Forecasting: principles and practice

Rob J Hyndman

3.4 Extras
1. Bagged ETS
2. Models for different frequencies
3. Ensuring forecasts stay within limits
4. Forecast combinations
5. Prediction intervals for aggregates
6. Backcasting
7. Missing values
8. Outliers
Bagged ETS

Algorithm: Generating bootstrapped series

bootstrap ← function(ts, num.boot) {
    lambda ← BoxCox.lambda(ts, min=0, max=1)
    ts.bc ← BoxCox(ts, lambda)
    if(ts is seasonal) {
        [trend, seasonal, remainder] ← stl(ts.bc)
    }
    else {
        seasonal ← 0
        [trend, remainder] ← loess(ts.bc)
    }
    recon.series[1] ← ts
    for(i in 2:num.boot) {
        boot.sample[i] ← MBB(remainder)
        recon.series.bc[i] ← trend + seasonal + boot.sample[i]
        recon.series[i] ← InvBoxCox(recon.series.bc[i], lambda)
    }
    return(recon.series)
Bagged ETS
Bagged ETS

![Graph showing the Bagged ETS over the years 1982 to 1992. The graph plots the M495 values with a y-axis ranging from 3000 to 6000 and an x-axis for the years 1982 to 1992. The data shows a general upward trend with fluctuations.]
Bagged ETS

![Graph showing time series data for Bagged ETS]


M495: 3000, 4000, 5000, 6000
Bagged ETS

```r
baggedETS(Mcomp::M3[[1896]]$x) %>%
  forecast %>% autoplot +
  xlab("Year") + ylab("M495")
```
Intervals show range of point forecasts
They are not prediction intervals
1. Bagged ETS
2. Models for different frequencies
3. Ensuring forecasts stay within limits
4. Forecast combinations
5. Prediction intervals for aggregates
6. Backcasting
7. Missing values
8. Outliers
Models for different frequencies

Models for annual data
- ETS, ARIMA, Dynamic regression
## Models for different frequencies

### Models for annual data
- ETS, ARIMA, Dynamic regression

### Models for quarterly data
- ETS, ARIMA/SARIMA, Dynamic regression, Dynamic harmonic regression, STL+ETS, STL+ARIMA
Models for different frequencies

Models for annual data
- ETS, ARIMA, Dynamic regression

Models for quarterly data
- ETS, ARIMA/SARIMA, Dynamic regression, Dynamic harmonic regression, STL+ETS, STL+ARIMA

Models for monthly data
- ETS, ARIMA/SARIMA, Dynamic regression, Dynamic harmonic regression, STL+ETS, STL+ARIMA
Models for different frequencies

Models for weekly data

- ARIMA/SARIMA, Dynamic regression, Dynamic harmonic regression, STL+ETS, STL+ARIMA, TBATS
Models for different frequencies

### Models for weekly data
- ARIMA/SARIMA, Dynamic regression, Dynamic harmonic regression, STL+ETS, STL+ARIMA, TBATS

### Models for daily, hourly and other sub-daily data
- ARIMA/SARIMA, Dynamic regression, Dynamic harmonic regression, STL+ETS, STL+ARIMA, TBATS
Outline

1. Bagged ETS
2. Models for different frequencies
3. Ensuring forecasts stay within limits
4. Forecast combinations
5. Prediction intervals for aggregates
6. Backcasting
7. Missing values
8. Outliers
Positive forecasts

```r
eggs %>%
  ets(model="AAN", damped=FALSE, lambda=0) %>%
  forecast(h=50, biasadj=TRUE) %>%
  autoplot()
```

Forecasts from ETS(A,A,N)
Suppose egg prices constrained to lie within $a = 50$ and $b = 400$.

Transform data using scaled logit transform:

$$y = \log \left( \frac{x - a}{b - x} \right),$$

where $x$ is on the original scale and $y$ is the transformed data. To reverse the transformation, we will use

$$x = \frac{(b - a)e^y}{1 + e^y} + a.$$
Forecasts constrained to an interval

```r
# Bounds
a <- 50
b <- 400

# Transform data and fit model
fit <- log((eggs-a)/(b-eggs)) %>% ets(model="AAN", damped=FALSE)
fic <- forecast(fit, h=50)

# Back-transform forecasts
fic[['mean']] <- (b-a)*exp(fic[['mean']]) / (1+exp(fic[['mean']])) + a
fic[['lower']] <- (b-a)*exp(fic[['lower']]) / (1+exp(fic[['lower']])) + a
fic[['upper']] <- (b-a)*exp(fic[['upper']]) / (1+exp(fic[['upper']])) + a
fic[['x']] <- eggs
autoplot(fic)
```
Forecasts constrained to an interval

Forecasts from ETS(A,A,N)
Outline

1. Bagged ETS
2. Models for different frequencies
3. Ensuring forecasts stay within limits
4. Forecast combinations
5. Prediction intervals for aggregates
6. Backcasting
7. Missing values
8. Outliers
Clemen (1989)

“The results have been virtually unanimous: combining multiple forecasts leads to increased forecast accuracy. ... In many cases one can make dramatic performance improvements by simply averaging the forecasts.”
Forecast combinations

```r
train <- window(auscafe, end=c(2012,9))
h <- length(auscafe) - length(train)
ETS <- forecast(ets(train), h=h)
ARIMA <- forecast(auto.arima(train, lambda=0, biasadj=TRUE), h=h)
STL <- stlf(train, lambda=0, h=h, biasadj=TRUE)
NNAR <- forecast(nnetar(train), h=h)
TBATS <- forecast(tbats(train, biasadj=TRUE), h=h)
Combination <- (ETS[["mean"]]
               + ARIMA[["mean"]]
               + STL[["mean"]]
               + NNAR[["mean"]]
               + TBATS[["mean"]])/5

autoplot(auscafe) +
    autolayer(ETS, series="ETS", PI=FALSE) +
    autolayer(ARIMA, series="ARIMA", PI=FALSE) +
    autolayer(STL, series="STL", PI=FALSE) +
    autolayer(NNAR, series="NNAR", PI=FALSE) +
    autolayer(TBATS, series="TBATS", PI=FALSE) +
    autolayer(Combination, series="Combination") +
    xlab("Year") + ylab("$ billion") +
    ggtitle("Australian monthly expenditure on eating out")
```
Forecast combinations

Australian monthly expenditure on eating out
Forecast combinations

\[
c(\text{ETS} = \text{accuracy}(\text{ETS, auscafe})["Test set","RMSE"]],
\text{ARIMA} = \text{accuracy}(\text{ARIMA, auscafe})["Test set","RMSE"]],
\text{STL-ETS} = \text{accuracy}(\text{STL, auscafe})["Test set","RMSE"]],
\text{NNAR} = \text{accuracy}(\text{NNAR, auscafe})["Test set","RMSE"]],
\text{TBATS} = \text{accuracy}(\text{TBATS, auscafe})["Test set","RMSE"]],
\text{Combination} =
\text{accuracy}(\text{Combination, auscafe})["Test set","RMSE"])
\]

<table>
<thead>
<tr>
<th></th>
<th>ETS</th>
<th>ARIMA</th>
<th>STL-ETS</th>
<th>NNAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>0.13700</td>
<td>0.12146</td>
<td>0.21446</td>
<td>0.29801</td>
</tr>
<tr>
<td>TBATS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.09406</td>
<td>0.07090</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Prediction intervals for aggregates

```r
# First fit a model to the data
fit <- ets(gas/1000)
# Forecast six months ahead
fc <- forecast(fit, h=6)
sum(fc[["mean"]][1:6])

## [1] 281.8

# Simulate 10000 future sample paths
nsim <- 10000
h <- 6
sim <- numeric(nsim)
for(i in seq_len(nsim))
  sim[i] <- sum(simulate(fit, future=TRUE, nsim=h))
mean(sim)

## [1] 281.9
```
Prediction intervals for aggregates

#80% interval:
`quantile(sim, prob=c(0.1, 0.9))`

## 10% 90%
## 262.9 300.9

#95% interval:
`quantile(sim, prob=c(0.025, 0.975))`

## 2.5% 97.5%
## 253.2 311.2
# Function to reverse time
reverse_ts <- function(y)
{
  ts(rev(y), start=tsp(y)[1L], frequency=frequency(y))
}

# Function to reverse a forecast
reverse_forecast <- function(object)
{
  h <- length(object[["mean"]])
  f <- frequency(object[["mean"]])
  object[["x"]]<- reverse_ts(object[["x"]])
  object[["mean"]]<- ts(rev(object[["mean"]]),
                           end=tsp(object[["x"]])[1L]-1/f, frequency=f)
  object[["lower"]]<- object[["lower"]][h:1L,]
  object[["upper"]]<- object[["upper"]][h:1L,]
  return(object)
}
euretail %>% reverse_ts() %>% auto.arima() %>% forecast() %>% reverse_forecast() -> bc
autoplot(bc) + ggtitle(paste("Backcasts from", bc[["method"]]))

Backcasts from ARIMA(1,1,2)(0,1,1)[4]

Time
level
80
95
Backcasts from ARIMA(1,1,2)(0,1,1)[4]

level
80
95
Outline

1. Bagged ETS
2. Models for different frequencies
3. Ensuring forecasts stay within limits
4. Forecast combinations
5. Prediction intervals for aggregates
6. Backcasting
7. Missing values
8. Outliers
Missing values

Functions which can handle missing values

- auto.arima()
- Arima()
- tslm()
- nnetar()

Models which cannot handle missing values

- ets()
- stl()
- stlf()
- tbats()
Missing values

Functions which can handle missing values

- auto.arima()
- Arima()
- tslm()
- nnetar()

Models which cannot handle missing values

- ets()
- stl()
- stlf()
- tbats()

What to do?

1. Model section of data after last missing value.
2. Estimate missing values with na.interp().
Missing values

**autoplot(gold)**

![Graph showing time series data for gold](image-url)
Missing values

gold %>% na.interp() %>%
autoplot(series="Interpolated") +
autolayer(gold, series="Original") +
scale_color_manual(
values=c(Interpolated="red", Original="gray"))
Outliers

**autoplot**(gold)
Outliers

\texttt{tsoutliers}(gold)

## $index
## [1] 770

## $replacements
## [1] 494.9
Outliers

\[
gold \ %>>\ tsclean() \ %>>\ autoplot()
\]