Code

Melbourne Datathon 2020: Source Code and Additional Analysis

Kasun Bandara

October 17, 2020

Hide

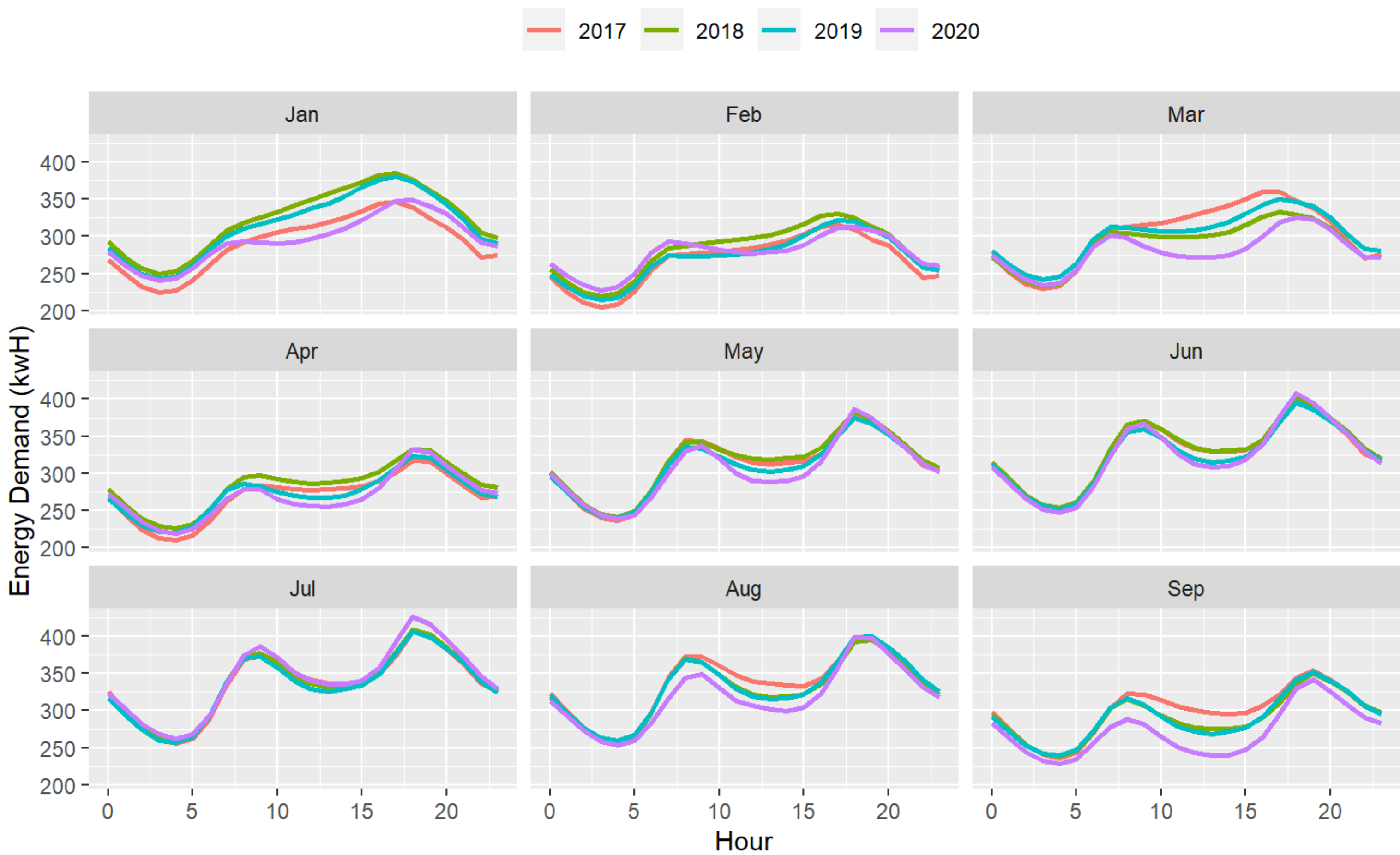
```
# Loading the requred Libraries.
library(vroom)
library(stringr)
library(sugrrants)
library(tsibble)
library(tidyverse)
library(feasts)
library(fable)
library(dplyr)
library(lubridate)
```

Energy Consumption Pattern Analysis

- This analysis is conducted only for the state of victoria, as one of the hardest impacted states from COVID19 in Australia. However, this analysis can be repeated to any other other state in Australia by changing the corresponding data files (energy, temperature, holidays).

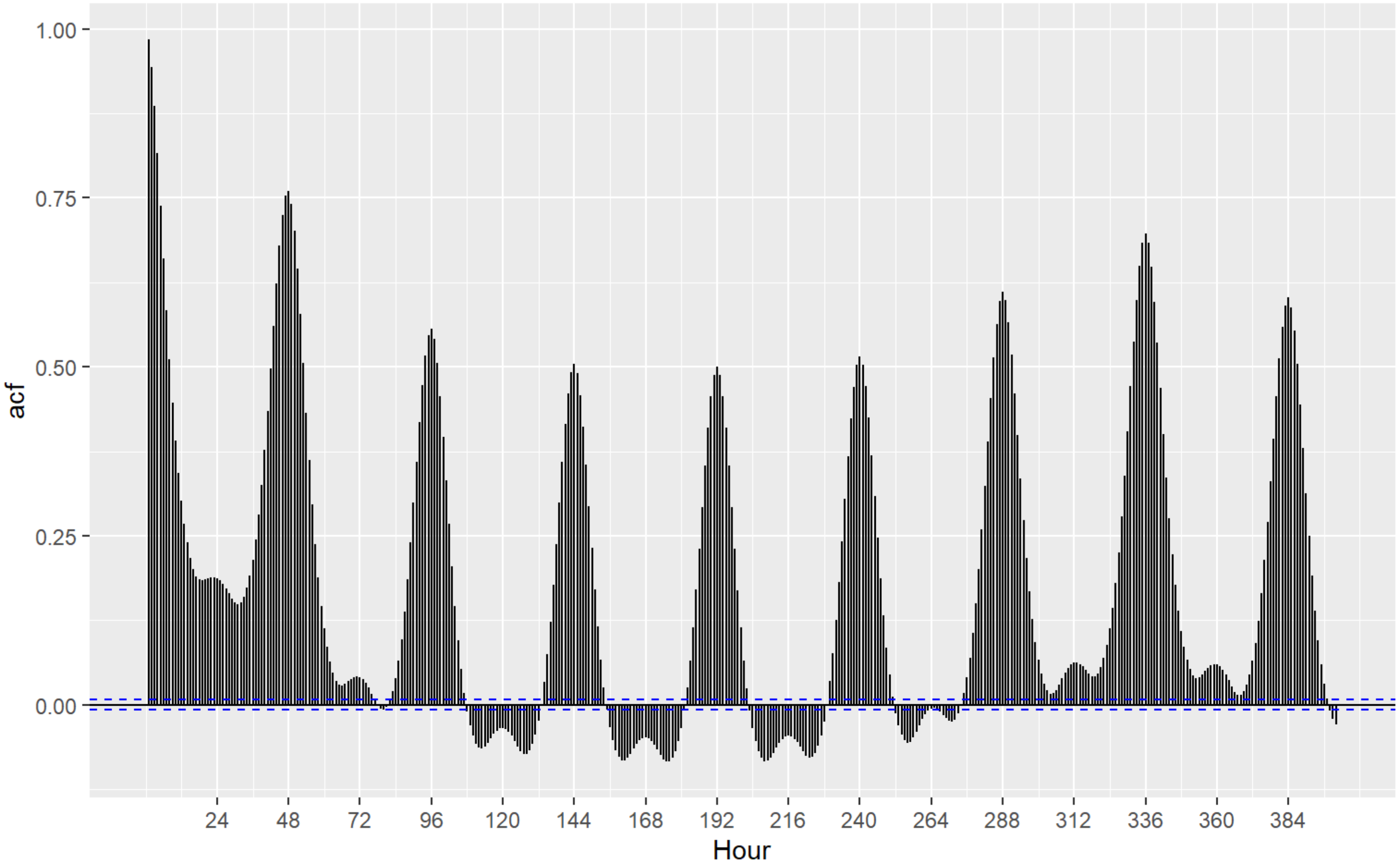
Hide

```
# Reading energy consumption data.
setwd("energy_data/")
vic_energy = list.files(pattern = "*.csv")
vic_energy_df <- vroom(vic_energy)

# Converting to time series data and mutating new variables for the analysis
vic_energy_hourly <-
  vic_energy_df %>% mutate(date = as_datetime(SETTLEMENTDATE, format = "%Y/%m/%d %H:%M:%S")) %>%
  select(date, TOTALDEMAND) %>% mutate(
    hour = hour(date),
    day = wday(date, label = TRUE),
    year = year(date),
    month_id = month(date),
    month_name = month(date, label = TRUE)
  )

# Selecting the first night months of each year (Because we only have data for 2020 till September)
vic_energy_hourly_filter <-
  vic_energy_hourly %>% filter(month_id %in% c(1:9))  %>% filter(month_id %in% c(1:9)) %>%
  select(-month_id) %>% group_by(hour, month_name, year) %>% summarise(total_energy = sum(TOTALDEMAND) /
                                                                          1e3)

# Generating Figure 1
ggplot(data = vic_energy_hourly_filter,
       mapping = aes(
         x = hour,
         y = total_energy,
         colour =  factor(year)
       )) +
  geom_line(size = 1.0) + guides(colour = guide_legend(title = "")) +
  theme(legend.position = "top") + facet_wrap(~ month_name) +
  ylab("Energy Demand (kwH)") + xlab("Hour")
```



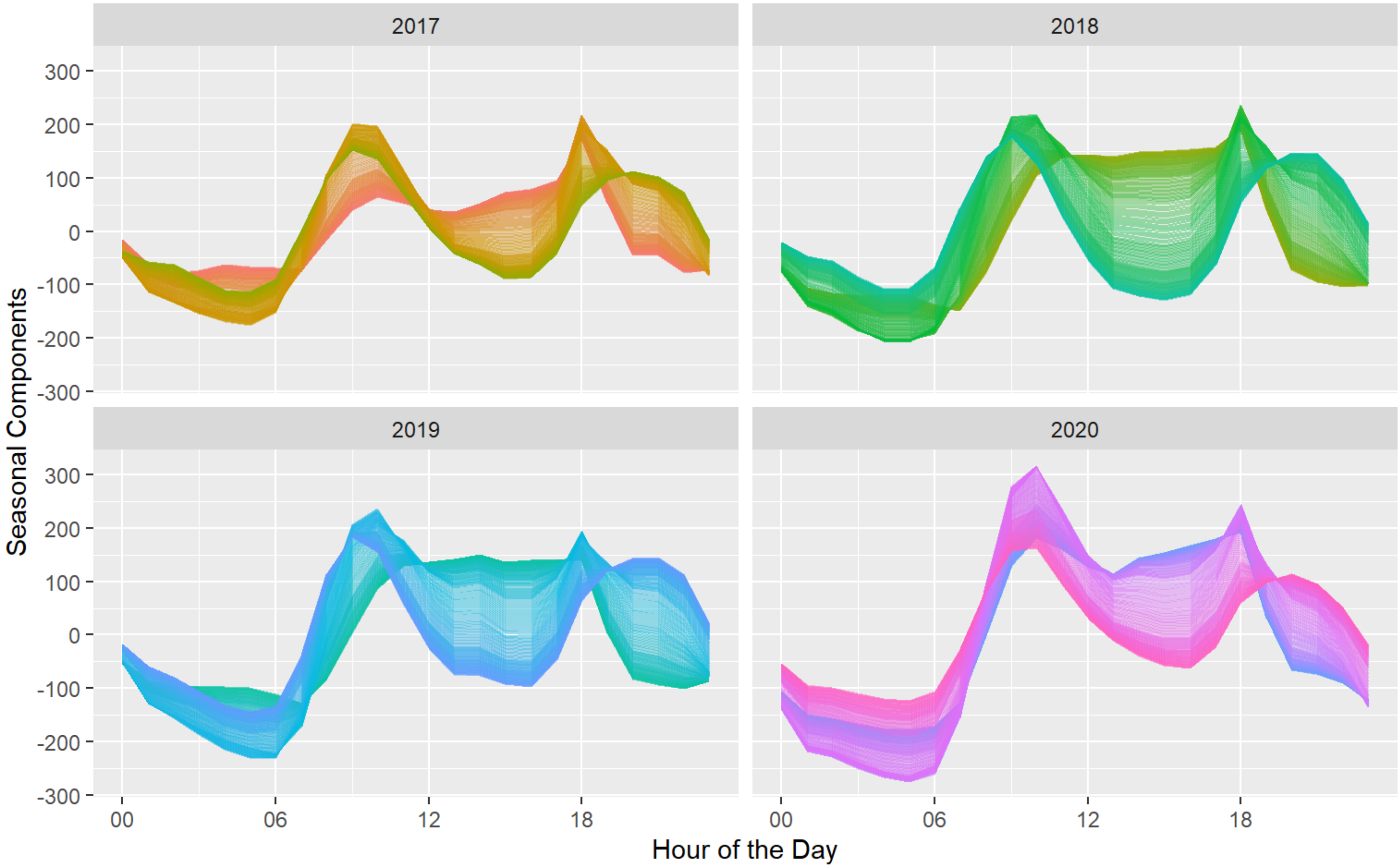
Hide

```
# Generating the autocorrelation plot for time series.
# Observation: The indication of multiple seasonalities, daily and weekly energy consumption seasonalities.
vic_energy_hourly %>% as_tsibble(index = date) %>% ACF(TOTALDEMAND / 1e3, lag_max = 400) %>% autoplot()  +
 xlab("Hour")
```


1.00
0.75
0.50
0.25
0.00
acf
24
48
72
96
120
144
168
192
216
240
264
288
312
336
360
384
Hour

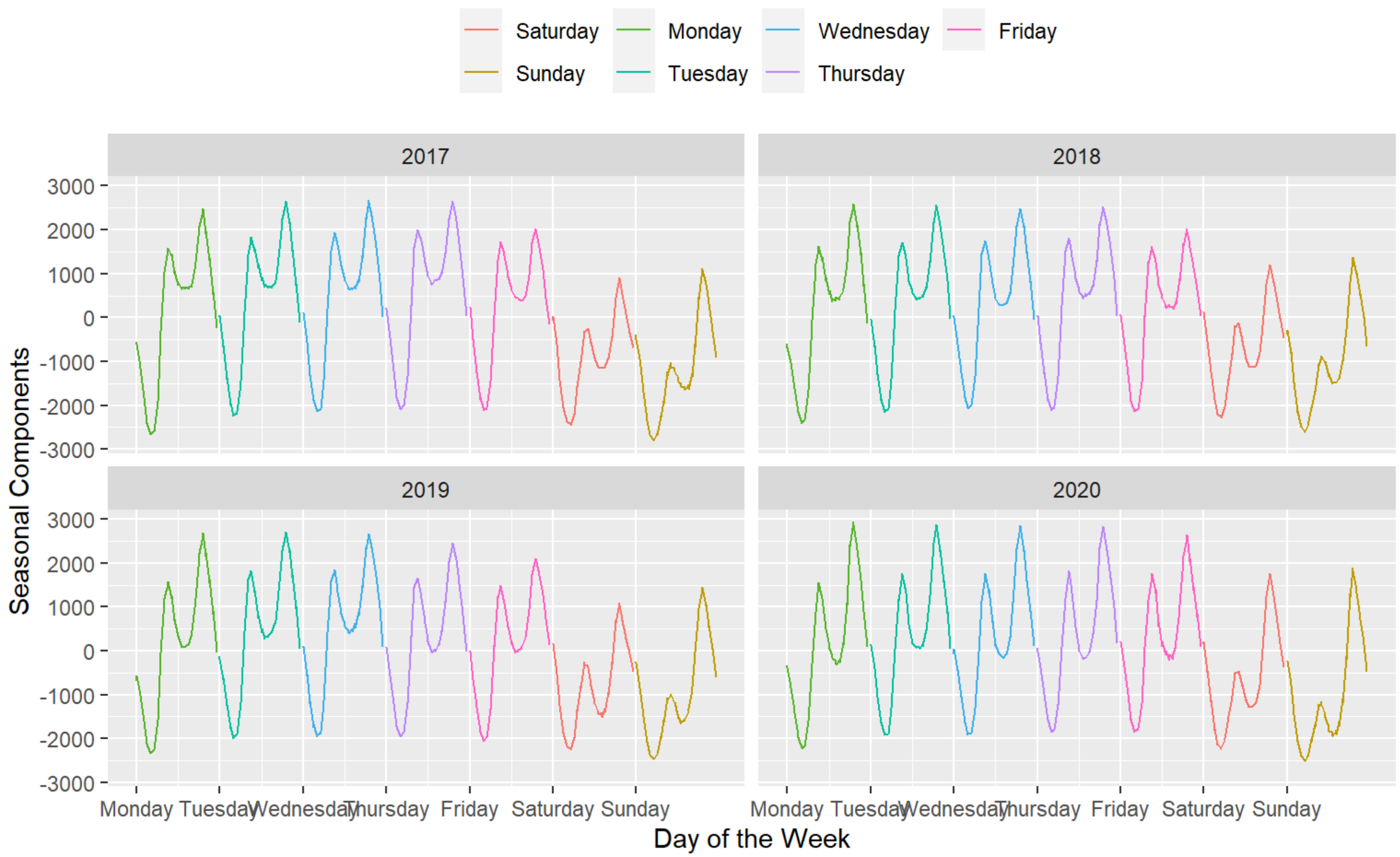
Hide

```
# Aggregating the time series to hourly wise, and filtering the months ranging from April - September for e
ach year.
# This is because the COVID19 restrictions started to impose in Victoria from April.
vic_energy_hourly_tsibble <-
  vic_energy_hourly %>% filter(month_id %in% c(4:9)) %>% mutate(hour_wise = round_date(date, "hour")) %>%
  group_by(hour_wise) %>% summarise(hourly_energy = sum(TOTALDEMAND)) %>% mutate(year = year(hour_wise)) %
>%
  as_tsibble(index = hour_wise, key = c(year))

# Applying the STL decomposition to extract the daily seasonality (Higher seasonal window is chosen for a b
etter visualisation)
vic_energy_daily_decomp <-
  vic_energy_hourly_tsibble %>% model(stl = STL(hourly_energy ~ season(window = 200))) %>% components() %>%
as_tibble() %>%
  select(hour_wise, year, season_day) %>% as_tsibble(index = hour_wise, key = c(year))

# Generating daily seasonal patterns for each year (Figure 2)
vic_energy_daily_decomp %>% gg_season(season_day, period = "day", alpha = 0.4) +
  facet_wrap( ~ year) + guides(colour = "none") +
  ylab("Seasonal Components") + xlab("Hour of the Day")
```


2017
2018
2019
2020
300
200
100
0
-100
-200
-300
Seasonal Components
00
06
12
18
Hour of the Day

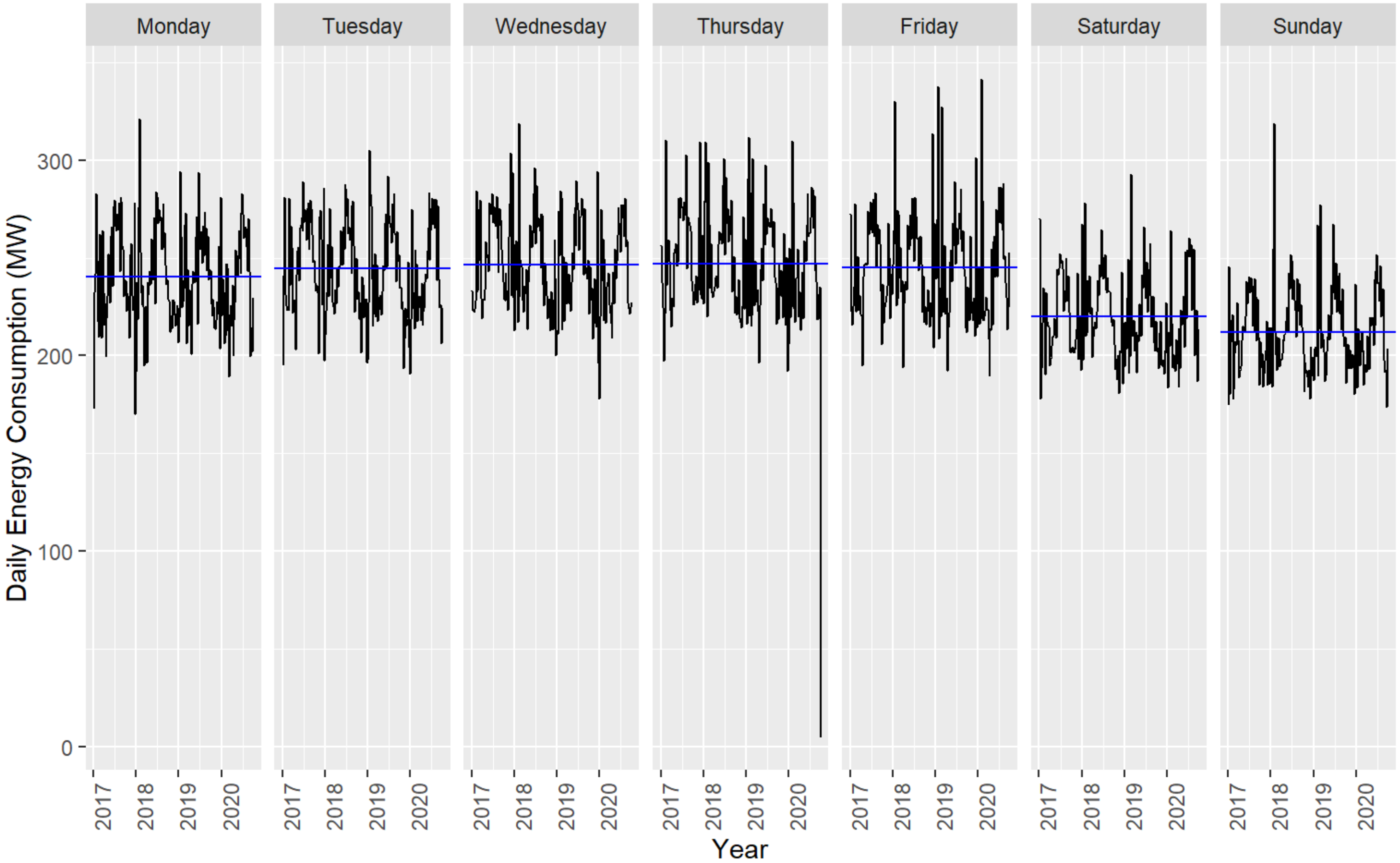
Hide

```
# Applying the STL decomposition to extract the weekly seasonality
vic_energy_weekly_decomp <-
  vic_energy_hourly_tsibble %>% model(stl = STL(hourly_energy ~ season(window = 60))) %>% components() %>%
 as_tibble() %>%
  select(hour_wise, year, season_week) %>% as_tsibble(index = hour_wise, key = c(year))

# Generating weekly seasonal patterns for each year (Figure 3)
vic_energy_weekly_decomp %>% gg_season(season_week, period = "week", facet_period = "day") +
  facet_wrap( ~ year) + theme(legend.position = "top") +
  ylab("Seasonal Components") + xlab("Day of the Week")
```



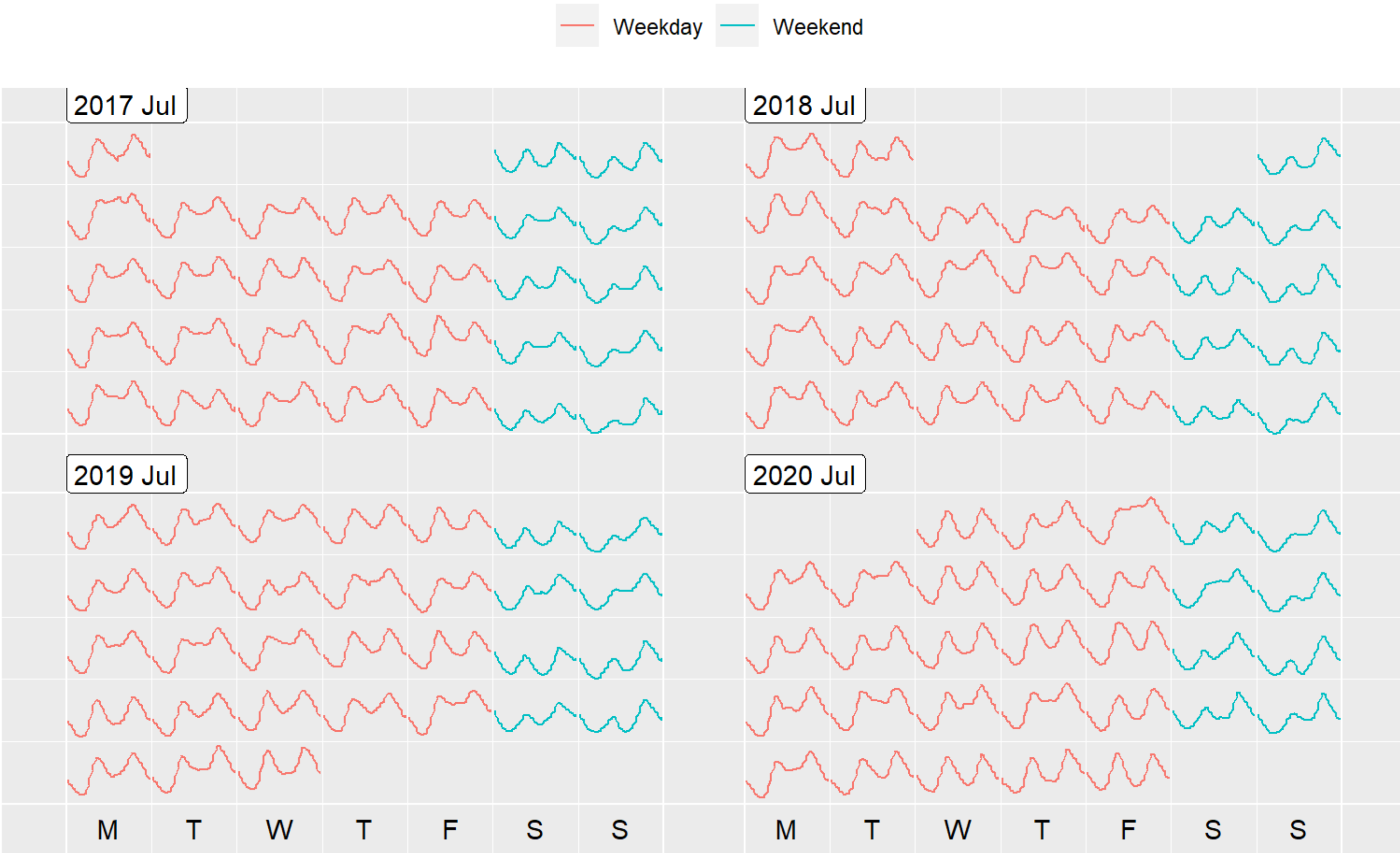
Hide

```
# Aggregating the half hourly time series to a daily time series.
vic_energy_daily_tsibble <-
  vic_energy_hourly %>% mutate(date = as_date(date)) %>% group_by(date) %>%
  summarise(daily_agg = sum(TOTALDEMAND) / 1e3)  %>% as_tsibble(index = date)

# Visualising the weekly consumption patterns across all years.
vic_energy_daily_tsibble %>% gg_subseries(daily_agg, period = "week") +
  ylab("Daily Energy Consumption (MW)") + xlab("Year")
```


Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Daily Energy Consumption (MW)
300
200
100
0
2017
2018
2019
2020
Year

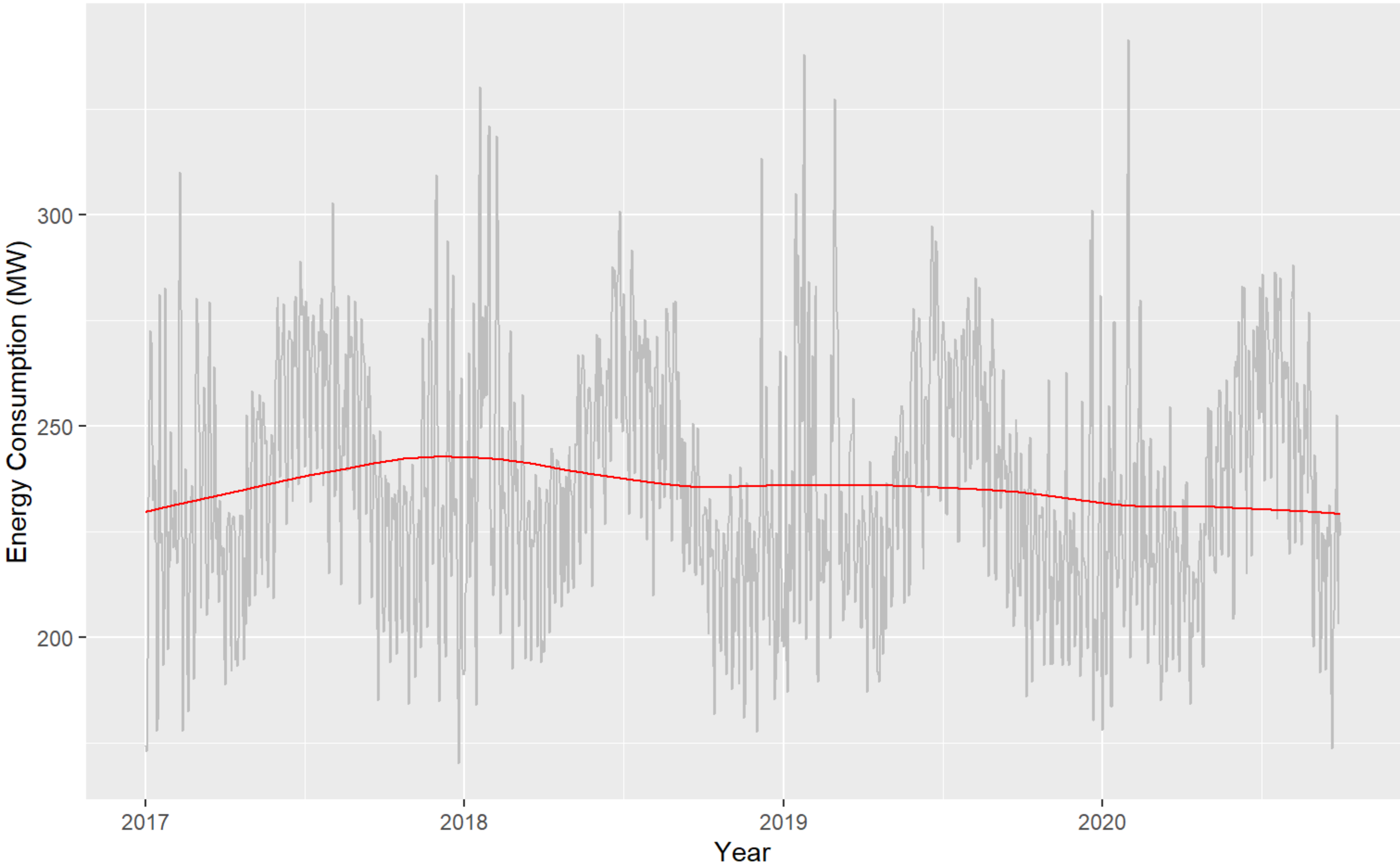
Hide

```
# Filtering the energy consumption relevant to July in each year
vic_energy_july <- vic_energy_hourly %>% filter(month_id == 7) %>%
  mutate(Time = hour) %>% mutate(Date = as.Date(date)) %>% mutate(Day = wday(Date, label = TRUE))

# Using frame_calendar() function to analyse the energy consumption in July.
# Mutating a new variable to denote the type of the day (Weekend or Weekday).
p <- vic_energy_july %>%
  mutate(Weekend = if_else(Day %in% c("Sat", "Sun"), "Weekend", "Weekday")) %>%
  frame_calendar(x = Time, y = TOTALDEMAND, date = Date) %>%
  ggplot(aes(
    x = .Time,
    y = .TOTALDEMAND,
    group = Date,
    colour = Weekend
  )) +
  geom_line() +
  guides(colour = guide_legend(title = "")) +
  theme(legend.position = "top")

# Generating Plot 4.
plot <- prettify(p)
plot
```


Weekday
Weekend
2017 Jul
2018 Jul
2019 Jul
2020 Jul
M
T
W
T
F
S
S
M
T
W
T
F
S
S

Hide

```
# Removing the record for October.
vic_energy_daily_tsibble = vic_energy_daily_tsibble[1:1369, ]

# Daily time series decomposition. A higher window is chosen to smooth the trend.
vic_energy_trend_decomp <- vic_energy_daily_tsibble %>%
  model(stl = STL(daily_agg ~ trend(window = 365)))

# Plotting the overall trend of the energy consumption over the years.
vic_energy_daily_tsibble %>%  autoplot(daily_agg, color = "grey") +
  autolayer(components(vic_energy_trend_decomp), trend, color = "red") +
  ylab("Energy Consumption (MW)") + xlab("Year")
```



- The overall trend in energy consumption over the past few years. Despite the population growth in Victoria [1], we see that the overall trend of energy demand is relatively constant (or follows a small negative trend). This can be attributed to the uptaking of solar panels in the households. According to [2], over the past few years, the number of solar panel installations have been increasing in Victoria, that could act as a reason to decrease the overall total energy demand in the national grid.

Energy Consumption vs Temperature

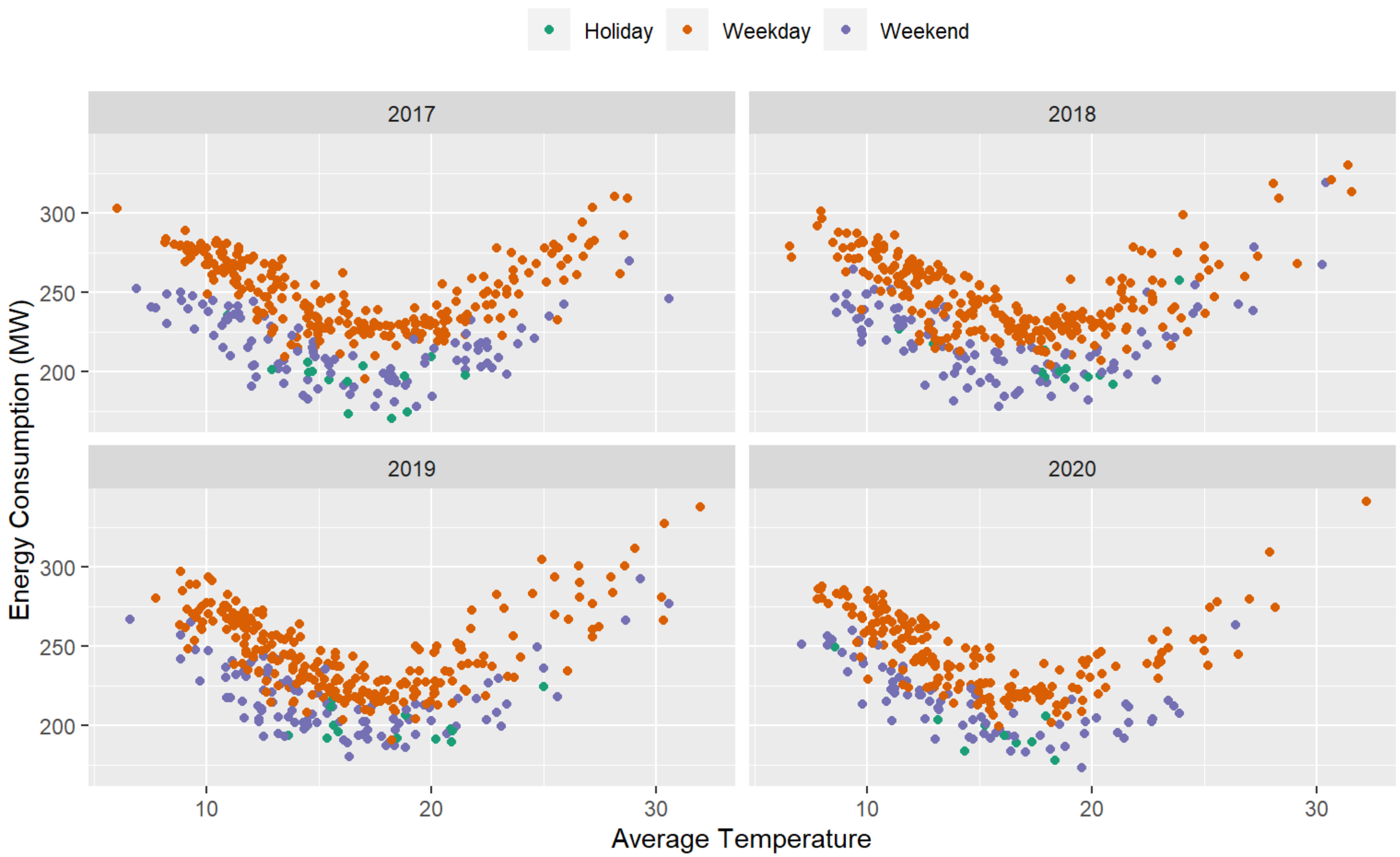
- Assumption: The bureau of meteorology station number used as temperature data: **086338: Weather Observations for Melbourne (Olympic Park)**, assuming this represents the temperature of the state of victoria.

Hide

```
# Reading the temperature max and min data.
df_weather_max <- read_csv("Temp/IDCJAC0010_086338_1800_Data.csv")
df_weather_min <- read_csv("Temp/IDCJAC0011_086338_1800_Data.csv")

df_weather_max$Month <- as.numeric(df_weather_max$Month)
df_weather_max$Day <- as.numeric(df_weather_max$Day)
df_weather_min$Month <- as.numeric(df_weather_min$Month)
df_weather_min$Day <- as.numeric(df_weather_min$Day)

# Filter max temperature data from 2017 Jan to 2020 Sep.
df_weather_max_select <-
  df_weather_max %>% select(Year, Month, Day, `Maximum temperature (Degree C)`) %>%
  filter(Year %in% c(2017:2020)) %>%
  filter(!(Year == 2020  & Month %in% c(10)))

# Filter min temperaturer data from 2017 Jan to 2020 Sep.
df_weather_min_select <-
  df_weather_min %>% select(Year, Month, Day, `Minimum temperature (Degree C)`) %>%
  filter(Year %in% c(2017:2020)) %>%
  filter(!(Year == 2020  & Month %in% c(10)))


df_weather_combined <-
  cbind(
    vic_energy_daily_tsibble,
    max_temp = df_weather_max_select$`Maximum temperature (Degree C)`,
    min_temp = df_weather_min_select$`Minimum temperature (Degree C)`
  )

# Computing the average temperature.
df_weather_vic <-
  df_weather_combined %>% mutate(year = year(date)) %>%
  mutate(avg_temp = (max_temp + min_temp) / 2) %>% as_tibble()
```

```
# Reading Victorian holidays files.
holiday_2018 <-
  read.csv("Holidays/australianpublicholidays-201718.csv")
holiday_2019 <-
  read.csv("Holidays/australian_public_holidays_2019.csv")
holiday_2020 <-
  read.csv("Holidays/australian_public_holidays_2020.csv")

colnames(holiday_2020) <- colnames(holiday_2018)
colnames(holiday_2019) <- colnames(holiday_2018)

vic_holidays <- rbind(holiday_2018, holiday_2019, holiday_2020)

vic_holidays$Date <- as.character(vic_holidays$Date)

# Filtering holidays specific to Victoria (including the national holidays)
vic_holidays_df <- vic_holidays %>% select(Date, Applicable.To) %>%
  filter(
    str_detect(Applicable.To, "VIC") |
      str_detect(Applicable.To, "vic") |
      str_detect(Applicable.To, "NAT")
  ) %>%
  mutate(date = as.Date(Date, "%Y%m%d")) %>% mutate(Holiday = TRUE) %>%
  filter(!(year(date) == 2020  &
             month(date) %in% c(10, 11, 12))) %>% select(date, Holiday)
# Joining the temperature data and holidays.
df_vic_weather_temp <-
  df_weather_vic %>% left_join(vic_holidays_df, by = "date") %>% replace_na(list(Holiday = FALSE))

# Creating a new column to denote date type (Weekend, Weekday, Holiday)
df_vic_weather_temp_final <- df_vic_weather_temp %>%
  mutate(Day_Type = case_when(
    Holiday ~ "Holiday",
```

```
    wday(date) %in%  2:6 ~ "Weekday",
    TRUE ~ "Weekend"
  ))

# Figure 5
df_vic_weather_temp_final %>%
  ggplot(aes(
    x = avg_temp,
    y = daily_agg,
    colour = factor(Day_Type)
  )) +
  geom_point() +
  scale_colour_brewer(palette = "Dark2") + facet_wrap( ~ year) +
  guides(colour = guide_legend(title = "")) +
  theme(legend.position = "top") +
  ylab("Energy Consumption (MW)") + xlab("Average Temperature")
```



Energy Forecasting

- External Variables: *Weekend/Weekday/Holiday (Holidays and Weekends are considered as the same)*, *Average Temperature (As a non linear function to the demand, as it follows a non-linear relationship accrording to Figure 5)*, *Day of the Week*, *Month*, *COVID Dummy Variable (indicates whether a particular date belongs to the COVID19 restriction period)*
- Model Selection: As a *single time series* is considered for this forecasting task, based on the recommendations given in [3] and [4], I avoid using deep learning/machine learning based forecasting models, instead use univariate ARIMA model and Time series regression (TSLM) model. I also use Naive Seasonal model as a baseline model to compare against the

proposed models.

- Varaints and Benchmarks: The ARIMA model without the COVID19 dummy variable (*ARIMA.NORMAL*), The ARIMA model with the COVID19 dummy variable (*ARIMA.COVID*), Time series Regression without the COVID19 dummy variable (*TSLM.NORMAL*), Time series Regression with the COVID19 dummy variable (*TSLM.COVID*), and Naive Seasonal model that does not include any exogenous variables (*NAIVE_SEASONAL*).
- Error Measures: *Mean Absolute Scaled Error (MASE), Root Mean Square Error (RMSE)*
- The order of the non-linear function (quadratic, cubic) to model temperature is determined by minimising the AICC model fitting error in ARIMA models (The best order is chosen as 2)

Hide

```
# Creating the exogenous variables to model.
df_vic_forecast <- df_vic_weather_temp_final %>%
  mutate(week_day = wday(date),
         month = month(date),) %>%
  select(date, daily_agg, avg_temp, week_day, month, Day_Type) %>% as_tsibble(index = date)

# Introducing COVID dummy variable to indicate the restrictions imposed (From the month of April)
df_vic_forecast_covid <- df_vic_forecast %>%
  mutate(Covid = case_when((year(date) == 2020 &
                              month %in% c(4:9)) ~ 1, TRUE ~ 0))

# Spliting the dataset to training and testing.
df_vic_forecast_covid_train <-
  df_vic_forecast_covid %>% filter(date < "2020-09-01")
df_vic_forecast_covid_test <-
  df_vic_forecast_covid %>% filter(date > "2020-08-31") %>% select(-daily_agg)


# Different models built using different orders of temperature. Comment out these code blocks to examine th
e individual models.


#df_vic_covid_fit1 <- df_vic_forecast_covid_train %>%
#  model(ARIMA.COVID = ARIMA((daily_agg) ~ avg_temp + (Day_Type == "Weekday") + week_day + month +  Covid),
#       ARIMA.NORMAL = ARIMA((daily_agg) ~ avg_temp + (Day_Type == "Weekday") + week_day + month),
#      TSLM.NORMAL = TSLM((daily_agg) ~ avg_temp + (Day_Type == "Weekday") + week_day + month),
#     TSLM.COVID = TSLM((daily_agg) ~ avg_temp + (Day_Type == "Weekday") + week_day + month + Covid),
#    Naive_Seasonal = SNAIVE(daily_agg))

#glance(df_vic_covid_fit1)


#df_vic_covid_fit3 <- df_vic_forecast_covid_train %>%
```

```
# model(ARIMA.COVID = ARIMA((daily_agg) ~ avg_temp + I(avg_temp^2) + I(avg_temp^3)+ (Day_Type == "Weekday")
+ week_day + month +  Covid),
#      ARIMA.NORMAL = ARIMA((daily_agg) ~ avg_temp + I(avg_temp^2) + I(avg_temp^3)+ (Day_Type == "Weekday")
+ week_day + month),
#     TSLM.NORMAL = TSLM((daily_agg) ~ avg_temp + I(avg_temp^2) + I(avg_temp^3) + (Day_Type == "Weekday") +
week_day + month),
#    TSLM.COVID = TSLM((daily_agg) ~ avg_temp + I(avg_temp^2) + I(avg_temp^3) + (Day_Type == "Weekday") + w
eek_day + month + Covid),
#    Naive_Seasonal = SNAIVE(daily_agg))

#glance(df_vic_covid_fit3)



# Model fitting for ARIMA.COVID, ARIMA.NORMAL, TSLM.NORMAL, TSLM.COVID, and NAIVE_SEASONAL.
df_vic_covid_fit2 <- df_vic_forecast_covid_train %>%
  model(
    ARIMA.COVID = ARIMA(
      daily_agg ~ avg_temp + I(avg_temp ^ 2) + (Day_Type == "Weekday") + week_day + month +  Covid
    ),
    ARIMA.NORMAL = ARIMA(
      daily_agg ~ avg_temp + I(avg_temp ^ 2) + (Day_Type == "Weekday") + week_day + month
    ),
    TSLM.NORMAL = TSLM(
      daily_agg ~ avg_temp + I(avg_temp ^ 2) + (Day_Type == "Weekday") + week_day + month
    ),
    TSLM.COVID = TSLM(
      daily_agg ~ avg_temp + I(avg_temp ^ 2) + (Day_Type == "Weekday") + week_day + month + Covid
    ),
    NAIVE_SEASONAL = SNAIVE(daily_agg)
  )
```

Hide

```
# Summarise the model fitting.
glance(df_vic_covid_fit2)
```

```
## # A tibble: 5 x 17
##   .model sigma2 log_lik   AIC  AICc   BIC ar_roots ma_roots r_squared
##   <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <list>   <list>       <dbl>
## 1 ARIMA~   76.6  -4796. 9617. 9617. 9685. <cpl [1~ <cpl [4~    NA
## 2 ARIMA~   76.6  -4796. 9615. 9616. 9678. <cpl [1~ <cpl [4~    NA
## 3 TSLM.~  163.   -5306. 6825. 6825. 6862. <NULL>   <NULL>       0.783
## 4 TSLM.~  162.   -5305. 6825. 6825. 6867. <NULL>   <NULL>       0.783
## 5 NAIVE~  671.      NA    NA    NA    NA  <NULL>   <NULL>      NA
## # ... with 8 more variables: adj_r_squared <dbl>, statistic <dbl>,
## #   p_value <dbl>, df <int>, CV <dbl>, deviance <dbl>, df.residual <int>,
## #   rank <int>
```

Hide

```
# Reporting the parameters for individual models.
df_vic_covid_fit2 %>% select(TSLM.COVID) %>% report()
```

```
## Series: daily_agg
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -56.4038  -8.6005   0.1256   8.8805  46.4836
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              403.84476    3.84631 104.995  < 2e-16 ***
## avg_temp                 -22.45302    0.41836 -53.670  < 2e-16 ***
## I(avg_temp^2)              0.61495    0.01164  52.815  < 2e-16 ***
## Day_Type == "Weekday"TRUE  30.76242    0.75050  40.989  < 2e-16 ***
## week_day                   0.80502    0.17433   4.618 4.25e-06 ***
## month                     -0.70554    0.10893  -6.477 1.31e-10 ***
## Covid                     -1.64808    1.15180  -1.431    0.153
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.75 on 1332 degrees of freedom
## Multiple R-squared: 0.7832,  Adjusted R-squared: 0.7822
## F-statistic: 801.8 on 6 and 1332 DF, p-value: < 2.22e-16
```

Hide

```
df_vic_covid_fit2 %>% select(ARIMA.COVID) %>% report()
```

```
## Series: daily_agg
## Model: LM w/ ARIMA(0,1,4)(2,0,0)[7] errors
##
## Coefficients:
##           ma1      ma2      ma3      ma4    sar1    sar2  avg_temp
##       -0.3403  -0.2230  -0.1154  -0.1170  0.0838  0.0401  -12.6767
## s.e.   0.0327   0.0302   0.0305   0.0278  0.0290  0.0280    0.4198
##       I(avg_temp^2)  Day_Type == "Weekday"TRUE  week_day    month   Covid
##              0.3847                    27.0847    1.1103  -0.3656  4.3784
## s.e.         0.0108                     0.5957    0.1325   0.3365  6.7282
##
## sigma^2 estimated as 76.61:  log likelihood=-4795.51
## AIC=9617.01   AICc=9617.29   BIC=9684.6
```

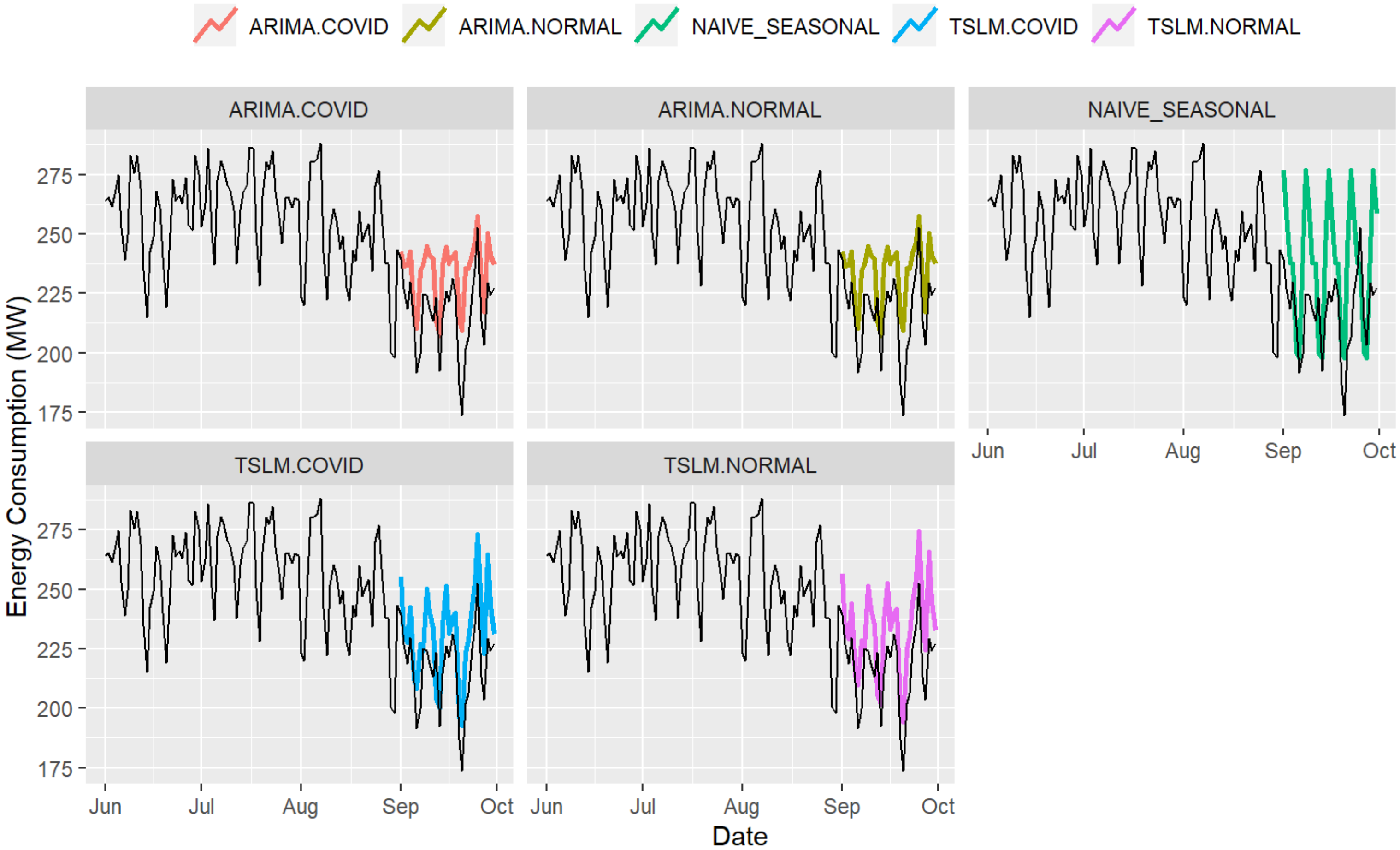
Hide

```
#df_vic_covid_fit2 %>% select(ARIMA.NORMAL) %>% report()
#df_vic_covid_fit2 %>% select(NAIVE_SEASONAL) %>% report()

# Generating forecasts for the models.
forecast <-
  df_vic_covid_fit2  %>% forecast(df_vic_forecast_covid_test)

# Evaluating the accuracy of the models.
accuracy(forecast, df_vic_forecast_covid) %>% as_tibble() %>% select(.model, RMSE, MASE)
```

```
## # A tibble: 5 x 3
##   .model          RMSE  MASE
##   <chr>          <dbl> <dbl>
## 1 ARIMA.COVID     19.6 0.970
## 2 ARIMA.NORMAL    19.6 0.971
## 3 NAIVE_SEASONAL  30.7 1.41
## 4 TSLM.COVID      18.4 0.906
## 5 TSLM.NORMAL     19.6 0.978
```

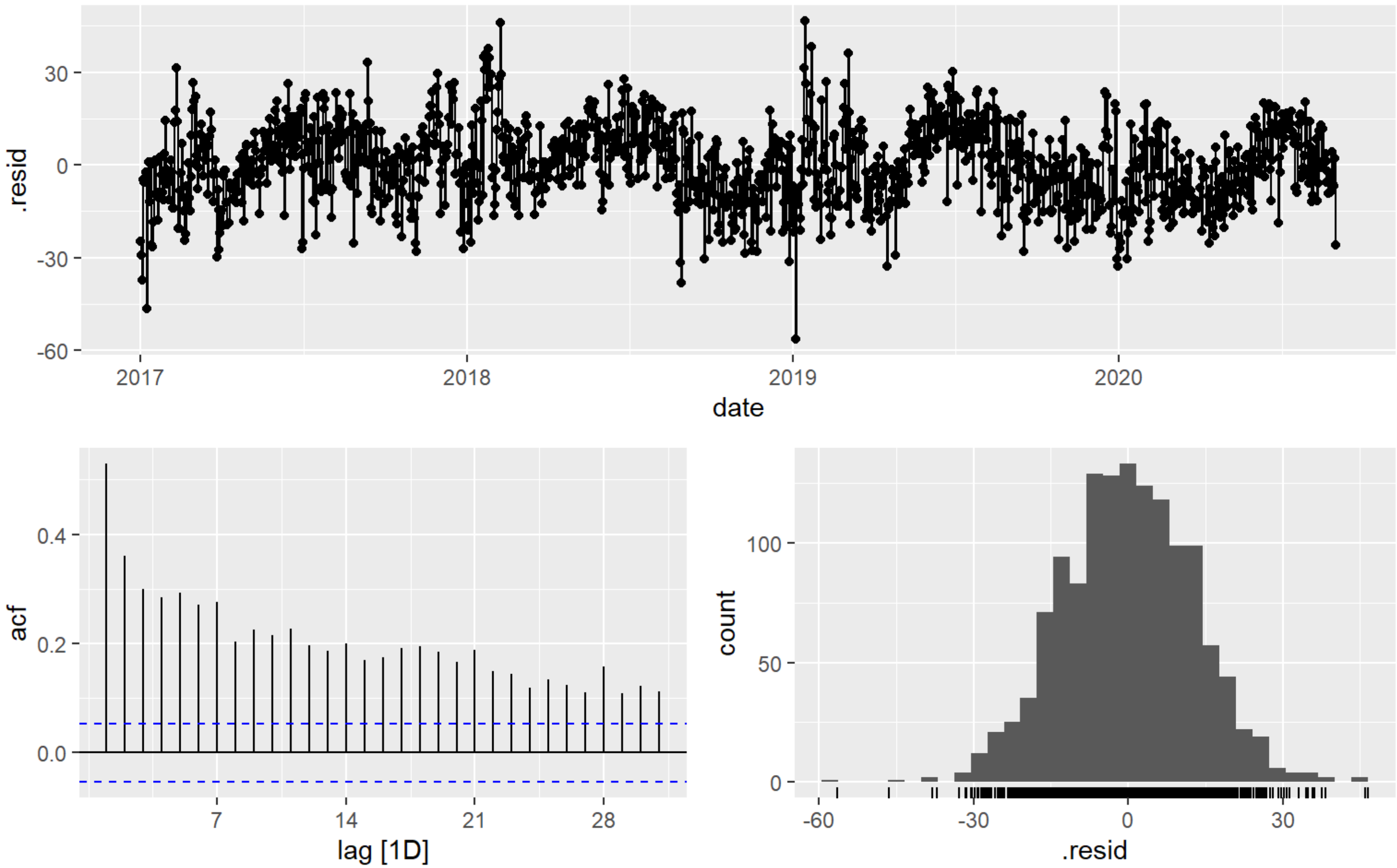
Hide

```
# Figure 6
forecast %>% autoplot(filter(df_vic_forecast_covid, (year(date) == 2020 &
                                                       month %in% c(6:9))),
                      level = NULL,
                      size = 1) +
  facet_wrap( ~ .model) +
  guides(colour = guide_legend(title = "")) +
  theme(legend.position = "top") +
  xlab("Date") + ylab("Energy Consumption (MW)")
```



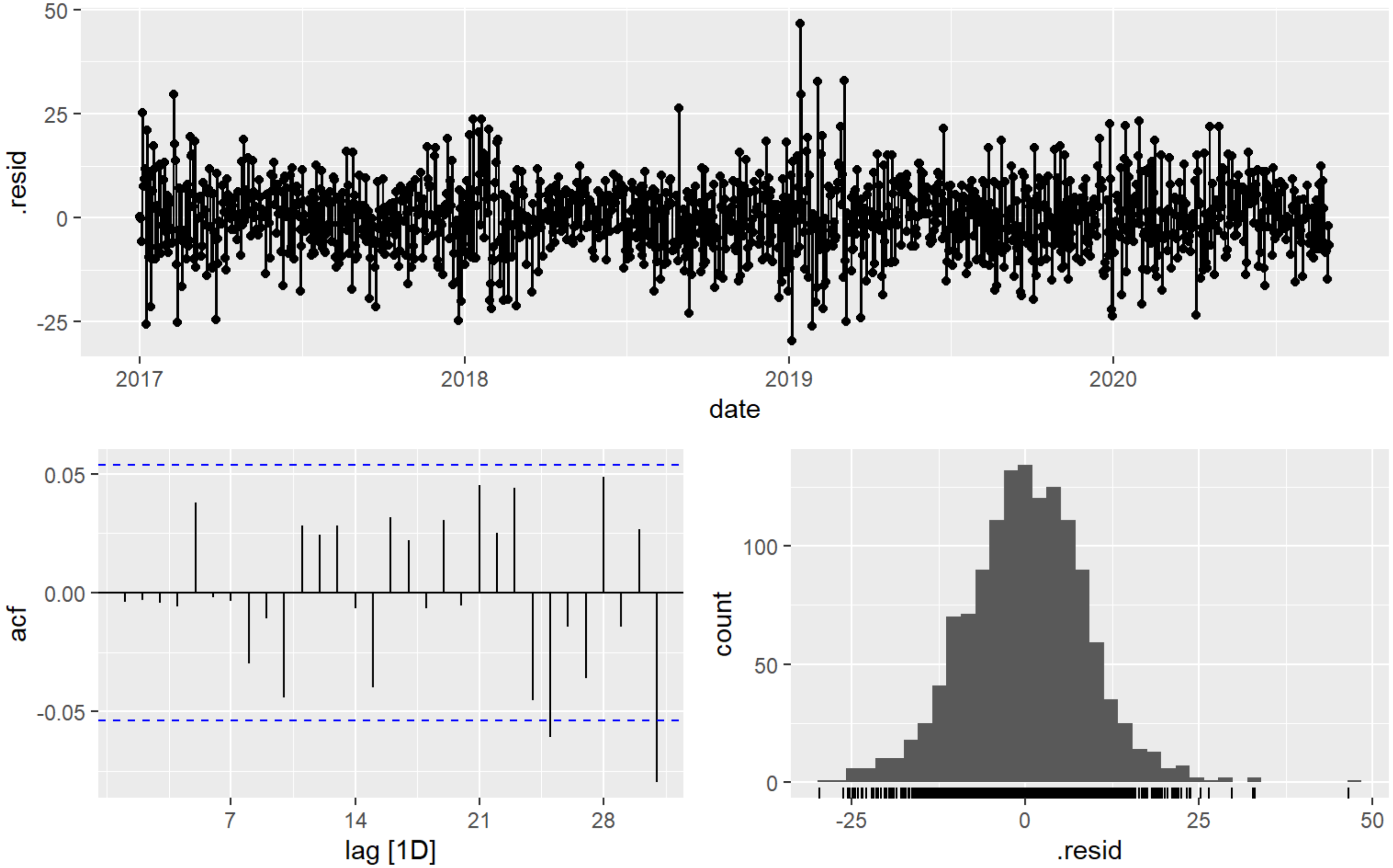
Hide

```
# Residual diagnostics for the best performing models.
df_vic_covid_fit2 %>% select(TSLM.COVID) %>% gg_tsresiduals()
```



Hide

```
df_vic_covid_fit2 %>% select(ARIMA.COVID) %>% gg_tsresiduals()
```



Hide

```
# A Portmanteau test is conducted using the Ljung-Box test.
df_vic_covid_fit2  %>% select(TSLM.COVID) %>% augment() %>%
  features(.resid, ljung_box, dof = 7, lag = 20)
```

```
## # A tibble: 1 x 3
##   .model     lb_stat lb_pvalue
##   <chr>        <dbl>     <dbl>
## 1 TSLM.COVID   1774.         0
```

Hide

```
df_vic_covid_fit2  %>% select(ARIMA.COVID) %>% augment() %>%
  features(.resid, ljung_box, dof = 12, lag = 20)
```

```
## # A tibble: 1 x 3
##   .model      lb_stat lb_pvalue
##   <chr>         <dbl>     <dbl>
## 1 ARIMA.COVID    14.6    0.0681
```

- According to residual distribution plots, we see that the mean residuals for both **TSLM.COVID** and **ARIMA.COVID** are closer to zero. However, in the **TSLM.COVID** residual ACF plot, it can be seen that the residuals are correlated, and there is still information left in the residuals, which can be used in generating forecasts. This can be mainly due to **TSLM** not handling the autocorrelations within a time series (only use external variables, no past lags used). On the other hand, in the **ARIMA.COVID** residual plot, we see that the residuals are mostly uncorrelated (except for the small spike in lag 32). Therefore, we can conclude that the proposed **ARIMA.COVID** satisfy the properties of a good forecasting model [5]
- I also extended the residual diagnostics by conducting a Ljung-Box test to assess the normality of the residuals (normality test). The Ljung-Box test for the **ARIMA.COVID** and **TSLM.COVID** returns the p-values of 0.0642 and 0 respectively. Confirming with previous observations from the residual analysis, we see that only **ARIMA.COVID** residual results are not significant (p-value greater than 0.05), and the residuals are not distinguishable from a white noise series (failing to reject the null hypothesis).
- Nevertheless, according to the forecast accuracy, **TSLM.COVID** performs the best, recording the lowest RMSE and MASE. Also, among ARIMA varaints, the **ARIMA.COVID** variant outperforms the **ARIMA.NORMAL**. This indicates the importance of accounting for the COVID19 restriction factor when forecasting energy consumption under current circumstances.

References

1. Victoria Population (http://www.population.net.au/population-of-victoria/#:~:text=Population%20Growth%20of%20Victoria,year%20to%20the%20overall%20population)
2. Solar Panel Growth (http://www.cleanenergyregulator.gov.au/RET/Forms-and-resources/Postcode-data-for-small-scale-installations#SGU--Solar-Deemed)
3. Criteria for classifying forecasting methods (https://www.sciencedirect.com/science/article/pii/S0169207019301529)
4. Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions (https://www.sciencedirect.com/science/article/abs/pii/S0169207020300996)
5. Forecasting: Principles and Practice (https://otexts.com/fpp3/diagnostics.html)