it is now possible to get the R garbage collector to do this via a *finalizer*.

Bibliography

Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer-Verlag, New York, 1996. 5, 6

P. J. Diggle. *Time Series: A Biostatistical Introduction*. Oxford University Press, Oxford, 1990. 4

J. Durbin and S. J. Koopman, editors. *Time Series Analysis by State Space Methods*. Oxford University Press, Oxford, 2001. ISBN 0-19-852354-8. 4, 5

G. Gardner, A. C. Harvey, and G. D. A. Phillips. Algorithm as154. an algorithm for exact maximum likelihood estimation of autoregressive-moving average models by means of kalman filtering. *Applied Statistics*, 29:311–322. 4

A. C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, 1989. 5

A. Pole, M. West, and J. Harrison. *Applied Bayesian Forecasting and Time Series Analysis*. Chapman & Hall, 1994. 5

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S-PLUS*. Springer-Verlag, New York, third edition, 1999. 2

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, New York, fourth edition, 2002. 2, 4

Mike West and Jeff Harrison. *Bayesian Forecasting and Dynamic Models*. Springer-Verlag, New York, second edition, 1997. ISBN 0-387-94725-6. 5

Brian D. Ripley
University of Oxford, UK
ripley@stats.ox.ac.uk

Naive Time Series Forecasting Methods

The Holt-Winters Method in package ts

by David Meyer

Exponential smoothing methods forecast time series by discounted past observations. They have become very popular because of their (relative) simplicity compared to their good overall performance. Common applications range from business tasks (e.g., forecasting of sales or stock fluctuations) to environmental studies (e.g., measurements of atmospheric components or rainfall data)—with typically no more *a priori* knowledge than the possible existence of trend of seasonal patterns. Such methods are sometimes also called *naive* because no covariates are used in the models, i.e., the data are assumed to be self-explaining. Their success is rooted in the fact that they belong to a class of *local* models which automatically adapt their parameters to the data during the estimation procedure and therefore implicitly account for (slow) structural changes in the training data. Moreover, because the influence of new data is controlled by hyperparameters, the effect is a smoothing of the original time series.

Among the simplest methods is the ordinary exponential smoothing, which assumes no trend and no seasonality. Holt's linear trend method (see Holt, 1957) and Winters extensions (see Winters, 1960) add a trend and a seasonal component (the latter either additive or multiplicative). Their methods are still surprisingly popular, although many extensions and more general frameworks do exist. We describe briefly the methods implemented in package **ts** and give some examples of their application.

Exponential smoothing

Let Y_t denote a univariate time series. Exponential smoothing assumes that the forecast $\hat{Y}$ for period $t+h$ based on period t is given by a variable level $\hat{a}$ at period t

$$\hat{Y}_{t+h} = \hat{a}_t \tag{1}$$

which is recursively estimated by a weighted average of the observed and the predicted value for Y_t:

$$\begin{aligned} \hat{a}_t &= \alpha Y_t + (1-\alpha)\hat{Y}_t \\ &= \alpha Y_t + (1-\alpha)\,\hat{a}_{t-1} \end{aligned}$$

$0 < \alpha < 1$ called the smoothing parameter; the smaller it is chosen, the less sensitive $\hat{Y}$ becomes for changes (i.e., the smoother the forecast time series will be). The initial value $\hat{a}_1$ is usually chosen as Y_1.

An equivalent formulation for $\hat{a}_t$ is the so called error-correction form:

$$\begin{aligned} \hat{a}_t &= \hat{a}_{t-1} + \alpha\,(Y_t - \hat{a}_{t-1}) \\ &= \hat{a}_{t-1} + \alpha\,(Y_t - \hat{Y}_t) \\ &= \hat{a}_{t-1} + \alpha\, e_t \end{aligned} \tag{2}$$

showing that $\hat{a}_t$ can be estimated by $\hat{a}_{t-1}$ plus an error e made in period t. (We will come back to this later on.)

Exponential smoothing can be seen as a special case of the Holt-Winters method with no trend and no seasonal component. As an example, we use the

Nile data for which an intercept-only model seems appropriate[1]. We will try to predict the values from 1937 on, choosing $\alpha = 0.2$:

```
library(ts)
data(Nile)
past   <- window(Nile, end   = 1936)
future <- window(Nile, start = 1937)
alpha  <- 0.2
```

To obtain a level-only model, we set the other hyperparameters β and γ (see the next sections) to 0:

```
model <- HoltWinters(past, alpha = alpha,
                     beta = 0, gamma = 0)
```

The object `model` contains an object of type `"HoltWinters"`. The fitted values (obtained by `fitted(model)`) should of course be identical[2] to those given by `filter`:

```
filter(alpha * past, filter = 1 - alpha,
       method = "recursive", init = past[1])
```

We predict 43 periods ahead (until the end of the series):

```
pred <- predict(model, n.ahead = 43)
```

The `plot` method for Holt-Winters objects shows the observed and fitted values, and optionally adds the predicted values (see figure 1):

```
plot(model, predicted.values = pred)
lines(future)
```

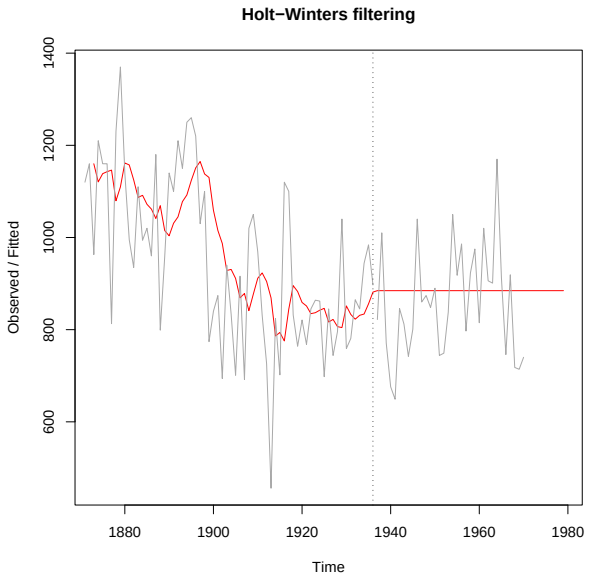

Figure 1: Exponential smoothing for the Nile data up to 1937, and the predictions in 1937 of the rest of the series. The original series is in shown in gray.

The coefficients (here only the level) are listed in the output produced by `print()` or can be extracted with `coef()`.

Holt's linear trend method

An obvious extension is to include an additional trend component, say $\hat{b}_t$. The forecast equation becomes:

$$\hat{Y}_{t+h} = \hat{a}_t + h \cdot \hat{b}_t$$

with updating formulas expressing similar ideas to those for exponential smoothing. There is an additional formula for the trend component:

$$\begin{aligned} \hat{a}_t &= \alpha Y_t + (1-\alpha)(\hat{a}_{t-1} + \hat{b}_{t-1}) \\ \hat{b}_t &= \beta\,(\hat{a}_t - \hat{a}_{t-1}) + (1-\beta)\,\hat{b}_{t-1} \end{aligned}$$

The use is a *local* fit of a straight line with the coefficients adjusted for new values. We now have one more parameter to choose: β. A straightforward approach to find the optimal values for both α and β is to look for the OLS estimates, i.e., the parameter combination minimizing the sum of squared errors of the one-step-ahead predictions. That is what `HoltWinters` does for all the unspecified parameters, illustrated below for the Australian residents data:

```
data(austres)
past   <- window(austres, start = c(1985, 1),
                          end = c(1989, 4))
future <- window(austres, start = c(1990, 1))
(model <- HoltWinters(past, gamma = 0))

Holt-Winters exponential smoothing with trend
and without seasonal component.

Call:
 HoltWinters(x = past, gamma = 0)

Smoothing parameters:
 alpha:  0.931416
 beta :  0.494141
 gamma:  0

Coefficients:
         [,1]
a 16956.68845
b    63.58486
```

The start values for $\hat{a}_t$ and $\hat{b}_t$ are $Y[2]$ and $Y[2] - Y[1]$, respectively; the parameters are estimated via `optim` using `method = "L-BFGS-B"`, restricting the parameters to the unit cube.

Another feature is the optional computation of confidence intervals for the predicted values, which by default are plotted along with them (see figure 2):

```
pred <- predict(model, n.ahead = 14,
                prediction.interval = TRUE)
plot(model, pred); lines(future)
```

[1] At least piecewise: note that one might expect a structural break in 1899 due to the construction of the first Ashwan dam

[2] Currently the initial value is only observation 2 as it is for the trend model (for which this is sensible). This will be changed in R 1.5.1.

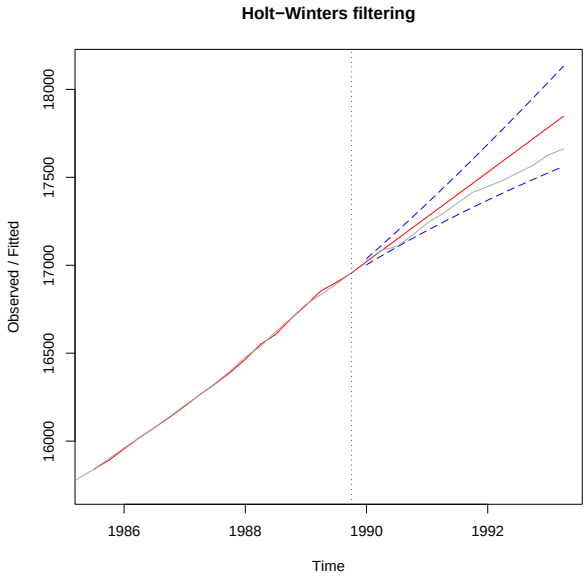

Figure 2: Holt's linear trend method applied to data on Autralian residents, with 95% confidence limits in dashed blue lines. Original data are plotted in gray, fitted and predicted values in red.

Winters' seasonal method

Winters extended the method to include a seasonal component. There are additive and multiplicative versions. In the following, we suppose seasonal trend components $\hat{s}_t$ and a period p (e.g., 4 for quarterly data).

Additive version

The additive model assumes the decomposition:

$$\hat{Y}_{t+h} = \hat{a}_t + h \cdot \hat{b}_t + \hat{s}_t$$

with an additional updating formula for the seasonal component:

$$\begin{aligned} \hat{a}_t &= \alpha\,(Y_t - \hat{s}_{t-p}) + (1-\alpha)(\hat{a}_{t-1} + \hat{b}_{t-1}) \\ \hat{b}_t &= \beta\,(\hat{a}_t - \hat{a}_{t-1}) + (1-\beta)\,\hat{b}_{t-1} \\ \hat{s}_t &= \gamma\,(Y_t - \hat{a}_t) + (1-\gamma)\,\hat{s}_{t-p} \end{aligned}$$

Multiplicative version

This version assumes a model with time-increasing seasonal component (i.e., the amplitude becomes larger with increasing t). The prediction formula becomes:

$$\hat{Y}_{t+h} = (\hat{a}_t + h \cdot \hat{b}_t)\,\hat{s}_t$$

and the updating formulas change accordingly:

$$\begin{aligned} \hat{a}_t &= \alpha\,(Y_t - \hat{s}_{t-p}) + (1-\alpha)(\hat{a}_{t-1} + \hat{b}_{t-1}) \\ \hat{b}_t &= \beta\,(\hat{a}_t - \hat{a}_{t-1}) + (1-\beta)\,\hat{b}_{t-1} \\ \hat{s}_t &= \gamma\,(Y_t / \hat{a}_t) + (1-\gamma)\,\hat{s}_{t-p} \end{aligned}$$

For automatic parameter selection, we need at least three complete cycles to estimate an initial seasonal component, done via a classical seasonal decomposition using moving averages performed by the new function `decompose()`. The intercept and trend are estimated by a simple regression on the extracted trend component.

As an example, we apply the multiplicative version to the UK gas consumption data for 1960–80 and predict the next six years.

```
data(UKgas)
past   <- window(UKgas, end   = c(1980, 4))
future <- window(UKgas, start = c(1981, 1))
model  <- HoltWinters(past, seasonal = "mult")
pred   <- predict(model, n.ahead = 24)
```

For clarity, we plot the original time series and the predicted values separately (see figure 3):

```
par(mfrow = c(2,1))
plot(UKgas, main = "Original Time Series")
plot(model, pred)
```

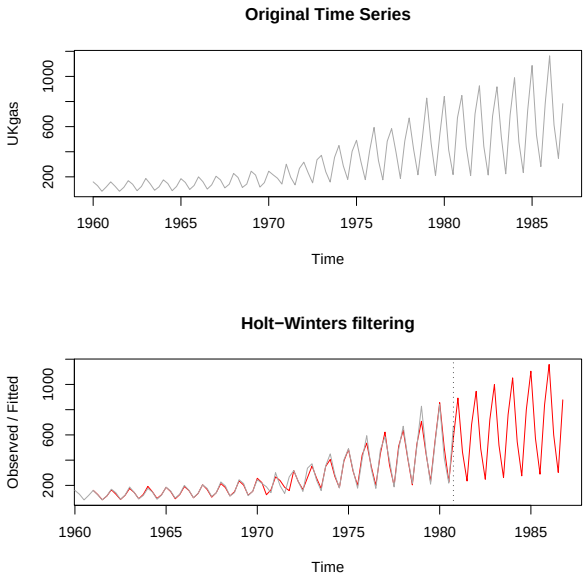

Figure 3: Winter's method applied to quarterly UK gas consumption data. The upper figure is the actual data, the lower the data (gray) and fited values (red) up to 1980Q4, the predictions (red) thereafter.

Conclusion

We have seen that the Holt-Winters method consists in a simple yet effective forecasting procedure, based on exponential moving averages, covering both trend and seasonal models. Many extensions exist however (see, e.g., Hyndman et al., 2001), and the various formulations can also be seen as special cases of a wider class of models. For example, all but the multiplicative Holt-Winters can be identified as

(restricted) SARIMA models. To motivate this, consider equation 2—the error-correction form of exponential smoothing. By plugging it into the forecasting equation 1, we almost directly obtain:

$$\begin{aligned}(1-L)\hat{Y}_t &= \alpha e_{t-1} \\ &= (1-\theta L)e_t\end{aligned}$$

($\theta = 1-\alpha$), which is (in terms of estimates) the standard form of an ARIMA(0,1,1)-process. Finally, we note that we actually face *state-space* models: given process Y_t, we try to estimate the underlying processes a_t, b_t and s_t which cannot be observed directly. But this is the story of structural time series and the function `structTS`, told in an another article by Brian D. Ripley ...

Bibliography

Holt, C. (1957). Forecasting seasonals and trends by exponentially weighted moving averages. *ONR Research Memorandum, Carnegie Institute 52*. 7

Hyndman, R., Koehler, A., Snyder, R., & Grose, S. (2001). A state space framework for automatic forecasting using exponential smoothing methods. *Journal of Forecasting*. To appear. 9

Winters, P. (1960). Forecasting sales by exponentially weighted moving averages. *Management Science*, **6**, 324–342. 7

David Meyer
Technische Universität Wien, Austria
David.Meyer@ci.tuwien.ac.at

Rmpi: Parallel Statistical Computing in R

by Hao Yu

Rmpi is an R interface (wrapper) to the Message-Passing Interface (MPI). MPI is a standard application interface (API) governed by the MPI forum (http://www.mpi-forum.org) for running parallel applications. Applicable computing environments range from dedicated Beowulf PC clusters to parallel supercomputers such as IBM's SP2. Performance and portability are the two main strengths of MPI. In this article, we demonstrate how the MPI API is implemented in **Rmpi**. In particular, we show how interactive R slaves are spawned and how we use them to do sophisticated MPI parallel programming beyond the "embarrassingly parallel".

Introduction

Put simply, parallel statistical computing means dividing a job into many small parts which will be executed simultaneously over a cluster of computers linked together in a network (LAN or WAN). Communications among the components is a crucial aspect of parallel computing. The message-passing model is highly suitable for such a task. There are two primary open-source message-passing models available: MPI and PVM (Parallel Virtual Machine, Geist et al., 1994). For PVM's implementation **rpvm** in R, see Li and Rossini, 2001.

Whether to use MPI or PVM largely depends on the underlying hardware structure and one's personal preference. Both models provide a parallel programming interface. The main difference is in how buffers are managed or used for sending and receiving data. Typically, in PVM, users are responsible for packing data into a system designated buffer on the sender side and unpacking from a system buffer on the receiver side. This reduces the user's responsibility for managing the buffer; on the other hand, this may create problems of running out of buffer space if large amounts of data are transferred. PVM can also lead to performance draining copies. MPI, by contrast, requires no buffering of data within the system. This allows MPI to be implemented on the widest range of platforms, with great efficiency. Perhaps Luke Tierney's package **snow** (Simple Network of Workstations, http://www.stat.umn.edu/~luke/R/cluster) will ultimately provide a unified interface to both **Rmpi** and **rpvm** so users can freely choose either one to implement parallel computing.

There are 128 routines in the MPI-1 standard and 287 functions in the combined MPI (MPI-1 and MPI-2) standard. MPI is rich in functionality; it is also capable of handling the diversity and complexity of today's high performance computers. Likely the environment with highest potential for MPI use is the Beowulf cluster, a high-performance massively parallel computer built primarily out of commodity hardware components.

One of the goals of **Rmpi** is to provide an extensive MPI API in the R environment. Using R as an interface, the user can either spawn C(++) or Fortran programs as children or join other MPI programs. Another goal is to provide an interactive R master and slave environment so MPI programming can be done completely in R. This was achieved by implementing a set of MPI API extensions specifically designed for the R or R slave environments.