Tourism Forecasting

2023-11-30

```r
source('load.R')
library(fpp3)
```

```
## -- Attaching packages ---------------------------------------------- fpp3 0.5 --

## v tibble      3.1.8     v tsibble     1.1.3
## v dplyr       1.1.0     v tsibbledata 0.4.1
## v tidyr       1.3.0     v feasts      0.3.1
## v lubridate   1.9.2     v fable       0.3.3
## v ggplot2     3.4.1     v fabletools  0.3.3

## -- Conflicts ------------------------------------------------- fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
```

```r
library(fable.prophet)
```

```
## Loading required package: Rcpp
```

Model Evaluation

Fit models on all series

```r
# load data
filepath = "tourism_monthly_dataset.tsf"
loaded_data <- convert_tsf_to_tsibble(filepath, "vistors", 'series_name', 'start_timestamp')

tsibble_data <- loaded_data[[1]]

head(tsibble_data)
```

```
## # A tsibble: 6 x 3 [1D]
## # Key:       series_name [1]
##   series_name start_timestamp  vistors
```

```
##   <chr>       <date>          <dbl>
## 1 T1         1979-01-01      1149.87
## 2 T1         1979-02-01      1053.800
## 3 T1         1979-03-01      1388.880
## 4 T1         1979-04-01      1783.370
## 5 T1         1979-05-01      1921.025
## 6 T1         1979-06-01      2704.945
```

```
# convert to tsibble
series.names = tsibble_data |> as_tibble() |> select(series_name) |> distinct()
series.names = series.names$series_name
tsibble_data = tsibble_data |>
  mutate(month = yearmonth(start_timestamp)) |>
  as_tsibble(index = month)

# forecast
forecast.single = function (series) {
  tourism.ts = tsibble_data |> filter(series_name == series)
  nb.all = nrow(tourism.ts)
  nb.test = 24
  nb.train = nb.all - nb.test

  tourism.train = tourism.ts |> filter(row_number() <= nb.train)
  tourism.test = tourism.ts |> filter(row_number() > nb.train)
  #tourism.train |> autoplot(vistors)

  #boxcox.lambda = tourism.train |> features(vistors, features = guerrero)
  #boxcox.lambda = boxcox.lambda$lambda_guerrero
  #tourism.train |> autoplot(box_cox(vistors, boxcox.lambda))
  #tourism.train |> model(STL(vistors)) |> components() |> autoplot()

  fit.all = tourism.train |> model(
    arima = ARIMA(vistors),
    ets = ETS(vistors),
    nnar = NNETAR(sqrt(vistors)),
    prophet = prophet(vistors ~ season(period = 12, order =2, type = 'multiplicative')),
    mean = MEAN(vistors),
    naive = NAIVE(vistors),
    snaive = SNAIVE(vistors),
    drift = RW(vistors ~ drift())
  )

  fcst.accu.all = NULL

  for (h in c(1,2,3,6,12,18,24)) {
    fcst = fit.all |> forecast(h = h, times = 100)
    fcst.accu = fcst |> accuracy(tourism.test) |>
      select(.model, series_name, RMSE, MAE, MAPE)
    fcst.accu$h = h
    if (is.null(fcst.accu.all)) {
      fcst.accu.all = fcst.accu
    } else {
      fcst.accu.all = rbind(fcst.accu.all, fcst.accu)
    }
```

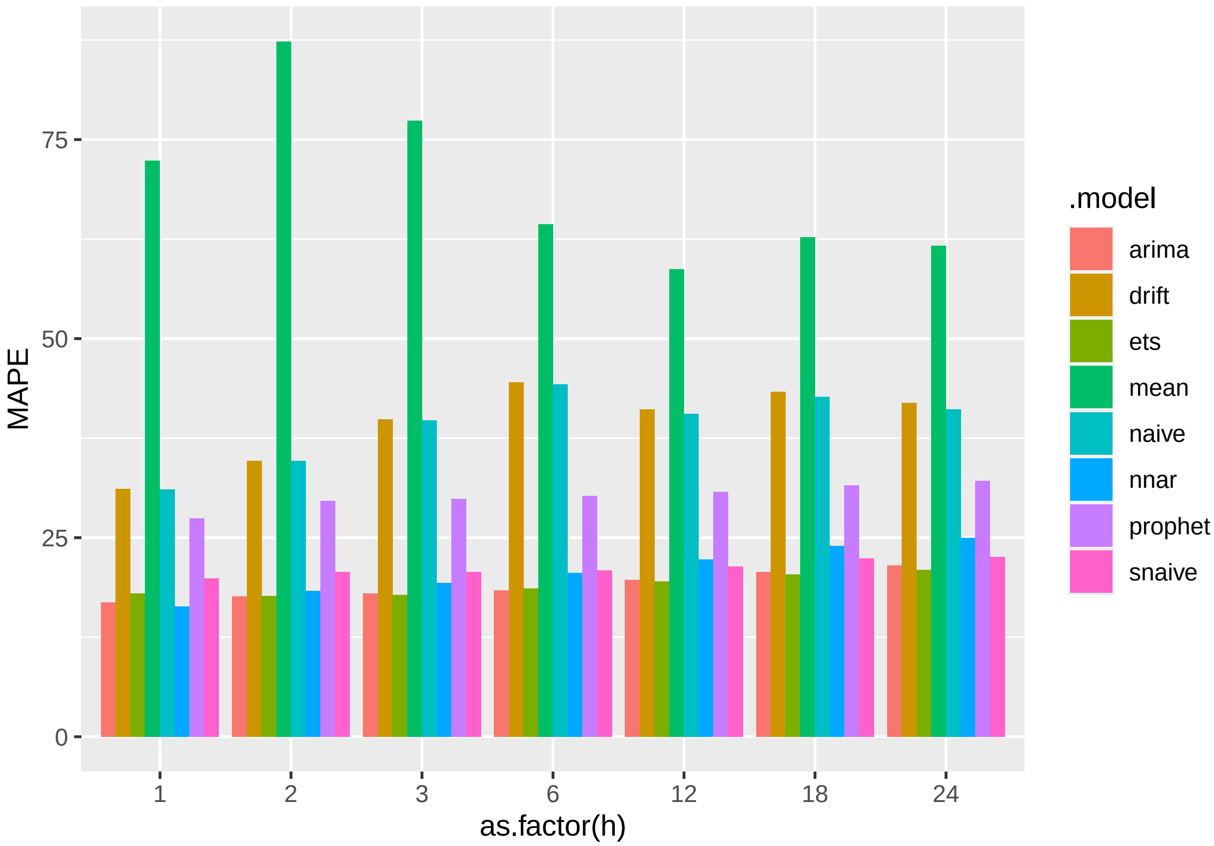
```
  }
  fcst.accu.all
}

## forecast all
accu.all.series = NULL
for (series_name in series.names) {
  if (is.null(accu.all.series)) {
    accu.all.series = forecast.single(series_name)
  } else {
    accu.all.series = rbind(accu.all.series, forecast.single(series_name))
  }
}
```

Forecast accuracy and model rank

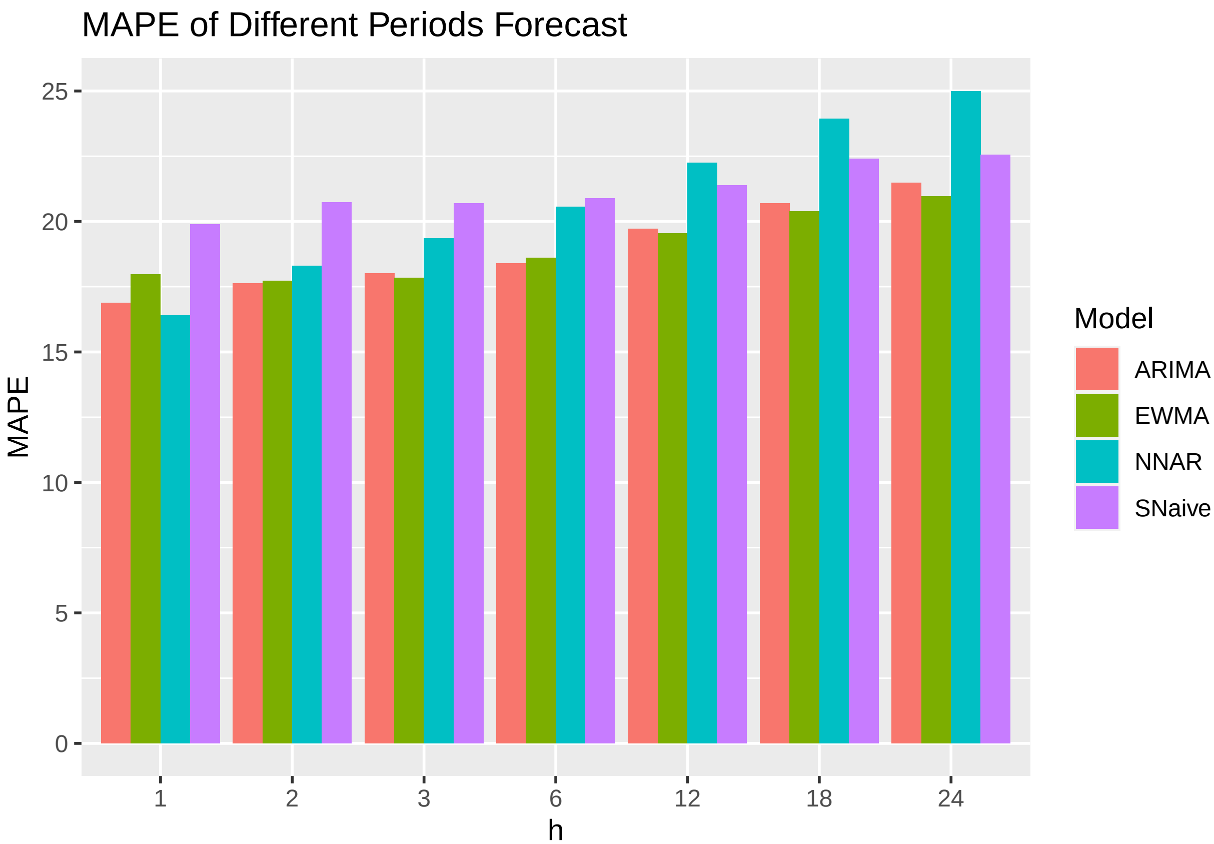
```
#save(accu.all.series, file = 'accu2.all.series.RObject')

accu.summarise = accu.all.series |> group_by(.model, h) |>
  summarise(RMSE = mean(RMSE), MAE = mean(MAE), MAPE=mean(MAPE))
```

```
## `summarise()` has grouped output by '.model'. You can override using the
## `.groups` argument.
```

```
# compare all models
ggplot(accu.summarise, mapping = aes(x = as.factor(h), y = MAPE)) +
  geom_bar(aes(fill = .model), stat = 'identity', position = 'dodge')
```




```r
accu.with.best.model = accu.summarise |> filter(
  .model != 'mean', .model != 'drift', .model != 'naive', .model != 'prophet'
)

accu.with.best.model$.model[accu.with.best.model$.model=='arima'] = 'ARIMA'
accu.with.best.model$.model[accu.with.best.model$.model=='ets'] = 'EWMA'
accu.with.best.model$.model[accu.with.best.model$.model=='snaive'] = 'SNaive'
accu.with.best.model$.model[accu.with.best.model$.model=='nnar'] = 'NNAR'

# compare ARIMA EWMA SNaive and NNAR
ggplot(accu.with.best.model, mapping = aes(x = as.factor(h), y = MAPE)) +
  geom_bar(aes(fill = .model), stat = 'identity', position = 'dodge') +
  labs(x = 'h', fill = 'Model') +
  ggtitle('MAPE of Different Periods Forecast')
```




```
# mapes table
mapes = pivot_wider(data = accu.summarise, names_from = h, values_from = 'MAPE', id_cols = .model)

# average model rank
rank.mat = apply(mapes |> ungroup() |> select(`1`, `2`, `3`, `6`, `12`, `18`, `24`), 2, rank)
model.rank = rowSums(rank.mat)/7
tibble(mapes |> select(.model), rank = model.rank)
```

```
## # A tibble: 8 x 2
##   .model      rank
##   <chr>      <dbl>
## 1 arima   1.714286
## 2 drift   7
## 3 ets     1.571429
## 4 mean    8
## 5 naive   6
## 6 nnar    3.142857
## 7 prophet 5
## 8 snaive  3.571429
```

T1 ETS and ARIMA model

```
# check EWMA parameter on T1 series
T1 = tsibble_data |> filter(series_name == 'T1')
```

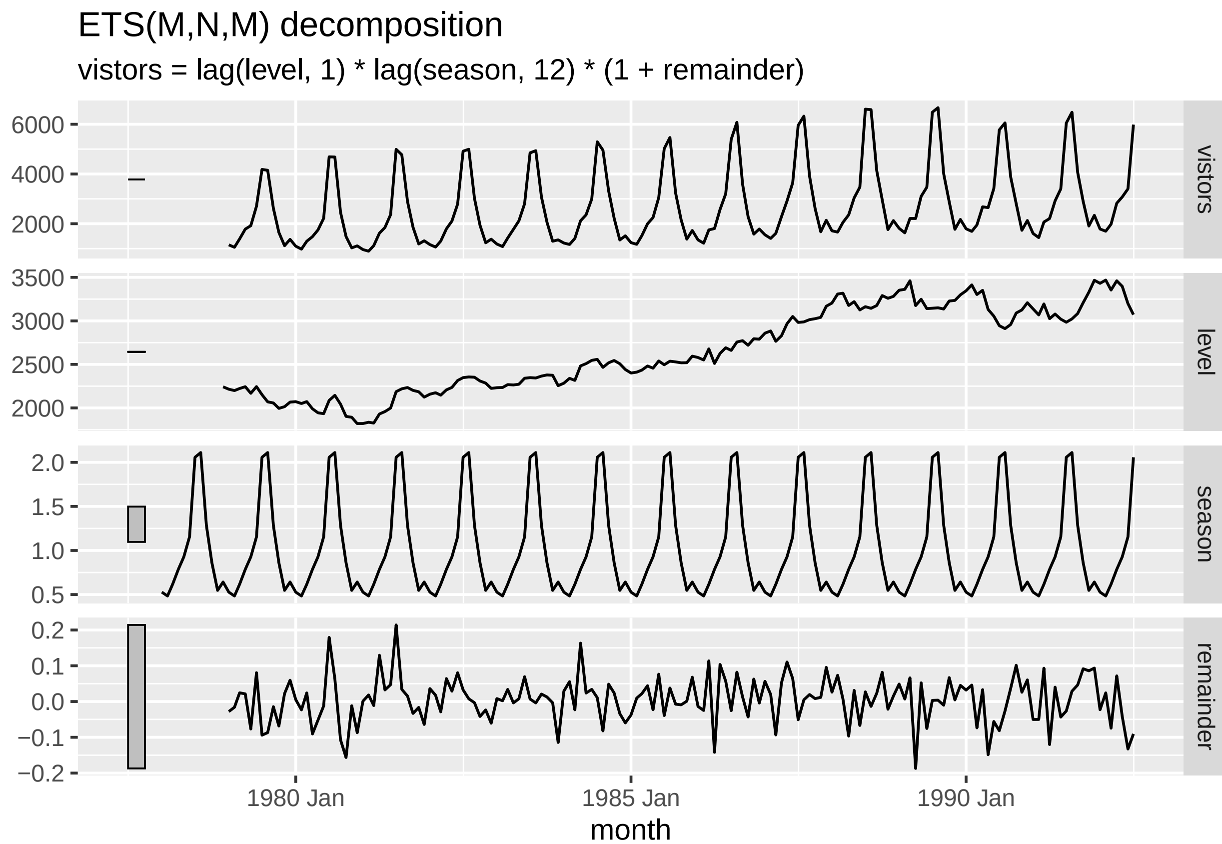
```r
nb.all = nrow(T1)
nb.test = 24
nb.train = nb.all - nb.test

T1.train = T1 |> filter(row_number() <= nb.train)
T1.test = T1 |> filter(row_number() > nb.train)

fit.best = T1.train |> model(
  ets = ETS(vistors),
  arima = ARIMA(vistors)
)
fcst.best = fit.best |> forecast(new_data = T1.test)

#ets model
report(fit.best |> select(arima))
```

```
## Series: vistors
## Model: ARIMA(1,0,3)(1,1,1)[12] w/ drift
##
## Coefficients:
##          ar1      ma1     ma2      ma3    sar1     sma1  constant
##       0.9276  -0.6868  0.1548  -0.1973  0.1562  -0.5721    5.2486
## s.e.  0.0588   0.1013  0.1102   0.0964  0.1783   0.1448    1.9518
##
## sigma^2 estimated as 39824:  log likelihood=-1011.88
## AIC=2039.76   AICc=2040.78   BIC=2063.9
```

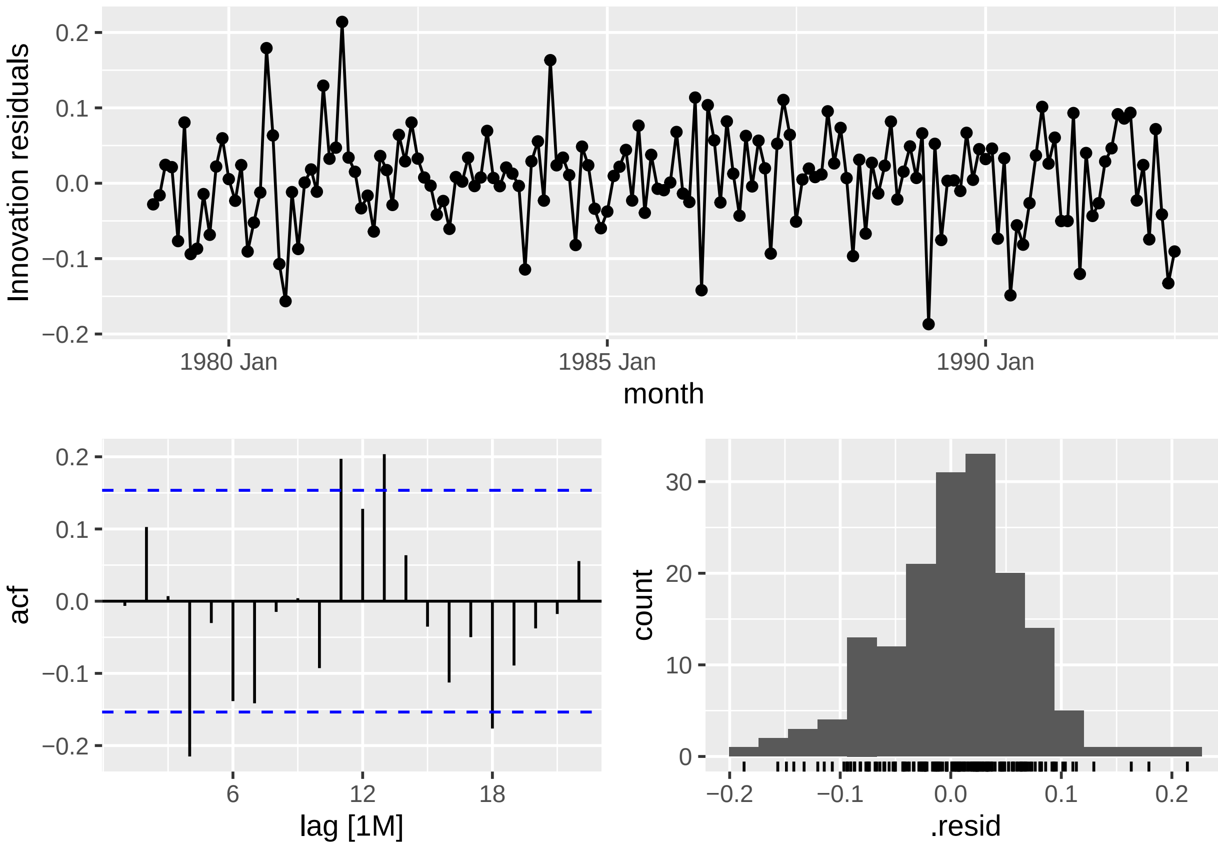
```r
report(fit.best |> select(ets))
```

```
## Series: vistors
## Model: ETS(M,N,M)
##   Smoothing parameters:
##     alpha = 0.4402505
##     gamma = 0.0001362852
##
##   Initial states:
##      l[0]      s[0]     s[-1]     s[-2]    s[-3]    s[-4]    s[-5]    s[-6]
##  2242.375 0.6422779 0.5473855 0.8614515 1.284841 2.111213 2.057443 1.154695
##      s[-7]     s[-8]    s[-9]    s[-10]    s[-11]
##  0.9274411 0.7855666 0.616549 0.4835589 0.5275775
##
##   sigma^2:  0.0044
##
##      AIC     AICc      BIC
## 2484.814 2488.080 2531.221
```

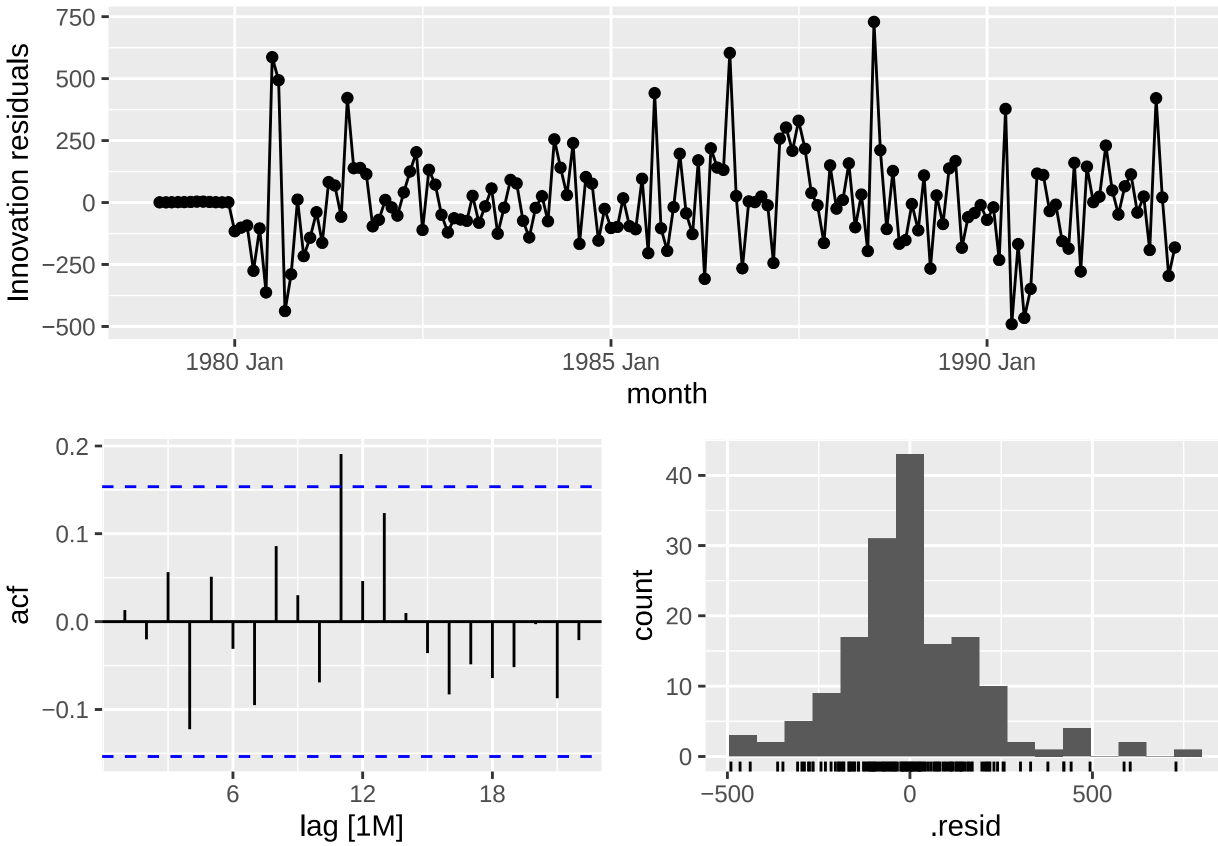
```r
# component and residual
fit.best |> select(ets) |> components() |> autoplot()
```



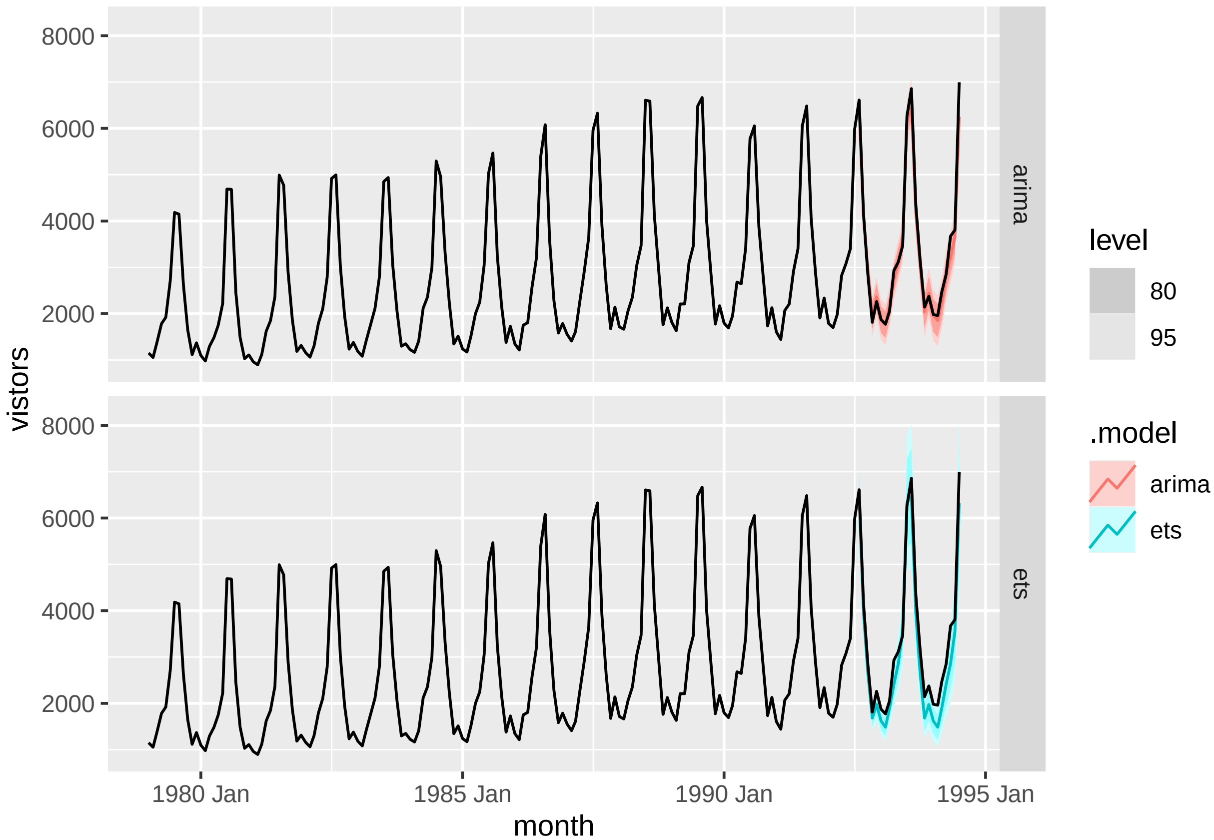

```
fit.best |> select(ets) |> gg_tsresiduals()
```




```
fit.best |> select(arima) |> gg_tsresiduals()
```




```r
#forecast plot
fcst.best |> autoplot(T1) + facet_grid(.model ~ .)
```



8000
6000
4000
2000
arima
ets
vistors
level
80
95
.model
arima
ets
1980 Jan
1985 Jan
1990 Jan
1995 Jan
month