Time Series in R 1.5.0

by Brian D. Ripley

R has shipped with a package **ts** since 0.65.0 in mid 1999; this package has been enhanced considerably in version 1.5.0. The original release brought together several existing functions written by Paul Gilbert, Martyn Plummer, Adrian Trapletti and myself and included interfaces to a Fortran package I wrote for teaching time series in the late 1970's. Improvements were promised at the time, one function being called `arima0` to signal its preliminary status.

There are contributed packages with other time series functionality, including **tseries** which has an econometric/financial slant, and bundle **dse**, a toolkit for working with state-space models of vector time series.

The new function `HoltWinters` is discussed in the following article by David Meyer.

One of the goals of package **ts** was to provide enough functionality to cover the time series chapter of Venables and Ripley (1999). Time-series analysis was part of the 'PLUS' of S-PLUS more than a decade ago, but unfortunately has been little updated since, and I had planned for a long time to provide better facilities to fit ARIMA processes. Indeed, this was on the list of new things we intended to cover in Venables and Ripley (1999), and it would have been embarrassing not to have done so by Venables and Ripley (2002). So one of the innovations for R 1.5.0 is function `arima`, a version of which appears in the **MASS** library section for S-PLUS.

Updating package **ts** took so long because of technical challenges which this article hopes to illuminate. There can be a lot more detail to statistical computing than one finds even in the most complete monographs. It is often said that the way to learn a subject is to be forced to teach a course on it: writing a package is a much more searching way to discover if one really understands a piece of statistical methodology!

Missing values

Missing values are an occupational hazard in some applications of time series: markets are closed for the day, the rain gauge jammed, a data point is lost, ... So it is very convenient if the time-series analysis software can handle missing values transparently. Prior to version 1.5.0, R's functions made little attempt to do so, but we did provide functions `na.contiguous` and `na.omit.ts` to extract non-missing stretches of a time series.

There were several technical problems in supporting arbitrary patterns of missing values:

- Most of the computations were done in Fortran, and R's API for missing values only covers C code: this was resolved by carefully translating code[1] to C.

- Some computations only have their standard statistical properties if the series is complete. The sample autocorrelation function as returned by `acf` is a valid autocorrelation for a stationary time series if the series is complete, but this is not necessarily true if correlations are only computed over non-missing pairs. Indeed, it is quite possible that the pattern of missingness may make missing one or both of *all* the pairs at certain lags. Even if most of the pairs are missing the sampling properties of the ACF will be affected, including the confidence limits which are plotted by default by `plot.acf`.

- Standard time-series operations propagate missingness. This is true of most filtering operations, in particular the differencing operations which form the basis of the 'Box–Jenkins' methodology. The archetypal Box–Jenkins 'airline' model[2] involves both differencing and seasonal differencing. For such a model one missing value in the original series creates three missing values in the transformed series.

Our approach has been to implement support for missing values where we know of a reasonably sound statistical approach, but expect the user to be aware of the pitfalls.

Presidents

One of R's original datasets is `presidents`, a quarterly time series of the Gallup polls of the approval rating of the US presidents from 1945 to 1974. It has 6 missing values, one at the beginning as well the last two quarters in each of 1948 and 1974. This seems a sufficiently complete series to allow a reasonable analysis, so we can look at the ACF and PACF (figure 1)

```
data(presidents)
acf(presidents, na.action = na.pass)
pacf(presidents, na.action = na.pass)
```

[1] The usual route to translate Fortran code is to `f2c -a -A` from http://www.netlib.org/f2c. Unfortunately this can generate illegal C, as happened here, and if speed is important idiomatic C can run substantially faster. So the automatic translation was re-written by hand.

[2] as fitted to a monthly series of numbers of international airline passengers, now available as dataset `AirPassengers` in package **ts**.

We need an `na.action` argument, as the default behaviour is to fail if missing values are present. Function `na.pass` returns its input unchanged.

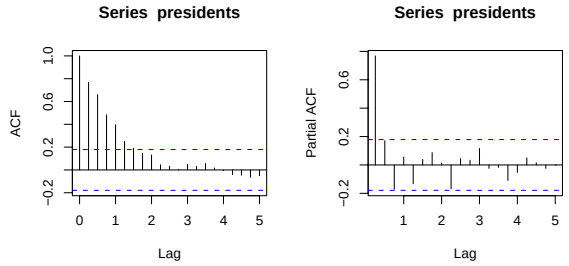

Figure 1: Autocorrelation and partial correlation plots of the `presidents` series. Note that the confidence limits shown do not take account of missingness.

The plots suggest an AR(1) or AR(3) model would be appropriate. We cannot use `ar`, but the new function `arima` will fit these models.

```
> (fit1 <- arima(presidents, c(1, 0, 0)))

Coefficients:
         ar1  intercept
      0.8242    56.1505
s.e.  0.0555     4.6434

sigma^2 estimated as 85.47:
log likelihood = -416.89,  aic = 839.78
> tsdiag(fit1)
> (fit3 <- arima(presidents, c(3, 0, 0)))

Coefficients:
         ar1     ar2      ar3  intercept
      0.7496  0.2523  -0.1890    56.2223
s.e.  0.0936  0.1140   0.0946     4.2845

sigma^2 estimated as 81.12:
log likelihood = -414.08,  aic = 838.16
> tsdiag(fit3)
```

This suggests a fairly clear preference for AR(3). Function `tsdiag` is a new generic function for diagnostic plots.

Fitting ARIMA models

There are several approximations to full maximum-likelihood fitting of ARIMA models in common use. For an ARMA model (with no differencing) the model implies a multivariate normal distribution for the observed series. A very common approximation is to ignore the determinant in the normal density. The 'conditional sum of squares' approach uses a likelihood conditional on the first few observations, and reduces the problem to minimizing a sum of squared residuals. The CSS approach has the same asymptotics as the full ML approach, and the textbooks often regard it as sufficient. It is not adequate for two reasons:

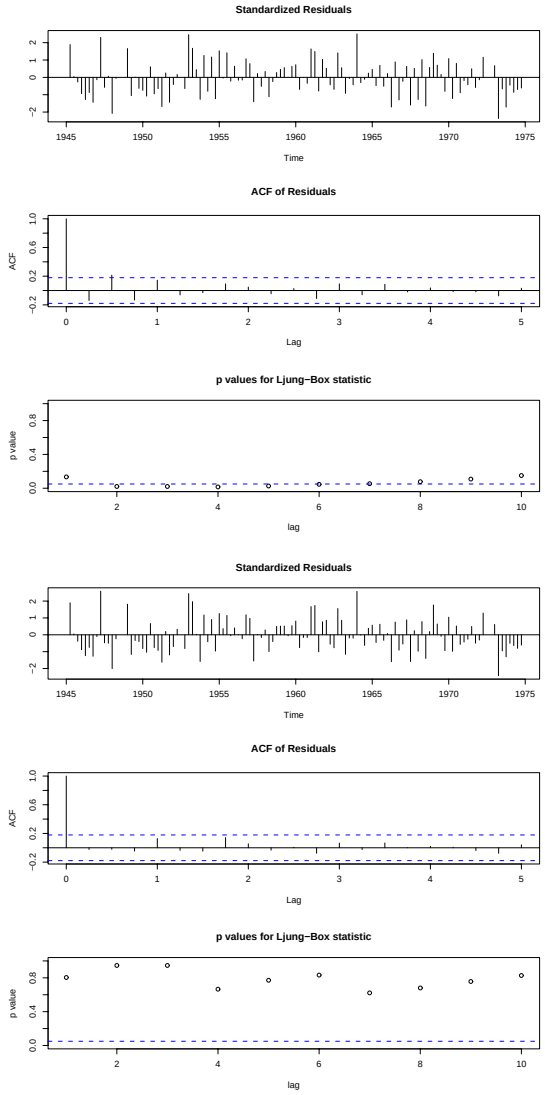

Figure 2: Diagnostic plots for AR(1) (upper) and AR(3) (lower) fits to the `presidents` series.

1. **Missing values.** The residuals are found by a recursive filter which cannot be adapted to handle missing values. The theory can be, but loses its main appeal: simplicity.

2. **Short series.** Both conditioning and discarding the determinant term are essentially end corrections. Unlike spatial statistics, edge effects are normally ignored in time series, in the mistaken belief that time series are long. For example, `AirPassengers` is of length 144: surely that is not short? Yes it is: we are fitting 12 linked series of length 12 each, one for each month, and the relevant end effect is that there are only 12 years. For pedagogical purposes `arima` and `arima0` include `method = "CSS"` so users can see for themselves the size of the effect.

The state-space approach ('Kalman filtering') has long been advocated as a good way to compute the

likelihood of an ARMA model even in the presence of missing values: computational details are given by Durbin and Koopman (2001), and the original `arima0` used Fortran code[3] published as long ago as Gardner et al.. The state-space approach used to be regarded (rightly) as slow, but that is no longer an issue for tasks likely to be done in R. It is important to use state-space models correctly, including initializing them at their stationary distribution or the end effects will return.

ARIMA models present a further challenge. The model is for a differenced version of the observed data rather than for the data themselves, so a likelihood is not actually defined. There are a number of approaches outlined in Durbin and Koopman (2001),[4] of which I find the most satisfactory is the 'diffuse prior' approach, and that is implemented in `arima`. This assumes that the initial values of the time series on which the traditional approach conditions have mean zero (as someone chose the units of observation) but a large variance. Again there are both theoretical and numerical issues, as well as practical ones: what should one do if every January observation is missing? The approach in `arima` can cope with missing values in the initial segment.

Even when one has defined a log-likelihood and found an algorithm to compute it, there remains the task of optimizing it. Yet again this is much harder than the books make out. Careful study shows that there are often multiple local maxima. Finding good starting values is nigh to impossible if the end effects are going to be important. Finding reliable test examples is difficult. In the early days of `arima0` someone reported that it gave different results from SPSS, and Bill Venables suggested on `R-help` that this might prove helpful to the SPSS developers! One good test is to reverse the series and see if the same model is fitted. All ARIMA processes are reversible and `help("AirPassengers")` provides an empirical demonstration. So if `arima` takes a long time to find a solution, please bear in mind that a reliable solution is worth waiting for.

UK lung deaths

Venables and Ripley (2002, §14.3) analyse a time series from Diggle (1990) on monthly deaths from bronchitis, emphysema and asthma in the UK, 1974–1979. As this has only 6 whole years, it is a fairly severe test of the ability to handle end effects.

Looking at ACFs of the seasonally differenced series suggests an ARIMA$((2,0,0)\times(0,1,0)_{12})$ model, which we can fit by `arima`:

```
data(deaths, package="MASS")
deaths.diff <- diff(deaths, 12)
## plots not shown here
acf(deaths.diff, 30); pacf(deaths.diff, 30)

> (deaths.arima1 <-
    arima(deaths, order = c(2,0,0),
          seasonal = list(order = c(0,1,0),
                          period = 12)) )
Coefficients:
        ar1     ar2
      0.118  -0.300
s.e.  0.126   0.125

sigma^2 = 118960:
log likelihood = -435.83,  aic = 877.66
> tsdiag(deaths.arima1, gof.lag = 30)
```

However the diagnostics indicate the need for a seasonal AR term. We can try this first without differencing

```
> (deaths.arima2 <-
    arima(deaths, order = c(2,0,0),
          list(order = c(1,0,0),
               period = 12)) )
Coefficients:
        ar1     ar2   sar1  intercept
      0.801  -0.231  0.361    2062.45
s.e.  0.446   0.252  0.426     133.90

sigma^2 = 116053:
log likelihood = -523.16,  aic = 1056.3
> tsdiag(deaths.arima2, gof.lag = 30)
```

The AICs are not comparable, as a differenced model is not an explanation of all the observations.

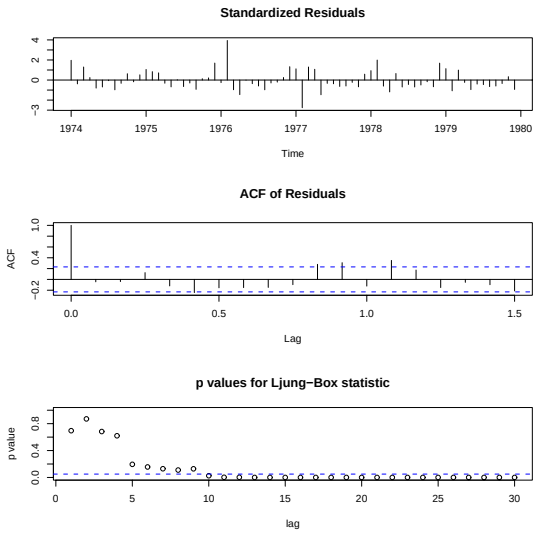

Figure 3: Diagnostic plots for `deaths.arima2`.

The diagnostics suggest that there is still seasonal structure in the residuals, so we next tried including both differencing and a seasonal AR term:

[3] Not only did this have to be translated to C to handle missing values, but several undocumented efficiency gains that resulted from assuming complete observation had to be undone.

[4] which in parts is an advertisement for non-free C code called **SsfPack** at `http://www.ssfpack.com` which links to a non-free Windows program called **Ox**.

```
> (deaths.arima3 <-
    arima(deaths, order = c(2,0,0),
          list(order = c(1,1,0),
               period = 12)) )
Coefficients:
       ar1    ar2   sar1
     0.293 -0.271 -0.571
s.e. 0.137  0.141  0.103

sigma^2 = 77145:
log likelihood = -425.22,  aic = 858.43
> tsdiag(deaths.arima3, gof.lag = 30)
```

for which the diagnostics plots look good.

Structural time series

Structural models of time series are an approach which is most closely linked to the group of Andrew Harvey (see in particular Harvey, 1989) but there are several closely related approaches, such as the dynamic linear models of Jeff Harrison and co-workers see (West and Harrison (1997); Pole et al. (1994) is a gentler introduction).

I have long thought that the approach had considerable merit, but believe its impact has been severely hampered by the lack of freely-available software.[5] It is beginning to appear in introductory time-series textbooks, for example Brockwell and Davis (1996, §8.5). As often happens, I wrote `StructTS` both for my own education and to help promote the methodology.

Experts such as Jim Durbin have long pointed out that all the traditional time-series models are just ways to parametrize the second-order properties of stationary time series (perhaps after filtering/differencing), and even AR models are not models of the real underlying mechanisms. (The new functions `ARMAacf`, `ARMAtoMA` and `acf2AR` allow one to move between the ACF representing the second-order properties and various parametrizations.) Structural time series are an attempt to model a plausible underlying mechanism for non-stationary time series.

The simplest structural model is a *local level*, which has an underlying level m_t which evolves by

$$\mu_{t+1} = \mu_t + \xi_t, \qquad \xi_t \sim N(0, \sigma_\xi^2)$$

The observations are

$$x_t = \mu_t + \epsilon_t, \qquad \epsilon_t \sim N(0, \sigma_\epsilon^2)$$

so there are two parameters, σ_ξ^2 and σ_ϵ^2, either of which could be zero. It is an ARIMA(0,1,1) model, but with restrictions on the parameter set.

The next step up is *local linear trend model* which has the same measurement equation, but with a time-varying slope in the dynamics for μ_t, given by

$$\begin{aligned} \mu_{t+1} &= \mu_t + \nu_t + \xi_t, & \xi_t &\sim N(0, \sigma_\xi^2) \\ \nu_{t+1} &= \nu_t + \zeta_t, & \zeta_t &\sim N(0, \sigma_\zeta^2) \end{aligned}$$

with three variance parameters. It is not uncommon to find $\sigma_\zeta^2 = 0$ (which reduces to the local level model) or $\sigma_\xi^2 = 0$, which ensures a smooth trend. This is a (highly) restricted ARIMA(0,2,2) model.

For a seasonal model we will normally use the so-called *Basic Structural Model* (BSM) for a seasonal time series. To be concrete, consider energy consumption measures quarterly, such as the series `UKgas` in package **ts**. This is based on a (hypothetical) decomposition of the series into a level, trend and a seasonal component. The measurement equation is

$$x_t = \mu_t + \gamma_t + \epsilon_t, \qquad \epsilon_t \sim N(0, \sigma_\epsilon^2)$$

where γ_t is a seasonal component with dynamics

$$\gamma_{t+1} = -(\gamma_t + \gamma_{t-1} + \gamma_{t-2}) + \omega_t, \quad \omega_t \sim N(0, \sigma_\omega^2)$$

The boundary case $\sigma_\omega^2 = 0$ corresponds to a deterministic (but arbitrary) seasonal pattern. (This is sometimes known as the 'dummy variable' version of the BSM.) There are now four variance parameters $(\sigma_\zeta^2, \sigma_\xi^2, \sigma_\omega^2, \sigma_\epsilon^2)$, one or more of which (but not all) can be zero.

These models are quite similar to those used in exponential smoothing and Holt–Winters forecasting (see the accompanying article by David Meyer); one important difference is that structural time series are handled in a formal statistical framework.

Estimation

Structural time series are fitted by maximum likelihood with a diffuse prior for those components (such as the overall level) which are not fully specified by the model. Using the state-space approach (as detailed in Durbin and Koopman, 2001) makes it easy to handle missing values.

What is not easy is the optimization, and I am now convinced that many of the published examples (e.g., Figure 8-4 in Brockwell and Davis (1996)) are incorrect. One issue is multiple local maxima of the log-likelihood, and some examples do seem to be sub-optimal local maxima. Optimization under non-negativity constraints can be tricky,[6] and it appears that other software either assumes that all the variances are positive or that $\sigma_\epsilon^2 > 0$.

We can illustrate this on the classic airline passengers dataset.

[5] Harvey's coworkers have produced a Windows package called STAMP, `http://stamp-software.com`, and **SsfPack** provides code in **Ox** for these models.

[6] Function `StructTS` uses `optim(method="L-BFGS-B")`, and it was this application that I had in mind when programming that.

```
data(AirPassengers)
## choose some sensible units on log scale
ap <- log10(AirPassengers) - 2
(fit <- StructTS(ap, type= "BSM"))
Call:
StructTS(x = ap, type = "BSM")

Variances:
   level     slope      seas   epsilon
0.000146  0.000000  0.000263  0.000000
```

Note that the measurement variance σ_ϵ^2 is estimated as zero.

Prediction and smoothing

One we have fitted a structural time series model, we can compute predictions of the components, that is the level, trend and seasonal pattern. There are two distinct bases on which we can do so. The `fitted` method for class `"StructTS"` displays the contemporaneous predictions, whereas the `tsSmooth` method displays the final predictions. That is, `fitted` shows the predictions of μ_t, ν_t and γ_t based on observing $x_1, \ldots, x_t$ for $t = 1, \ldots, T$, whereas `tsSmooth` gives predictions of μ_t, ν_t and γ_t based on observing the whole time series $x_1, \ldots, x_T$ (the 'fixed interval smoother' in Kalman filter parlance).

Looking at the fitted values shows how the information about the trend and seasonal patterns (in particular) builds up: it is like preliminary estimates of inflation etc given out by national statistical services. The smoothed values are the final revisions after all the information has been received and incorporated.

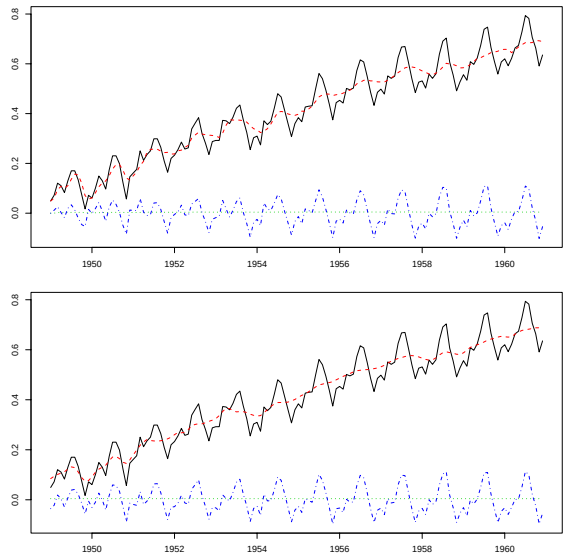

Figure 4: Fitted values (top) and smoothed values (bottom) of a BSM fit to the `AirPassengers` data, on $\log_{10}$ scale. The original series is solid black, the level is dashed red, the slope is dotted green and the seasonal component is dash-dotted blue.

We can illustrate this for the `AirPassengers` data by

```
plot(cbind(ap, fitted(fit)),
   plot.type = "single", lty=1:4, col=1:4)
plot(cbind(ap, tsSmooth(fit)),
   plot.type = "single", lty=1:4, col=1:4)
```

Note the way that fluctuations which are contemporaneously ascribed to level changes are revised to be part of the (changing) seasonal component as the latter changes shape around 1952–4. It is instructive to compare this with Figure 8-4 in Brockwell and Davis (1996).

Implementation

The tools available to the R programmer have moved on considerably since 1999. Like much of the statistical functionality in R, the internals of package **ts** were written in C or Fortran, interfaced via `.C` or `.Fortran`. Nowadays much new code is being written using C code interfaced with `.Call` which allows the R objects to be manipulated at C level, and in particular for the return object to be constructed in C code. The approach via `.Call` can also be much more efficient as it allows the programmer much more control over the copying of objects.

The first version of `arima` was written using the general-purpose Kalman filter code in the (new) functions `KalmanLike` and friends. This proved to be far too slow for seasonal ARIMA models, the time being spent in multiplying sparse matrices (and this was confirmed by profiling). The code was supplanted by special-purpose code, first for ARMA and then for ARIMA processes: having the slower reference implementation was very useful.

The original `arima0` code had dynamically allocated global C arrays to store information passed down and retrieved by `.C` calls. This had been flagged as a problem in the planned move to a threaded version of R. A more elegant approach was needed. `arima` passes around an R list which represents the state space model, but `arima0` only works at C level, so the obvious approach is to store all the information (which includes the data) in a C structure rather than in global variables. The remaining issue was to associate the instance of the structure with the instance of `arima0` (since in a threaded version of R more than one could be running simultaneously or at least interleaved). This was solved by the use of *external references*. This is a little-documented area of R (but see `http://developer.r-project.org/simpleref.html` and `http://developer.r-project.org/references.html`) that allows us to return *via* `.Call` an R object (of type `EXTPTRSXP`) that contains a pointer to our structure. We chose to de-allocate the arrays in the structure in the `on.exit` function of `arima0`, but

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