Exam January 12, 2024

Anthony Tricarico

Table of contents

> **Warning**
>
> The following document should not be viewed as a solution to the exercises, but rather as a resource that can help you think about how you can solve similar problems. For issues, always refer back to the original solution provided by your Professor in class.

1 Exercise 1

Holt's Linear Trend Method is one of the first models developed by Holt which expands on the classical exponential smoothing methods to include the possibility to smooth out time series that presented a trend component. As the name suggests, this method can be used with time series with a trend component that do not present a seasonal component.

> **Tip**
>
> Check out section 8.2 of the book to learn more about the actual equations involved in this method.

2 Exercise 2

A prediction interval is an interval that with a certain level of confidence will contain the true value of the forecast variable at times $T+1, T+2, \ldots, T+h$ where h is the time horizon of the forecasts (i.e., how many periods into the future you want to produce the forecasts for). Usually predictions intervals get larger and larger the farther away they get from the last observation T in the dataset.

With large enough datasets we can assume the distribution of the residuals to be normal and we can compute the prediction intervals by using reference values from the standard normal distribution (Z scores) which will vary depending on the level of confidence (1-α) that we want to assign to the interval.

However, when the assumption of normality for the residuals cannot be assumed to be respected, it would be better to use bootstrap-based prediction intervals as they will be more accurate in the estimates they produce.

3 Exercise 3

We start by importing the necessary packages.

```
library(fpp3)
```

```
Registered S3 method overwritten by 'tsibble':
  method               from
  as_tibble.grouped_df dplyr

-- Attaching packages --------------------------------------- fpp3 1.0.1 --

v tibble      3.2.1     v tsibble     1.1.5
v dplyr       1.1.4     v tsibbledata 0.4.1
v tidyr       1.3.1     v feasts      0.4.1
v lubridate   1.9.3     v fable       0.4.1
v ggplot2     3.5.1

-- 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()
```

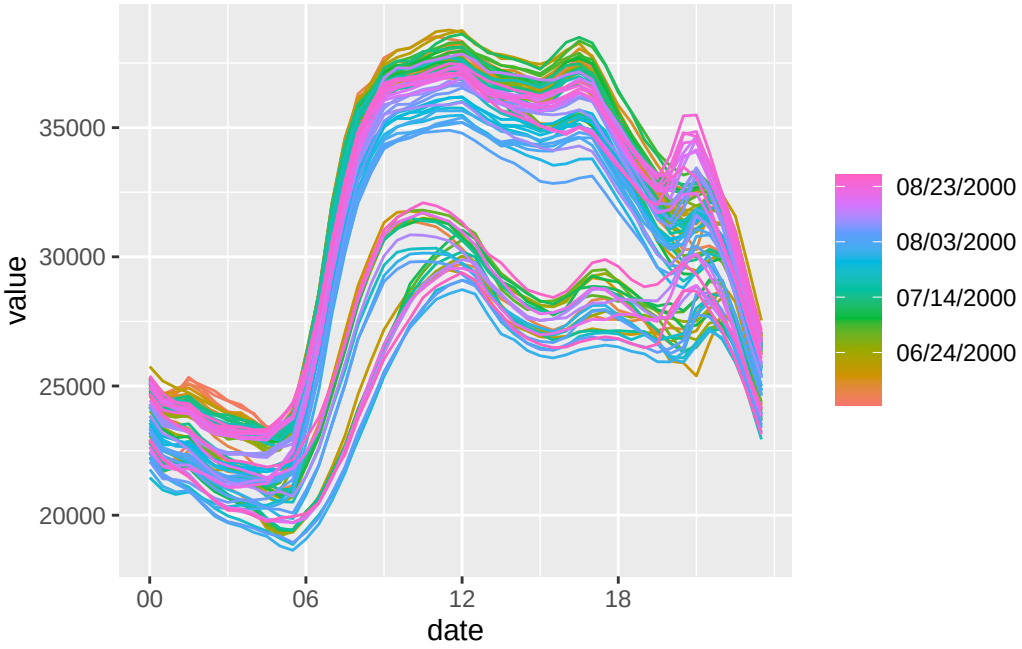
```
library(timetk)
```

```
df <- as_tsibble(taylor_30_min, index = date)
```

3.1 Checking daily seasonality

```
gg_season(df, period = "1d")
```

```
Plot variable not specified, automatically selected `y = value`
```

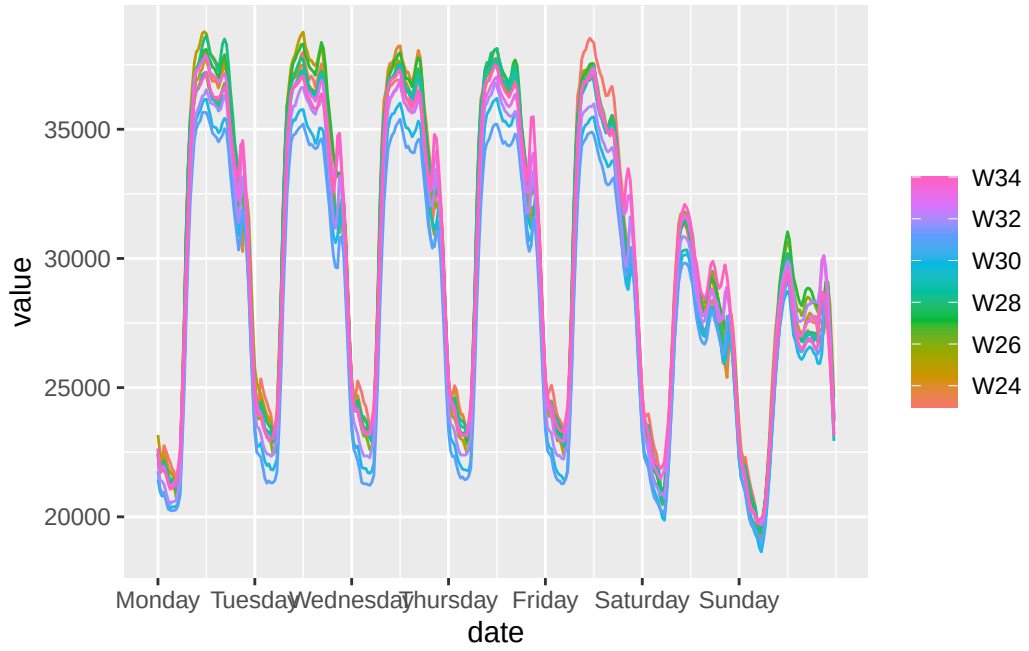


From this plot we see that there is a strong seasonality as every day from 6 to 12 there is a surge in electricity demand which lasts up until 18 and then decreases. Since this pattern is repeated every day, this is a strong evidence of the presence of daily seasonality.

3.2 Checking weekly seasonality

```
gg_season(df, period = "1w")
```

```
Plot variable not specified, automatically selected `y = value`
```



The daily pattern that was visible before, is also reflected in this plot. However, when we look at the weekly seasonality we can only say that there is less electricity demand on average on weekend days compared to the other days of the week. Still it is a patter indicative of weekly seasonality.

4 Exercise 4

As per usual, we start by importing the necessary packages. If you have troubles loading the packages, you can use `install.packages("name_of_dependency")` substituting the name inside the quotation marks with the name of the package whose absence is preventing the correct loading of the packages needed.

```
library(Ecfun)
```

```
Attaching package: 'Ecfun'
```

```
The following object is masked from 'package:base':

    sign
```

```
library(Ecdat)
```

```
Attaching package: 'Ecdat'
```

```
The following object is masked from 'package:datasets':

    Orange
```

```r
df <- as_tsibble(Hstarts)
head(df)
```

```
# A tsibble: 6 x 3 [1Q]
# Key:       key [1]
    index key   value
    <qtr> <chr> <dbl>
1 1960 Q1 hs     7.99
2 1960 Q2 hs     8.84
3 1960 Q3 hs     8.95
4 1960 Q4 hs     8.99
5 1961 Q1 hs     8.39
6 1961 Q2 hs     9.09
```

We are asked to use the non-seasonally adjusted data so we filter the `df` to include only those rows having `key == "hs"`

```r
df_hs <- df %>%
  filter(key == "hs")
```

4.1 Point a)

We can create a train test split based on the index (date) of a `tsibble` in the following way:

```r
train <- df_hs %>%
  filter_index("1960 Q1" ~ "1994 Q4")

test <- df_hs %>%
  filter_index("1995 Q1" ~ "2001 Q4")
```

4.2 Point b)

```r
stl_train <- train %>%
  model(
    "STL" = STL(value ~ trend() + season())
  )
```

Notice that in the code above it is possible to change the `window` parameter inside the `trend` and `season` components but I left the default values to see how they perform.

4.3 Point c)

```
mod_season_adj <- components(stl_train) %>%
  select(index, season_adjust) %>%
  model(
    "nai_adj" = NAIVE(season_adjust),
    "drift_adj" = RW(season_adjust ~ drift())
  )

mod_season_comp <- components(stl_train) %>%
  select(index, season_year) %>%
  model(
    "snai_comp" = SNAIVE(season_year)
  )
```

```
pred_adj <- forecast(mod_season_adj, h = nrow(test))
pred_comp <- forecast(mod_season_comp, h = nrow(test))
```

4.4 Point d)

In the last point we put everything together. First we put together the estimated seasonally adjusted time series with the estimated trend and then we get the actual estimated prediction which is the result of the sum between these two estimated components.

```
actual_pred_adj <- pred_adj %>%
  select(index, season_adjust, .mean) %>%
  mutate(.mean = .mean + pred_comp$.mean)
```

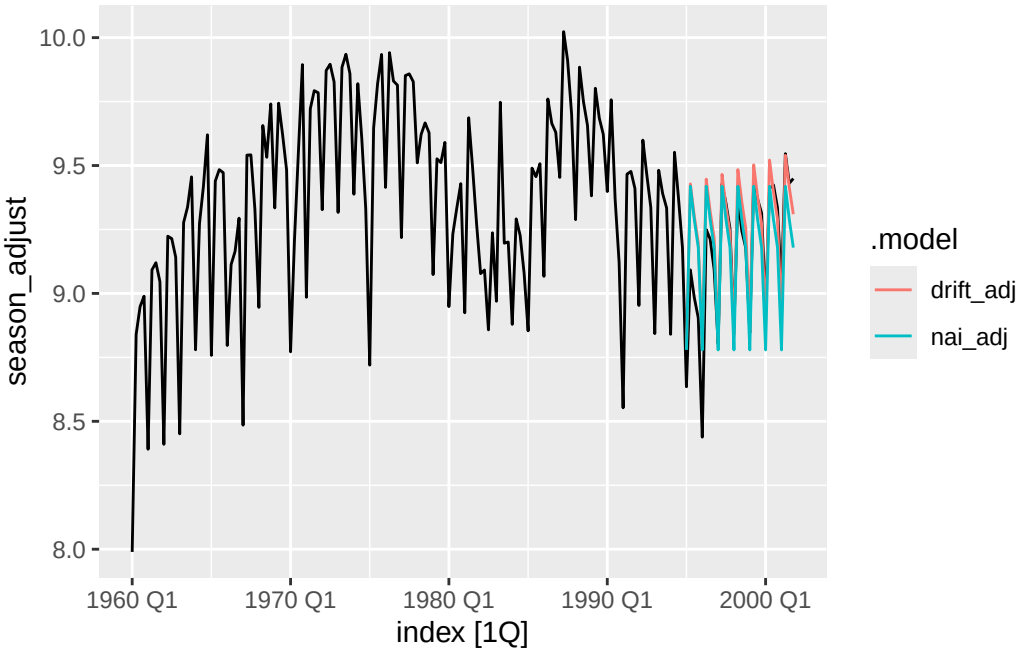
```
names(df_hs) <- c("index", "key", "season_adjust")
accuracy(actual_pred_adj, df_hs)
```

```
# A tibble: 2 x 10
  .model    .type        ME  RMSE   MAE     MPE  MAPE  MASE RMSSE   ACF1
  <chr>     <chr>     <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>  <dbl>
1 drift_adj Test  -0.0672   0.261 0.201 -0.818   2.24 0.992 0.968 0.0149
2 nai_adj   Test   0.000397 0.272 0.228 -0.0863  2.51 1.13  1.01  0.130
```

Just out of curiosity we can plot out the results from the two models against the actual data.

```
autoplot(df_hs) +
  geom_line(data = actual_pred_adj, aes(x = index, y = .mean, color = .model))
```

```
Plot variable not specified, automatically selected `.vars = season_adjust`
```



We see that the best model is the drift model as it has the lowest RMSE.