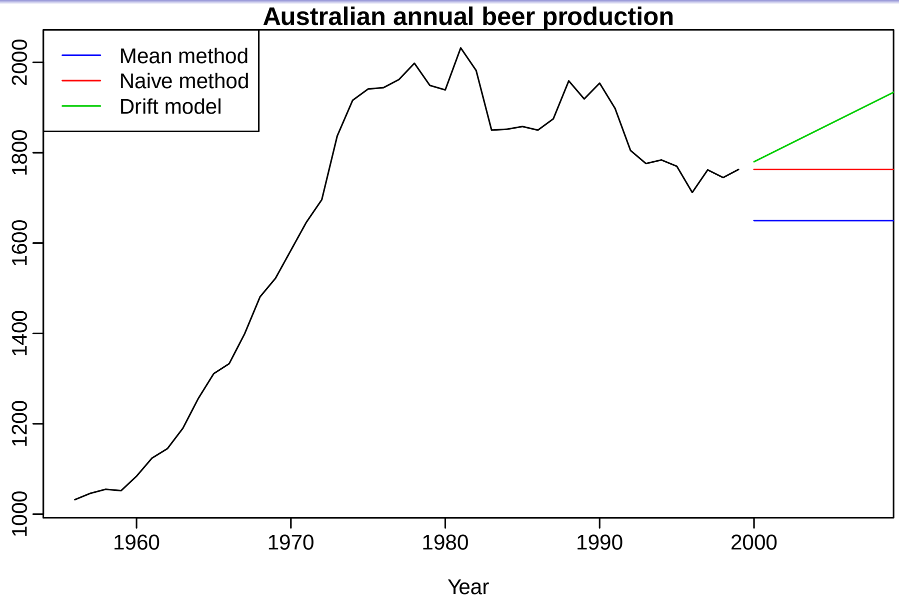
where q is a stable measure of the scale of the time series $\{y_t\}$.

For seasonal time series,

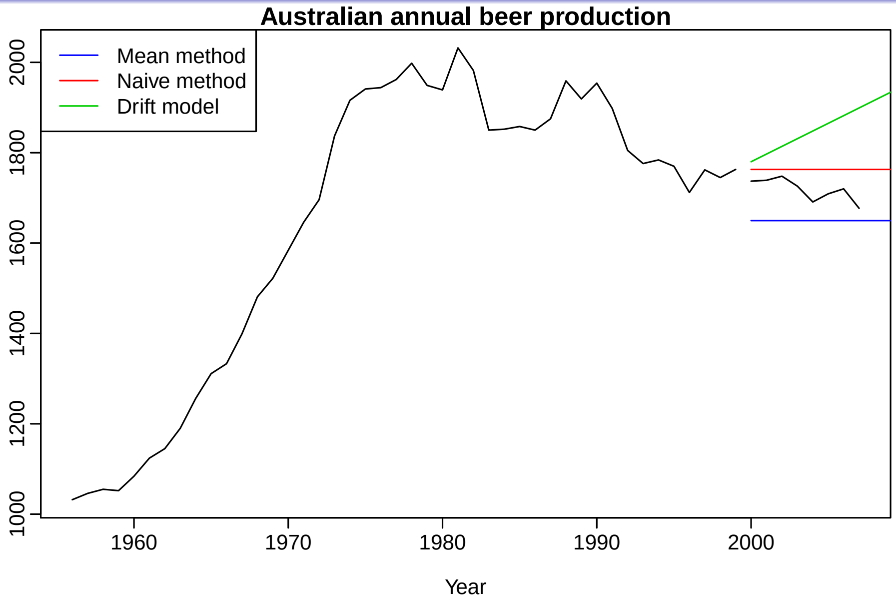
$$q = (n-m)^{-1} \sum_{t=m+1}^{n} |y_t - y_{t-m}|$$

works well. Then MASE is equivalent to MAE relative to a seasonal naive method.

Measures of forecast accuracy



Measures of forecast accuracy



Measures of forecast accuracy

Mean method

RMSE	MAE	MAPE	MASE
72.4223	68.6477	3.9775	1.5965

Naïve method

RMSE	MAE	MAPE	MASE
50.2382	44.6250	2.6156	1.0378

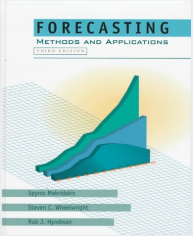
Drift method

RMSE	MAE	MAPE	MASE
134.6788	121.1250	7.0924	2.8169

Outline

Exponential smoothing

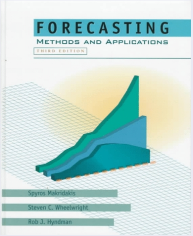
Classic Reference



Makridakis, Wheelwright and Hyndman (1998) *Forecasting: methods and applications*, 3rd ed., Wiley: NY.

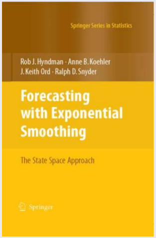
Exponential smoothing

Classic Reference



Makridakis, Wheelwright and Hyndman (1998) *Forecasting: methods and applications*, 3rd ed., Wiley: NY.

Current Reference



Hyndman, Koehler, Ord and Snyder (2008) *Forecasting with exponential smoothing: the state space approach*, Springer-Verlag: Berlin.

Exponential smoothing methods

Trend Component		Seasonal Component N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	A,A	A,M
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

Exponential smoothing methods

Trend Component		Seasonal Component N (None)	A (Additive)	M (Multiplicative)
N	(None)	**N,N**	N,A	N,M
A	(Additive)	A,N	A,A	A,M
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

(N,N): Simple exponential smoothing

Exponential smoothing methods

Trend Component		Seasonal Component N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	**A,N**	A,A	A,M
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

(N,N): Simple exponential smoothing

(A,N): Holt’s linear method

Exponential smoothing methods

Trend Component		Seasonal Component N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	**A,A**	A,M
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

(N,N): Simple exponential smoothing
(A,N): Holt's linear method
(A,A): Additive Holt-Winters' method

Exponential smoothing methods

Trend Component		Seasonal Component N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	A,A	**A,M**
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

(N,N): Simple exponential smoothing
(A,N): Holt's linear method
(A,A): Additive Holt-Winters' method
(A,M): Multiplicative Holt-Winters' method

Exponential smoothing methods

Trend Component		Seasonal Component N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	A,A	A,M
A_d	(Additive damped)	A_d,N	A_d,A	**A_d,M**
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

(N,N): Simple exponential smoothing
(A,N): Holt's linear method
(A,A): Additive Holt-Winters' method
(A,M): Multiplicative Holt-Winters' method
(A_d,M): Damped multiplicative Holt-Winters' method

Exponential smoothing methods

		Seasonal Component		
Trend Component		N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	A,A	A,M
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

(N,N): Simple exponential smoothing
(A,N): Holt's linear method
(A,A): Additive Holt-Winters' method
(A,M): Multiplicative Holt-Winters' method
(A_d,M): Damped multiplicative Holt-Winters' method

There are 15 separate exponential smoothing methods.

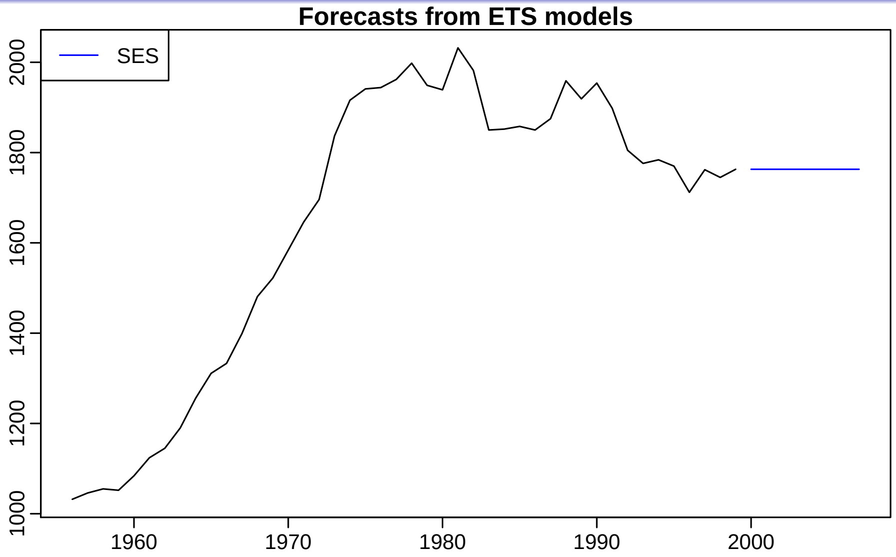
R functions

- `HoltWinters()` implements methods (N,N), (A,N), (A,A) and (A,M). Initial states are selected heuristically. Smoothing parameters are estimated by minimizing MSE.

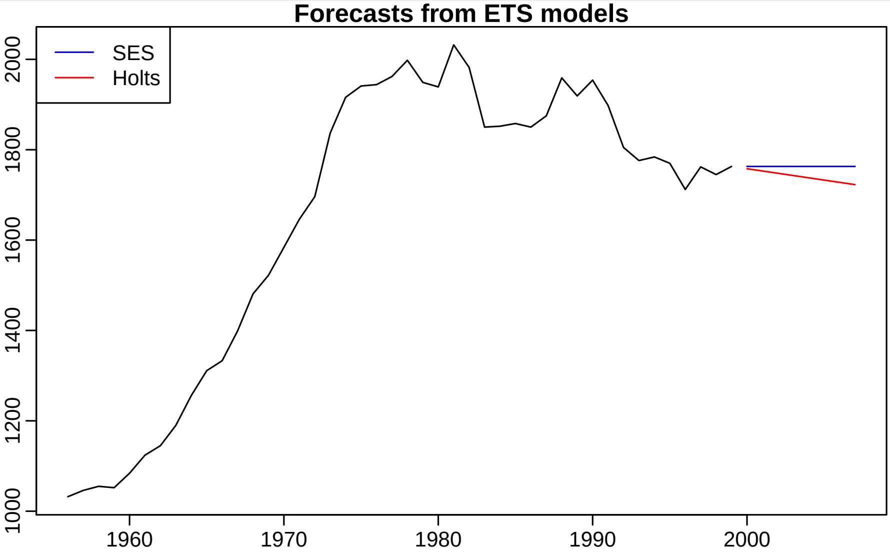
R functions

- `HoltWinters()` implements methods (N,N), (A,N), (A,A) and (A,M). Initial states are selected heuristically. Smoothing parameters are estimated by minimizing MSE.
- Use `ets(x,model="ZMA",damped=FALSE)` to estimate parameters for the (M,A) method. All other methods similarly. Initial states and smoothing parameters are estimated using maximum likelihood estimation (see later).

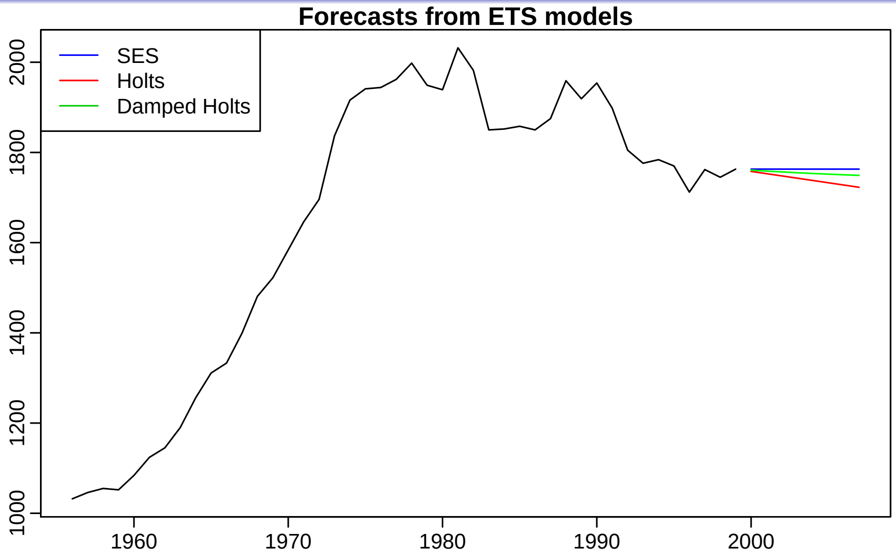
Exponential smoothing



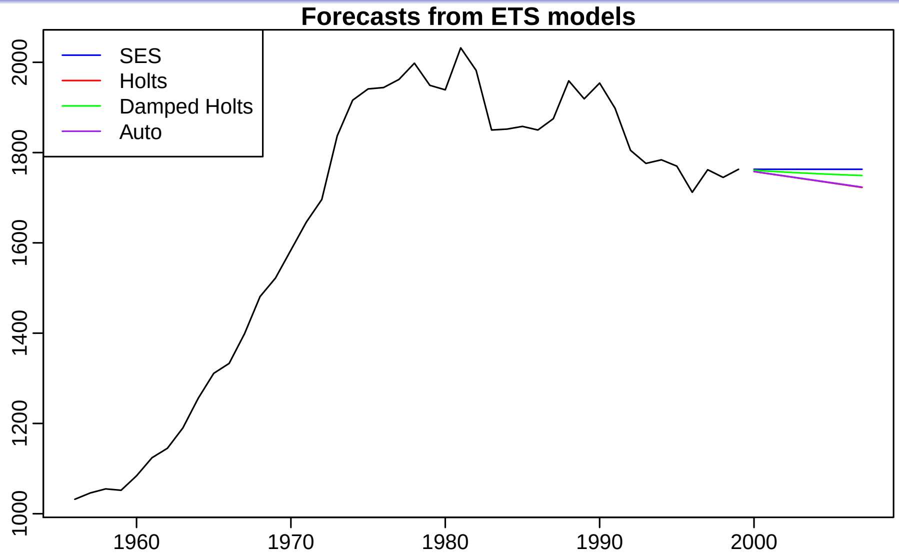
Exponential smoothing



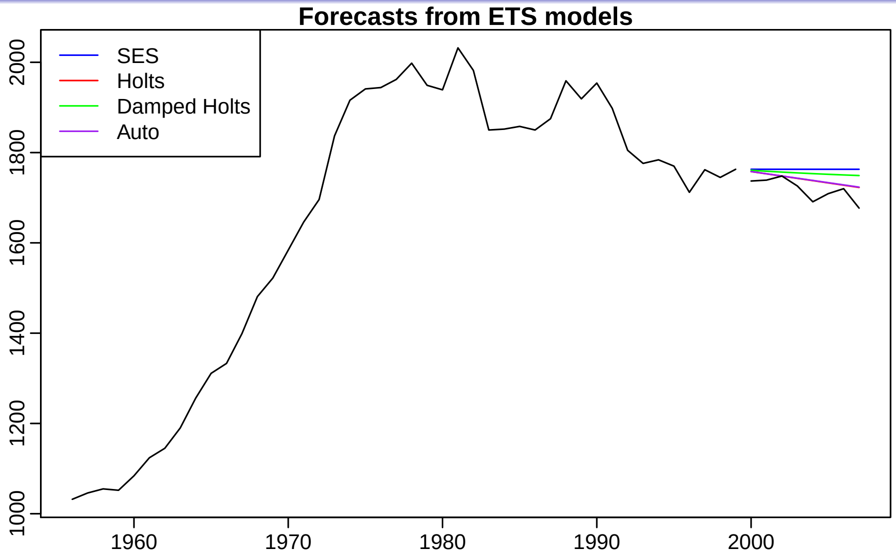
Exponential smoothing



Exponential smoothing



Exponential smoothing



Measures of forecast accuracy

Mean method

RMSE	MAE	MAPE	MASE
72.4223	68.6477	3.9775	1.5965

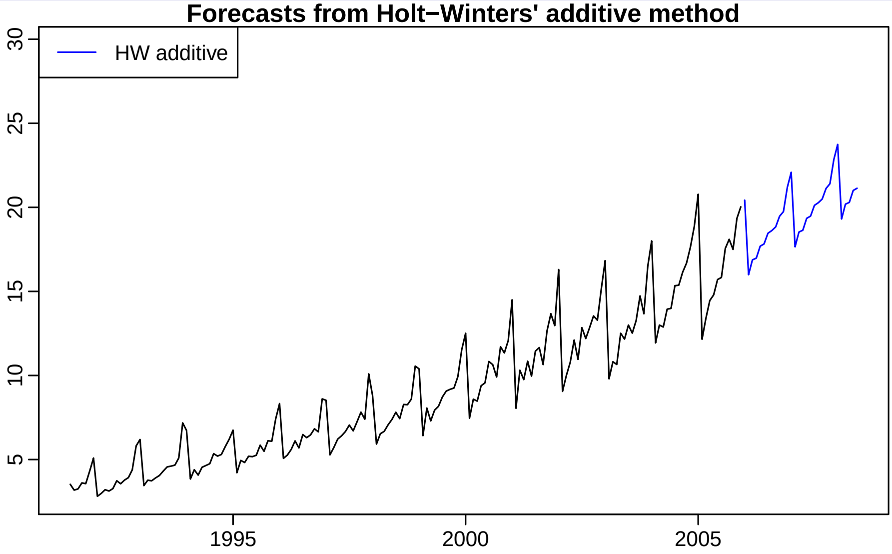
Naïve method

RMSE	MAE	MAPE	MASE
50.2382	44.6250	2.6156	1.0378

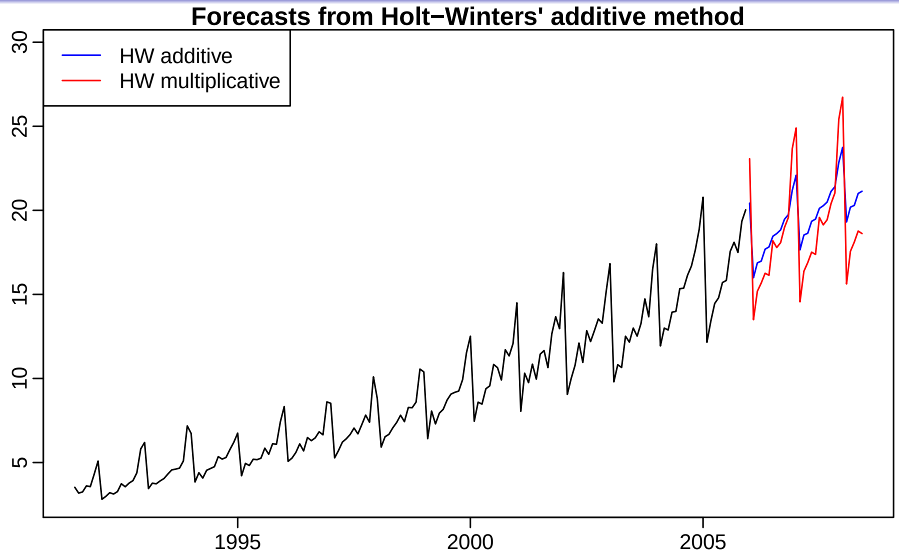
Auto ETS model

RMSE	MAE	MAPE	MASE
27.3974	22.4014	1.3150	0.5210

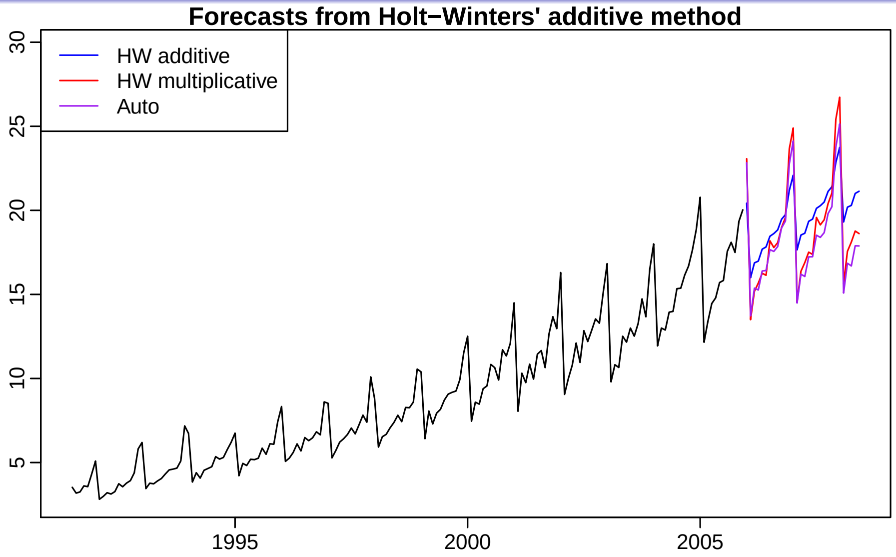
Exponential smoothing



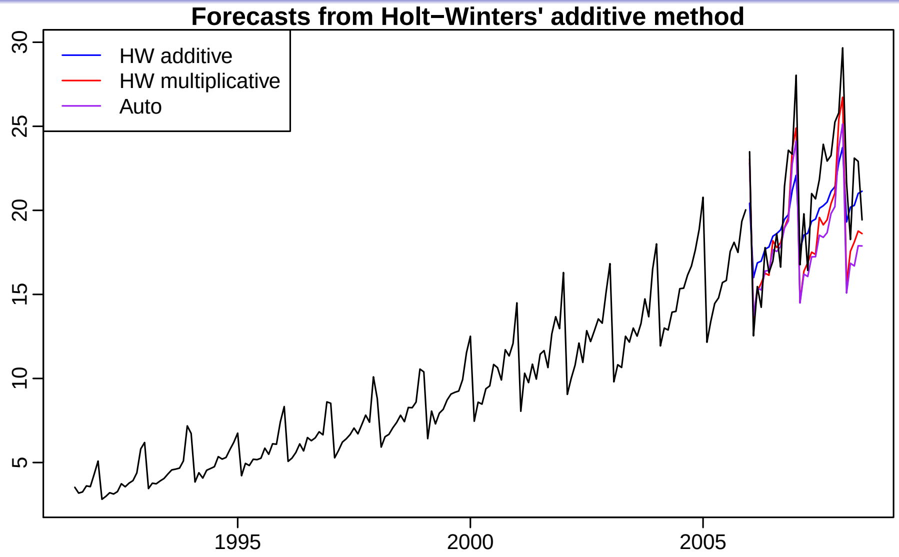
Exponential smoothing



Exponential smoothing



Exponential smoothing



Exponential smoothing

		Seasonal Component		
Trend Component		N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	A,A	A,M
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

Exponential smoothing

		Seasonal Component		
Trend Component		N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	A,A	A,M
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

General notation **ETS**(*Error,Trend,Seasonal*)

Exponential smoothing

Trend Component		Seasonal Component N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	A,A	A,M
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

General notation **ETS**(*Error,Trend,Seasonal*)

Exponen**T**ial **S**moothing

Exponential smoothing

Trend Component		Seasonal Component: N (None)	A (Additive)	M (Multiplicative)
N	(None)	**N,N**	N,A	N,M
A	(Additive)	A,N	A,A	A,M
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

General notation **ETS**(*Error,Trend,Seasonal*)
Exponen**T**ial **S**moothing

ETS(A,N,N): Simple exponential smoothing with additive errors

Exponential smoothing

		Seasonal Component		
Trend Component		N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	**A,N**	A,A	A,M
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

General notation **ETS**(*Error,Trend,Seasonal*)
Exponen**T**ial **S**moothing

ETS(A,A,N): Holt's linear method with additive errors

Exponential smoothing

Trend Component		Seasonal Component N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	**A,A**	A,M
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

General notation **ETS**(*Error,Trend,Seasonal*)
Exponen**T**ial **S**moothing

ETS(A,A,A): Additive Holt-Winters' method with additive errors

Exponential smoothing

Trend Component		Seasonal Component N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	A,A	**A,M**
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

General notation **ETS**(*Error,Trend,Seasonal*)
Exponen**T**ial **S**moothing

ETS(M,A,M): Multiplicative Holt-Winters' method with multiplicative errors

Exponential smoothing

		Seasonal Component		
Trend Component		N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	A,A	A,M
A_d	(Additive damped)	**A_d,N**	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

General notation **ETS**(*Error,Trend,Seasonal*)
Exponen**T**ial **S**moothing

ETS(A,A_d,N): Damped trend method with additive errors

Exponential smoothing

Trend Component		Seasonal Component N (None)	A (Additive)	M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	A,A	A,M
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

General notation **ETS**(*Error,Trend,Seasonal*)
Exponen**T**ial **S**moothing

There are 30 separate models in the ETS framework

Exponential smoothing

```
fit <- ets(a10train)
fit2 <- ets(a10train,model="MMM",damped=FALSE)
fcast1 <- forecast(fit, h=30)
fcast2 <- forecast(fit2, h=30)
```

Exponential smoothing

```
fit <- ets(a10train)
fit2 <- ets(a10train,model="MMM",damped=FALSE)
fcast1 <- forecast(fit, h=30)
fcast2 <- forecast(fit2, h=30)
```

```
ets(y, model="ZZZ", damped=NULL, alpha=NULL,
    beta=NULL, gamma=NULL, phi=NULL,
    additive.only=FALSE,
    lower=c(rep(0.0001,3),0.80),
    upper=c(rep(0.9999,3),0.98),
    opt.crit=c("lik","amse","mse","sigma"), nmse=3,
    bounds=c("both","usual","admissible"),
    ic=c("aic","aicc","bic"), restrict=TRUE)
```

Exponential smoothing

```
> fit
ETS(M,Ad,M)

  Smoothing parameters:
    alpha = 0.303
    beta  = 0.0205
    gamma = 1e-04
    phi   = 0.9797
  Initial states:
    l = 3.2875
    b = 0.0572
    s=0.9224 0.9252 0.8657 0.8763 0.7866 1.3149
           1.2457 1.0641 1.0456 0.989 0.9772 0.9874

  sigma:  0.0541

     AIC     AICc      BIC
636.3248 640.2479 690.0287
```

Exponential smoothing

```
> fit2
ETS(M,M,M)

  Smoothing parameters:
    alpha = 0.3899
    beta  = 0.0766
    gamma = 1e-04

  Initial states:
    l = 3.3312
    b = 1.0057
    s=0.923 0.9302 0.8569 0.8681 0.7796 1.334
           1.2548 1.0505 1.0392 0.9867 0.9783 0.9988

  sigma:  0.0565

     AIC     AICc      BIC
652.3502 655.8151 702.8951
```

Forecast accuracy

```
> accuracy(fcast1,a10test)
      RMSE        MAE       MAPE       MASE
 3.3031418  2.7067125  12.3566798  2.6505174

> accuracy(fcast2,a10test)
      RMSE        MAE       MAPE       MASE
 3.0001797  2.4143619  11.1033867  2.3642363
```

Exponential smoothing

`ets()` function

- Automatically chooses a model by default using the AIC, AICc or BIC.

Exponential smoothing

`ets()` function

- Automatically chooses a model by default using the AIC, AICc or BIC.
- Can handle any combination of trend, seasonality and damping

Exponential smoothing

`ets()` function

- Automatically chooses a model by default using the AIC, AICc or BIC.
- Can handle any combination of trend, seasonality and damping
- Produces prediction intervals for every model

Exponential smoothing

`ets()` function

- Automatically chooses a model by default using the AIC, AICc or BIC.
- Can handle any combination of trend, seasonality and damping
- Produces prediction intervals for every model
- Ensures the parameters are admissible (equivalent to invertible)

Exponential smoothing

`ets()` function

- Automatically chooses a model by default using the AIC, AICc or BIC.
- Can handle any combination of trend, seasonality and damping
- Produces prediction intervals for every model
- Ensures the parameters are admissible (equivalent to invertible)
- Produces an object of class `ets`.

Exponential smoothing

`ets()` function

- Automatically chooses a model by default using the AIC, AICc or BIC.
- Can handle any combination of trend, seasonality and damping
- Produces prediction intervals for every model
- Ensures the parameters are admissible (equivalent to invertible)
- Produces an object of class `ets`.

Exponential smoothing

`ets()` function

- Automatically chooses a model by default using the AIC, AICc or BIC.
- Can handle any combination of trend, seasonality and damping
- Produces prediction intervals for every model
- Ensures the parameters are admissible (equivalent to invertible)
- Produces an object of class `ets`.

Automatic ETS algorithm due to Hyndman et al (IJF, 2002), updated in Hyndman et al (2008).

Exponential smoothing

`ets` objects

- **Methods:** `coef()`, `plot()`, `summary()`, `residuals()`, `fitted()`, `simulate()` and `forecast()`

Exponential smoothing

`ets` objects

- **Methods:** `coef()`, `plot()`, `summary()`, `residuals()`, `fitted()`, `simulate()` and `forecast()`
- `plot()` function shows time plots of the original time series along with the extracted components (level, growth and seasonal).

Exponential smoothing

`ets()` function also allows refitting model to new data set.

```
> fit <- ets(a10train)

> test <- ets(a10test, model = fit)

> accuracy(test)
    RMSE       MAE      MAPE      MASE
 1.51968  1.28391  6.50663  0.41450

> accuracy(forecast(fit,30), a10test)
    RMSE       MAE      MAPE      MASE
 3.30314  2.70672 12.35668  2.65052
```

Automatic forecasting

Why use an automatic procedure?

1. Most users are not very expert at fitting time series models.

Automatic forecasting

Why use an automatic procedure?

1. Most users are not very expert at fitting time series models.
2. Most experts cannot beat the best automatic algorithms.

Automatic forecasting

Why use an automatic procedure?

1. Most users are not very expert at fitting time series models.
2. Most experts cannot beat the best automatic algorithms.
3. Many businesses and industries need thousands of forecasts every week/month.

Automatic forecasting

Why use an automatic procedure?

1. Most users are not very expert at fitting time series models.
2. Most experts cannot beat the best automatic algorithms.
3. Many businesses and industries need thousands of forecasts every week/month.
4. Some multivariate forecasting methods depend on many univariate forecasts.

Outline

Transformations to stabilize the variance

If the data show different variation at different levels of the series, then a transformation can be useful.

Transformations to stabilize the variance

If the data show different variation at different levels of the series, then a transformation can be useful.

Denote original observations as $y_1, \dots, y_n$ and transformed observations as $w_1, \dots, w_n$.

Transformations to stabilize the variance

If the data show different variation at different levels of the series, then a transformation can be useful.

Denote original observations as $y_1, \dots, y_n$ and transformed observations as $w_1, \dots, w_n$.

$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (y_t^\lambda - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

Transformations to stabilize the variance

If the data show different variation at different levels of the series, then a transformation can be useful.

Denote original observations as $y_1, \dots, y_n$ and transformed observations as $w_1, \dots, w_n$.

$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (y_t^\lambda - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

- $\lambda = 1$: (No substantive transformation)

Transformations to stabilize the variance

If the data show different variation at different levels of the series, then a transformation can be useful.

Denote original observations as $y_1, \dots, y_n$ and transformed observations as $w_1, \dots, w_n$.

$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (y_t^\lambda - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

- $\lambda = 1$: (No substantive transformation)
- $\lambda = \frac{1}{2}$: (Square root plus linear transformation)

Transformations to stabilize the variance

If the data show different variation at different levels of the series, then a transformation can be useful.

Denote original observations as $y_1, \dots, y_n$ and transformed observations as $w_1, \dots, w_n$.

$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (y_t^\lambda - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

- $\lambda = 1$: (No substantive transformation)
- $\lambda = \frac{1}{2}$: (Square root plus linear transformation)
- $\lambda = 0$: (Natural logarithm)

Transformations to stabilize the variance

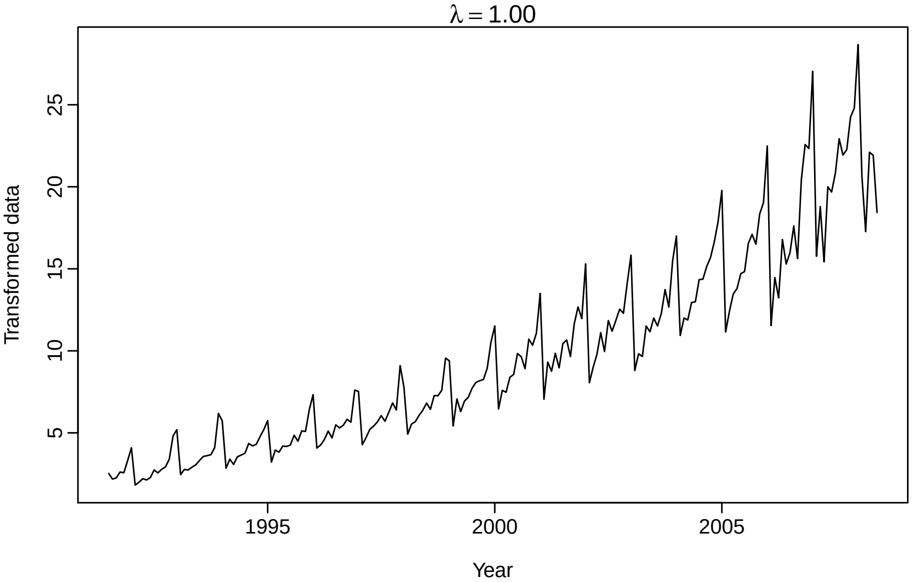
If the data show different variation at different levels of the series, then a transformation can be useful.

Denote original observations as $y_1, \dots, y_n$ and transformed observations as $w_1, \dots, w_n$.

$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (y_t^\lambda - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

- $\lambda = 1$: (No substantive transformation)
- $\lambda = \frac{1}{2}$: (Square root plus linear transformation)
- $\lambda = 0$: (Natural logarithm)
- $\lambda = -1$: (Inverse plus 1)

Box-Cox transformations



Back-transformation

We must reverse the transformation (or *back-transform*) to obtain forecasts on the original scale. The reverse Box-Cox transformations are given by

$$y_t = \begin{cases} \exp(w_t), & \lambda = 0; \\ (\lambda W_t + 1)^{1/\lambda}, & \lambda \neq 0. \end{cases}$$

Back-transformation

We must reverse the transformation (or *back-transform*) to obtain forecasts on the original scale. The reverse Box-Cox transformations are given by

$$y_t = \begin{cases} \exp(w_t), & \lambda = 0; \\ (\lambda W_t + 1)^{1/\lambda}, & \lambda \neq 0. \end{cases}$$

```
lam <- BoxCox.lambda(a10) # = 0.131
fit <- ets(a10, additive=TRUE, lambda=lam)
plot(forecast(fit))
plot(forecast(fit),include=60)
```

Outline

R functions

- The `arima()` function in the **stats** package provides seasonal and non-seasonal ARIMA model estimation including covariates.

R functions

- The `arima()` function in the **stats** package provides seasonal and non-seasonal ARIMA model estimation including covariates.
- However, it does not allow a constant unless the model is stationary

R functions

- The `arima()` function in the **stats** package provides seasonal and non-seasonal ARIMA model estimation including covariates.
- However, it does not allow a constant unless the model is stationary
- It does not return everything required for `forecast()`.

R functions

- The `arima()` function in the **stats** package provides seasonal and non-seasonal ARIMA model estimation including covariates.
- However, it does not allow a constant unless the model is stationary
- It does not return everything required for `forecast()`.
- It does not allow re-fitting a model to new data.

R functions

- The `arima()` function in the **stats** package provides seasonal and non-seasonal ARIMA model estimation including covariates.
- However, it does not allow a constant unless the model is stationary
- It does not return everything required for `forecast()`.
- It does not allow re-fitting a model to new data.
- Use the `Arima()` function in the **forecast** package which acts as a wrapper to `arima()`.

R functions

- The `arima()` function in the **stats** package provides seasonal and non-seasonal ARIMA model estimation including covariates.
- However, it does not allow a constant unless the model is stationary
- It does not return everything required for `forecast()`.
- It does not allow re-fitting a model to new data.
- Use the `Arima()` function in the **forecast** package which acts as a wrapper to `arima()`.
- Or use the `auto.arima()` function in the **forecast** package.

R functions

```
auto.arima(x, d = NA, D = NA, max.p = 5, max.q = 5,
    max.P = 2, max.Q = 2, max.order = 5,
    start.p = 2, start.q = 2,
    start.P = 1, start.Q = 1,
    stationary = FALSE, ic = c("aic", "aicc", "bic"),
    stepwise = TRUE, trace = FALSE,
    approximation = (length(x)>100 | frequency(x)>12),
    xreg = NULL, test = c("kpss", "adf", "pp"),
    seasonal.test = c("ocsb", "ch"),
    allowdrift = TRUE, lambda = NULL)
```

R functions

```
auto.arima(x, d = NA, D = NA, max.p = 5, max.q = 5,
    max.P = 2, max.Q = 2, max.order = 5,
    start.p = 2, start.q = 2,
    start.P = 1, start.Q = 1,
    stationary = FALSE, ic = c("aic", "aicc", "bic"),
    stepwise = TRUE, trace = FALSE,
    approximation = (length(x)>100 | frequency(x)>12),
    xreg = NULL, test = c("kpss", "adf", "pp"),
    seasonal.test = c("ocsb", "ch"),
    allowdrift = TRUE, lambda = NULL)
```

Automatic ARIMA algorithm due to Hyndman and Khandakar (JSS, 2008).

ARIMA vs ETS

- Myth that ARIMA models are more general than exponential smoothing.

ARIMA vs ETS

- Myth that ARIMA models are more general than exponential smoothing.
- Linear exponential smoothing models all special cases of ARIMA models.

ARIMA vs ETS

- Myth that ARIMA models are more general than exponential smoothing.
- Linear exponential smoothing models all special cases of ARIMA models.
- Non-linear exponential smoothing models have no equivalent ARIMA counterparts.

ARIMA vs ETS

- Myth that ARIMA models are more general than exponential smoothing.
- Linear exponential smoothing models all special cases of ARIMA models.
- Non-linear exponential smoothing models have no equivalent ARIMA counterparts.
- Many ARIMA models have no exponential smoothing counterparts.

ARIMA vs ETS

- Myth that ARIMA models are more general than exponential smoothing.
- Linear exponential smoothing models all special cases of ARIMA models.
- Non-linear exponential smoothing models have no equivalent ARIMA counterparts.
- Many ARIMA models have no exponential smoothing counterparts.
- ETS models all non-stationary. Models with seasonality or non-damped trend (or both) have two unit roots; all other models have one unit root.

Outline

Difficult seasonality

High frequency data

- `ets()` has maximum period 24

Difficult seasonality

High frequency data

- `ets()` has maximum period 24
- `Arima()` has maximum period 350, but usually runs out of memory if period > 200.

Difficult seasonality

High frequency data

- `ets()` has maximum period 24
- `Arima()` has maximum period 350, but usually runs out of memory if period > 200.
- `stl()` allows a decomposition of any frequency.

Difficult seasonality

High frequency data

- `ets()` has maximum period 24
- `Arima()` has maximum period 350, but usually runs out of memory if period > 200.
- `stl()` allows a decomposition of any frequency.

Difficult seasonality

High frequency data

- `ets()` has maximum period 24
- `Arima()` has maximum period 350, but usually runs out of memory if period > 200.
- `stl()` allows a decomposition of any frequency.

Multiple seasonal periods

- `dshw()` will allow two seasonal periods.

Difficult seasonality

High frequency data

- `ets()` has maximum period 24
- `Arima()` has maximum period 350, but usually runs out of memory if period > 200.
- `stl()` allows a decomposition of any frequency.

Multiple seasonal periods

- `dshw()` will allow two seasonal periods.
- Some new functions coming soon!

Outline

forecast() function

- Takes a time series or time series model as its main argument

forecast() function

- Takes a time series or time series model as its main argument
- Methods for objects of class `ts`, `ets`, `Arima`, `ar`, `HoltWinters`, `fracdiff`, `StructTS`, `stl`, etc.

forecast() function

- Takes a time series or time series model as its main argument
- Methods for objects of class `ts`, `ets`, `Arima`, `ar`, `HoltWinters`, `fracdiff`, `StructTS`, `stl`, etc.
- Output as class `forecast`.

forecast() function

- Takes a time series or time series model as its main argument
- Methods for objects of class `ts`, `ets`, `Arima`, `ar`, `HoltWinters`, `fracdiff`, `StructTS`, `stl`, etc.
- Output as class `forecast`.
- If first argument is class `ts`, returns forecasts from automatic ETS algorithm if non-seasonal or seasonal period is less than 13. Otherwise uses `stlf()`.

forecast package

```
> forecast(a10)

         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Jul 2008       24.09965 22.27328 25.91349 21.17243 26.95919
Aug 2008       24.04731 22.14468 25.99928 21.13311 27.06038
Sep 2008       24.30525 22.32954 26.33671 21.25731 27.46929
Oct 2008       25.92093 23.70871 28.15997 22.45784 29.31921
Nov 2008       26.82656 24.50077 29.29492 23.30437 30.61327
Dec 2008       31.24163 28.40841 34.08156 27.04162 35.87970
Jan 2009       33.48664 30.33970 36.69329 28.78131 38.33964
Feb 2009       20.21047 18.20746 22.25788 17.18718 23.39883
Mar 2009       22.13953 19.91242 24.39955 18.83605 25.63848
Apr 2009       22.43394 20.05864 24.83489 18.95948 26.20000
May 2009       24.03782 21.42388 26.65545 20.12685 28.21718
Jun 2009       23.79650 21.09909 26.54213 19.76471 28.14571
Jul 2009       26.03920 23.05168 29.07854 21.48762 30.95417
Aug 2009       25.94239 22.81384 29.24078 21.27523 31.01868
Sep 2009       26.18084 23.01920 29.62207 21.32134 31.56815
```

Outline

Cross-validation

Standard cross-validation

A more sophisticated version of training/test sets.

- Select one observation for test set, and use remaining observations in training set. Compute error on test observation.

Cross-validation

Standard cross-validation

A more sophisticated version of training/test sets.

- Select one observation for test set, and use remaining observations in training set. Compute error on test observation.
- Repeat using each possible observation as the test set.

Cross-validation

Standard cross-validation

A more sophisticated version of training/test sets.

- Select one observation for test set, and use remaining observations in training set. Compute error on test observation.
- Repeat using each possible observation as the test set.
- Compute accuracy measure over all errors.

Cross-validation

Standard cross-validation

A more sophisticated version of training/test sets.

- Select one observation for test set, and use remaining observations in training set. Compute error on test observation.
- Repeat using each possible observation as the test set.
- Compute accuracy measure over all errors.
- Does not work for time series because we cannot use future observations to build a model.

Time series cross-validation

Assume k is the minimum number of observations for a training set.

- Select observation $k + i + 1$ for test set, and use observations at times $1, 2, \ldots, k + i$ to estimate model. Compute error on forecast for time $k + i$.

Time series cross-validation

Assume k is the minimum number of observations for a training set.

- Select observation $k+i+1$ for test set, and use observations at times $1, 2, \ldots, k+i$ to estimate model. Compute error on forecast for time $k+i$.
- Repeat for $i = 0, 1, \ldots, n-k-1$ where n is total number of observations.

Time series cross-validation

Assume k is the minimum number of observations for a training set.

- Select observation $k + i + 1$ for test set, and use observations at times $1, 2, \ldots, k + i$ to estimate model. Compute error on forecast for time $k + i$.
- Repeat for $i = 0, 1, \ldots, n - k - 1$ where n is total number of observations.
- Compute accuracy measure over all errors.

Time series cross-validation

Assume k is the minimum number of observations for a training set.

- Select observation $k + i + 1$ for test set, and use observations at times $1, 2, \dots, k + i$ to estimate model. Compute error on forecast for time $k + i$.
- Repeat for $i = 0, 1, \dots, n - k - 1$ where n is total number of observations.
- Compute accuracy measure over all errors.

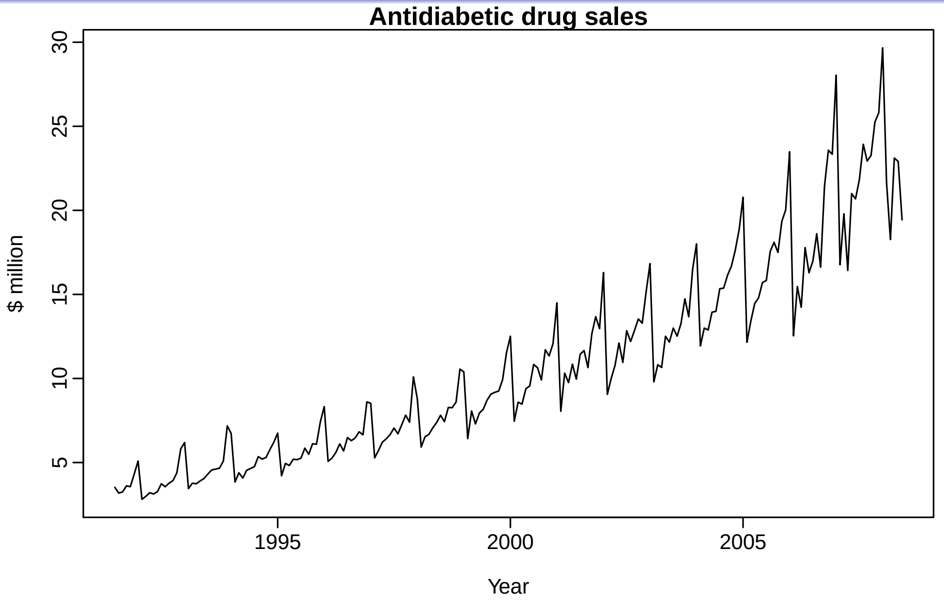
Time series cross-validation

Assume k is the minimum number of observations for a training set.

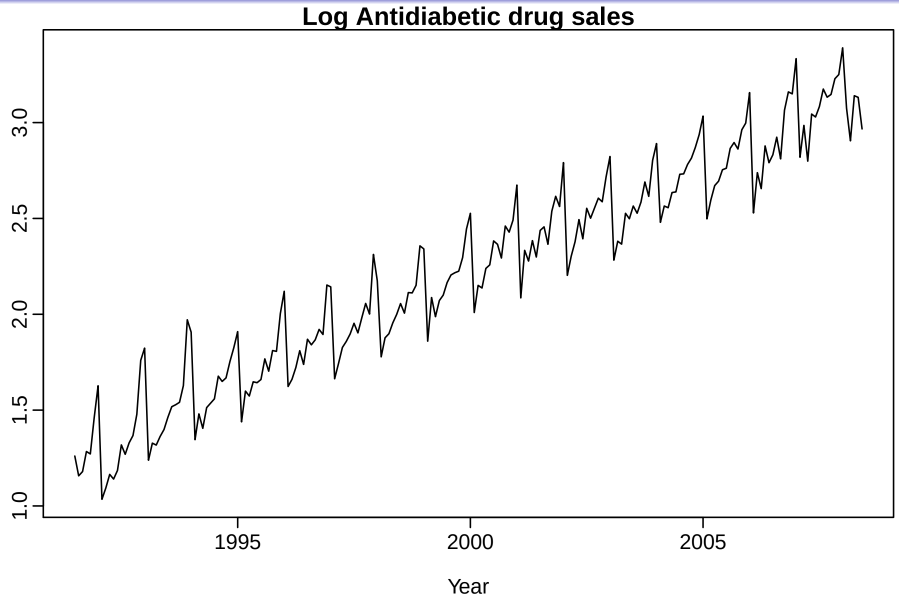
- Select observation $k+i+1$ for test set, and use observations at times $1, 2, \ldots, k+i$ to estimate model. Compute error on forecast for time $k+i$.
- Repeat for $i = 0, 1, \ldots, n-k-1$ where n is total number of observations.
- Compute accuracy measure over all errors.

Also called **rolling forecasting origin** because the origin $(k+i-1)$ at which forecast is based rolls forward in time.

Example: Pharmaceutical sales



Example: Pharmaceutical sales



Example: Pharmaceutical sales

Which of these models is best?

1. Linear model with trend and seasonal dummies applied to log data.

Example: Pharmaceutical sales

Which of these models is best?

1. Linear model with trend and seasonal dummies applied to log data.
2. ARIMA model applied to log data

Example: Pharmaceutical sales

Which of these models is best?

1. Linear model with trend and seasonal dummies applied to log data.
2. ARIMA model applied to log data
3. ETS model applied to original data

Example: Pharmaceutical sales

Which of these models is best?

1. Linear model with trend and seasonal dummies applied to log data.
2. ARIMA model applied to log data
3. ETS model applied to original data

Example: Pharmaceutical sales

Which of these models is best?

1. Linear model with trend and seasonal dummies applied to log data.
2. ARIMA model applied to log data
3. ETS model applied to original data

- Set $k = 48$ as minimum training set.

Example: Pharmaceutical sales

Which of these models is best?

1. Linear model with trend and seasonal dummies applied to log data.
2. ARIMA model applied to log data
3. ETS model applied to original data

- Set $k = 48$ as minimum training set.
- Forecast 12 steps ahead based on data to time $k + i - 1$ for $i = 1, 2, \ldots, 156$.

Example: Pharmaceutical sales

Which of these models is best?

1. Linear model with trend and seasonal dummies applied to log data.
2. ARIMA model applied to log data
3. ETS model applied to original data

- Set $k = 48$ as minimum training set.
- Forecast 12 steps ahead based on data to time $k + i - 1$ for $i = 1, 2, \ldots, 156$.
- Compare MAE values for each forecast horizon.

Example: Pharmaceutical sales

```
k <- 48
n <- length(a10)
mae1 <- mae2 <- mae3 <- matrix(NA,n-k-1,12)
for(i in 1:(n-k-1))
{
  xshort <- window(a10,end=1995+5/12+i/12)
  xnext <- window(a10,start=1995+(6+i)/12,end=1996+(5+i)/12)
  fit1 <- tslm(xshort ~ trend + season, lambda=0)
  fcast1 <- forecast(fit1,h=12)
  fit2 <- auto.arima(xshort, lambda=0)
  fcast2 <- forecast(fit2,h=12)
  fit3 <- ets(xshort)
  fcast3 <- forecast(fit3,h=12)
  mae1[i,] <- c(abs(fcast1$mean-xnext),rep(NA,12-length(xnext)))
  mae2[i,] <- c(abs(fcast2$mean-xnext),rep(NA,12-length(xnext)))
  mae3[i,] <- c(abs(fcast3$mean-xnext),rep(NA,12-length(xnext)))
}

plot(1:12,colSums(mae3,na.rm=TRUE),type="l",col=4,xlab="horizon",ylab="MAE")
lines(1:12,colSums(mae2,na.rm=TRUE),type="l",col=3)
lines(1:12,colSums(mae1,na.rm=TRUE),type="l",col=2)
legend("topleft",legend=c("LM","ARIMA","ETS"),col=2:4,lty=1)
```

Example: Pharmaceutical sales