11.1 Complex seasonality

So far, we have considered relatively simple seasonal patterns such as quarterly and monthly data. However, higher frequency time series often exhibit more complicated seasonal patterns. For example, daily data may have a weekly pattern as well as an annual pattern. Hourly data usually has three types of seasonality: a daily pattern, a weekly pattern, and an annual pattern. Even weekly data can be challenging to forecast as it typically has an annual pattern with seasonal period of $365.25/7 \approx 52.179$ on average.

Such multiple seasonal patterns are becoming more common with high frequency data recording. Further examples where multiple seasonal patterns can occur include call volume in call centres, daily hospital admissions, requests for cash at ATMs, electricity and water usage, and access to computer web sites.

Most of the methods we have considered so far are unable to deal with these seasonal complexities. Even the `ts` class in R can only handle one type of seasonality, which is usually assumed to take integer values.

To deal with such series, we will use the `msts` class which handles multiple seasonality time series. This allows you to specify all of the frequencies that might be relevant. It is also flexible enough to handle non-integer frequencies.

Despite this flexibility, we don't necessarily want to include all of these frequencies — just the ones that are likely to be present in the data. For example, if we have only 180 days of data, we may ignore the annual seasonality. If the data are measurements of a natural phenomenon (e.g., temperature), we can probably safely ignore any weekly seasonality.

The top panel of Figure 11.1 shows the number of retail banking call arrivals per 5-minute interval between 7:00am and 9:05pm each weekday over a 33 week period. The bottom panel shows the first three weeks of the same time series. There is a strong daily seasonal pattern with frequency 169 (there are 169 5-minute intervals per day), and a

weak weekly seasonal pattern with frequency $169 \times 5 = 845$. (Call volumes on Mondays tend to be higher than the rest of the week.) If a longer series of data were available, we may also have observed an annual seasonal pattern.

```
p1 <- autoplot(calls) +
  ylab("Call volume") + xlab("Weeks") +
  scale_x_continuous(breaks=seq(1,33,by=2))
p2 <- autoplot(window(calls, end=4)) +
  ylab("Call volume") + xlab("Weeks") +
  scale_x_continuous(minor_breaks = seq(1,4,by=0.2))
gridExtra::grid.arrange(p1,p2)
```

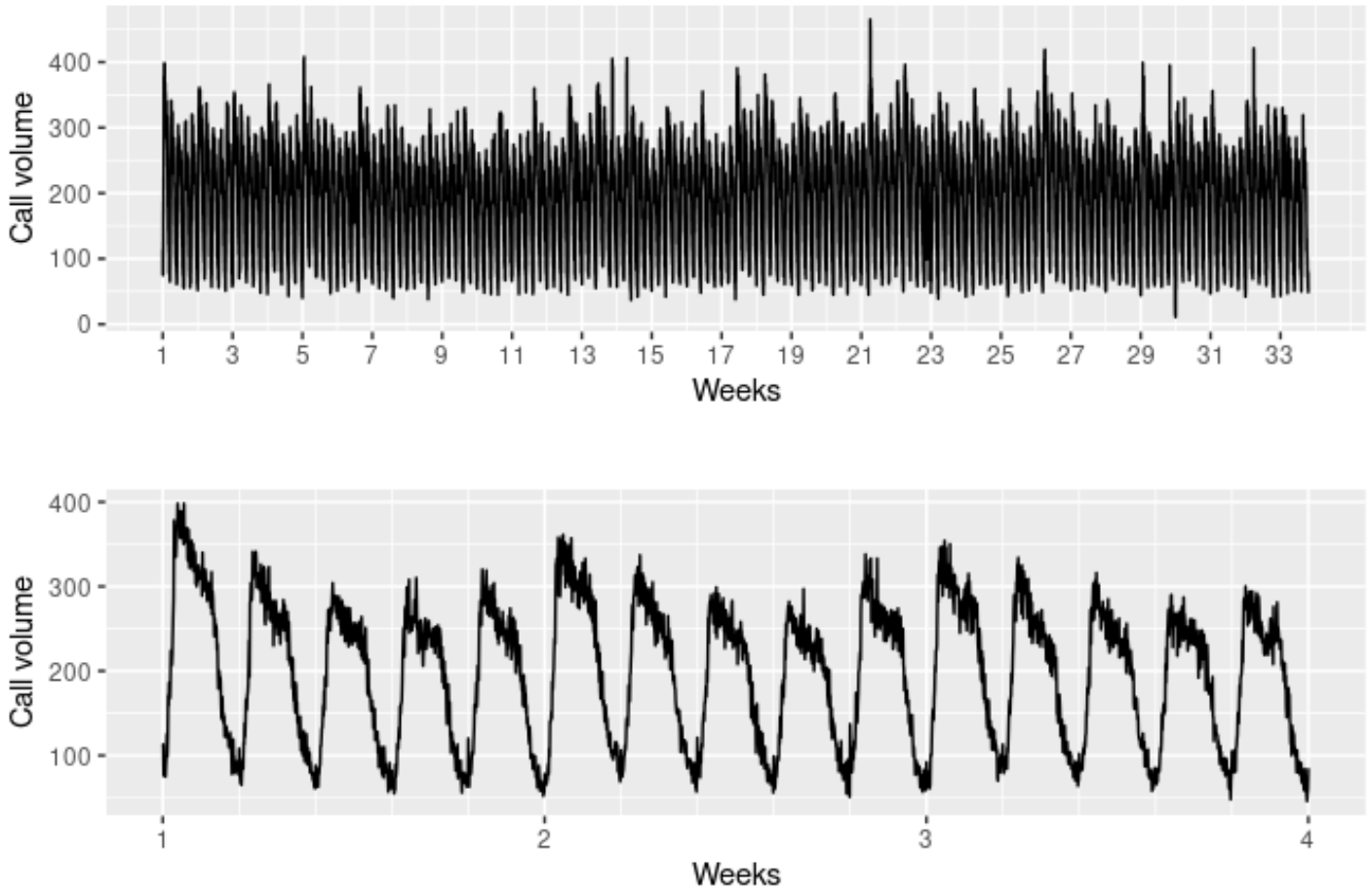

Figure 11.1: Five-minute call volume handled on weekdays between 7:00am and 9:05pm in a large North American commercial bank. Top panel shows data from 3 March 2003 for 164 days.

STL with multiple seasonal periods

The `mstl()` function is a variation on `stl()` designed to deal with multiple seasonality. It will return multiple seasonal components, as well as a trend and remainder component.

```
calls %>% mstl() %>%
  autoplot() + xlab("Week")
```

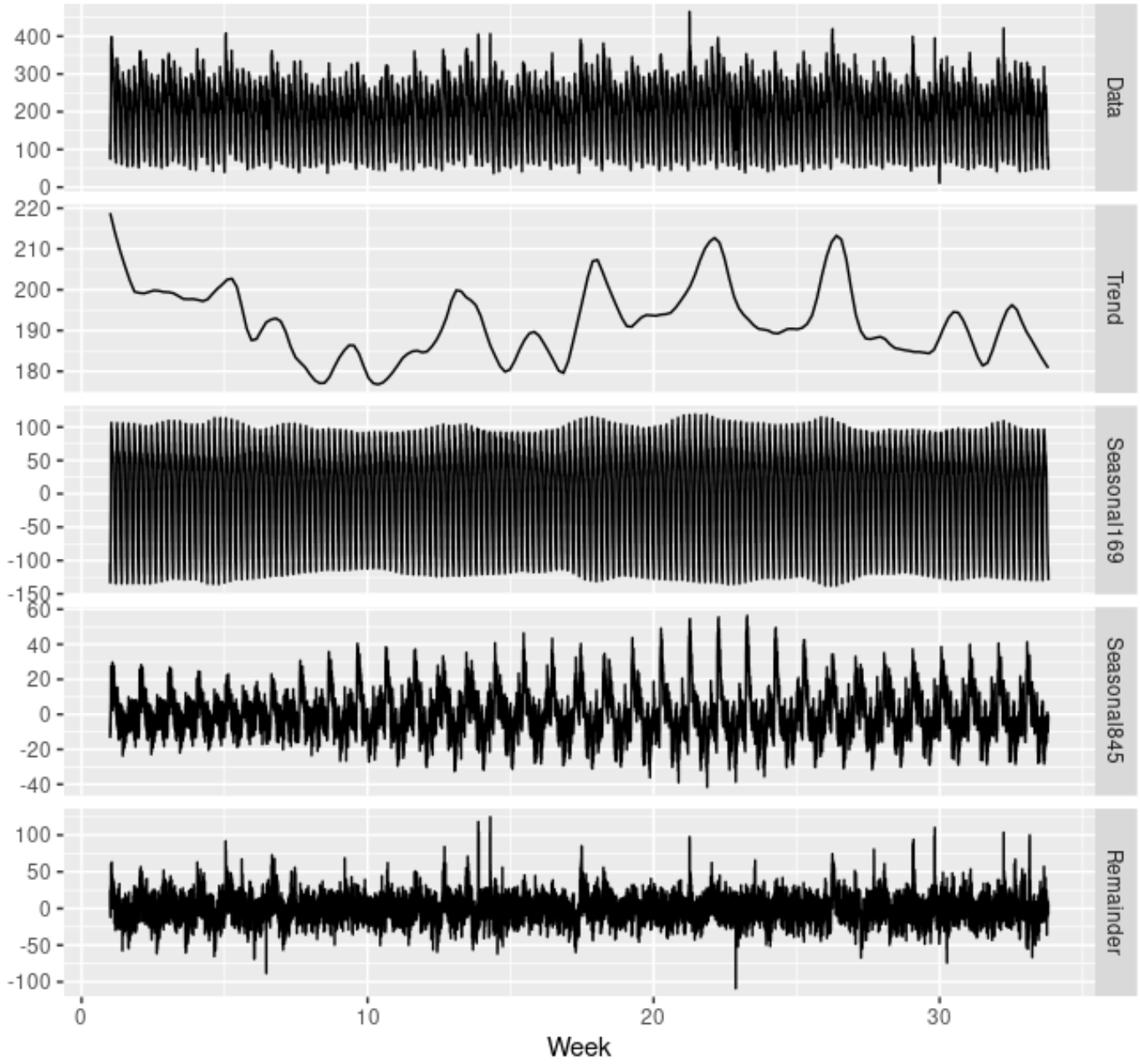

Figure 11.2: Multiple STL for the call volume data.

There are two seasonal patterns shown, one for the time of day (the third panel), and one for the time of week (the fourth panel). To properly interpret this graph, it is important to notice the vertical scales. In this case, the trend and the weekly seasonality

have relatively narrow ranges compared to the other components, because there is little trend seen in the data, and the weekly seasonality is weak.

The decomposition can also be used in forecasting, with each of the seasonal components forecast using a seasonal naïve method, and the seasonally adjusted data forecasting using ETS (or some other user-specified method). The `stlf()` function will do this automatically.

```
calls %>%  stlf() %>%
  autoplot() + xlab("Week")
```

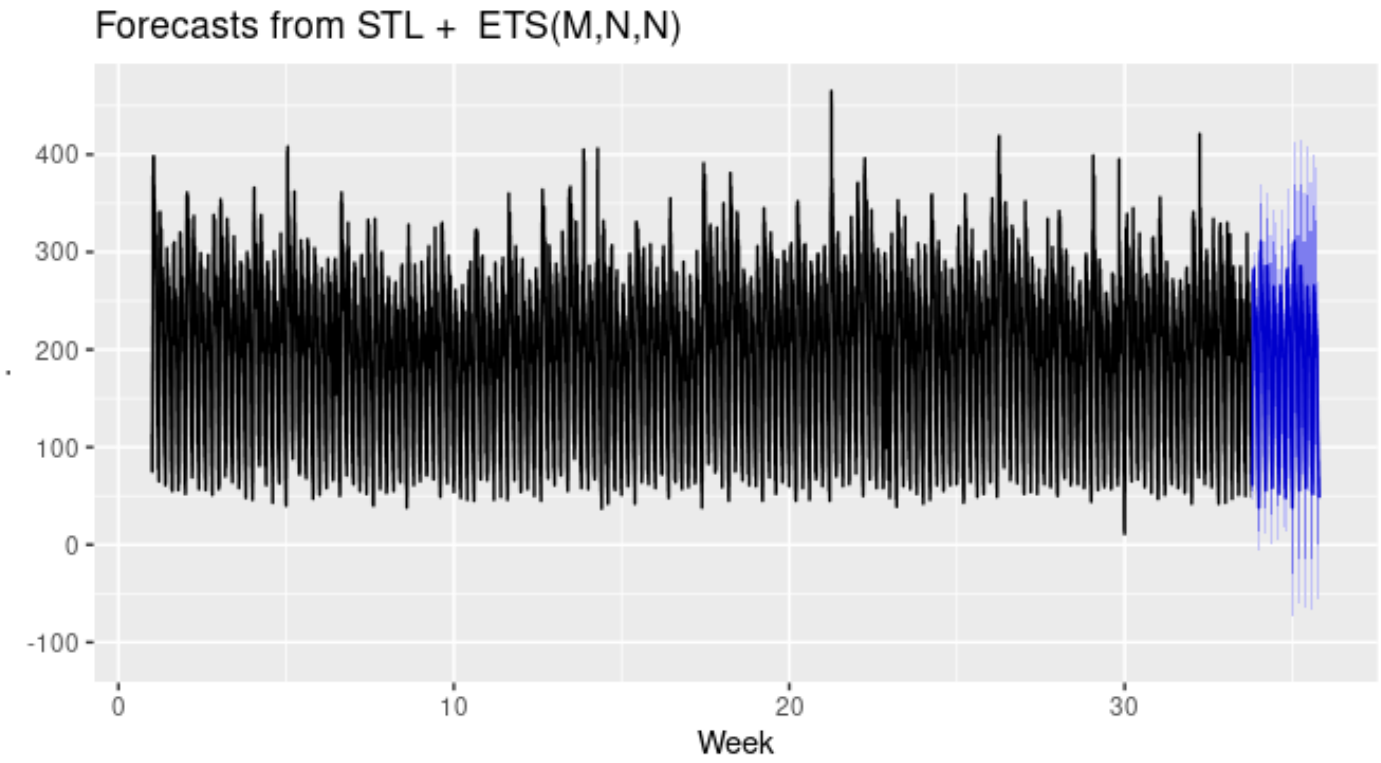

Figure 11.3: Multiple STL for the call volume data.

Dynamic harmonic regression with multiple seasonal periods

With multiple seasonalities, we can use Fourier terms as we did in earlier chapters (see Sections 5.4 and 9.5). Because there are multiple seasonalities, we need to add Fourier terms for each seasonal period. In this case, the seasonal periods are 169 and 845, so the Fourier terms are of the form

$$\sin\left(\frac{2\pi kt}{169}\right), \quad \cos\left(\frac{2\pi kt}{169}\right), \quad \sin\left(\frac{2\pi kt}{845}\right), \quad \text{and} \quad \cos\left(\frac{2\pi kt}{845}\right),$$

for $k = 1, 2, \ldots$. The `fourier()` function can generate these for you.

We will fit a dynamic harmonic regression model with an ARMA error structure. The total number of Fourier terms for each seasonal period have been chosen to minimise the AICc. We will use a log transformation (`lambda=0`) to ensure the forecasts and prediction intervals remain positive.

```
fit <- auto.arima(calls, seasonal=FALSE, lambda=0,
         xreg=fourier(calls, K=c(10,10)))
fit %>%
  forecast(xreg=fourier(calls, K=c(10,10), h=2*169)) %>%
  autoplot(include=5*169) +
    ylab("Call volume") + xlab("Weeks")
```

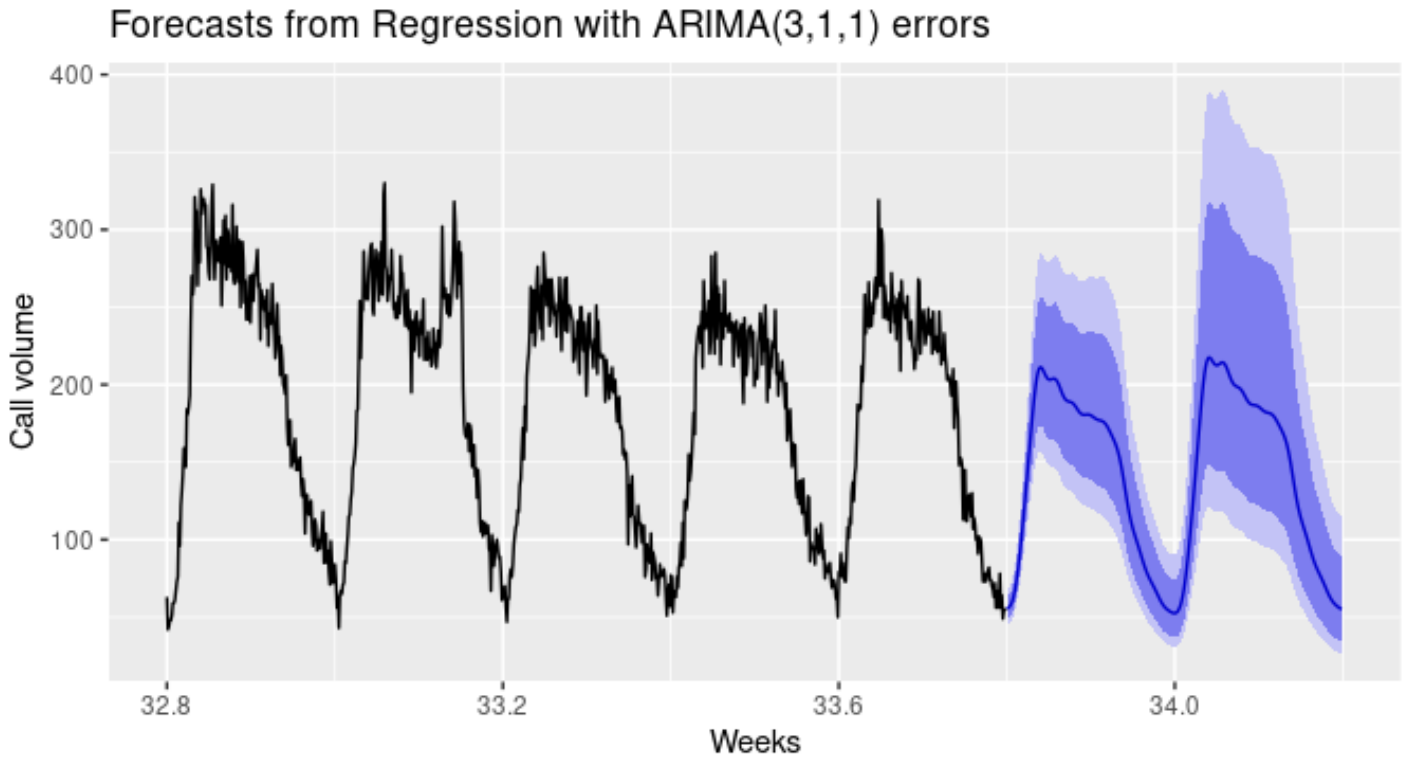

Figure 11.4: Forecasts from a dynamic harmonic regression applied to the call volume data.

This is a large model, containing 40 parameters: 4 ARMA coefficients, 20 Fourier coefficients for frequency 169, and 16 Fourier coefficients for frequency 845. We don't use all the Fourier terms for frequency 845 because there is some overlap with the terms of frequency 169 (since $845 = 5 \times 169$).

TBATS models

An alternative approach developed by De Livera, Hyndman, & Snyder (2011) uses a combination of Fourier terms with an exponential smoothing state space model and a Box-Cox transformation, in a completely automated manner. As with any automated modelling framework, there may be cases where it gives poor results, but it can be a useful approach in some circumstances.

A TBATS model differs from dynamic harmonic regression in that the seasonality is allowed to change slowly over time in a TBATS model, while harmonic regression terms force the seasonal patterns to repeat periodically without changing. One drawback of TBATS models, however, is that they can be slow to estimate, especially with long time series. Hence, we will consider a subset of the `calls` data to save time.

```
calls %>%
  subset(start=length(calls)-2000) %>%
  tbats() -> fit2
fc2 <- forecast(fit2, h=2*169)
autoplot(fc2, include=5*169) +
  ylab("Call volume") + xlab("Weeks")
```

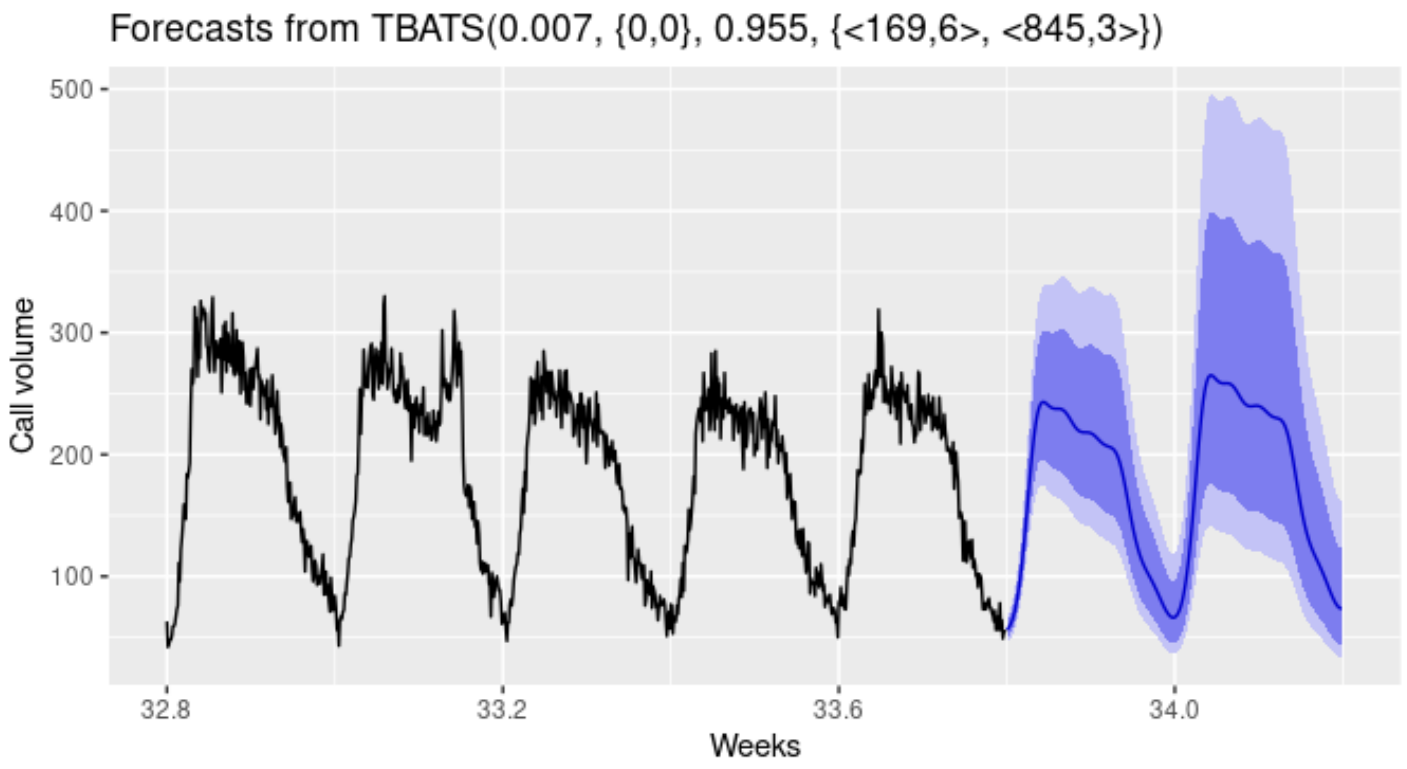

Figure 11.5: Forecasts from a TBATS model applied to the call volume data.

Here the prediction intervals appear to be much too wide – something that seems to happen quite often with TBATS models unfortunately.

Complex seasonality with covariates

TBATS models do not allow for covariates, although they can be included in dynamic harmonic regression models. One common application of such models is electricity demand modelling.

Figure 11.6 shows half-hourly electricity demand in Victoria, Australia, during 2014, along with temperatures for the same period for Melbourne (the largest city in Victoria).

```
autoplot(elecdemand[,c("Demand","Temperature")],
    facet=TRUE) +
  scale_x_continuous(minor_breaks=NULL,
    breaks=2014+
      cumsum(c(0,31,28,31,30,31,30,31,31,30,31,30))/365,
    labels=month.abb) +
  xlab("Time") + ylab("")
```

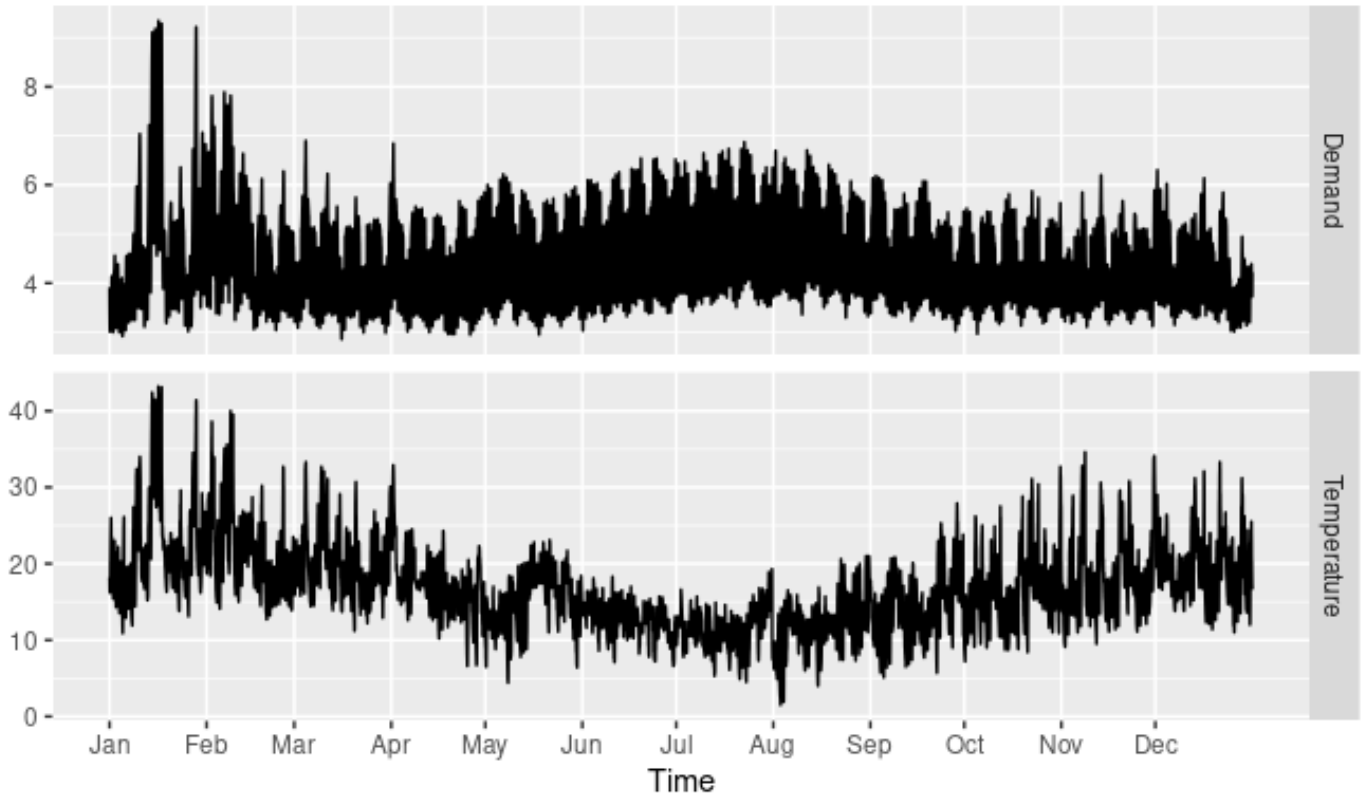

Figure 11.6: Half-hourly electricity demand and corresponding temperatures in 2014, Victoria, Australia.

Plotting electricity demand against temperature (Figure 11.7) shows that there is a nonlinear relationship between the two, with demand increasing for low temperatures (due to heating) and increasing for high temperatures (due to cooling).

```
elecdemand %>%
  as.data.frame() %>%
  ggplot(aes(x=Temperature, y=Demand)) + geom_point() +
    xlab("Temperature (degrees Celsius)") +
    ylab("Demand (GW)")
```

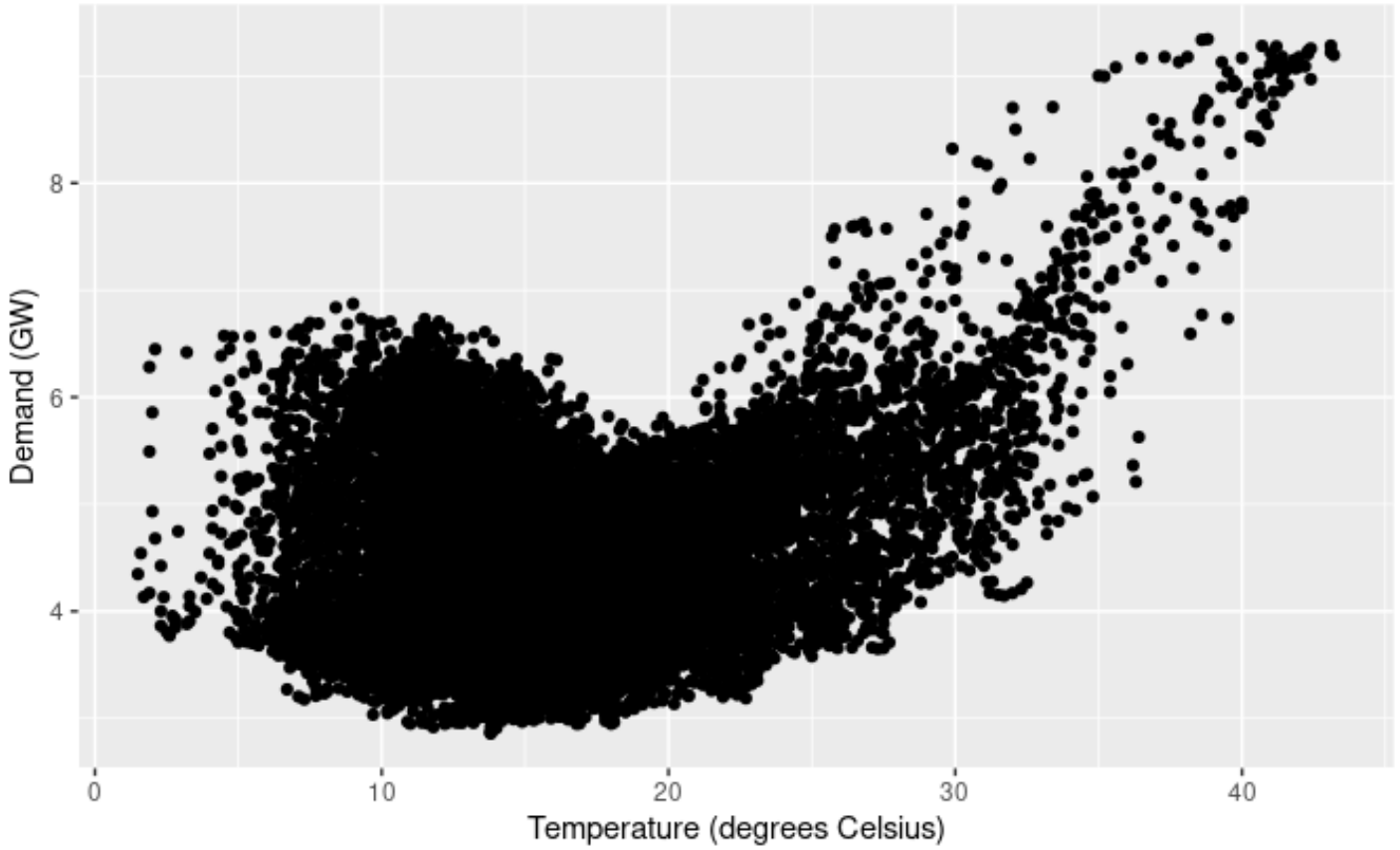

Figure 11.7: Half-hourly electricity demand for Victoria, plotted against temperatures for the same times in Melbourne, the largest city in Victoria.

We will fit a regression model with a piecewise linear function of temperature (containing a knot at 18 degrees), and harmonic regression terms to allow for the daily seasonal pattern.

```
cooling <- pmax(elecdemand[,"Temperature"], 18)
fit <- auto.arima(elecdemand[,"Demand"],
         xreg = cbind(fourier(elecdemand, c(10,10,0)),
                heating=elecdemand[,"Temperature"],
                cooling=cooling))
```

Forecasting with such models is difficult because we require future values of the predictor variables. Future values of the Fourier terms are easy to compute, but future temperatures are, of course, unknown. If we are only interested in forecasting up to a week ahead, we could use temperature forecasts obtain from a meteorological model. Alternatively, we could use scenario forecasting (Section 4.5) and plug in possible temperature patterns. In the following example, we have used a repeat of the last two days of temperatures to generate future possible demand values.

```
temps <- subset(elecdemand[,"Temperature"],
          start=NROW(elecdemand)-2*48+1)
fc <- forecast(fit,
        xreg=cbind(fourier(temps, c(10,10,0)),
          heating=temps, cooling=pmax(temps,18)))
autoplot(fc, include=14*48)
```

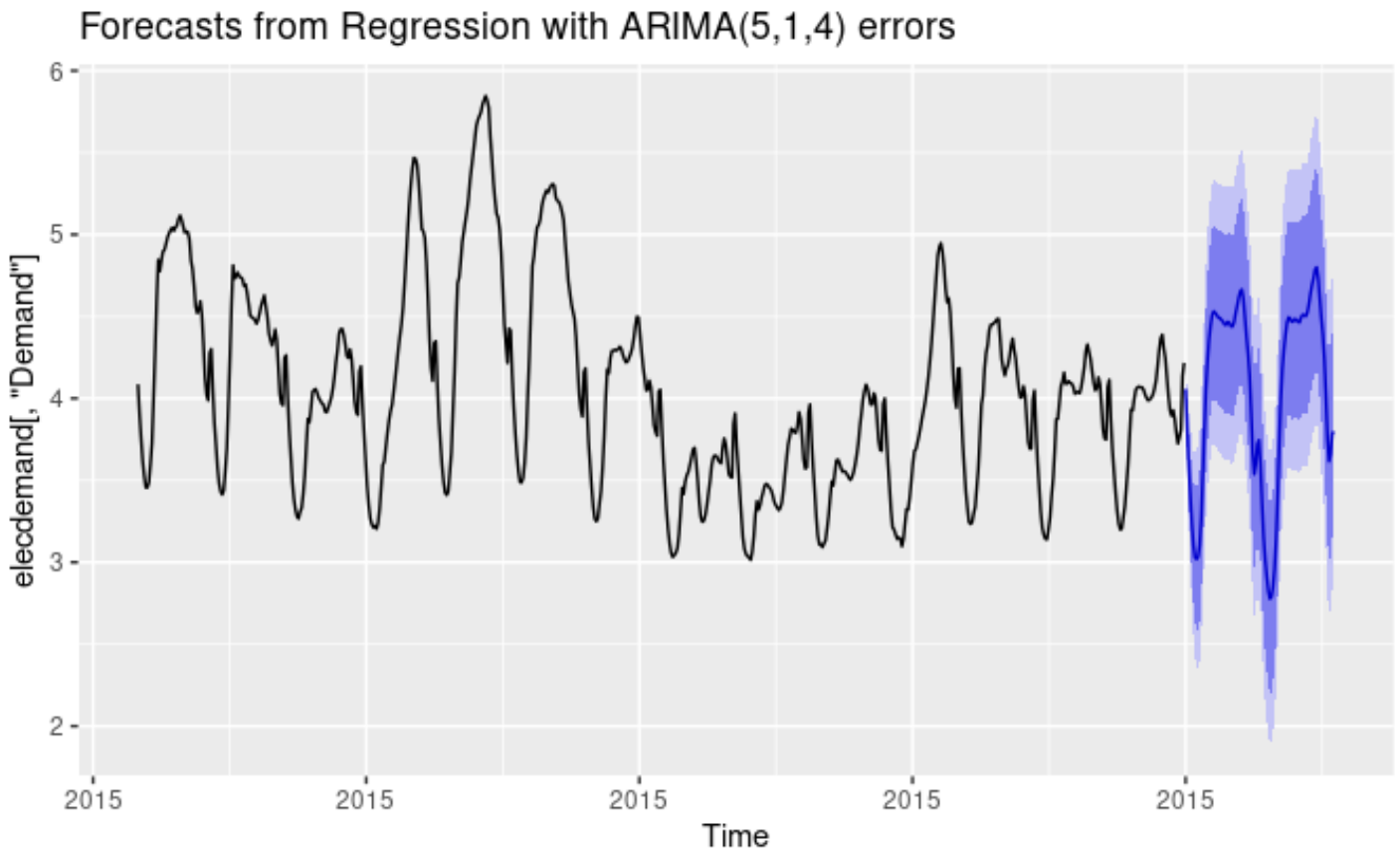

Figure 11.8: Forecasts from a dynamic harmonic regression model applied to half-hourly electricity demand data.

Although the short-term forecasts look reasonable, this is a crude model for a complicated process. The residuals demonstrate that there is a lot of information that has not been captured with this model.

```
checkresiduals(fc)
```

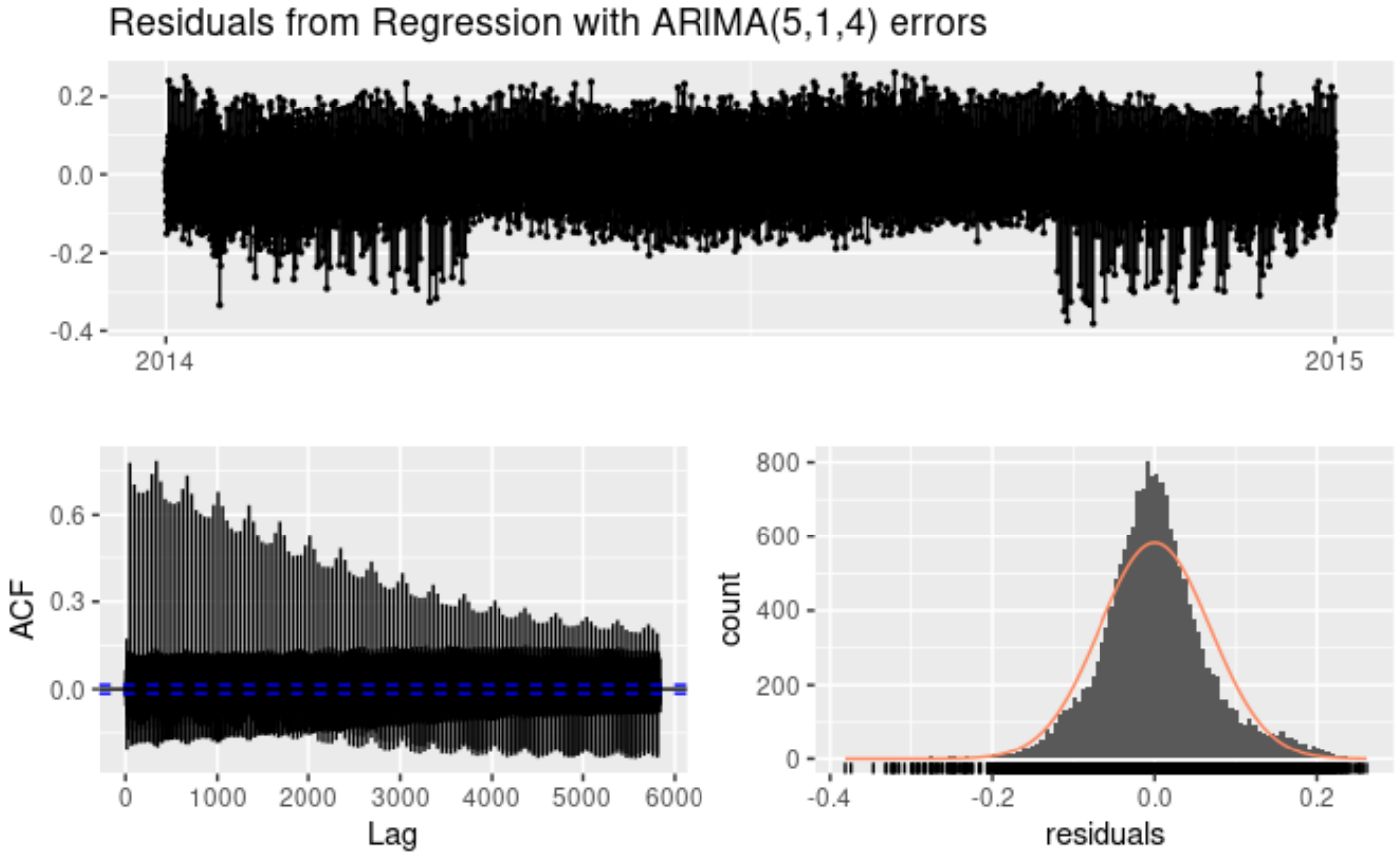

Figure 11.9: Residual diagnostics for the dynamic harmonic regression model.

```
#>
#>  Ljung-Box test
#>
#> data:  Residuals from Regression with ARIMA(5,1,4) errors
#> Q* = 738412, df = 3455, p-value <2e-16
#>
#> Model df: 49.   Total lags used: 3504
```

More sophisticated versions of this model which provide much better forecasts are described in Hyndman & Fan (2010) and Fan & Hyndman (2012).