ST5209/X Assignment 2

Teng Siyuan

Submission

1. Render the document to get a .pdf printout.
2. Submit both the .qmd and .pdf files to Canvas.

```
Warning: package 'ggplot2' was built under R version 4.5.2
```

```
Warning: package 'readr' was built under R version 4.5.2
```

Question 1 (Forecast evaluation, Q5.12 in FPP)

`tourism` contains quarterly visitor nights (in thousands) from 1998 to 2017 for 76 regions of Australia.

a. Extract data from the Gold Coast region using `filter()` and aggregate total overnight trips using `summarise()`. Call this new dataset `gc_tourism`.

```
gc_tourism<-tourism|>
  filter(Region=="Gold Coast")|>
  summarise(Trips=sum(Trips))
gc_tourism
```

```
# A tsibble: 80 x 2 [1Q]
   Quarter Trips
     <qtr> <dbl>
 1 1998 Q1  827.
 2 1998 Q2  681.
 3 1998 Q3  839.
 4 1998 Q4  820.
 5 1999 Q1  987.
 6 1999 Q2  751.
```

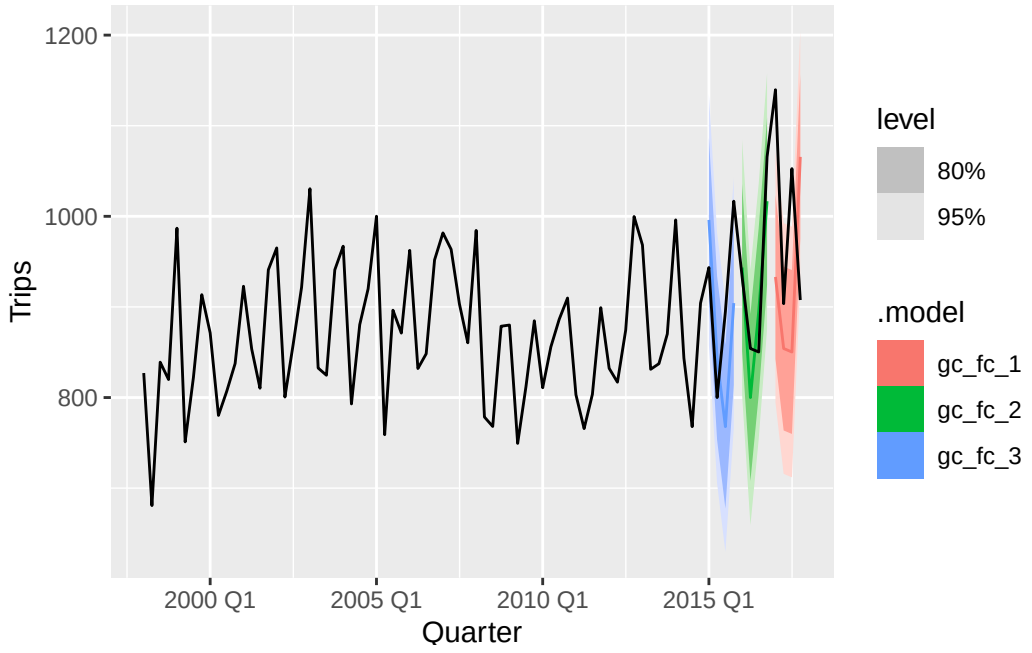
```
 7 1999 Q3  822.
 8 1999 Q4  914.
 9 2000 Q1  871.
10 2000 Q2  780.
# i 70 more rows
```

b. Using `slice()` or `filter()`, create three training sets for this data excluding the last 1, 2 and 3 years. For example, `gc_train_1 <- gc_tourism |> slice(1:(n()-4))`.

```
gc_train_1 <- gc_tourism |> slice(1:(n()-4))
gc_train_2 <- gc_tourism |> slice(1:(n()-8))
gc_train_3 <- gc_tourism |> slice(1:(n()-12))
```

c. Compute and plot one year of forecasts for each training set using the seasonal naive (`SNAIVE()`) method. Call these `gc_fc_1`, `gc_fc_2` and `gc_fc_3`, respectively. (Hint: You may combine the mable objects from each model into a single mable using `bind_cols`)

```
gc_fc<-bind_cols(
  gc_train_1|> model(gc_fc_1=SNAIVE(Trips)),
  gc_train_2|> model(gc_fc_2=SNAIVE(Trips)),
  gc_train_3|> model(gc_fc_3=SNAIVE(Trips))
)  |>
  forecast(h=4)
autoplot(gc_fc,gc_tourism)
```



d. Use `accuracy()` to compare the test set forecast accuracy using MASE. What can you conclude about the relative performance of the different models? Explain your answer.

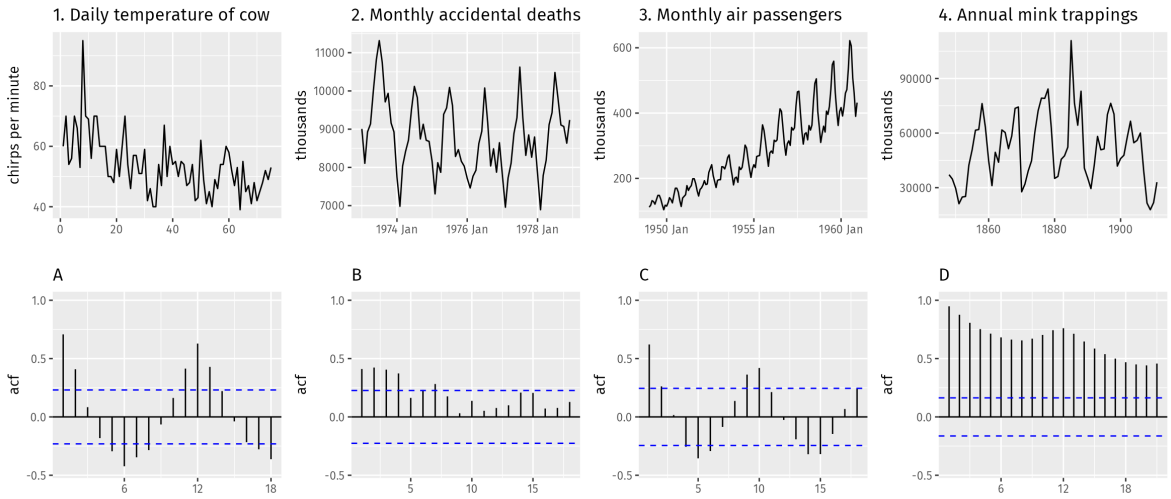
```
gc_fc|>accuracy(gc_tourism)
```

```
# A tibble: 3 x 10
  .model   .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE   ACF1
  <chr>    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
1 gc_fc_1 Test   75.1 167.  154.   6.36 15.1  2.66  2.36  -0.410
2 gc_fc_2 Test   12.0  43.1  39.5  1.14  4.32 0.670 0.599 -0.792
3 gc_fc_3 Test   35.8  91.4  83.9  3.56  9.07 1.46  1.30   0.239
```

We can find that the gc_fc_2 shows the best forecast accuracy. The difference is partly sample-dependent and the second training set happens to align best with the seasonal structure in the test period.

Question 2 (ACF plots, Q2.9 in FPP)

The following time plots and ACF plots correspond to four different time series. Match each time plot in the first row with one of the ACF plots in the second row.



1-B; 2-A; 3-D; 4-C

Question 3 (Time series classification)

We will investigate this dataset, which contains 600 synthetic control charts for an industrial process. There are 6 types of control charts, and our goal is to build a model to classify them

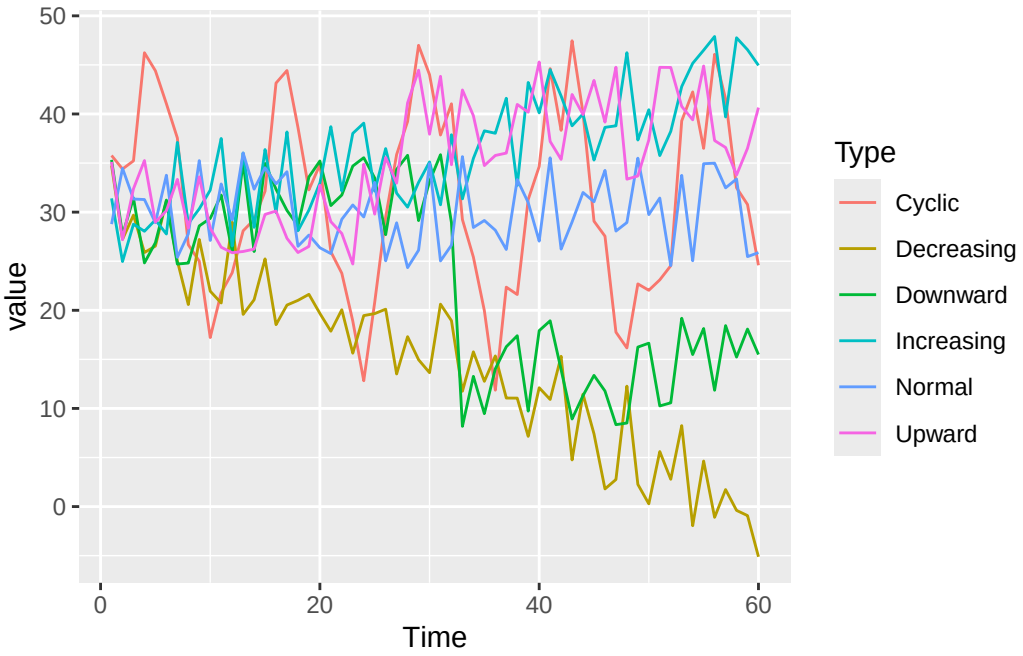
correctly.

We have already processed the raw data into a convenient, labeled form. Run the following code snippet to load it and create train and test sets.

```
ccharts <- read_rds("../_data/CLEANED/ccharts.rds")
ccharts_train <- ccharts[["train"]]
ccharts_test <- ccharts[["test"]]
```

a. Make a time plot of one time series from each category in the training set. Note that the category is recorded under the `Type` column.

```
ccharts_train |>
  group_by(Type) |>
  filter(id == first(id)) |>
  ungroup() |>
  ggplot(aes(x = Time, y = value, colour = Type, group = Type)) +
  geom_line()
```



b. Compute all time series features for both the training and test set using the snippet. What is the difference between `acf1` and `stl_e_acf1` for Increasing, Decreasing, Upward, and Downward types? Why is there a difference?

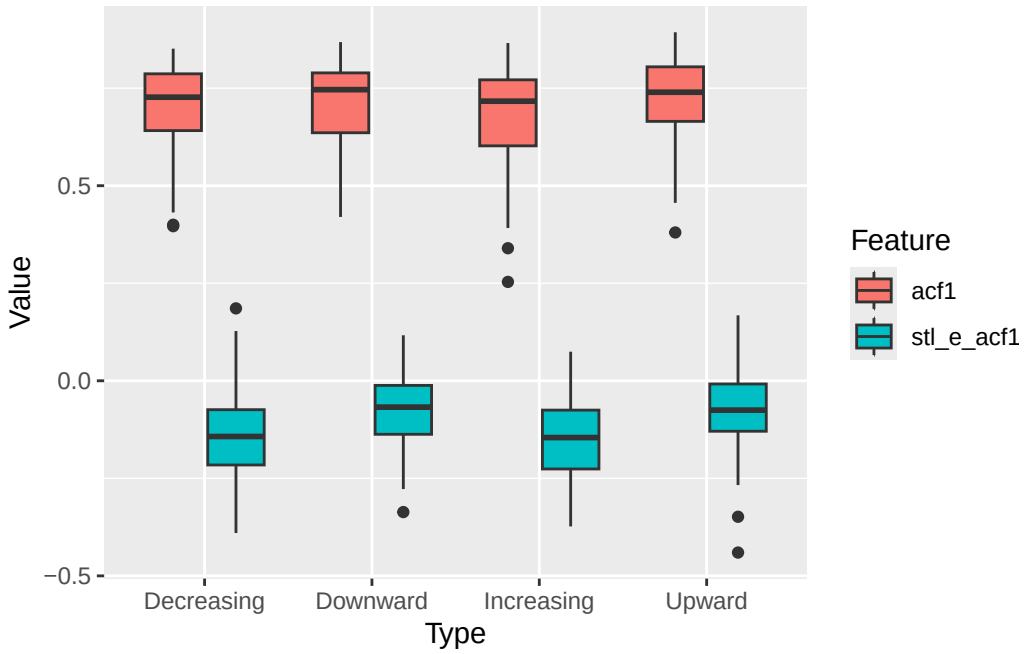
```
# Change to eval: TRUE in order to run
train_feats <- ccharts_train |> features(value, feature_set(pkgs = "feasts"))
```

```
Warning: 486 errors (1 unique) encountered for feature 6
[486] The `urca` package must be installed to use this functionality. It can be installed wi

Warning: 486 errors (1 unique) encountered for feature 7
[486] The `urca` package must be installed to use this functionality. It can be installed wi

Warning: 486 errors (1 unique) encountered for feature 8
[486] The `urca` package must be installed to use this functionality. It can be installed wi

Warning: 486 errors (1 unique) encountered for feature 20
[486] The `fracdiff` package must be installed to use this functionality. It can be installe
```

```
test_feats <- ccharts_test |> features(value, feature_set(pkgs = "feasts"))
```

```
Warning: 114 errors (1 unique) encountered for feature 6
[114] The `urca` package must be installed to use this functionality. It can be installed wi

Warning: 114 errors (1 unique) encountered for feature 7
[114] The `urca` package must be installed to use this functionality. It can be installed wi

Warning: 114 errors (1 unique) encountered for feature 8
[114] The `urca` package must be installed to use this functionality. It can be installed wi

Warning: 114 errors (1 unique) encountered for feature 20
[114] The `fracdiff` package must be installed to use this functionality. It can be installe
```

```
select_types<-c("Increasing","Decreasing","Upward","Downward")

train_feats_filtered<-train_feats|>
  filter(Type %in% select_types)

train_feats_long <- train_feats_filtered |>
  select(Type, acf1, stl_e_acf1) |>
  pivot_longer(cols = c(acf1, stl_e_acf1), names_to = "Feature", values_to = "Value")

train_feats_long|>
  ggplot(aes(x=Type,y=Value,fill=Feature))+
  geom_boxplot()
```



For these four types, the values of acf1 are consistently higher. This is because the acf1 is computed on the original series and is inflated by the presence of trend, while the stl_e_acf1 is computed on the STL remainder after removing the trend.

c. Investigate the relationship between the following features and `Type`. Pick two features whose scatter plot gives a good separation between all 6 chart types.

   i. `linearity`

   ii. `trend_strength`

   iii. `acf1`

   iv. `stl_e_acf1`

   v. `shift_level_max`

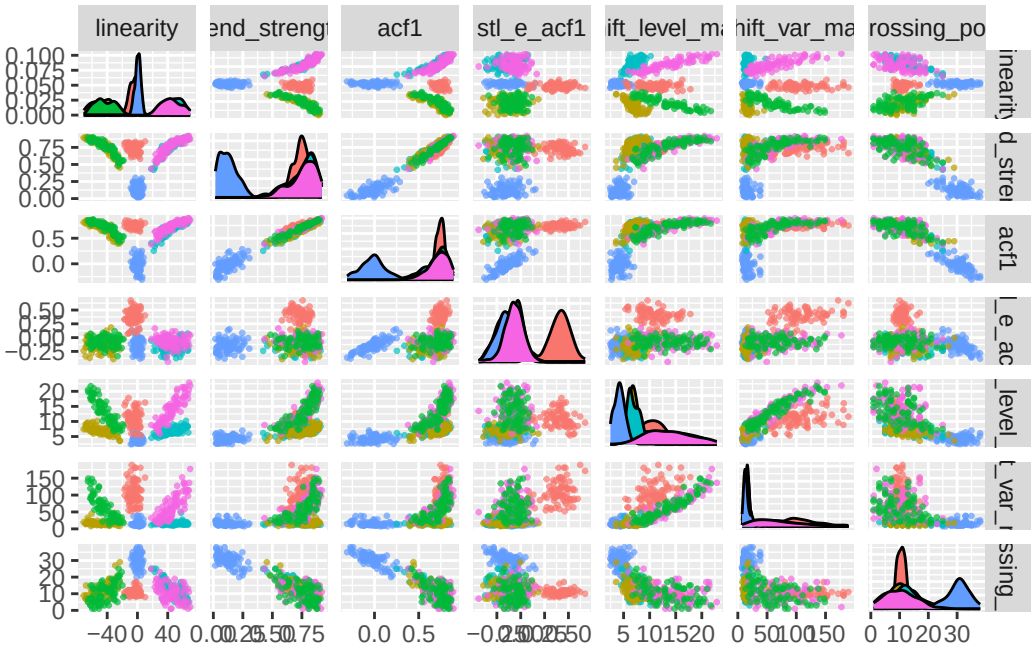
   vi. `shift_var_max`

   vii. `n_crossing_points`

   Make the scatter plot and explain why these features are able to separate the different chart types.

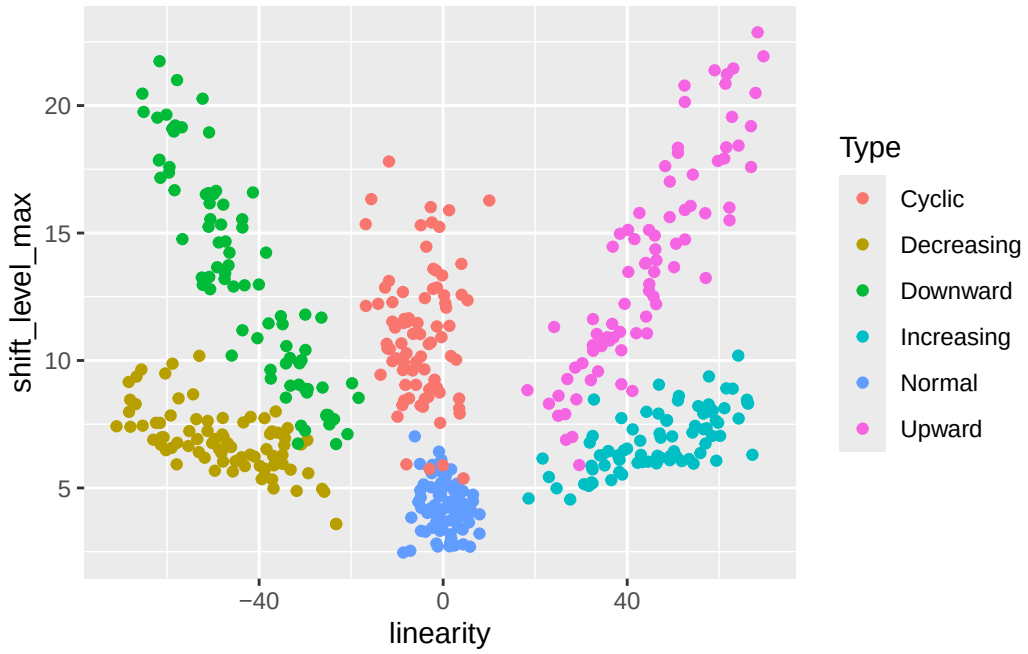
```
selected_feats<-c("linearity","trend_strength","acf1","stl_e_acf1","shift_level_max","sh
train_feat_selected<-train_feats|>
  select(Type,all_of(selected_feats))
```

```
library(GGally)
train_feat_selected|>
  ggpairs(
    columns = 2:ncol(train_feat_selected),
    lower=list(continuous=wrap("points",size=0.5,alpha=0.7)),
    upper=list(continuous=wrap("points",size=0.5,alpha=0.7)),
    mapping=aes(color=Type)
  )
```



We can find that linearity and shift_level_max can separate these six types.

```
ggplot(train_feats,aes(x=linearity,y=shift_level_max,color=Type))+geom_point()
```



The combination of linearity and shift_level_max provides good separation because linearity captures the direction and strength of systematic trends, while shift_level_max captures the magnitude of sudden level shifts. Different chart types exhibit distinct patterns in these two aspects, leading to clear clustering in the scatter plot.

d. Install the `caret` package and use the following snippet to fit a k-nearest neighbors model on the two features you have selected (substitute `X` and `Y` for the two features you selected in c), and then predict on the test set. What percentage of the test examples are correctly classified? Write code to compute this value. (You should get $> 90\%$ accuracy)

```
# Change to eval: TRUE in order to run
library(caret)
```

```
Loading required package: lattice
```

```
Attaching package: 'caret'
```

```
The following objects are masked from 'package:fabletools':

    MAE, RMSE
```

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

    lift
```

```
knn_fit <- train(Type ~ linearity + shift_level_max, data = train_feats, method = "knn")
test_pred<-predict(knn_fit, newdata = test_feats)
```

```
correct_pre<-sum(test_pred==test_feats$Type)
total_pre<-length(test_pred)
pre_accuracy<-correct_pre/total_pre
library(scales)
```

```
Attaching package: 'scales'

The following object is masked from 'package:purrr':

    discard

The following object is masked from 'package:readr':

    col_factor
```

```
percent(pre_accuracy,accuracy = 0.01)|>
print()
```

```
[1] "92.98%"
```

Question 4 (Periodograms)

Consider the `vic_elec` dataset from the `tsibbledata` package.

a. Compute and plot the periodogram for `Demand`. What frequencies and periods do the 7 highest peaks correspond to? *Hint: You may use the* `periodogram` *function provided in* `plot_util.R` *and change the* `max_freq` *setting to see at different resolutions.*

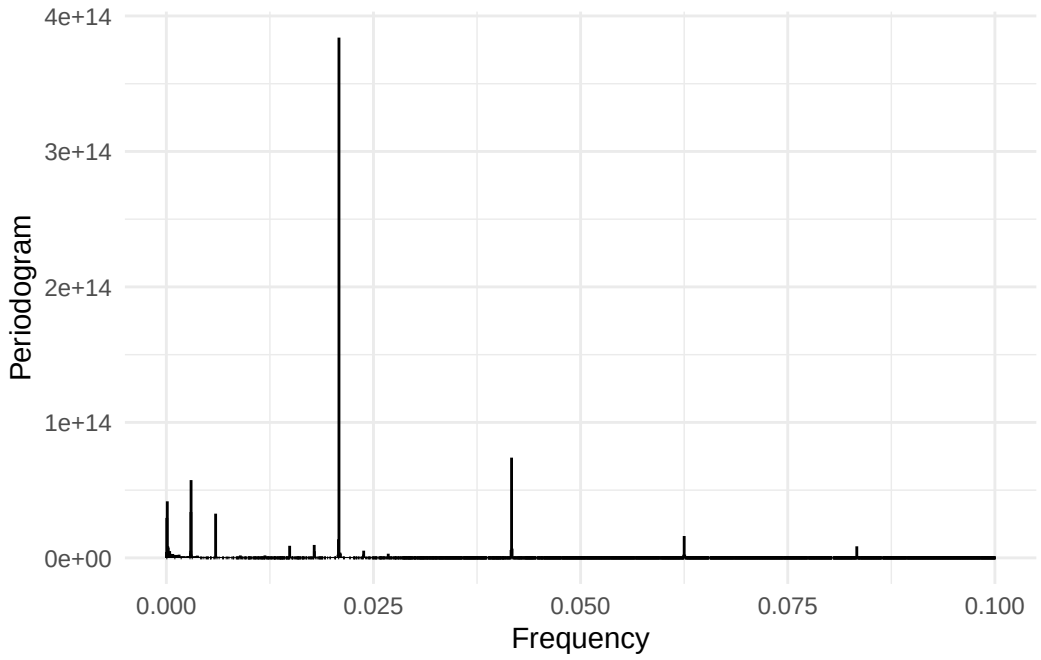
```
source("../_code/plot_util.R")
```

```
Loading required package: rlang
```

```
Attaching package: 'rlang'

The following objects are masked from 'package:purrr':

    %@%, flatten, flatten_chr, flatten_dbl, flatten_int, flatten_lgl,
    flatten_raw, invoke, splice

Loading required package: magrittr


Attaching package: 'magrittr'

The following object is masked from 'package:rlang':

    set_names

The following object is masked from 'package:purrr':

    set_names

The following object is masked from 'package:tidyr':

    extract
```

```
p<-vic_elec|>
  pull(Demand)|>
  periodogram(max_freq=0.1)
p
```

```
Warning: Removed 47347 rows containing missing values or values outside the scale range
(`geom_segment()`).
```




```
d <- ggplot_build(p)$data[[1]] |>
  transmute(freq = x, power = y)
top7 <- d |>
  filter(freq > 0) |>
  arrange(desc(power)) |>
  slice(1:7)
top7
```

```
          freq        power
1 2.083333e-02 3.840182e+14
2 4.166667e-02 7.400681e+13
3 2.984337e-03 5.741843e+13
4 1.140511e-04 4.175582e+13
5 2.965328e-03 3.386078e+13
6 5.949665e-03 3.264529e+13
7 5.702555e-05 2.951500e+13
```

```
top7_out <- top7 |>
  mutate(
    period_hhours   = 1 / freq,
    period_hours = period_hhours * 0.5,
    period_days  = period_hhours / 48
  )
```

```
top7_out
```

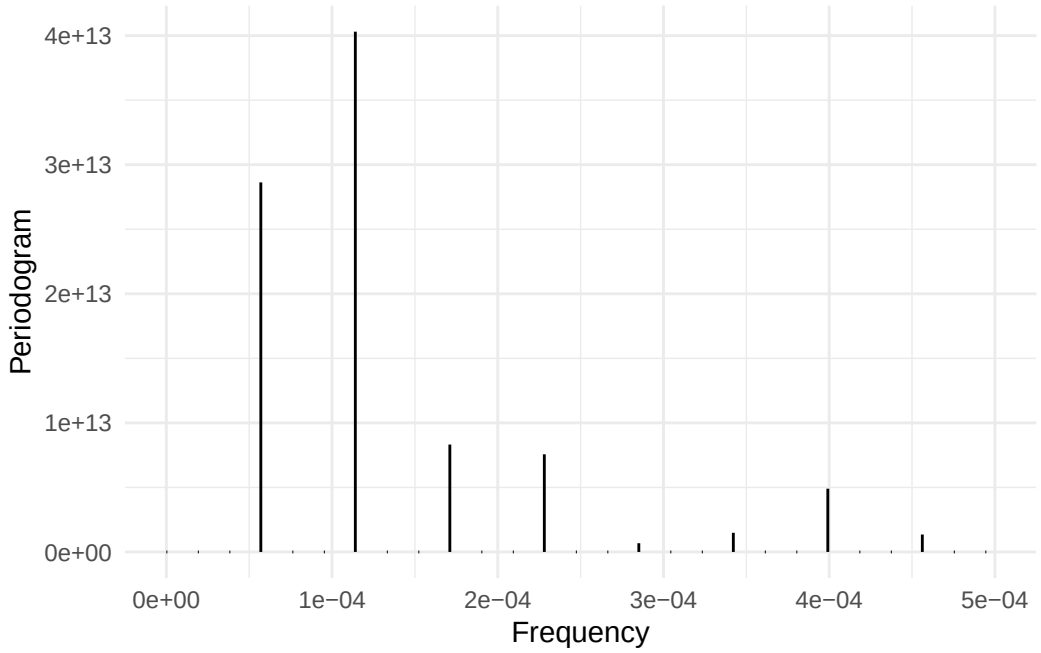
```
          freq        power period_hhours period_hours period_days
1 2.083333e-02 3.840182e+14       48.0000     24.00000    1.000000
2 4.166667e-02 7.400681e+13       24.0000     12.00000    0.500000
3 2.984337e-03 5.741843e+13      335.0828    167.54140    6.980892
4 1.140511e-04 4.175582e+13     8768.0000   4384.00000  182.666667
5 2.965328e-03 3.386078e+13      337.2308    168.61538    7.025641
6 5.949665e-03 3.264529e+13      168.0767     84.03834    3.501597
7 5.702555e-05 2.951500e+13    17536.0000   8768.00000  365.333333
```

b. Perform an STL decomposition for `Demand`. Plot the periodogram for `season_year`, `season_week`, and `season_day`. Which of the original periodogram peaks appear in each of these periodograms?

```
vic_elec_decomp<-vic_elec|>
  model(STL(Demand))|>
  components()
```

```
vic_elec_decomp|>
  pull(season_year)|>
  periodogram(max_freq = 0.0005)
```

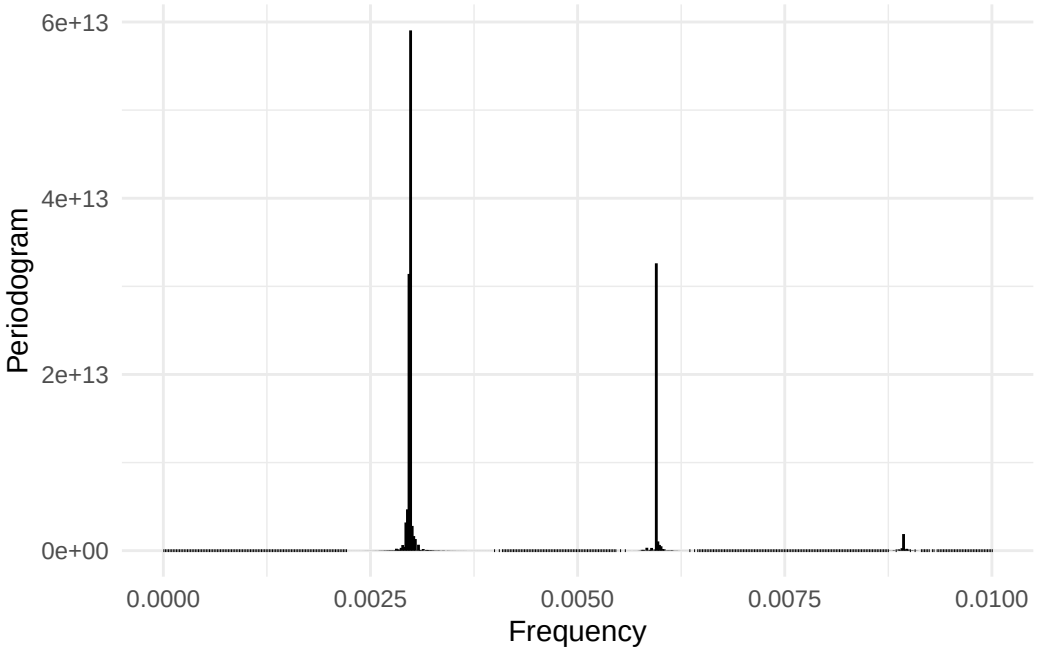
```
Warning: Removed 52581 rows containing missing values or values outside the scale range
(`geom_segment()`).
```



We can the original 4th and 7th highest peak.

```
vic_elec_decomp|>
  pull(season_week)|>
  periodogram(max_freq = 0.01)
```

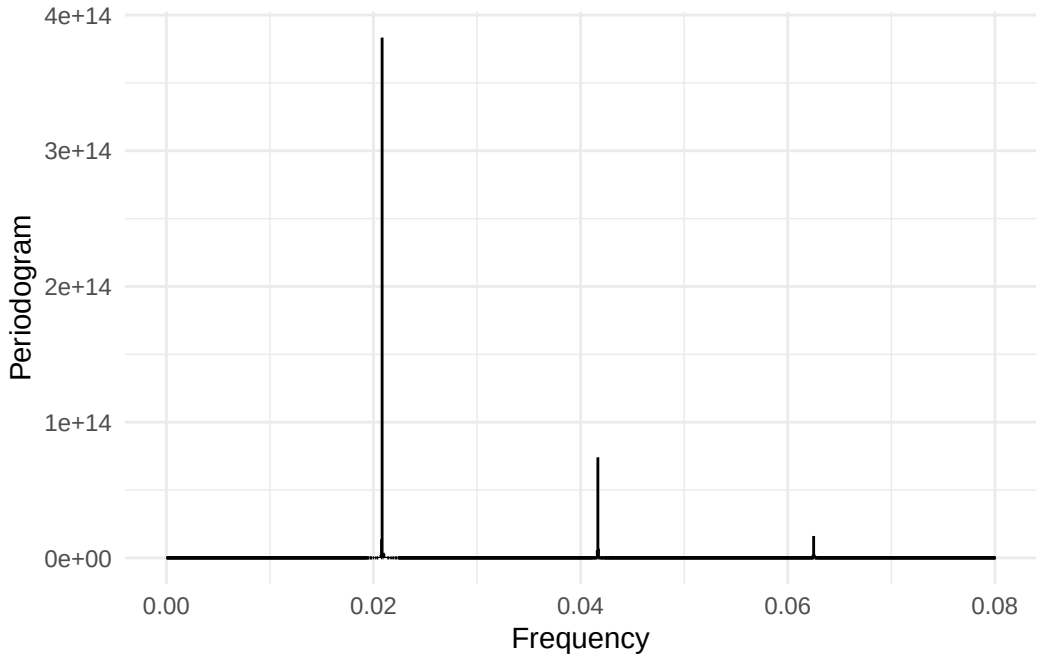
```
Warning: Removed 52081 rows containing missing values or values outside the scale range
(`geom_segment()`).
```



We can find the original 3rd, 5th and 6th highest peak.

```
vic_elec_decomp|>
  pull(season_day)|>
  periodogram(max_freq = 0.08)
```

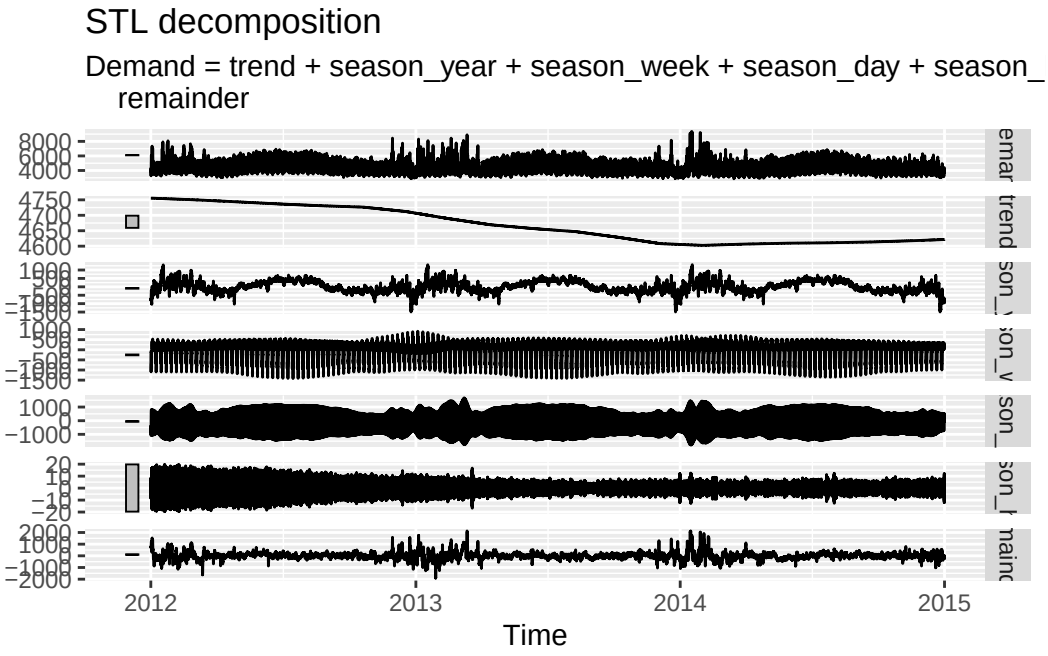
```
Warning: Removed 48399 rows containing missing values or values outside the scale range
(`geom_segment()`).
```



We can find the original 1st and 2nd highest peak.

c. Based on the peak heights, which is the strongest seasonal component? Does it agree with what you see in a time series decomposition plot?

```
vic_elec_decomp |>
  autoplot()
```



From b, we know that the daily cycle is the strongest seasonal components, and this aligns with the decomposition plot, where the seasonal_day component exhibits the most pronounced and repetitive fluctuations.