5.8 Nonlinear regression

Although the linear relationship assumed so far in this chapter is often adequate, there are many cases in which a nonlinear functional form is more suitable. To keep things simple in this section we assume that we only have one predictor x.

The simplest way of modelling a nonlinear relationship is to transform the forecast variable y and/or the predictor variable x before estimating a regression model. While this provides a non-linear functional form, the model is still linear in the parameters. The most commonly used transformation is the (natural) logarithm (see Section 3.2).

A **log-log** functional form is specified as

$$\log y = \beta_0 + \beta_1 \log x + \varepsilon.$$

In this model, the slope β_1 can be interpreted as an elasticity: β_1 is the average percentage change in y resulting from a 1% increase in x. Other useful forms can also be specified. The **log-linear** form is specified by only transforming the forecast variable and the **linear-log** form is obtained by transforming the predictor.

Recall that in order to perform a logarithmic transformation to a variable, all of its observed values must be greater than zero. In the case that variable x contains zeros, we use the transformation $\log(x+1)$; i.e., we add one to the value of the variable and then take logarithms. This has a similar effect to taking logarithms but avoids the problem of zeros. It also has the neat side-effect of zeros on the original scale remaining zeros on the transformed scale.

There are cases for which simply transforming the data will not be adequate and a more general specification may be required. Then the model we use is

$$y = f(x) + \varepsilon$$

where f is a nonlinear function. In standard (linear) regression, $f(x) = \beta_0 + \beta_1 x$. In the specification of nonlinear regression that follows, we allow f to be a more flexible nonlinear function of x, compared to simply a logarithmic or other transformation.

One of the simplest specifications is to make f **piecewise linear**. That is, we introduce points where the slope of f can change. These points are called **knots**. This can be achieved by letting $x_{1,t} = x$ and introducing variable $x_{2,t}$ such that

$$x_{2,t} = (x - c)_+ = \begin{cases} 0 & x < c \\ (x - c) & x \geq c \end{cases}$$

The notation $(x - c)_+$ means the value $x - c$ if it is positive and 0 otherwise. This forces the slope to bend at point c. Additional bends can be included in the relationship by adding further variables of the above form.

An example of this follows by considering $x = t$ and fitting a piecewise linear trend to a time series.

Piecewise linear relationships constructed in this way are a special case of **regression splines**. In general, a linear regression spline is obtained using

$$x_1 = x \quad x_2 = (x - c_1)_+ \quad \dots \quad x_k = (x - c_{k-1})_+$$

where $c_1, \dots, c_{k-1}$ are the knots (the points at which the line can bend). Selecting the number of knots ($k - 1$) and where they should be positioned can be difficult and somewhat arbitrary. Some automatic knot selection algorithms are available in some software, but are not yet widely used.

A smoother result can be obtained using piecewise cubics rather than piecewise lines. These are constrained to be continuous (they join up) and smooth (so that there are no sudden changes of direction, as we see with piecewise linear splines). In general, a cubic regression spline is written as

$$x_1 = x \quad x_2 = x^2 \quad x_3 = x^3 \quad x_4 = (x - c_1)_+ \quad \dots \quad x_k = (x - c_{k-3})_+.$$

Cubic splines usually give a better fit to the data. However, forecasts of y become unreliable when x is outside the range of the historical data.

Forecasting with a nonlinear trend

In Section 5.4 fitting a linear trend to a time series by setting $x = t$ was introduced. The simplest way of fitting a nonlinear trend is using quadratic or higher order trends obtained by specifying

$$x_{1,t} = t, \quad x_{2,t} = t^2, \quad \dots.$$

However, it is not recommended that quadratic or higher order trends be used in forecasting. When they are extrapolated, the resulting forecasts are often unrealistic.

A better approach is to use the piecewise specification introduced above and fit a piecewise linear trend which bends at some point in time. We can think of this as a nonlinear trend constructed of linear pieces. If the trend bends at time τ, then it can be specified by simply replacing $x = t$ and $c = \tau$ above such that we include the predictors,

$$\begin{aligned} x_{1,t} &= t \\ x_{2,t} = (t-\tau)_+ &= \begin{cases} 0 & t < \tau \\ (t-\tau) & t \geq \tau \end{cases} \end{aligned}$$

in the model. If the associated coefficients of $x_{1,t}$ and $x_{2,t}$ are β_1 and β_2, then β_1 gives the slope of the trend before time τ, while the slope of the line after time τ is given by $\beta_1 + \beta_2$. Additional bends can be included in the relationship by adding further variables of the form $(t-\tau)_+$ where τ is the "knot" or point in time at which the line should bend.

Example: Boston marathon winning times

The top panel of Figure 5.20 shows the Boston marathon winning times (in minutes). The course was lengthened (from 24.5 miles to 26.2 miles) in 1924, which led to a jump in the winning times, so we only consider data from that date onwards. The time series shows a general downward trend as the winning times have been improving over the years. The bottom panel shows the residuals from fitting a linear trend to the data. The plot shows an obvious nonlinear pattern which has not been captured by the linear trend. There is also some heteroscedasticity, with decreasing variation over time.

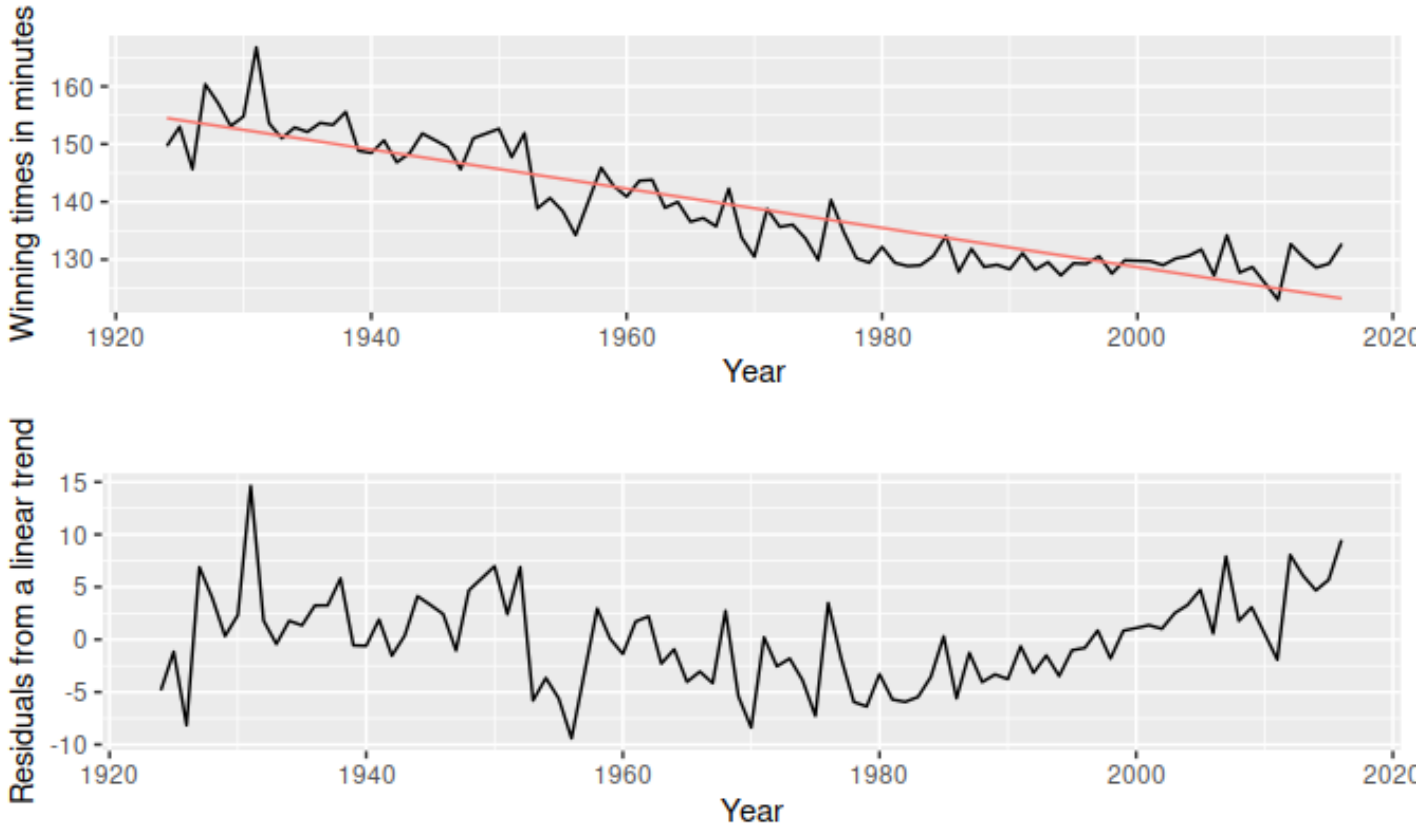

Figure 5.20: Fitting a linear trend to the Boston marathon winning times is inadequate

Fitting an exponential trend (equivalent to a log-linear regression) to the data can be achieved by transforming the y variable so that the model to be fitted is,

$$\log y_t = \beta_0 + \beta_1 t + \varepsilon_t.$$

This also addresses the heteroscedasticity. The fitted exponential trend and forecasts are shown in Figure 5.21. Although the exponential trend does not seem to fit the data much better than the linear trend, it gives a more sensible projection in that the winning times will decrease in the future but at a decaying rate rather than a fixed linear rate.

The plot of winning times reveals three different periods. There is a lot of volatility in the winning times up to about 1950, with the winning times barely declining. After 1950 there is a near-linear decrease in times, followed by a flattening out after the 1980s, with the suggestion of an upturn towards the end of the sample. To account for these changes, we specify the years 1950 and 1980 as knots. We should warn here that subjective identification of knots can lead to over-fitting, which can be detrimental to the forecast performance of a model, and should be performed with caution.

```
boston_men <- window(marathon, start=1924)
h <- 10
fit.lin <- tslm(boston_men ~ trend)
fcasts.lin <- forecast(fit.lin, h = h)
fit.exp <- tslm(boston_men ~ trend, lambda = 0)
fcasts.exp <- forecast(fit.exp, h = h)

t <- time(boston_men)
t.break1 <- 1950
t.break2 <- 1980
tb1 <- ts(pmax(0, t - t.break1), start = 1924)
tb2 <- ts(pmax(0, t - t.break2), start = 1924)

fit.pw <- tslm(boston_men ~ t + tb1 + tb2)
t.new <- t[length(t)] + seq(h)
tb1.new <- tb1[length(tb1)] + seq(h)
tb2.new <- tb2[length(tb2)] + seq(h)

newdata <- cbind(t=t.new, tb1=tb1.new, tb2=tb2.new) %>%
  as.data.frame()
fcasts.pw <- forecast(fit.pw, newdata = newdata)

fit.spline <- tslm(boston_men ~ t + I(t^2) + I(t^3) +
  I(tb1^3) + I(tb2^3))
fcasts.spl <- forecast(fit.spline, newdata = newdata)

autoplot(boston_men) +
  autolayer(fitted(fit.lin), series = "Linear") +
  autolayer(fitted(fit.exp), series = "Exponential") +
  autolayer(fitted(fit.pw), series = "Piecewise") +
  autolayer(fitted(fit.spline), series = "Cubic Spline") +
  autolayer(fcasts.pw, series="Piecewise") +
  autolayer(fcasts.lin, series="Linear", PI=FALSE) +
  autolayer(fcasts.exp, series="Exponential", PI=FALSE) +
  autolayer(fcasts.spl, series="Cubic Spline", PI=FALSE) +
  xlab("Year") + ylab("Winning times in minutes") +
```

```
  ggtitle("Boston Marathon") +
  guides(colour = guide_legend(title = " "))
```

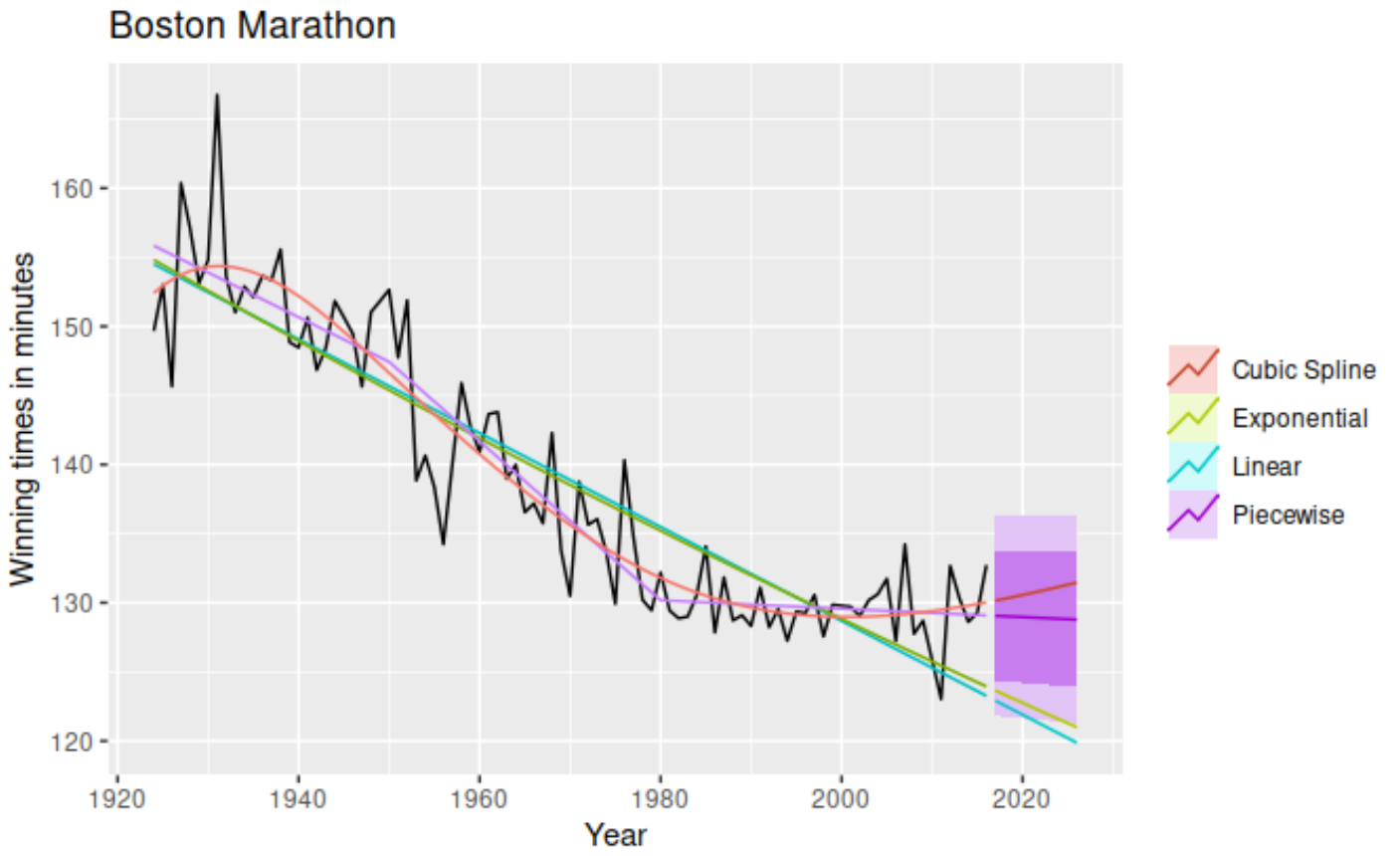

Figure 5.21: Projecting forecasts from a linear, exponential, piecewise linear trends and a cubic spline for the Boston marathon winning times

Figure 5.21 above shows the fitted lines and forecasts from linear, exponential, piecewise linear, and cubic spline trends. The best forecasts appear to come from the piecewise linear trend, while the cubic spline gives the best fit to the historical data but poor forecasts.

There is an alternative formulation of cubic splines (called **natural cubic smoothing splines**) that imposes some constraints, so the spline function is linear at the end, which usually gives much better forecasts without compromising the fit. In Figure 5.22, we have used the `splinef()` function to produce the cubic spline forecasts. This uses many more knots than we used in Figure 5.21, but the coefficients are constrained to prevent over-fitting, and the curve is linear at both ends. This has the added advantage that knot selection is not subjective. We have also used a log transformation (`lambda=0`) to handle the heteroscedasticity.

```
boston_men %>%
  splinef(lambda=0) %>%
  autoplot()
```

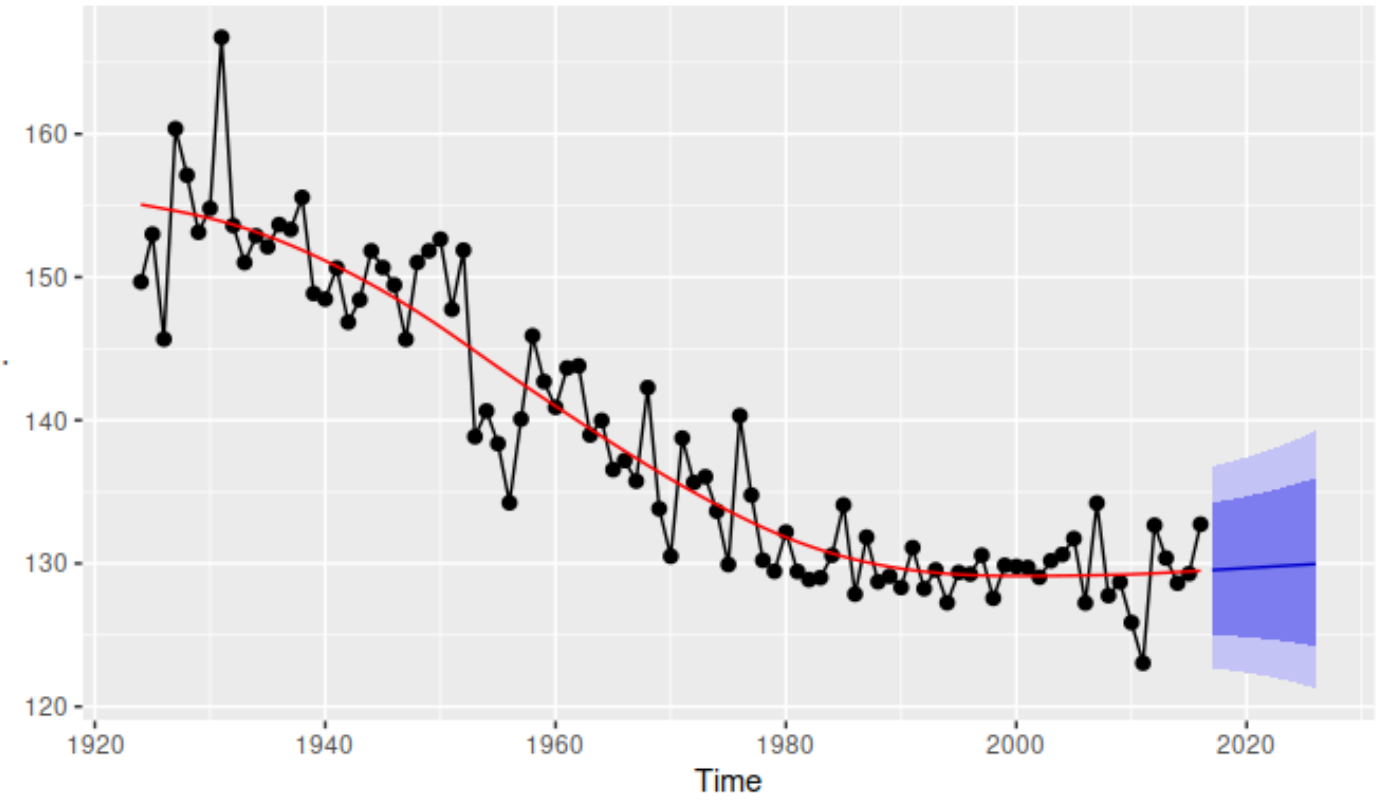

Figure 5.22: Natural cubic smoothing splines applied to the marathon data. The forecasts are a linear projection of the trend at the end of the observed data.

The residuals plotted in Figure 5.23 show that this model has captured the trend well, although there is some heteroscedasticity remaining. The wide prediction interval associated with the forecasts reflects the volatility observed in the historical winning times.

```
boston_men %>%
  splinef(lambda=0) %>%
  checkresiduals()
```

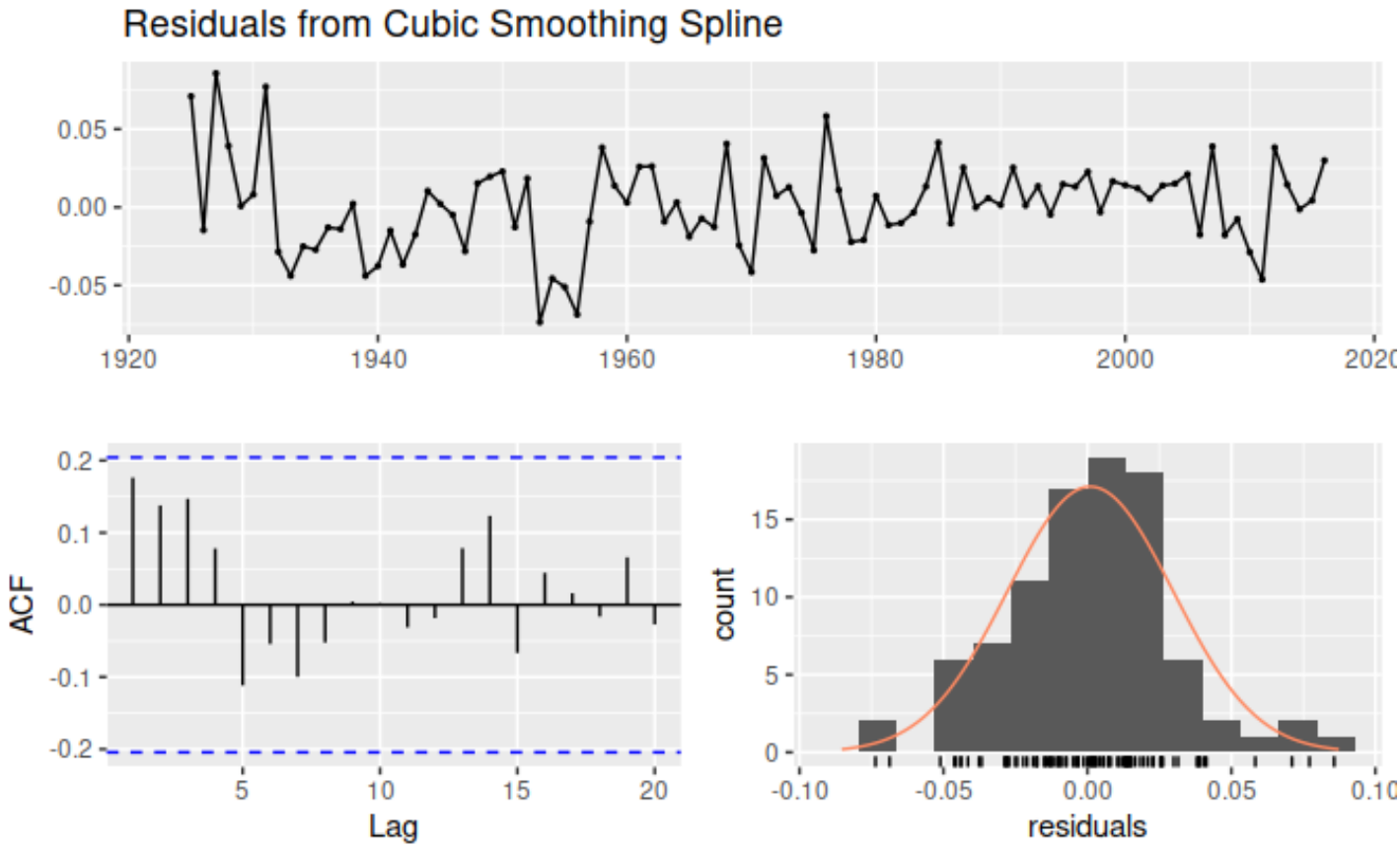

Figure 5.23: Residuals from the cubic spline trend.