Lab6_THieF_a24kimwu

2025-10-06

Temporal Hierarchies in R

Data & packages

```r
pckg <- c("thief","MAPA","tsutils","abind")
for (i in 1:length(pckg)){
  if(!(pckg[i] %in% rownames(installed.packages()))){
    install.packages(pckg[i])
  }
  library(pckg[i],character.only = TRUE)
 }
```

```
## Loading required package: forecast
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## Loading required package: parallel
```

```
## Loading required package: RColorBrewer
```

```
## Loading required package: smooth
```

```
## Loading required package: greybox
```

```
## Package "greybox", v2.0.5 loaded.
```

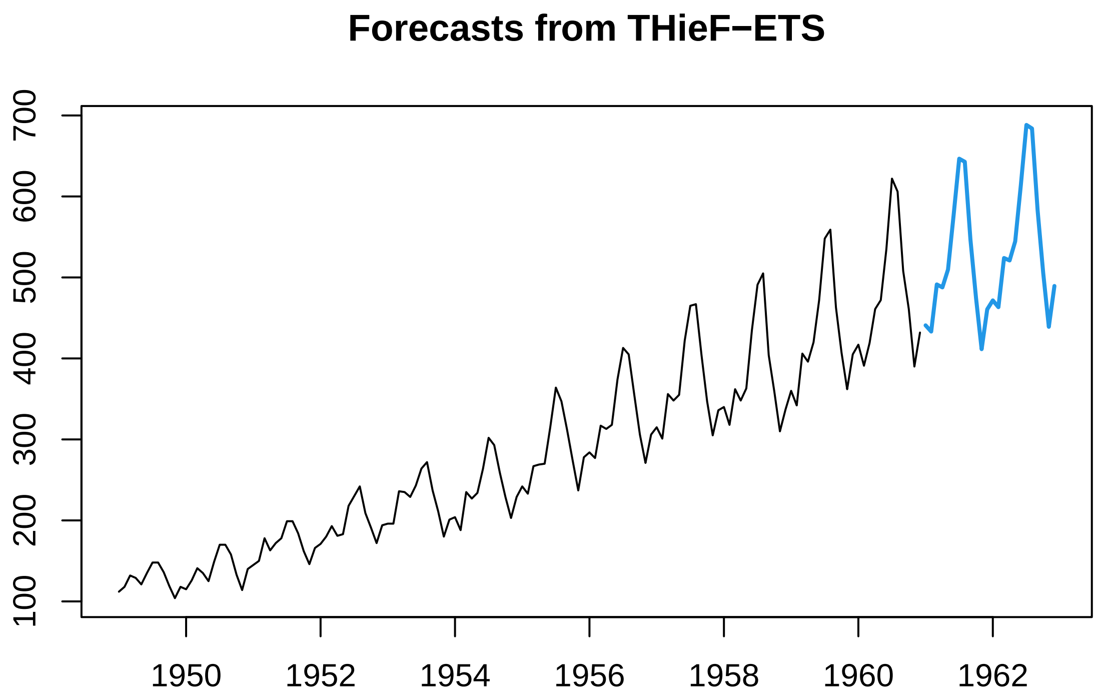
```
## This is package "smooth", v4.3.0
```

```r
y <- AirPassengers
```

Temporal hierarchies using the thief package

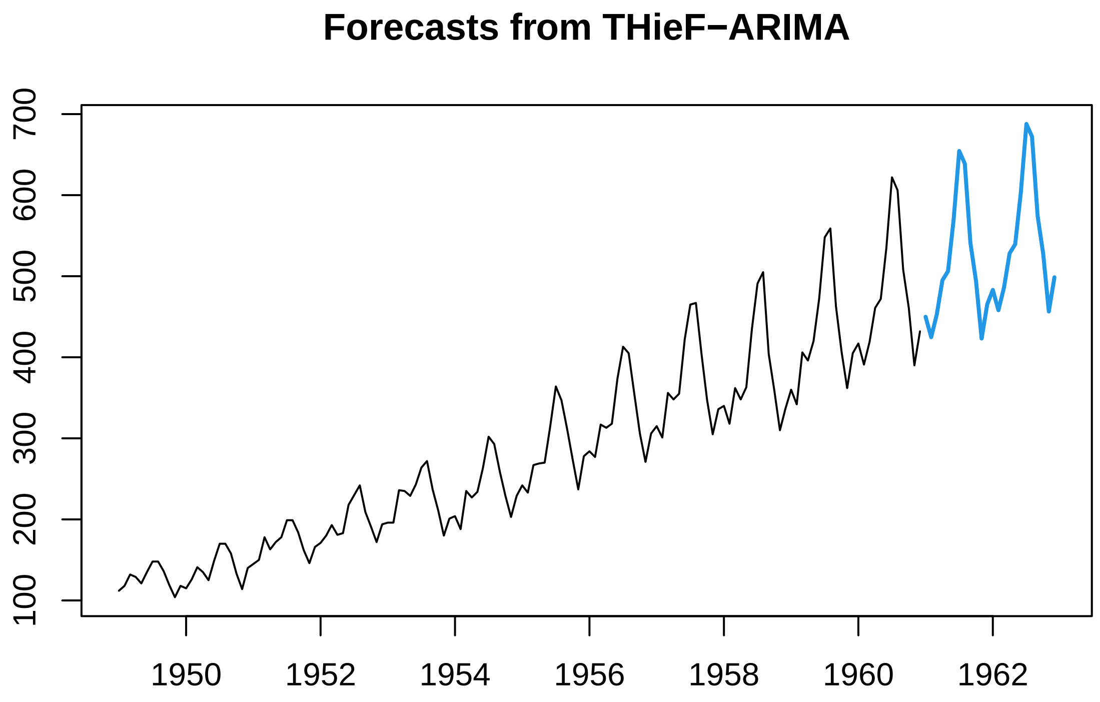
Using the function thief() the whole process is automated:

```r
frc1 <- thief(y)
plot(frc1)
```



By default uses ETS as the base forecast, we can change that using usemodel:

```r
frc2 <- thief(y,usemodel="arima")
plot(frc2)
```



Manual implementation of THieF

First, we construct the summing matrix S, used to map the hierarchy and pick up from that the useful aggregation levels.

```
S <- tsutils::Sthief(y) # Get the S matrix
ff <- frequency(y) # Get sampling frequency of target series
AL <- ff/(1:ff) # Calculate frequencies of various aggregation levels
AL <- AL[AL %% 1 == 0] # And exclude those that would not be integer
k <- length(AL) # Find how many are left
```

Next we create the temporally aggregated time series:

```
Y <- MAPA::tsaggr(y,AL)[[1]]
```

We set the target forecast horizon, and calculate the corresponding forecast horizons for all temporal aggregation levels:

```
hrz <- 16 # Target horizon
hAggr <- (ceiling(hrz/ff)*ff)/AL
hAggr
```

```
## [1]  2  4  6  8 12 24
```

Now we produce all the base forecasts, i.e., the independent forecasts for each level of the temporal hierarchy.

```
frc <- mse <- list()
for (i in 1:k){
 yTemp <- Y[[i]]
 fit <- ets(yTemp)
 mse[[i]] <- fit$mse
 frcTemp <- forecast(fit,h=hAggr[i])$mean
 # Re-structure forecasts
 frc[[i]] <- matrix(frcTemp,ncol=hAggr[1]) # Organised as column per year
 }
```

Next, we re/arrange the forecasts into a matrix. First comes the most aggregate, and the rest follow.

```
frcAll <- abind(frc,along=1)
frcAll
```

```
##             [,1]      [,2]
##  [1,] 6117.6361 6596.1292
##  [2,] 3007.6633 3250.5458
##  [3,] 3129.1045 3371.9871
##  [4,] 1833.0574 1941.5106
##  [5,] 2393.3701 2532.2736
##  [6,] 1898.4091 2006.5253
##  [7,] 1355.3595 1459.7158
##  [8,] 1593.3745 1713.7677
##  [9,] 1887.3399 2027.3332
## [10,] 1383.1478 1483.8981
## [11,]  867.1471  929.9355
## [12,]  963.5666 1032.5139
## [13,] 1094.7423 1172.1632
## [14,] 1332.7008 1425.8650
## [15,] 1037.7271 1109.4450
## [16,]  881.9057  942.1689
## [17,]  441.8018  459.0139
## [18,]  434.1186  450.6333
## [19,]  496.6300  515.0797
## [20,]  483.2375  500.7700
## [21,]  483.9914  501.1423
## [22,]  551.0244  570.0974
## [23,]  613.1797  633.9130
## [24,]  609.3648  629.4938
## [25,]  530.5408  547.6630
## [26,]  463.0332  477.6340
## [27,]  402.7478  415.1573
## [28,]  451.9694  465.5780
```

Now we estimate the G matrix that contains the combination weighs. We wil use here two easy approximations, the structural and the variance scaling.

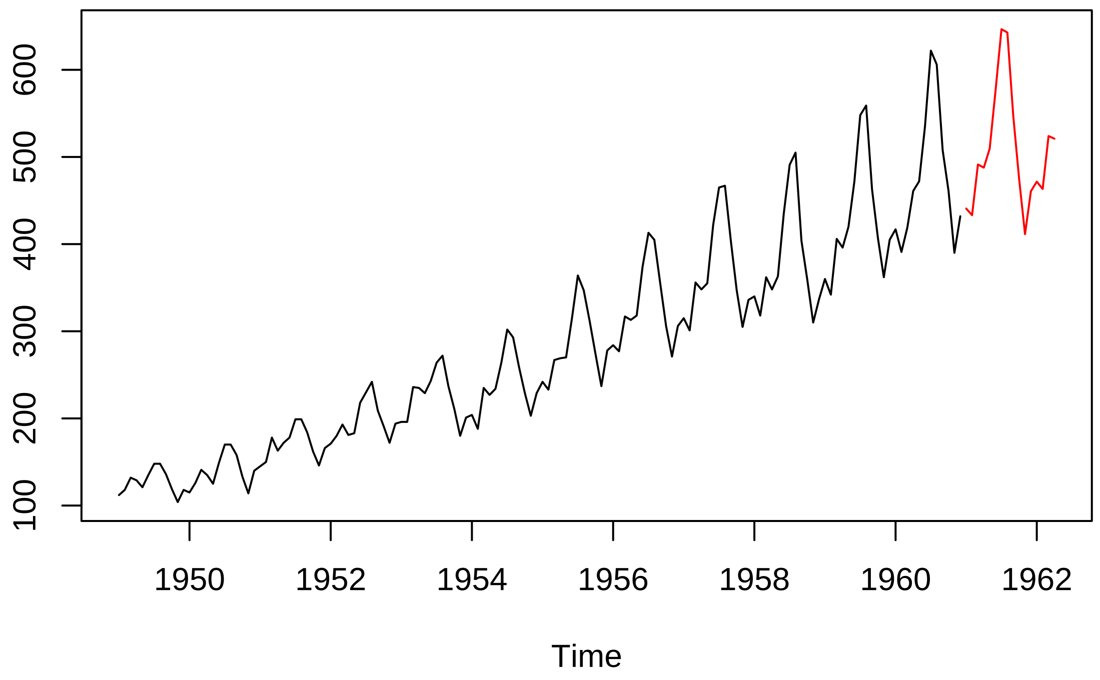
```
# Structural:
W <- diag(1/rowSums(S))
Gstr <- solve(t(S)%*%W%*%S)%*%t(S)%*%W
# Variance:
mse <- unlist(mse)
W <- diag(1/mse[rep((1:k),rev(AL))])
Gvar <- solve(t(S)%*%W%*%S)%*%t(S)%*%W
```

Now we reconclile (combine) the forecasts. For this example we use the Structural scaling, but we only need to replace G to obtain any other result.

```
# Create the bottom level forecasts
frcBRec <- Gstr %*% frcAll
frcFinal <- as.numeric(frcBRec)[1:hrz]
# We can also translate this into a time series object
frcFinal <- ts(frcFinal,frequency=frequency(y),start=end(y)[1] + deltat(y)*end(y)[2])
frcFinal
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 1961 440.8650 433.1818 491.3771 487.7213 509.7371 576.7701 646.6260 642.8111
## 1962 471.6625 463.2819 524.0035 520.9690
##           Sep      Oct      Nov      Dec
## 1961 547.0800 474.7431 411.4239 460.6455
## 1962
```

```
ts.plot(y,frcFinal,col=c("black","red"))
```



Or we can use the S-matrix to generate the forecasts for the complete hierarchy.

```
frcARec <- S %*% frcBRec # Which has the same structure as frcAll
```