MONASH University

MONASH BUSINESS SCHOOL

Forecasting: principles and practice

Rob J Hyndman

3.4 Extras

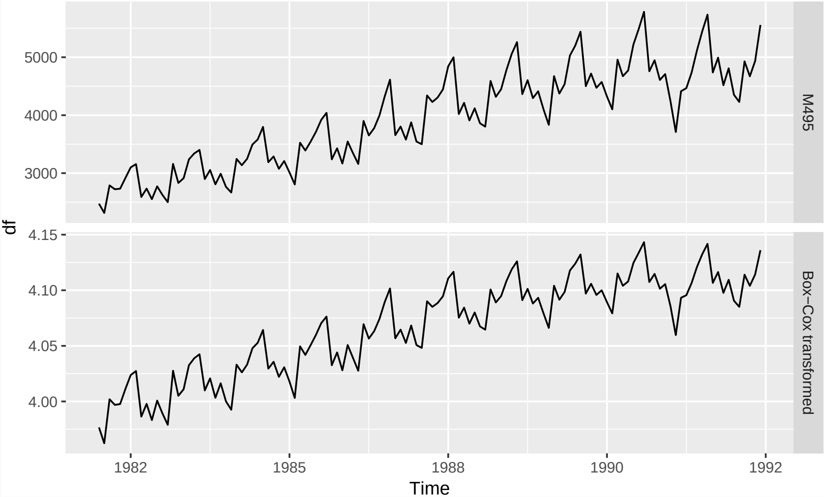
Outline

Bagged ETS

Algorithm: Generating bootstrapped series

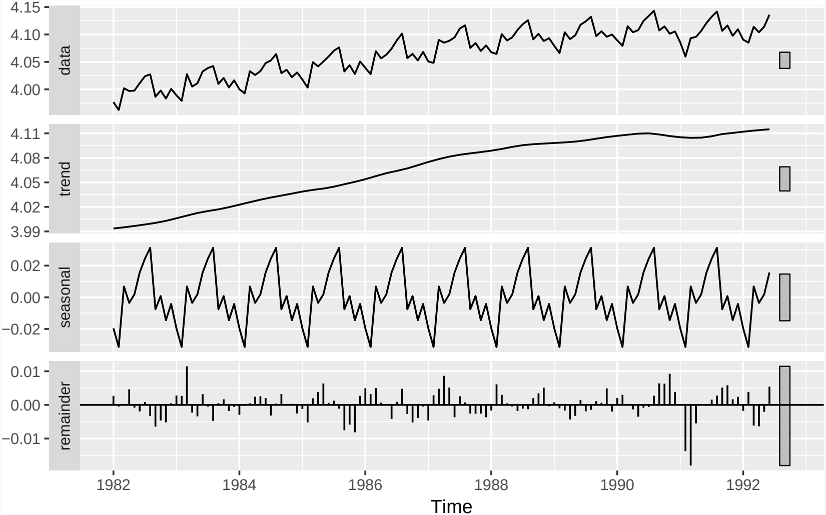
```
bootstrap ← function(ts, num.boot) {
  lambda ← BoxCox.lambda(ts, min=0, max=1)
  ts.bc ← BoxCox(ts, lambda)
  if(ts is seasonal) {
    [trend, seasonal, remainder] ← stl(ts.bc)
  }
  else {
    seasonal ← 0
    [trend, remainder] ← loess(ts.bc)
  }
  recon.series[1] ← ts
  for(i in 2:num.boot) {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



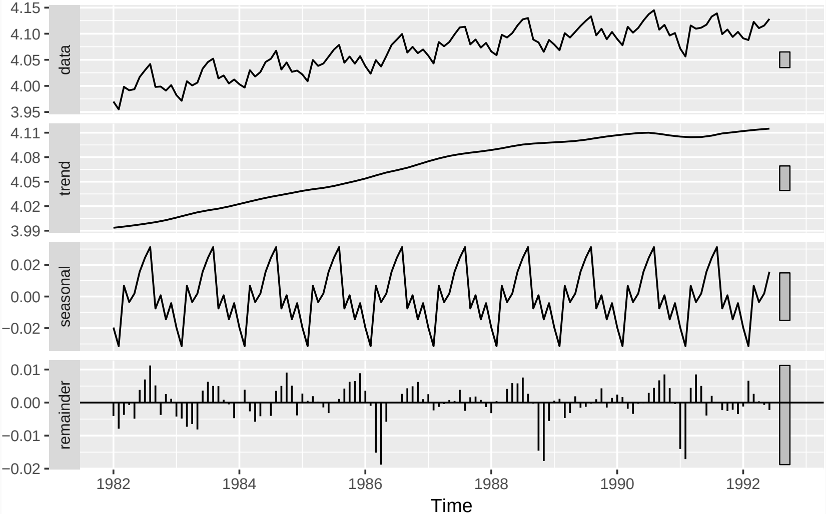
5000
4000
3000
4.15
4.10
4.05
4.00
df
M495
Box−Cox transformed
1982
1985
1988
1990
1992
Time

Bagged ETS



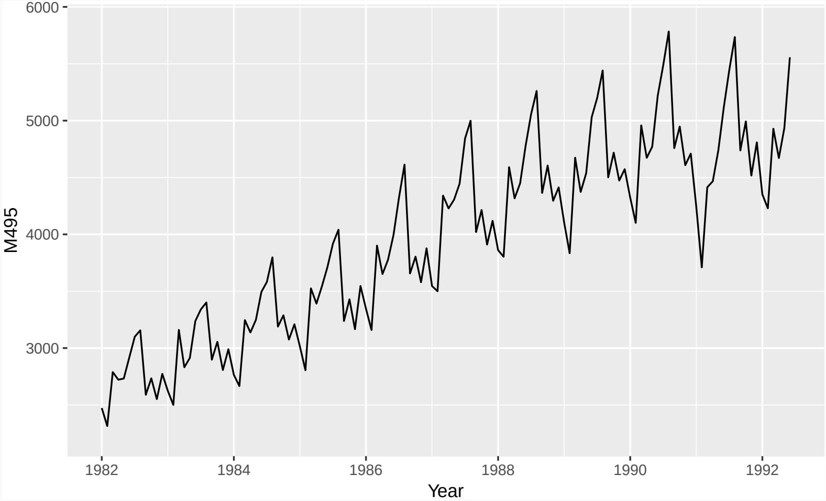
data
trend
seasonal
remainder
4.15
4.10
4.05
4.00
4.11
4.08
4.05
4.02
3.99
0.02
0.00
−0.02
0.01
0.00
−0.01
1982
1984
1986
1988
1990
1992
Time

Bagged ETS

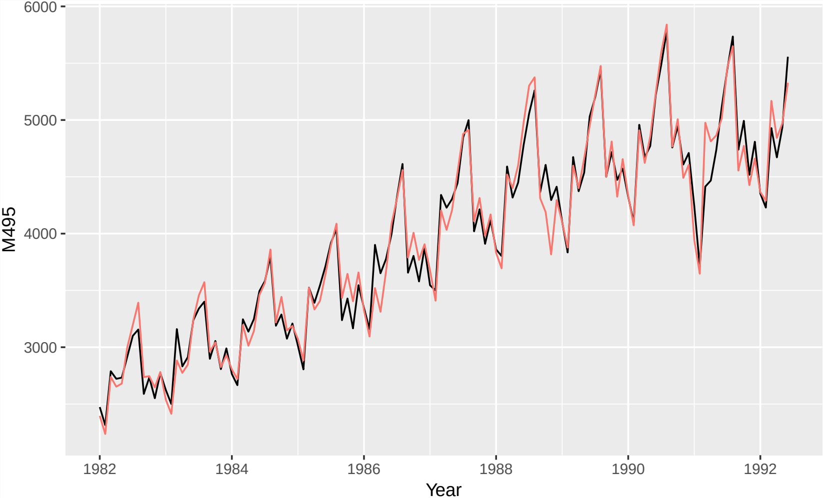


data
trend
seasonal
remainder
4.15
4.10
4.05
4.00
3.95
4.11
4.08
4.05
4.02
3.99
0.02
0.00
−0.02
0.01
0.00
−0.01
−0.02
1982
1984
1986
1988
1990
1992
Time

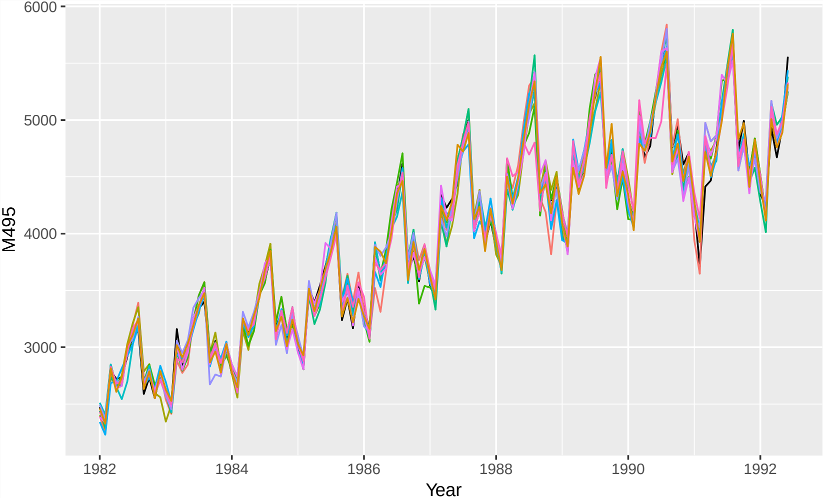
Bagged ETS



Bagged ETS

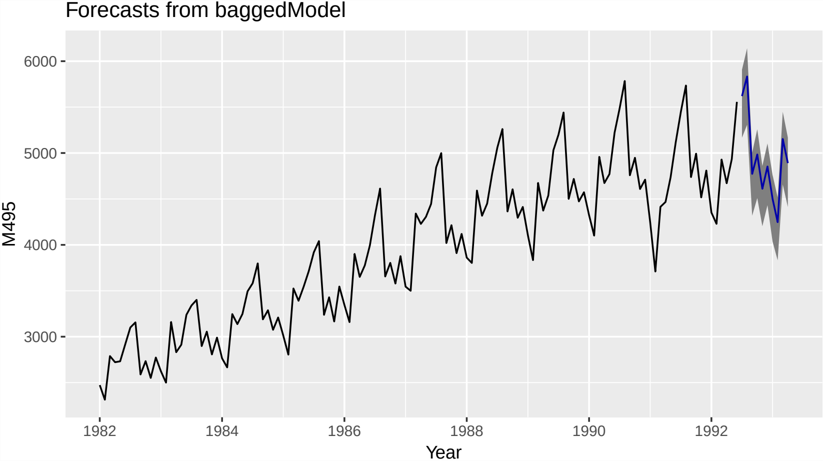


Bagged ETS

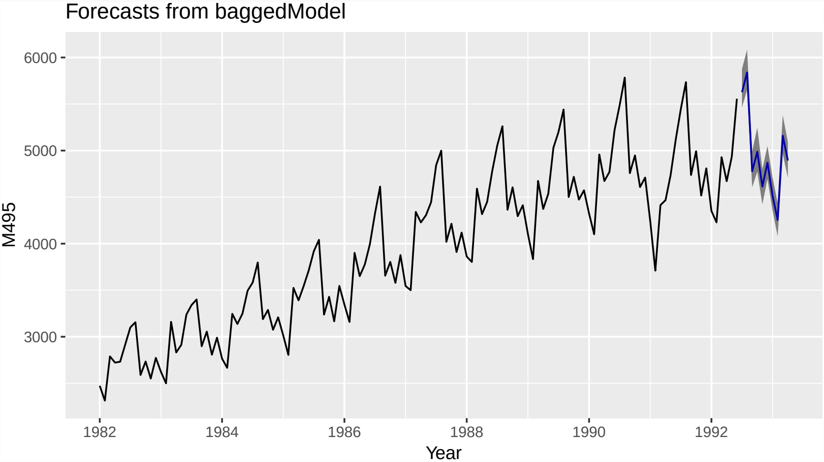


Bagged ETS

```
baggedETS(Mcomp::M3[[1896]]$x) %>%
  forecast %>% autoplot +
  xlab("Year") + ylab("M495")
```



Bagged ETS



- Intervals show range of point forecasts
- They are not prediction intervals

Outline

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

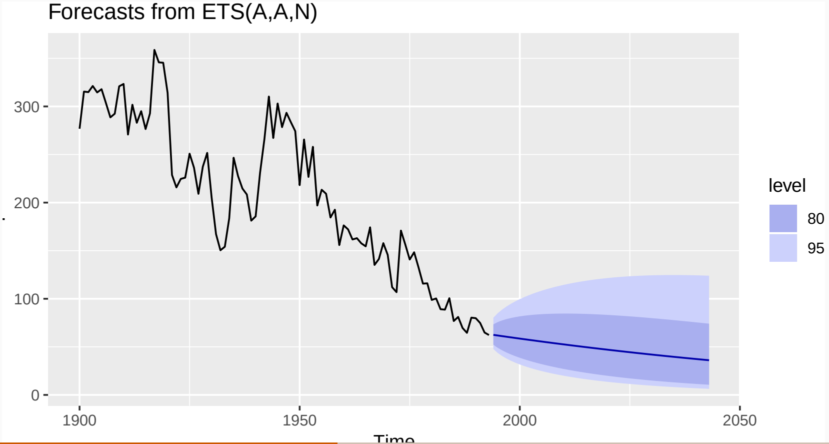
Models for daily, hourly and other sub-daily data

- ARIMA/SARIMA, Dynamic regression, Dynamic harmonic regression, STL+ETS, STL+ARIMA, TBATS

Outline

Positive forecasts

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



Forecasts constrained to an interval

Suppose egg prices constrained to lie within $a = 50$ and $b = 400$.

Transform data using scaled logit transform:

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

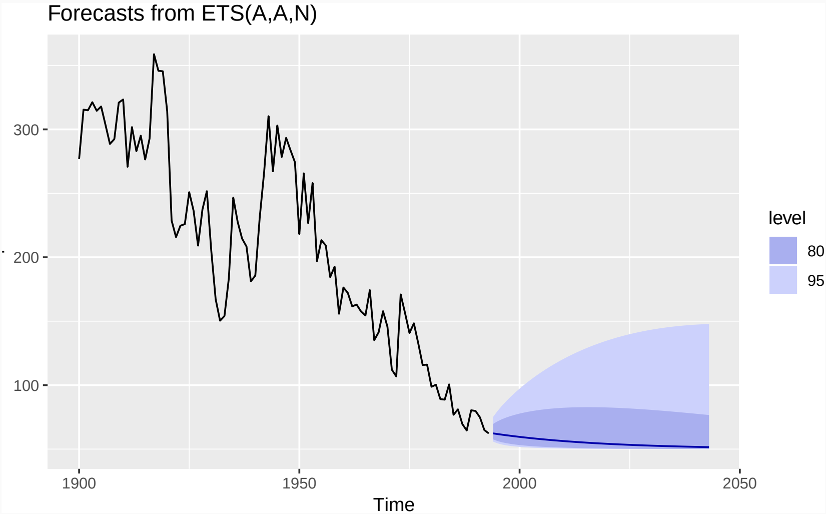
where x is on the original scale and y is the transformed data. To reverse the transformation, we will use

$$x = \frac{(b-a)e^y}{1+e^y} + a.$$

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)
fc <- forecast(fit, h=50)
# Back-transform forecasts
fc[["mean"]] <- (b-a)*exp(fc[["mean"]]) /
  (1+exp(fc[["mean"]])) + a
fc[["lower"]] <- (b-a)*exp(fc[["lower"]]) /
 (1+exp(fc[["lower"]])) + a
fc[["upper"]] <- (b-a)*exp(fc[["upper"]]) /
 (1+exp(fc[["upper"]])) + a
fc[["x"]] <- eggs
autoplot(fc)
```

Forecasts constrained to an interval



Outline

Forecast combinations

Clemen (1989)

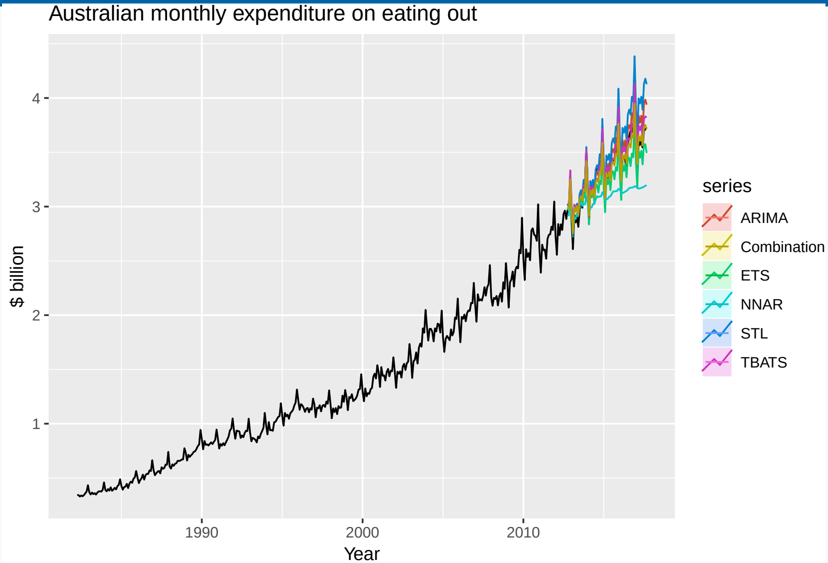
"The results have been virtually unanimous: combining multiple forecasts leads to increased forecast accuracy. . . . In many cases one can make dramatic performance improvements by simply averaging the forecasts."

Forecast combinations

```
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



Forecast combinations

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

```
##         ETS       ARIMA     STL-ETS        NNAR
##     0.13700     0.12146     0.21446     0.32886
##       TBATS Combination
##     0.09406     0.07199
```

Outline

Prediction intervals for aggregates

```
# 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] 281.7
```

Prediction intervals for aggregates

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

```
##   10%   90%
## 262.9 300.8
```

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

```
##  2.5% 97.5%
## 253.3 310.8
```

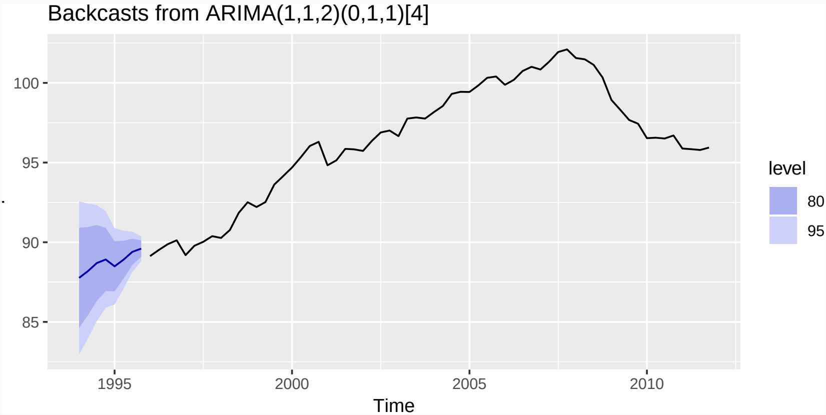
Outline

Backcasting

```r
# 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)
}
```

Backcasting

```
euretail %>% reverse_ts() %>%
  auto.arima() %>%
  forecast() %>% reverse_forecast() -> bc
autoplot(bc) +
  ggtitle(paste("Backcasts from",bc[["method"]]))
```



Outline

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

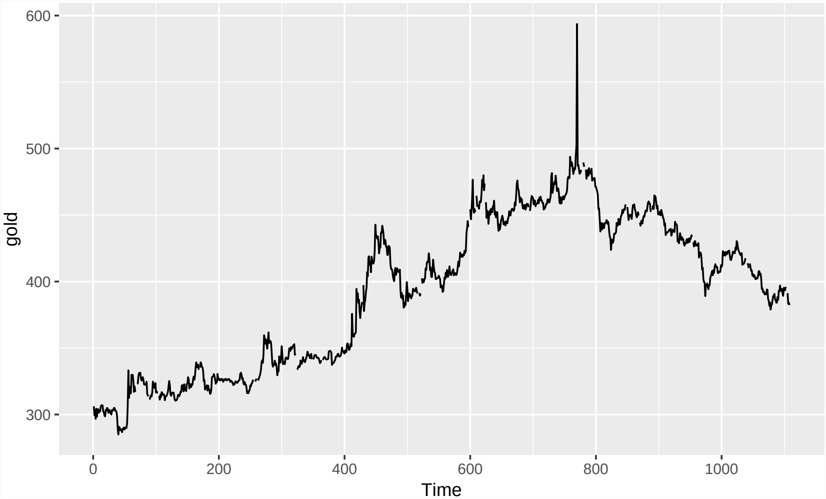
- `ets()`
- `stl()`
- `stlf()`
- `tbats()`

What to do?

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

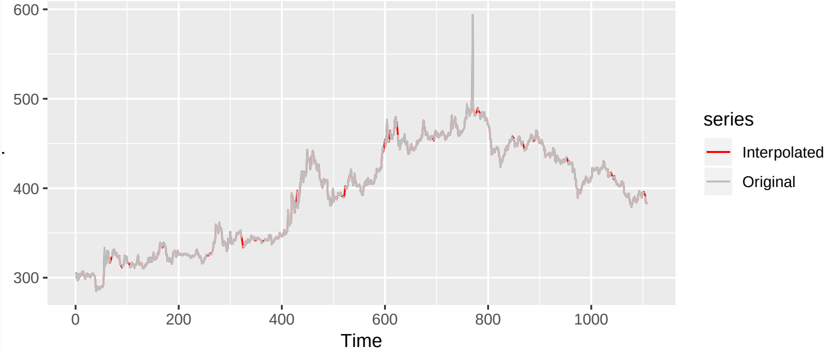
Missing values

```r
autoplot(gold)
```



Missing values

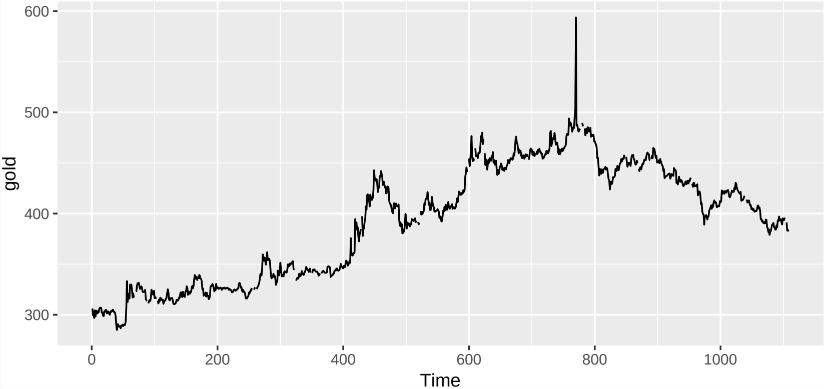
```
gold %>% na.interp() %>%
  autoplot(series="Interpolated") +
    autolayer(gold, series="Original") +
    scale_color_manual(
      values=c(Interpolated="red",Original="gray"))
```



Outline

Outliers

```r
autoplot(gold)
```



Outliers

```r
tsoutliers(gold)
```

```
## $index
## [1] 770
##
## $replacements
## [1] 494.9
```

Outliers

```r
gold %>% tsclean() %>% autoplot()
```