Time series and forecasting in R

Rob J Hyndman

29 June 2008

MONASH University

Outline

1. Time series objects
2. Basic time series functionality
3. The forecast package
4. Exponential smoothing
5. ARIMA modelling
6. More from the forecast package
7. Time series packages on CRAN

Outline

1. Time series objects
2. Basic time series functionality
3. The forecast package
4. Exponential smoothing
5. ARIMA modelling
6. More from the forecast package
7. Time series packages on CRAN

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.

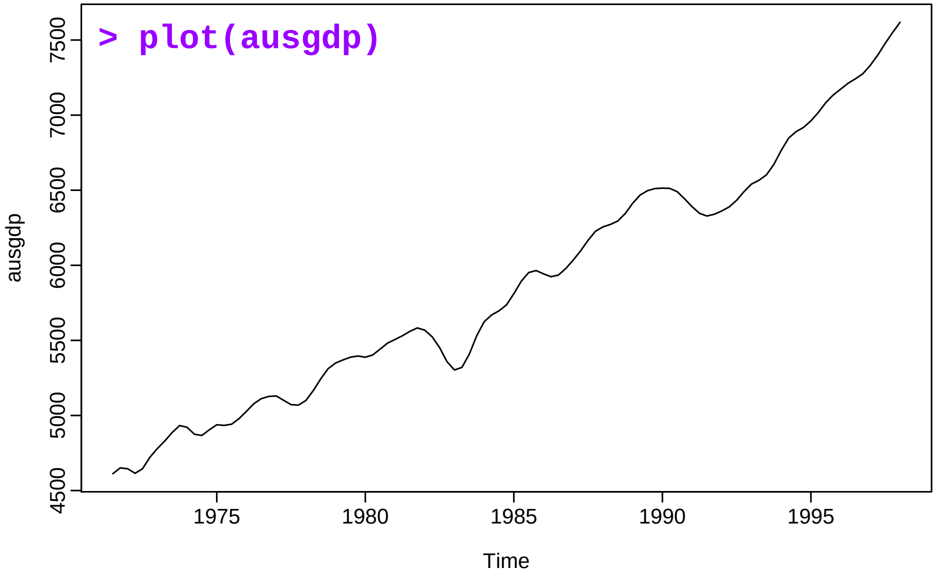
Australian GDP

```
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
1978 5100 5166 5244 5312
```

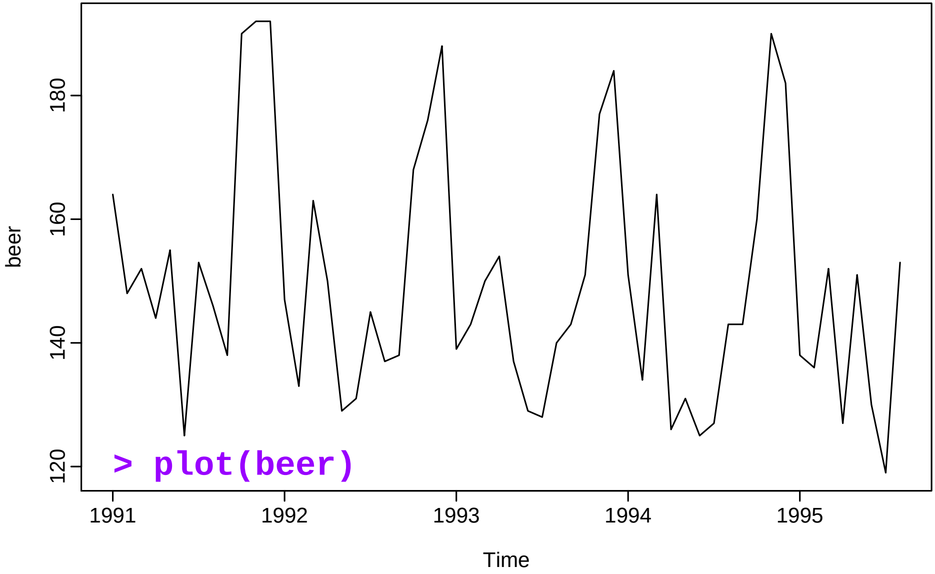
Australian GDP



Australian beer production

```
> beer
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1991 164 148 152 144 155 125 153 146 138 190 192 192
1992 147 133 163 150 129 131 145 137 138 168 176 188
1993 139 143 150 154 137 129 128 140 143 151 177 184
1994 151 134 164 126 131 125 127 143 143 160 190 182
1995 138 136 152 127 151 130 119 153
```

Australian beer production

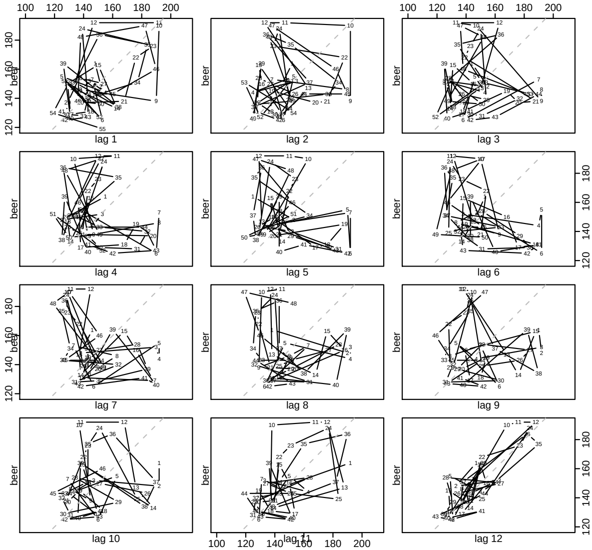


Outline

1. Time series objects
2. **Basic time series functionality**
3. The forecast package
4. Exponential smoothing
5. ARIMA modelling
6. More from the forecast package
7. Time series packages on CRAN

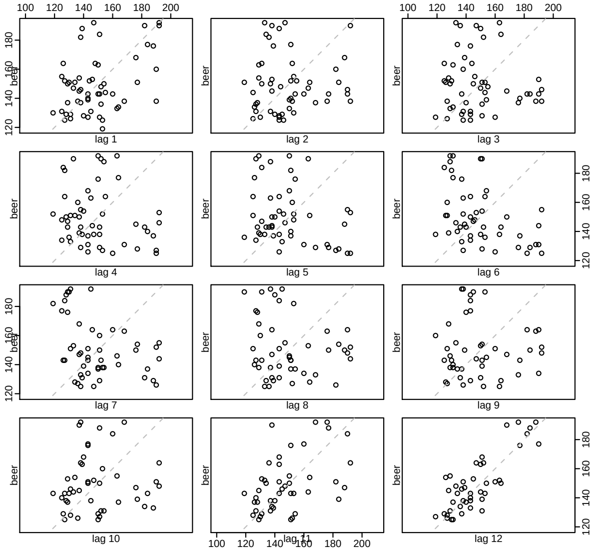
Lag plots

```
> lag.plot(beer,lags=12)
```



Lag plots

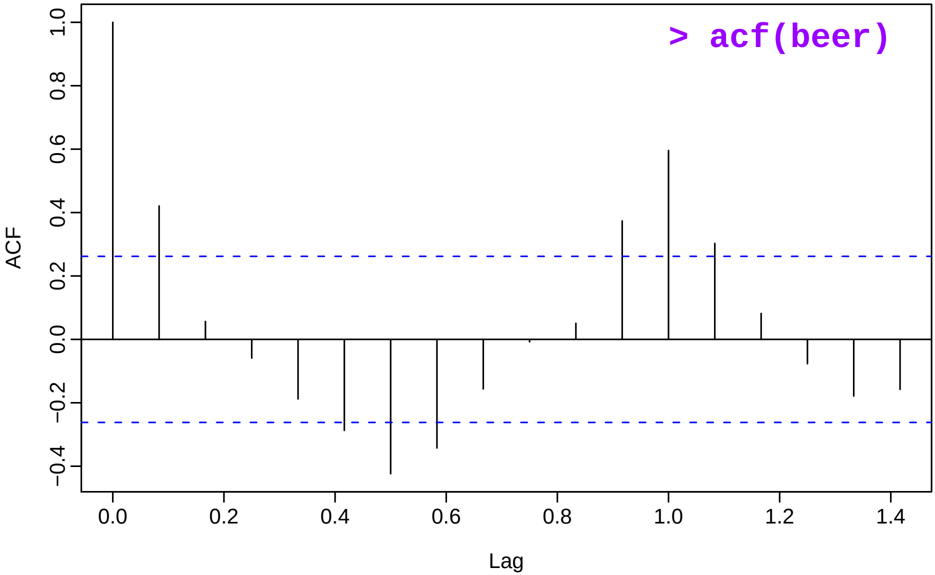
```
> lag.plot(beer,lags=12,do.lines=FALSE)
```



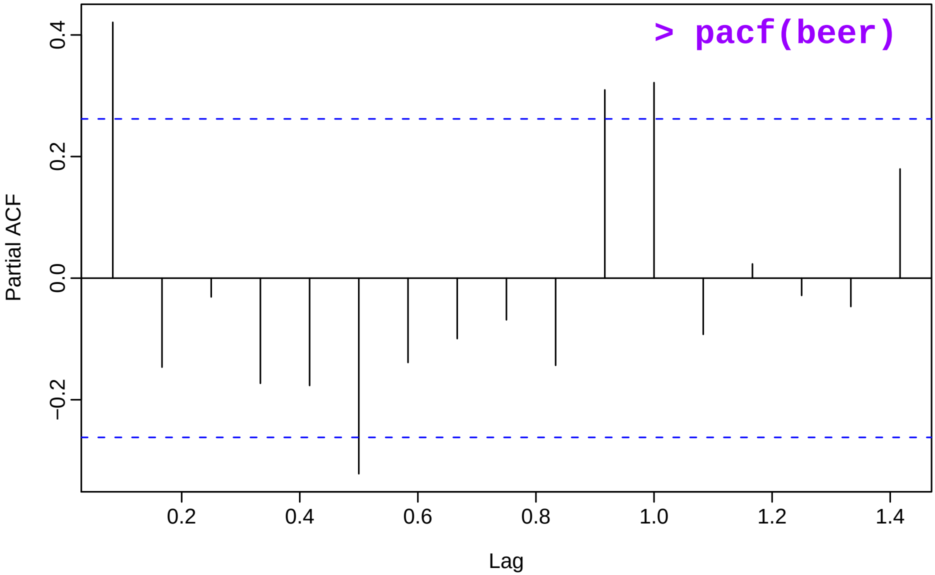
Lag plots

```
lag.plot(x, lags = 1, layout = NULL,
    set.lags = 1:lags, main = NULL,
    asp = 1, diag = TRUE,
    diag.col = "gray", type = "p",
    oma = NULL, ask = NULL,
    do.lines = (n <= 150), labels = do.lines,
    ...)
```

ACF



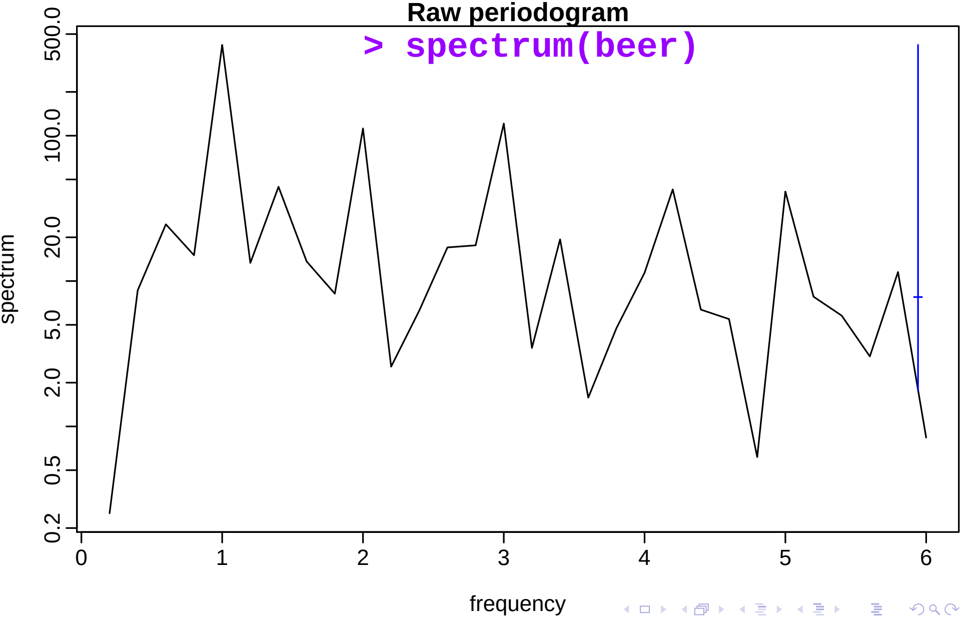
PACF



ACF/PACF

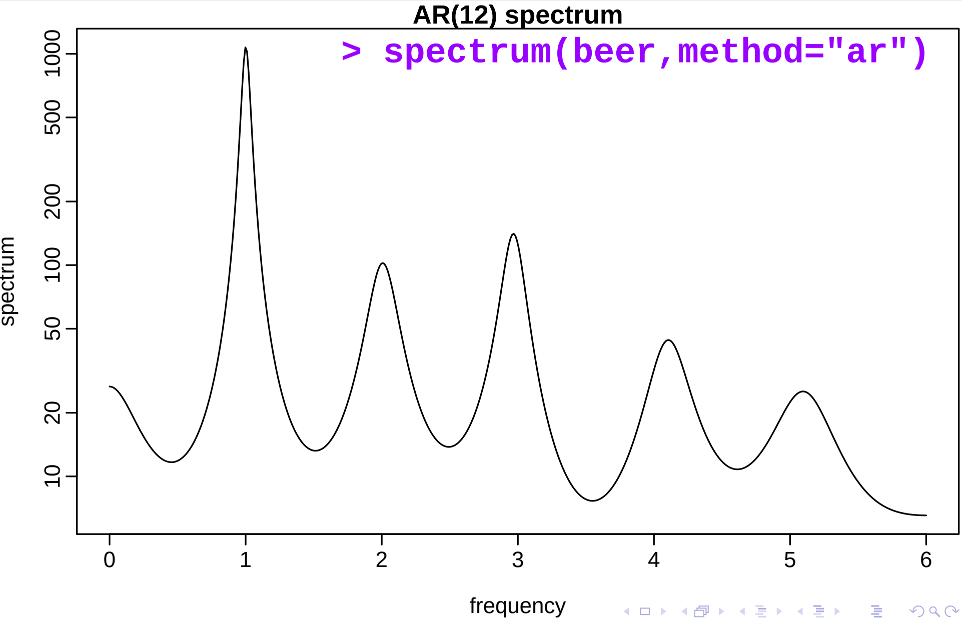
```
acf(x, lag.max = NULL,
    type = c("correlation", "covariance", "partial"),
    plot = TRUE, na.action = na.fail, demean = TRUE, ...)

pacf(x, lag.max, plot, na.action, ...)

ARMAacf(ar = numeric(0), ma = numeric(0), lag.max = r,
    pacf = FALSE)
```

Spectrum


Raw periodogram
> spectrum(beer)
spectrum
500.0
100.0
20.0
5.0
2.0
0.5
0.2
0
1
2
3
4
5
6
frequency

Spectrum

```
> spectrum(beer,method="ar")
```

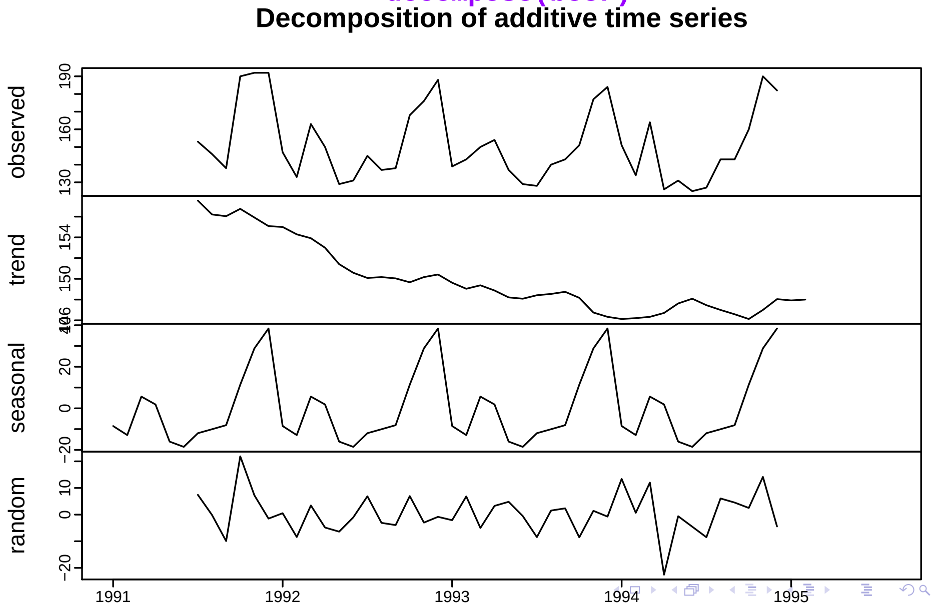


Spectrum

```
spectrum(x, ..., method = c("pgram", "ar"))

spec.pgram(x, spans = NULL, kernel, taper = 0.1,
    pad = 0, fast = TRUE, demean = FALSE,
    detrend = TRUE, plot = TRUE,
    na.action = na.fail, ...)

spec.ar(x, n.freq, order = NULL, plot = TRUE,
    na.action = na.fail,
    method = "yule-walker", ...)
```

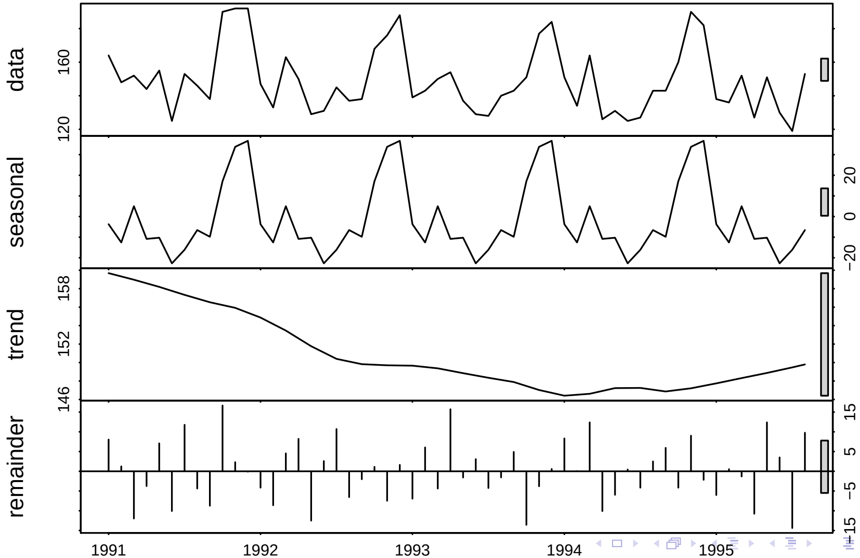
Classical decomposition

`decompose(beer)`



STL decomposition

```
plot(stl(beer,s.window="periodic"))
```



Decomposition

```
decompose(x, type = c("additive", "multiplicative"),
    filter = NULL)

stl(x, s.window, s.degree = 0,
    t.window = NULL, t.degree = 1,
    l.window = nextodd(period), l.degree = t.degree,
    s.jump = ceiling(s.window/10),
    t.jump = ceiling(t.window/10),
    l.jump = ceiling(l.window/10),
    robust = FALSE,
    inner = if(robust)  1 else 2,
    outer = if(robust) 15 else 0,
    na.action = na.fail)
```

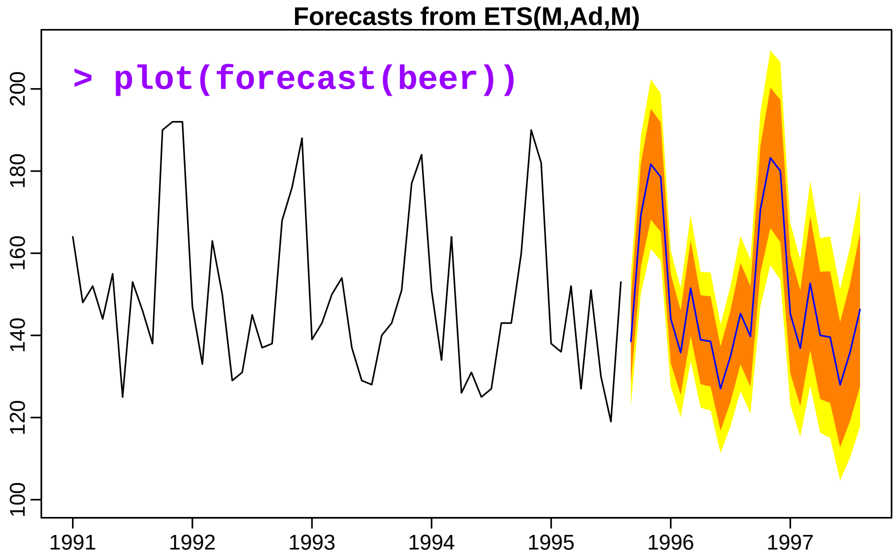
Outline

1. Time series objects
2. Basic time series functionality
3. **The forecast package**
4. Exponential smoothing
5. ARIMA modelling
6. More from the forecast package
7. Time series packages on CRAN

forecast package

```
> forecast(beer)
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Sep 1995       138.5042 128.2452 148.7632 122.8145 154.1940
Oct 1995       169.1987 156.6506 181.7468 150.0081 188.3894
Nov 1995       181.6725 168.1640 195.1810 161.0131 202.3320
Dec 1995       178.5394 165.2049 191.8738 158.1461 198.9327
Jan 1996       144.0816 133.2492 154.9140 127.5148 160.6483
Feb 1996       135.7967 125.4937 146.0996 120.0396 151.5537
Mar 1996       151.4813 139.8517 163.1110 133.6953 169.2673
Apr 1996       138.9345 128.1106 149.7584 122.3808 155.4882
May 1996       138.5279 127.5448 149.5110 121.7307 155.3250
Jun 1996       127.0269 116.7486 137.3052 111.3076 142.7462
Jul 1996       134.9452 123.7716 146.1187 117.8567 152.0337
Aug 1996       145.3088 132.9658 157.6518 126.4318 164.1858
Sep 1996       139.7348 127.4679 152.0018 120.9741 158.4955
Oct 1996       170.6709 155.2397 186.1020 147.0709 194.2708
Nov 1996       183.2204 166.1298 200.3110 157.0826 209.3582
Dec 1996       180.0290 162.6798 197.3783 153.4957 206.5624
Jan 1997       145.2589 130.7803 159.7374 123.1159 167.4019
```

forecast package



forecast package

```
> summary(forecast(beer))

Forecast method: ETS(M,Ad,M)

  Smoothing parameters:
    alpha = 0.0267
    beta  = 0.0232
    gamma = 0.025
    phi   = 0.98

  Initial states:
    l = 162.5752
    b = -0.1598
    s = 1.1979 1.2246 1.1452 0.9354 0.9754 0.9068
        0.8523 0.9296 0.9342 1.0160 0.9131 0.9696

  sigma:  0.0578

     AIC     AICc      BIC
499.0295 515.1347 533.4604
```

forecast package

- Automatic exponential smoothing state space modelling.

forecast package

- Automatic exponential smoothing state space modelling.
- Automatic ARIMA modelling

forecast package

- Automatic exponential smoothing state space modelling.
- Automatic ARIMA modelling
- Forecasting intermittent demand data using Croston's method

forecast package

- Automatic exponential smoothing state space modelling.
- Automatic ARIMA modelling
- Forecasting intermittent demand data using Croston's method
- Forecasting using Theta method

forecast package

- Automatic exponential smoothing state space modelling.
- Automatic ARIMA modelling
- Forecasting intermittent demand data using Croston's method
- Forecasting using Theta method
- Forecasting methods for most time series modelling functions including `arima()`, `ar()`, `StructTS()`, `ets()`, and others.

forecast package

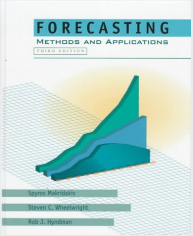
- Automatic exponential smoothing state space modelling.
- Automatic ARIMA modelling
- Forecasting intermittent demand data using Croston's method
- Forecasting using Theta method
- Forecasting methods for most time series modelling functions including `arima()`, `ar()`, `StructTS()`, `ets()`, and others.
- Part of the **forecasting** bundle along with **fma**, **expsmooth** and **Mcomp**.

Outline

1. Time series objects
2. Basic time series functionality
3. The forecast package
4. **Exponential smoothing**
5. ARIMA modelling
6. More from the forecast package
7. Time series packages on CRAN

Exponential smoothing

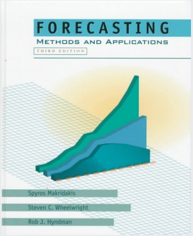
Classic Reference



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

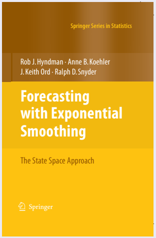
Exponential smoothing

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

- Until recently, there has been no stochastic modelling framework incorporating likelihood calculation, prediction intervals, etc.

Exponential smoothing

- Until recently, there has been no stochastic modelling framework incorporating likelihood calculation, prediction intervals, etc.
- Ord, Koehler & Snyder (JASA, 1997) and Hyndman, Koehler, Snyder and Grose (IJF, 2002) showed that all ES methods (including non-linear methods) are optimal forecasts from innovation state space models.

Exponential smoothing

- Until recently, there has been no stochastic modelling framework incorporating likelihood calculation, prediction intervals, etc.
- Ord, Koehler & Snyder (JASA, 1997) and Hyndman, Koehler, Snyder and Grose (IJF, 2002) showed that all ES methods (including non-linear methods) are optimal forecasts from innovation state space models.
- Hyndman et al. (2008) provides a comprehensive and up-to-date survey of the area.

Exponential smoothing

- Until recently, there has been no stochastic modelling framework incorporating likelihood calculation, prediction intervals, etc.
- Ord, Koehler & Snyder (JASA, 1997) and Hyndman, Koehler, Snyder and Grose (IJF, 2002) showed that all ES methods (including non-linear methods) are optimal forecasts from innovation state space models.
- Hyndman et al. (2008) provides a comprehensive and up-to-date survey of the area.
- The **forecast** package implements the framework of HKSO.

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

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**$(Error, Trend, Seasonal)$

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

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,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

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_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

Innovations state space models

No trend or seasonality and multiplicative errors

Example: ETS(M,N,N)

$$
\begin{aligned}
y_t &= \ell_{t-1}(1+\varepsilon_t) \\
\ell_t &= \alpha y_t + (1-\alpha)\ell_{t-1} \\
&= \ell_{t-1}(1+\alpha\varepsilon_t)
\end{aligned}
$$

$0 \le \alpha \le 1$

ε_t is white noise with mean zero.

Innovations state space models

No trend or seasonality and multiplicative errors

Example: ETS(M,N,N)

$$
\begin{aligned}
y_t &= \ell_{t-1}(1+\varepsilon_t) \\
\ell_t &= \alpha y_t + (1-\alpha)\ell_{t-1} \\
&= \ell_{t-1}(1+\alpha\varepsilon_t)
\end{aligned}
$$

$0 \leq \alpha \leq 1$

ε_t is white noise with mean zero.

All exponential smoothing models can be written using analogous state space equations.

Innovation state space models

Let $\mathbf{x}_t = (\ell_t, b_t, s_t, s_{t-1}, \dots, s_{t-m+1})$ and $\varepsilon_t \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$.

Example: Holt-Winters' multiplicative seasonal method

Example: ETS(M,A,M)

$$
\begin{aligned}
Y_t &= (\ell_{t-1} + b_{t-1})s_{t-m}(1 + \varepsilon_t) \\
\ell_t &= \alpha(y_t/s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
b_t &= \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1} \\
s_t &= \gamma(y_t/(\ell_{t-1} + b_{t-1})) + (1-\gamma)s_{t-m}
\end{aligned}
$$

where $0 \le \alpha \le 1, \quad 0 \le \beta \le \alpha, \quad 0 \le \gamma \le 1 - \alpha$ and m is the period of seasonality.

Exponential smoothing

From Hyndman et al. (2008):

- Apply each of 30 methods that are appropriate to the data. Optimize parameters and initial values using MLE (or some other criterion).

Exponential smoothing

From Hyndman et al. (2008):

- Apply each of 30 methods that are appropriate to the data. Optimize parameters and initial values using MLE (or some other criterion).
- Select best method using AIC:

$$\text{AIC} = -2\log(\text{Likelihood}) + 2p$$

where $p = \#$ parameters.

Exponential smoothing

From Hyndman et al. (2008):

- Apply each of 30 methods that are appropriate to the data. Optimize parameters and initial values using MLE (or some other criterion).
- Select best method using AIC:

$$\text{AIC} = -2\log(\text{Likelihood}) + 2p$$

  where $p = \#$ parameters.
- Produce forecasts using best method.

Exponential smoothing

From Hyndman et al. (2008):

- Apply each of 30 methods that are appropriate to the data. Optimize parameters and initial values using MLE (or some other criterion).
- Select best method using AIC:

$$\text{AIC} = -2\log(\text{Likelihood}) + 2p$$

  where $p = \#$ parameters.
- Produce forecasts using best method.
- Obtain prediction intervals using underlying state space model.

Exponential smoothing

From Hyndman et al. (2008):

- Apply each of 30 methods that are appropriate to the data. Optimize parameters and initial values using MLE (or some other criterion).
- Select best method using AIC:

$$\text{AIC} = -2\log(\text{Likelihood}) + 2p$$

  where $p = \#$ parameters.
- Produce forecasts using best method.
- Obtain prediction intervals using underlying state space model.

Exponential smoothing

From Hyndman et al. (2008):

- Apply each of 30 methods that are appropriate to the data. Optimize parameters and initial values using MLE (or some other criterion).
- Select best method using AIC:

$$\text{AIC} = -2\log(\text{Likelihood}) + 2p$$

  where $p = \#$ parameters.
- Produce forecasts using best method.
- Obtain prediction intervals using underlying state space model.

Method performed very well in M3 competition.

Exponential smoothing

```
fit <- ets(beer)
fit2 <- ets(beer,model="MNM",damped=FALSE)
fcast1 <- forecast(fit, h=24)
fcast2 <- forecast(fit2, h=24)
```

Exponential smoothing

```
fit <- ets(beer)
fit2 <- ets(beer,model="MNM",damped=FALSE)
fcast1 <- forecast(fit, h=24)
fcast2 <- forecast(fit2, h=24)
```

```
ets(y, model="ZZZ", damped=NULL, alpha=NULL, beta=NULL,
    gamma=NULL, phi=NULL, additive.only=FALSE,
    lower=c(rep(0.01,3), 0.8), upper=c(rep(0.99,3),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 = 0.0267
    beta  = 0.0232
    gamma = 0.025
    phi   = 0.98

  Initial states:
    l = 162.5752
    b = -0.1598
    s = 1.1979 1.2246 1.1452 0.9354 0.9754 0.9068
        0.8523 0.9296 0.9342 1.016 0.9131 0.9696

  sigma:  0.0578

     AIC     AICc      BIC
499.0295 515.1347 533.4604
```

Exponential smoothing

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

  Smoothing parameters:
    alpha = 0.247
    gamma = 0.01

  Initial states:
    l = 168.1208
    s = 1.2417 1.2148 1.1388 0.9217 0.9667 0.8934
        0.8506 0.9182 0.9262 1.049 0.9047 0.9743

  sigma:  0.0604

     AIC     AICc      BIC
500.0439 510.2878 528.3988
```

Exponential smoothing

`ets()` function

- Automatically chooses a model by default using the AIC

Exponential smoothing

`ets()` function

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

Exponential smoothing

`ets()` function

- Automatically chooses a model by default using the AIC
- 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
- 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
- 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` objects

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

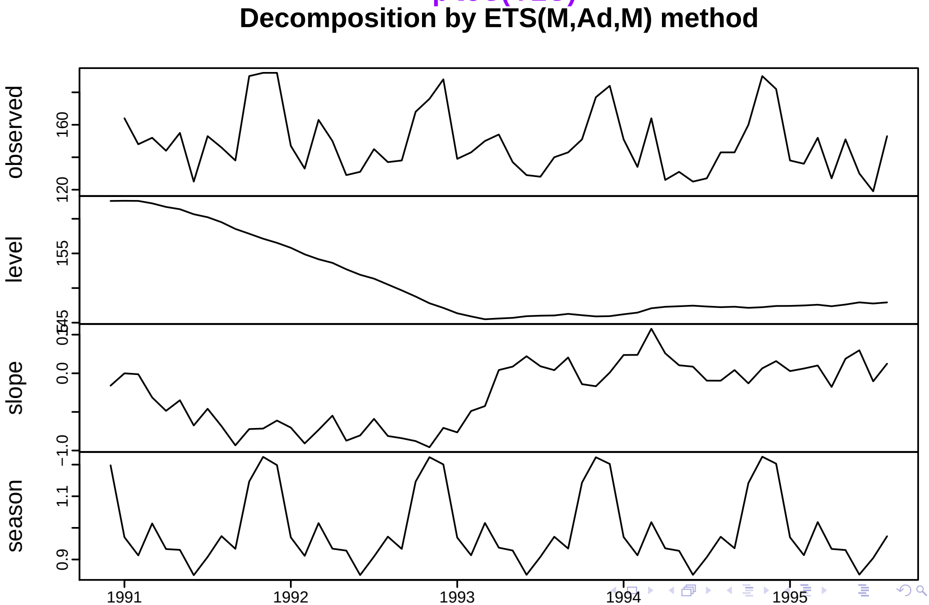
Exponential smoothing

`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

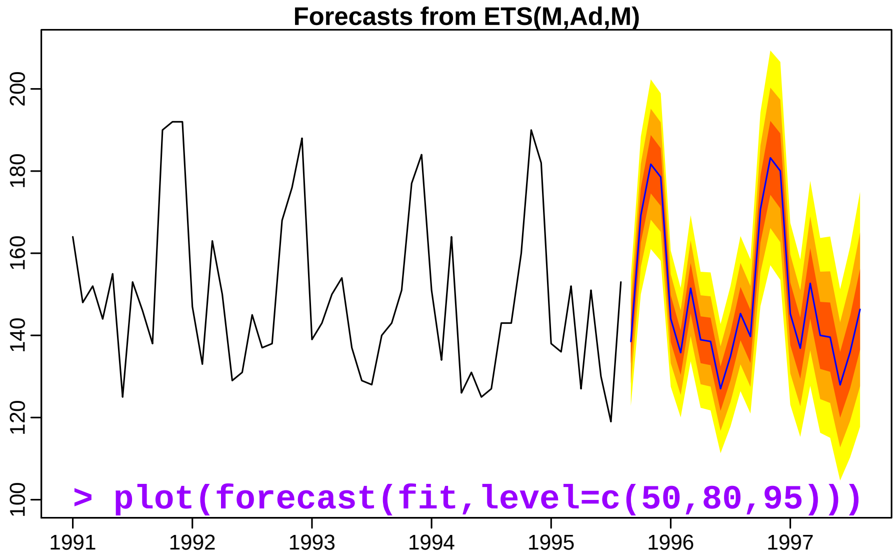
`plot(fit)`



Goodness-of-fit

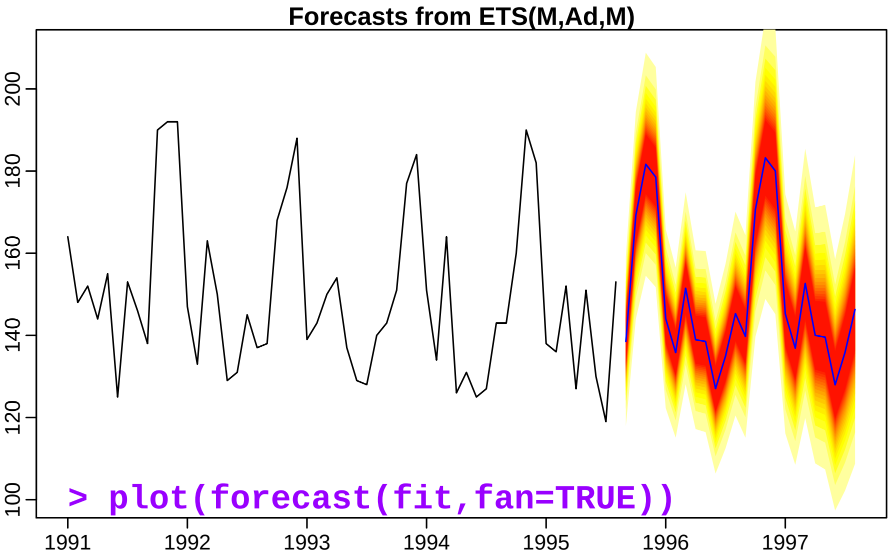
```
> accuracy(fit)
     ME    RMSE     MAE     MPE    MAPE    MASE
 0.0774  8.4156  7.0331 -0.2915  4.7883  0.4351

> accuracy(fit2)
     ME    RMSE     MAE     MPE    MAPE    MASE
-1.3884  9.0015  7.3303 -1.1945  5.0237  0.4535
```

Forecast intervals




```
> plot(forecast(fit,level=c(50,80,95)))
```

Forecast intervals




```
> plot(forecast(fit,fan=TRUE))
```

Exponential smoothing

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

```
> usfit <- ets(usnetelec[1:45])
> test <- ets(usnetelec[46:55], model = usfit)

> accuracy(test)
     ME     RMSE      MAE      MPE     MAPE     MASE
-4.3057 58.1668 43.5241 -0.1023   1.1758   0.5206

> accuracy(forecast(usfit,10), usnetelec[46:55])
      ME      RMSE       MAE      MPE      MAPE      MASE      ACF1 Theil's U
46.36580 65.55163 49.83883 1.25087 1.35781 0.72895 0.08899    0.73725
```

forecast package

forecast() function

- Takes either a time series as its main argument, or a time series model.

forecast package

forecast() function

- Takes either a time series as its main argument, or a time series model.
- Methods for objects of class `ts`, `ets`, `arima`, `HoltWinters`, `StructTS`, `ar` and others.

forecast package

forecast() function

- Takes either a time series as its main argument, or a time series model.
- Methods for objects of class `ts`, `ets`, `arima`, `HoltWinters`, `StructTS`, `ar` and others.
- If argument is `ts`, it uses `ets` model.

forecast package

forecast() function

- Takes either a time series as its main argument, or a time series model.
- Methods for objects of class `ts`, `ets`, `arima`, `HoltWinters`, `StructTS`, `ar` and others.
- If argument is `ts`, it uses `ets` model.
- Calls `predict()` when appropriate.

forecast package

forecast() function

- Takes either a time series as its main argument, or a time series model.
- Methods for objects of class `ts`, `ets`, `arima`, `HoltWinters`, `StructTS`, `ar` and others.
- If argument is `ts`, it uses `ets` model.
- Calls `predict()` when appropriate.
- Output as class `forecast`.

forecast package

forecast class contains

- Original series
- Point forecasts
- Prediction intervals
- Forecasting method used
- Forecasting model information
- Residuals
- One-step forecasts for observed data

forecast package

forecast class contains

- Original series
- Point forecasts
- Prediction intervals
- Forecasting method used
- Forecasting model information
- Residuals
- One-step forecasts for observed data

Methods applying to the forecast class:

- print
- plot
- summary

Outline

1 Time series objects

2 Basic time series functionality

3 The forecast package

4 Exponential smoothing

5 **ARIMA modelling**

6 More from the forecast package

7 Time series packages on CRAN

ARIMA modelling

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

ARIMA modelling

- 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

ARIMA modelling

- 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()`

ARIMA modelling

- 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.

ARIMA modelling

- 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.
- So I prefer the `Arima()` function in the **forecast** package which acts as a wrapper to `arima()`.

ARIMA modelling

- 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.
- So I prefer the `Arima()` function in the **forecast** package which acts as a wrapper to `arima()`.
- Even better, the `auto.arima()` function in the **forecast** package.

ARIMA modelling

```
> fit <- auto.arima(beer)
> fit
Series: beer
ARIMA(0,0,0)(1,0,0)[12] with non-zero mean

Coefficients:
        sar1  intercept
      0.8431   152.1132
s.e.  0.0590     5.1921

sigma^2 estimated as 122.1:  log likelihood = -221.44
AIC = 448.88   AICc = 449.34   BIC = 454.95
```

How does auto.arima() work?

A seasonal ARIMA process

$$\Phi(B^m)\phi(B)(1 - B^m)^D(1 - B)^d y_t = c + \Theta(B^m)\theta(B)\varepsilon_t$$

Need to select appropriate orders: p, q, P, Q, D, d

How does auto.arima() work?

A seasonal ARIMA process

$$\Phi(B^m)\phi(B)(1 - B^m)^D(1 - B)^d y_t = c + \Theta(B^m)\theta(B)\varepsilon_t$$

Need to select appropriate orders: p, q, P, Q, D, d

Use Hyndman and Khandakar (JSS, 2008) algorithm:

- Select no. differences d and D via unit root tests.
- Select p, q, P, Q by minimising AIC.
- Use stepwise search to traverse model space.

How does auto.arima() work?

$$\text{AIC} = -2\log(L) + 2(p + q + P + Q + k)$$

where L is the maximised likelihood fitted to the *differenced* data, $k = 1$ if $c \neq 0$ and $k = 0$ otherwise.

How does auto.arima() work?

$$\text{AIC} = -2\log(L) + 2(p + q + P + Q + k)$$

where L is the maximised likelihood fitted to the *differenced* data, $k = 1$ if $c \neq 0$ and $k = 0$ otherwise.

Step 1: Select current model (with smallest AIC) from:

ARIMA$(2, d, 2)(1, D, 1)_m$
ARIMA$(0, d, 0)(0, D, 0)_m$
ARIMA$(1, d, 0)(1, D, 0)_m$ if seasonal
ARIMA$(0, d, 1)(0, D, 1)_m$

How does auto.arima() work?

$$\text{AIC} = -2\log(L) + 2(p + q + P + Q + k)$$

where L is the maximised likelihood fitted to the *differenced* data, $k = 1$ if $c \neq 0$ and $k = 0$ otherwise.

Step 1: Select current model (with smallest AIC) from:
ARIMA$(2, d, 2)(1, D, 1)_m$
ARIMA$(0, d, 0)(0, D, 0)_m$
ARIMA$(1, d, 0)(1, D, 0)_m$ if seasonal
ARIMA$(0, d, 1)(0, D, 1)_m$

Step 2: Consider variations of current model:

- vary one of p, q, P, Q from current model by ± 1
- p, q both vary from current model by ± 1.
- P, Q both vary from current model by ± 1.
- Include/exclude c from current model

Model with lowest AIC becomes current model.

How does auto.arima() work?

$$\text{AIC} = -2\log(L) + 2(p + q + P + Q + k)$$

where L is the maximised likelihood fitted to the *differenced* data, $k = 1$ if $c \neq 0$ and $k = 0$ otherwise.

Step 1: Select current model (with smallest AIC) from:
$\text{ARIMA}(2, d, 2)(1, D, 1)_m$
$\text{ARIMA}(0, d, 0)(0, D, 0)_m$
$\text{ARIMA}(1, d, 0)(1, D, 0)_m$ if seasonal
$\text{ARIMA}(0, d, 1)(0, D, 1)_m$

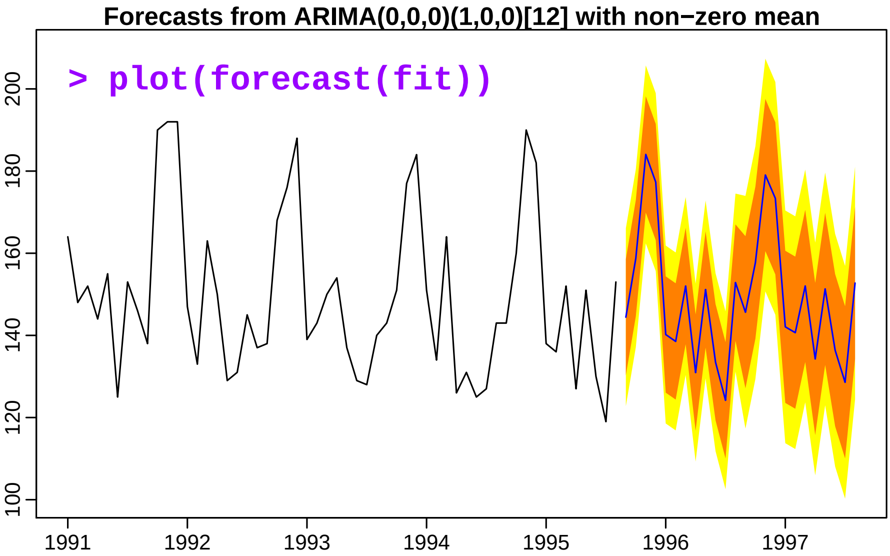
Step 2: Consider variations of current model:

- vary one of p, q, P, Q from current model by ± 1
- p, q both vary from current model by ± 1.
- P, Q both vary from current model by ± 1.
- Include/exclude c from current model

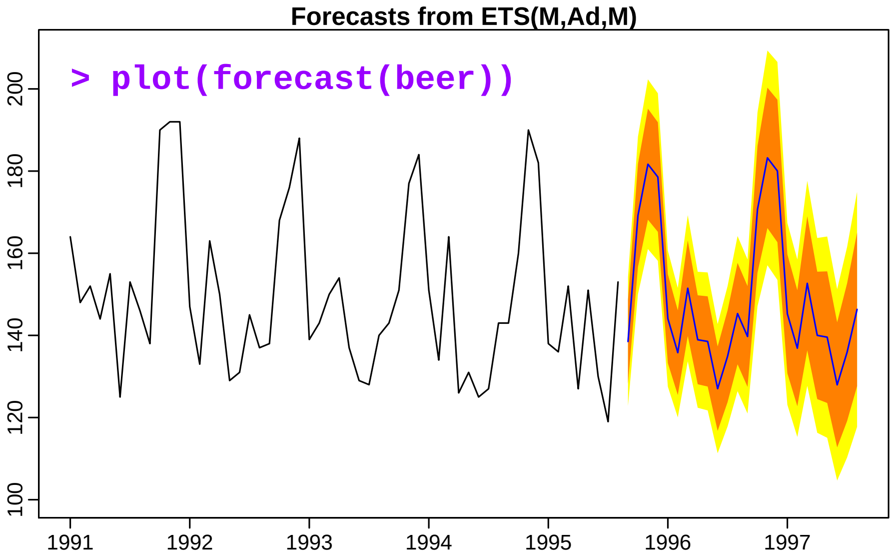
Model with lowest AIC becomes current model.

Repeat Step 2 until no lower AIC can be found.

ARIMA modelling



ARIMA modelling



ARIMA vs ETS

- Myth that ARIMA models more general than exponential smoothing.

ARIMA vs ETS

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

ARIMA vs ETS

- Myth that ARIMA models 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 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 which have no exponential smoothing counterparts.

ARIMA vs ETS

- Myth that ARIMA models 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 which 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—that is, non-seasonal models with either no trend or damped trend—have one unit root.

Outline

1. Time series objects
2. Basic time series functionality
3. The forecast package
4. Exponential smoothing
5. ARIMA modelling
6. **More from the forecast package**
7. Time series packages on CRAN

Other forecasting functions

croston() implements Croston's (1972) method for intermittent demand forecasting.

theta() provides forecasts from the Theta method.

splinef() gives cubic-spline forecasts, based on fitting a cubic spline to the historical data and extrapolating it linearly.

meanf() returns forecasts based on the historical mean.

rwf() gives "naïve" forecasts equal to the most recent observation assuming a random walk model.

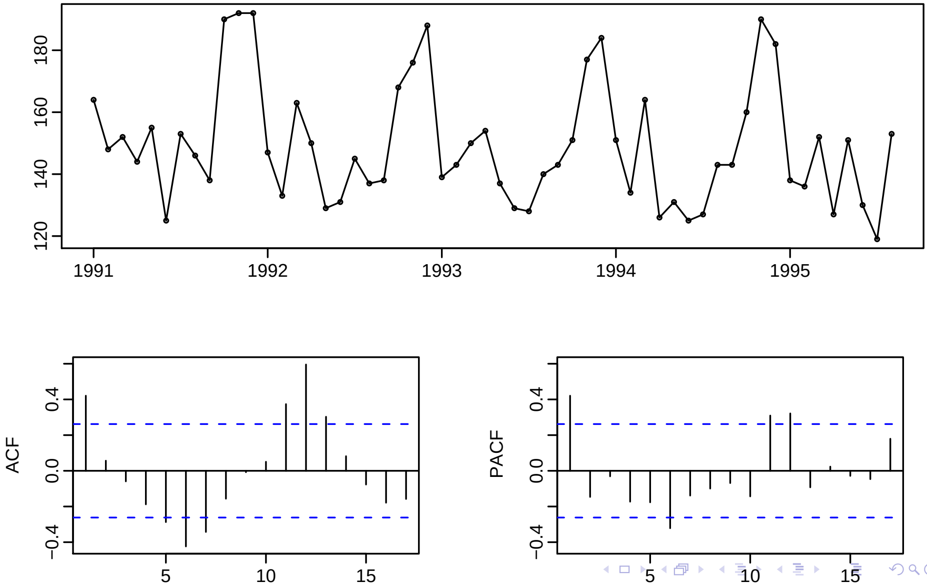
Other plotting functions

tsdisplay() provides a time plot along with an ACF and PACF.

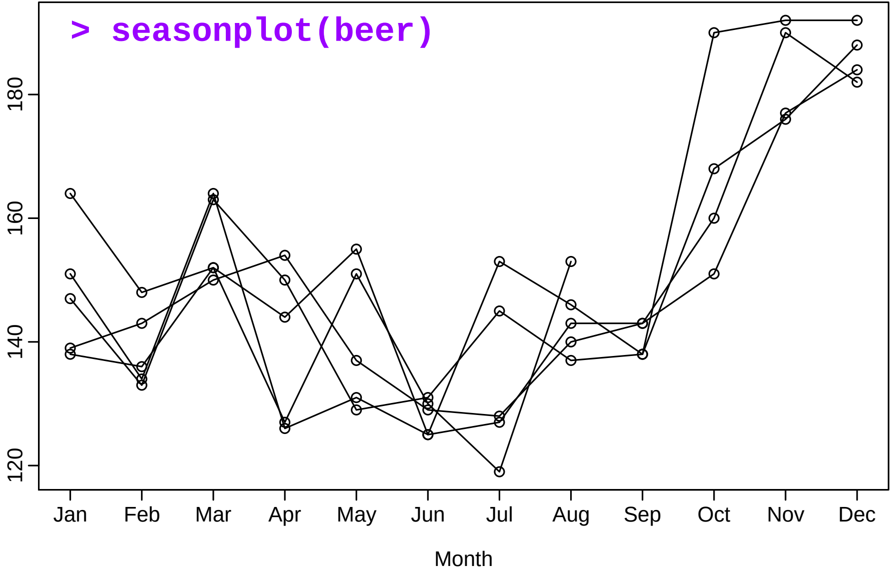
seasonplot() produces a seasonal plot.

tsdisplay

```
> tsdisplay(beer)
```



seasonplot


> seasonplot(beer)
180
160
140
120
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month

Outline

1 Time series objects

2 Basic time series functionality

3 The forecast package

4 Exponential smoothing

5 ARIMA modelling

6 More from the forecast package

7 **Time series packages on CRAN**

Basic facilities

stats Contains substantial time series capabilities including the `ts` class for regularly spaced time series. Also ARIMA modelling, structural models, time series plots, acf and pacf graphs, classical decomposition and STL decomposition.

Forecasting and univariate modelling

forecast Lots of univariate time series methods including automatic ARIMA modelling, exponential smoothing via state space models, and the forecast class for consistent handling of time series forecasts. Part of the **forecasting** bundle.

tseries GARCH models and unit root tests.

FitAR Subset AR model fitting

partsm Periodic autoregressive time series models

pear Periodic autoregressive time series models

Forecasting and univariate modelling

ltsa	Methods for linear time series analysis
dlm	Bayesian analysis of Dynamic Linear Models.
timsac	Time series analysis and control
fArma	ARMA Modelling
fGarch	ARCH/GARCH modelling
BootPR	Bias-corrected forecasting and bootstrap prediction intervals for autoregressive time series
gsarima	Generalized SARIMA time series simulation
bayesGARCH	Bayesian Estimation of the GARCH(1,1) Model with t innovations

Resampling and simulation

boot Bootstrapping, including the block bootstrap with several variants.

meboot Maximum Entropy Bootstrap for Time Series

Decomposition and filtering

robfilter	Robust time series filters
mFilter	Miscellaneous time series filters useful for smoothing and extracting trend and cyclical components.
ArDec	Autoregressive decomposition
wmtsa	Wavelet methods for time series analysis based on Percival and Walden (2000)
wavelets	Computing wavelet filters, wavelet transforms and multiresolution analyses
signalextraction	Real-time signal extraction (direct filter approach)
bspec	Bayesian inference on the discrete power spectrum of time series

Unit roots and cointegration

tseries Unit root tests and methods for computational finance.

urca Unit root and cointegration tests

uroot Unit root tests including methods for seasonal time series

Nonlinear time series analysis

nlts	R functions for (non)linear time series analysis
tseriesChaos	Nonlinear time series analysis
RTisean	Algorithms for time series analysis from nonlinear dynamical systems theory.
tsDyn	Time series analysis based on dynamical systems theory
BAYSTAR	Bayesian analysis of threshold autoregressive models
fNonlinear	Nonlinear and Chaotic Time Series Modelling
bentcableAR	Bent-Cable autoregression

Dynamic regression models

dynlm Dynamic linear models and time series regression

dyn Time series regression

tpr Regression models with time-varying coefficients.

Multivariate time series models

mAr Multivariate AutoRegressive analysis

vars VAR and VEC models

MSBVAR Markov-Switching Bayesian Vector Autoregression Models

tsfa Time series factor analysis

dse Dynamic system equations including multivariate ARMA and state space models.

brainwaver Wavelet analysis of multivariate time series

Functional data

far Modelling Functional AutoRegressive processes

Continuous time data

cts Continuous time autoregressive models

sde Simulation and inference for stochastic differential equations.

Irregular time series

zoo Infrastructure for both regularly and irregularly spaced time series.

its Another implementation of irregular time series.

fCalendar Chronological and Calendarical Objects

fSeries Financial Time Series Objects

xts Provides for uniform handling of R's different time-based data classes

Time series data

fma Data from Makridakis, Wheelwright and Hyndman (1998) *Forecasting: methods and applications*. Part of the **forecasting** bundle.

expsmooth Data from Hyndman, Koehler, Ord and Snyder (2008) *Forecasting with exponential smoothing*. Part of the **forecasting** bundle.

Mcomp Data from the M-competition and M3-competition. Part of the **forecasting** bundle.

FinTS R companion to Tsay (2005) *Analysis of financial time series* containing data sets, functions and script files required to work some of the examples.

TSA R functions and datasets from Cryer and Chan (2008) *Time series analysis with applications in R*

TSdbi Common interface to time series databases

fame Interface for FAME time series databases

fEcofin Ecofin - Economic and Financial Data Sets

Miscellaneous

hydrosanity	Graphical user interface for exploring hydrological time series
pastecs	Regulation, decomposition and analysis of space-time series.
RSEIS	Seismic time series analysis tools
paleoTS	Modeling evolution in paleontological time-series
GeneTS	Microarray Time Series and Network Analysis
fractal	Fractal Time Series Modeling and Analysis