11.4 Bootstrapping and bagging

Bootstrapping time series

In the preceding section, and in Section 3.5, we bootstrap the residuals of a time series in order to simulate future values of a series using a model.

More generally, we can generate new time series that are similar to our observed series, using another type of bootstrap.

First, the time series is Box-Cox-transformed, and then decomposed into trend, seasonal and remainder components using STL. Then we obtain shuffled versions of the remainder component to get bootstrapped remainder series. Because there may be autocorrelation present in an STL remainder series, we cannot simply use the re-draw procedure that was described in Section 3.5. Instead, we use a "blocked bootstrap," where contiguous sections of the time series are selected at random and joined together. These bootstrapped remainder series are added to the trend and seasonal components, and the Box-Cox transformation is reversed to give variations on the original time series.

Some examples are shown in Figure 11.16 for the monthly expenditure on retail debit cards in Iceland, from January 2000 to August 2013.

```
bootseries <- bld.mbb.bootstrap(debitcards, 10) %>%
  as.data.frame() %>% ts(start=2000, frequency=12)
autoplot(debitcards) +
  autolayer(bootseries, colour=TRUE) +
  autolayer(debitcards, colour=FALSE) +
  ylab("Bootstrapped series") + guides(colour="none")
```

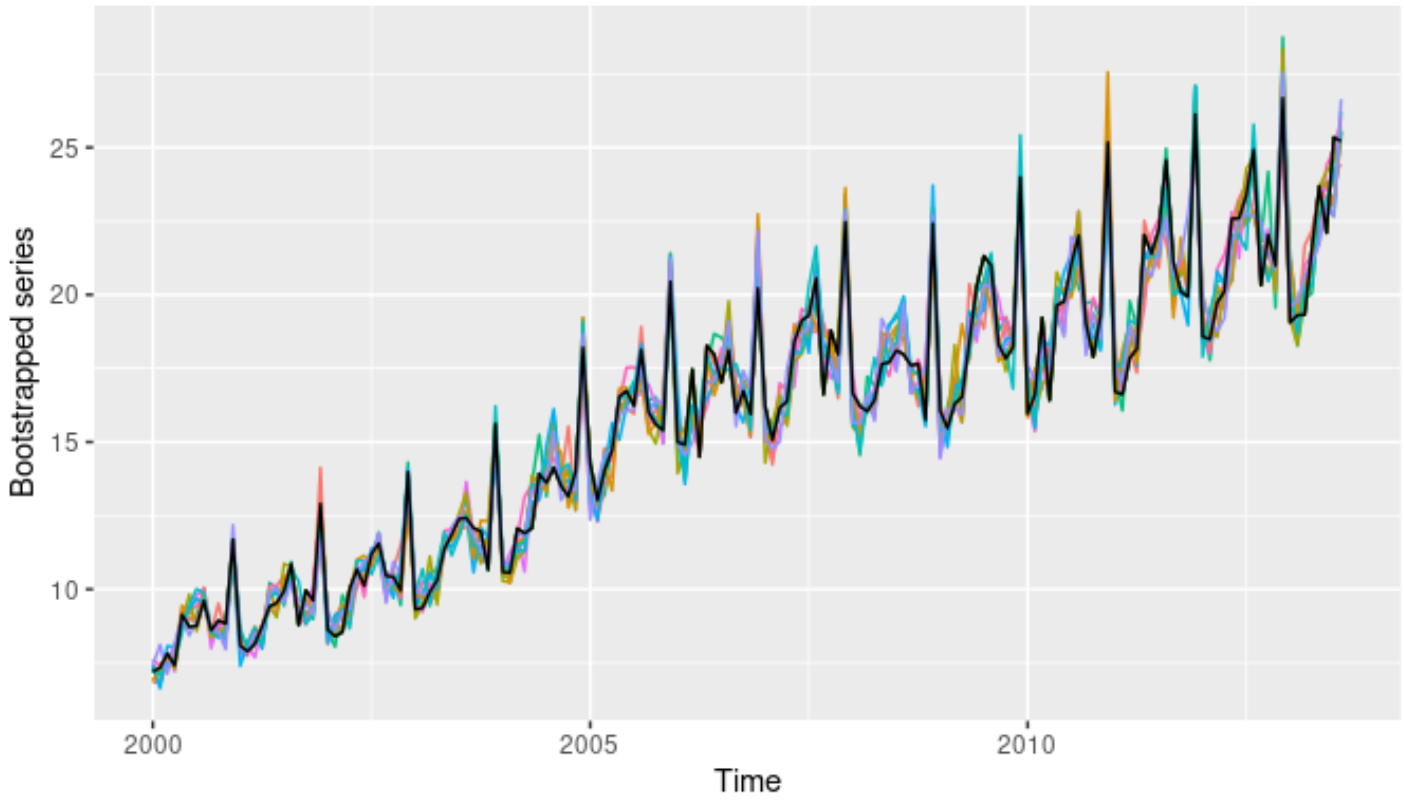

Figure 11.16: Ten bootstrapped versions of monthly expenditure on retail debit cards in Iceland.

This type of bootstrapping can be useful in two ways. First it helps us to get a better measure of forecast uncertainty, and second it provides a way of improving our point forecasts using "bagging."

Prediction intervals from bootstrapped series

Almost all prediction intervals from time series models are too narrow. This is a well-known phenomenon and arises because they do not account for all sources of uncertainty. Hyndman, Koehler, Snyder, & Grose (2002) measured the size of the problem by computing the actual coverage percentage of the prediction intervals on test data, and found that for ETS models, nominal 95% intervals may only provide coverage between 71% and 87%. The difference is due to missing sources of uncertainty.

There are at least four sources of uncertainty in forecasting using time series models:

1. The random error term;

2. The parameter estimates;
3. The choice of model for the historical data;
4. The continuation of the historical data generating process into the future.

When we produce prediction intervals for time series models, we generally only take into account the first of these sources of uncertainty. Even if we ignore the model uncertainty and the uncertainty due to changing data generating processes (sources 3 and 4), and we just try to allow for parameter uncertainty as well as the random error term (sources 1 and 2), there are no algebraic solutions apart from some simple special cases.

We can use bootstrapped time series to go some way towards overcoming this problem. We demonstrate the idea using the `debitcards` data. First, we simulate many time series that are similar to the original data, using the block-bootstrap described above.

```
nsim <- 1000L
sim <- bld.mbb.bootstrap(debitcards, nsim)
```

For each of these series, we fit an ETS model and simulate one sample path from that model. A different ETS model may be selected in each case, although it will most likely select the same model because the series are similar. However, the estimated parameters will be different. Therefore the simulated sample paths will allow for model uncertainty and parameter uncertainty, as well as the uncertainty associated with the random error term. This is a time-consuming process as there are a large number of time series to model.

```
h <- 36L
future <- matrix(0, nrow=nsim, ncol=h)
for(i in seq(nsim))
  future[i,] <- simulate(ets(sim[[i]]), nsim=h)
```

Finally, we take the means and quantiles of these simulated sample paths to form point forecasts and prediction intervals.

```
start <- tsp(debitcards)[2]+1/12
simfc <- structure(list(
    mean = ts(colMeans(future), start=start, frequency=12),
    lower = ts(apply(future, 2, quantile, prob=0.025),
               start=start, frequency=12),
    upper = ts(apply(future, 2, quantile, prob=0.975),
               start=start, frequency=12),
    level=95),
  class="forecast")
```

These prediction intervals will be larger than those obtained from an ETS model applied directly to the original data.

```
etsfc <- forecast(ets(debitcards), h=h, level=95)
autoplot(debitcards) +
  ggtitle("Monthly retail debit card usage in Iceland") +
  xlab("Year") + ylab("million ISK") +
  autolayer(simfc, series="Simulated") +
  autolayer(etsfc, series="ETS")
```

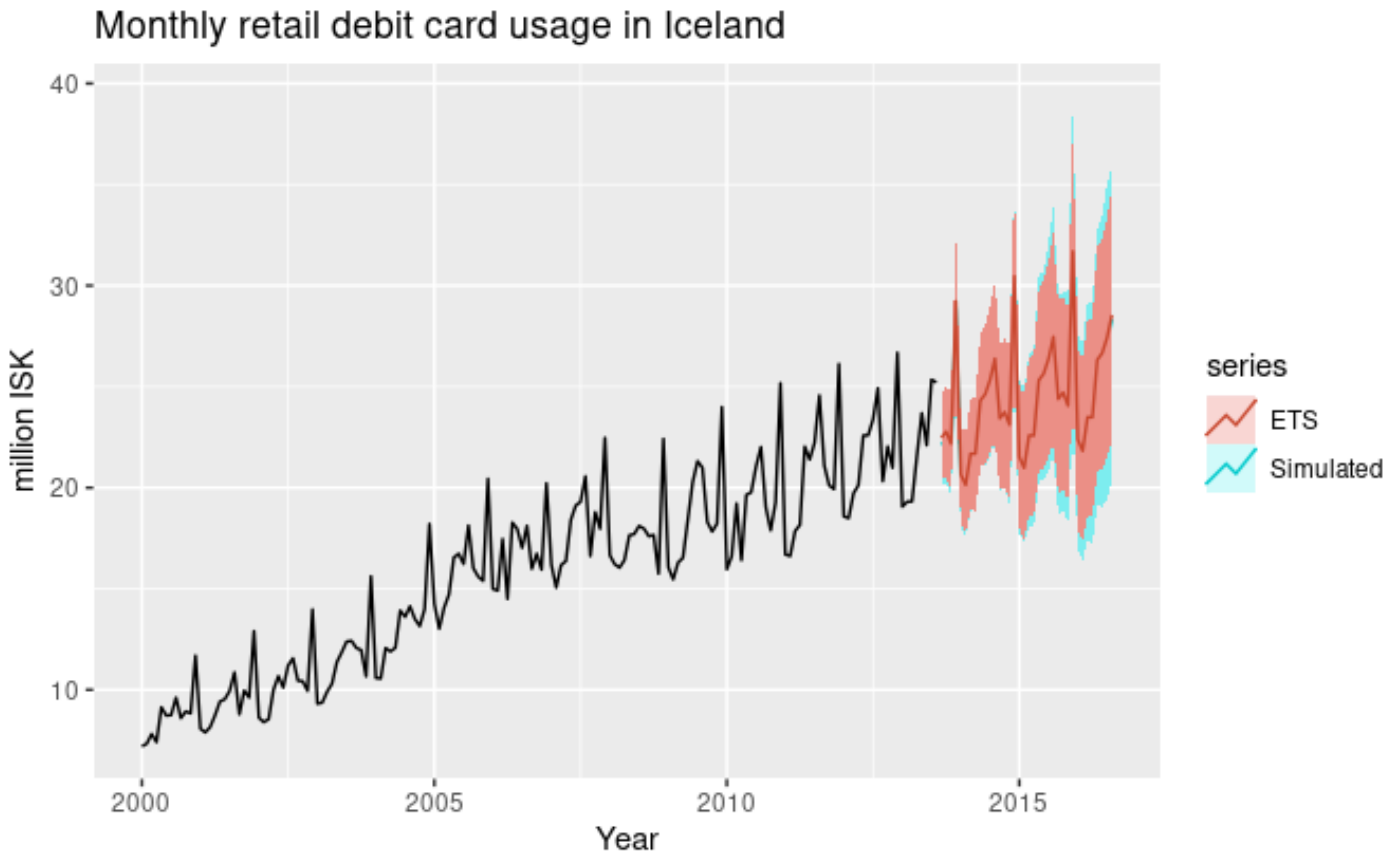

Figure 11.17: Forecasts of Iceland debit card usage using an ETS model with regular prediction intervals along with prediction intervals computed using simulations that allow for model and parameter uncertainty.

Bagged ETS forecasts

Another use for these bootstrapped time series is to improve forecast accuracy. If we produce forecasts from each of the additional time series, and average the resulting forecasts, we get better forecasts than if we simply forecast the original time series directly. This is called "bagging" which stands for "**b**ootstrap **agg**regatin**g**."

We could simply average the simulated future sample paths computed earlier. However, if our interest is only in improving point forecast accuracy, and not in also obtaining improved prediction intervals, then it is quicker to average the point forecasts from each series. The speed improvement comes about because we do not need to produce so many simulated series.

We will use `ets()` to forecast each of these series. Figure 11.18 shows ten forecasts obtained in this way.

```
sim <- bld.mbb.bootstrap(debitcards, 10) %>%
  as.data.frame() %>%
  ts(frequency=12, start=2000)
fc <- purrr::map(as.list(sim),
            function(x){forecast(ets(x))[["mean"]]}) %>%
      as.data.frame() %>%
      ts(frequency=12, start=start)
autoplot(debitcards) +
  autolayer(sim, colour=TRUE) +
  autolayer(fc, colour=TRUE) +
  autolayer(debitcards, colour=FALSE) +
  ylab("Bootstrapped series") +
  guides(colour="none")
```

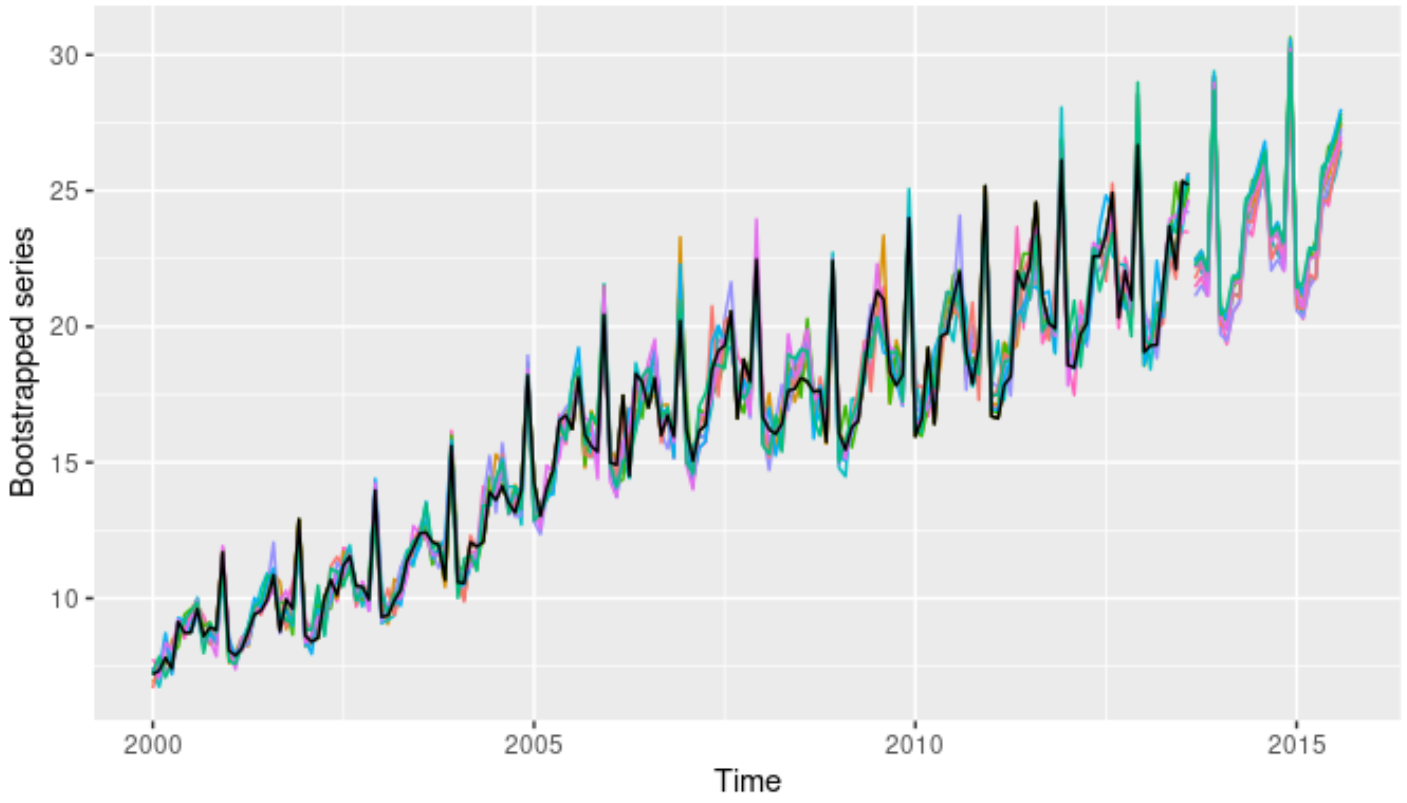

Figure 11.18: Forecasts of the ten bootstrapped series obtained using ETS models.

The average of these forecasts gives the bagged forecasts of the original data. The whole procedure can be handled with the `baggedETS()` function. By default, 100 bootstrapped series are used, and the length of the blocks used for obtaining bootstrapped residuals is set to 24 for monthly data. The resulting forecasts are shown in Figure 11.19.

```
etsfc <- debitcards %>% ets() %>% forecast(h=36)
baggedfc <- debitcards %>% baggedETS() %>% forecast(h=36)
autoplot(debitcards) +
  autolayer(baggedfc, series="BaggedETS", PI=FALSE) +
  autolayer(etsfc, series="ETS", PI=FALSE) +
  guides(colour=guide_legend(title="Forecasts"))
```

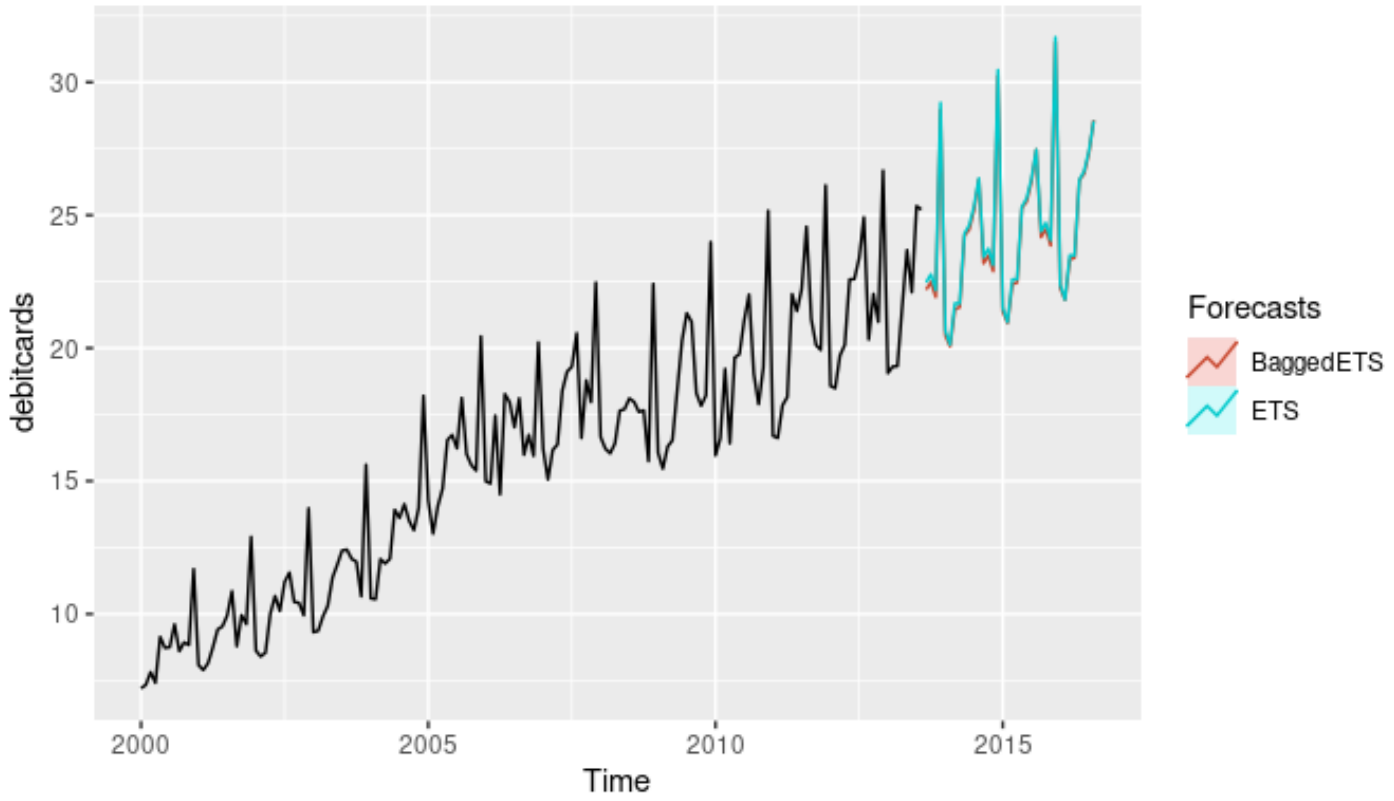

Figure 11.19: Comparing bagged ETS forecasts (the average of 100 bootstrapped forecast) and ETS applied directly to the data.

In this case, it makes little difference. Bergmeir, Hyndman, & Benítez (2016) show that, on average, bagging gives better forecasts than just applying `ets()` directly. Of course, it is slower because a lot more computation is required.