Using fpp3 package

This R file is an exploration into the fpp3 package and time series graphics, forecasting, and regression modelling.

```
library(fpp3)
```

```
## -- Attaching packages --------------------------------------------- fpp3 0.4.0 --
```

```
## v tibble      3.0.3     v tsibble     1.0.1
## v dplyr       1.0.2     v tsibbledata 0.3.0
## v tidyr       1.1.2     v feasts      0.2.2
## v lubridate   1.7.9     v fable       0.3.1
## v ggplot2     3.3.2
```

```
## -- 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()
```

We will use the vic_elec dataset

- this is a half-hourly electricity demand for Victoria, Austrailia from January 1, 2012 through December 31, 2014.
- Data Source: Australian Energy Market Operator & is contained in the tsibble library in R
- Demand = Total electricity demand in MW
- Temperature = Temperature of Melbourne, Austrailia
- Holiday = Indicator if that day is a public holiday

```
# reassign vic_elec as v
v <- vic_elec
summary(v)
```

```
##       Time                          Demand      Temperature
##  Min.   :2012-01-01 00:00:00   Min.   :2858   Min.   : 1.50
##  1st Qu.:2012-09-30 22:52:30   1st Qu.:3969   1st Qu.:12.30
##  Median :2013-07-01 22:45:00   Median :4635   Median :15.40
##  Mean   :2013-07-01 22:45:00   Mean   :4665   Mean   :16.27
##  3rd Qu.:2014-04-01 23:37:30   3rd Qu.:5244   3rd Qu.:19.40
##  Max.   :2014-12-31 23:30:00   Max.   :9345   Max.   :43.20
##       Date             Holiday
##  Min.   :2012-01-01   Mode :logical
##  1st Qu.:2012-09-30   FALSE:51120
##  Median :2013-07-01   TRUE :1488
##  Mean   :2013-07-01
##  3rd Qu.:2014-04-01
##  Max.   :2014-12-31
```

- vic_elec is an object called tsibble in R. A tsibble transforms tidy data frames (tibble) into a multivariate time series. For example, in vic_elec both Demand, Temperature, and Holiday are included in the time series dataframe. These are known as the “Key” variables.

Manipulating the data (examples):

The below are meant to be examples of functions which may be used to subset or alter the data:

```
# Using the distinct function lets us vew the categories of each variable or combinations of var
iables:
v %>% distinct(Holiday)
```

Holiday <lgl>
TRUE
FALSE

2 rows

```
# We can see Holiday is categorical of either TRUE or FALSE
```

```
# we can use the filter function to extract portions of the data; for example lets say we want t
o view only the data with Holiday = TRUE:
vholiday <- v %>% filter(Holiday =='TRUE')
head(vholiday, n=3L)
```

Time <dttm>	**Demand** <dbl>	**Temperature** <dbl>	**Date** <date>	**Holiday** <lgl>
2012-01-01 00:00:00	4382.825	21.40	2012-01-01	TRUE
2012-01-01 00:30:00	4263.366	21.05	2012-01-01	TRUE

Time <dttm>	Demand <dbl>	Temperature <dbl>	Date <date>	Holiday <lgl>
2012-01-01 01:00:00	4048.966	20.70	2012-01-01	TRUE

3 rows

```
tail(vholiday, n=3L)
```

Time <dttm>	Demand <dbl>	Temperature <dbl>	Date <date>	Holiday <lgl>
2014-12-26 22:30:00	3626.037	16.0	2014-12-26	TRUE
2014-12-26 23:00:00	3571.262	15.8	2014-12-26	TRUE
2014-12-26 23:30:00	3579.650	15.7	2014-12-26	TRUE

3 rows

```
# we can add on the select function to subset certain columns in the dataset.  Note the use of t
he pipe operator %>%:
vholiday1 <- v %>% filter(Holiday =='TRUE') %>%
  select(Demand, Time)
head(vholiday1, 3L)
```

Demand <dbl>	Time <dttm>
4382.825	2012-01-01 00:00:00
4263.366	2012-01-01 00:30:00
4048.966	2012-01-01 01:00:00

3 rows

```
# we can add the summarise function to convert the units of Demand to MW/100:

vholiday2 <- v %>% filter(Holiday =='TRUE') %>%
  select(Demand, Time) %>%
  summarise(Demand= (Demand/100))
head(vholiday2)
```

Time <dttm>	Demand <dbl>
2012-01-01 00:00:00	43.82825
2012-01-01 00:30:00	42.63366
2012-01-01 01:00:00	40.48966

	Time <dttm>	Demand <dbl>
	2012-01-01 01:30:00	38.77563
	2012-01-01 02:00:00	40.36230
	2012-01-01 02:30:00	38.65597

6 rows

```
# or the mutate function could be used to add a column with Demand in MW/100:

vholiday3 <- v %>% filter(Holiday =='TRUE') %>%
  select(Demand, Time) %>%
  mutate(Demand1= (Demand/100))
head(vholiday2)
```

	Time <dttm>	Demand <dbl>
	2012-01-01 00:00:00	43.82825
	2012-01-01 00:30:00	42.63366
	2012-01-01 01:00:00	40.48966
	2012-01-01 01:30:00	38.77563
	2012-01-01 02:00:00	40.36230
	2012-01-01 02:30:00	38.65597

6 rows

Time Series Regression Models and Forecasts; There are 2 examples below:

Steps are:

1. Data Preparation
2. Visualize
3. Define a model (Specify)
4. Check the model's performance (Evaluate)
5. Produce Forecasts

Example 1:

Step 1: Data Preparation. Let's alter the structure to summarize the demand for each day so it shows the total demand on a daily basis instead of every 30 mintues. Then, we'll refer to the maximum temperature for each day. Note that total daily demand is divided

by 10000 to facilitate some of the plotting scales. We'll filter the time period to 2012 and 2013 so we can train a model and validate the forecast against the observed values from early 2014:

```
v2 <- vic_elec %>%
  filter(year(Time) == 2013) %>%
 index_by(Date = as_date(Time)) %>%
  summarise(
    Demand = sum(Demand/10000),
    Temperature = max(Temperature),
    Holiday = any(Holiday)
  ) %>%
  mutate(Day_Cat = case_when(
    Holiday ~ "Holiday",
    wday(Date) %in% 2:6 ~ "Weekday",
    TRUE ~ "Weekend"
  ))
head(v2, 7L)
```

Date <date>	Demand <dbl>	Temperature <dbl>	Holiday <lgl>	Day_Cat <chr>
2013-01-01	17.59020	24.5	TRUE	Holiday
2013-01-02	19.56539	21.9	FALSE	Weekday
2013-01-03	24.28752	36.7	FALSE	Weekday
2013-01-04	29.43749	40.6	FALSE	Weekday
2013-01-05	22.05377	34.7	FALSE	Weekend
2013-01-06	19.64079	25.2	FALSE	Weekend
2013-01-07	25.26280	29.3	FALSE	Weekday

7 rows

```
tail(v2, 3L)
```

Date <date>	Demand <dbl>	Temperature <dbl>	Holiday <lgl>	Day_Cat <chr>
2013-12-29	16.81784	20.2	FALSE	Weekend
2013-12-30	18.29488	21.9	FALSE	Weekday
2013-12-31	18.43879	25.1	FALSE	Weekday

3 rows

Step 2. Visualize:

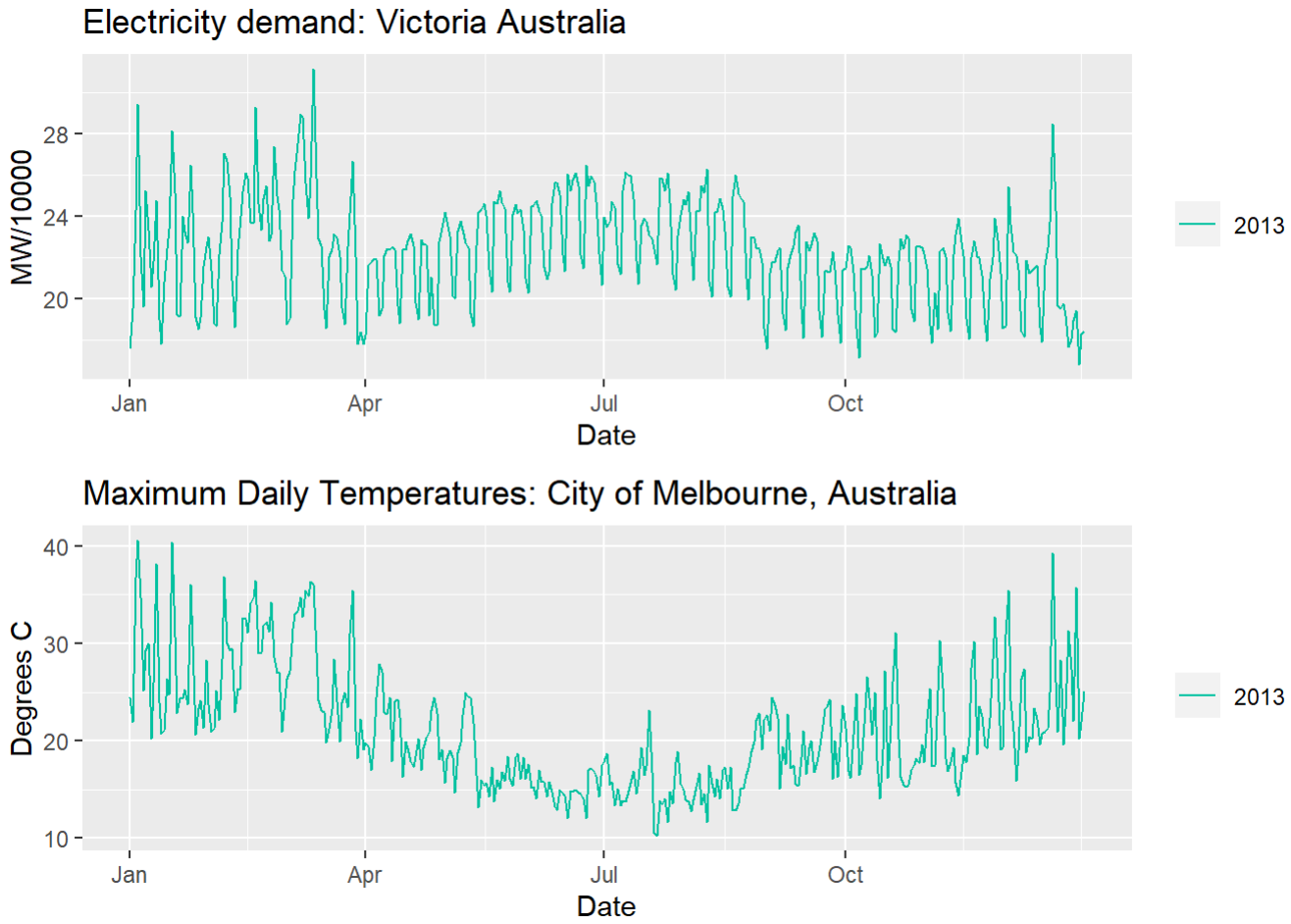
- A look at daily electricity demand and maximum daily temperature over each year:

```
p1 <- v2 %>% gg_season(Demand, period = "year") +
  theme(legend.position = "right") +
  labs(y="MW/10000", title="Electricity demand: Victoria Australia")
p2 <- v2 %>% gg_season(Temperature, period = "year") +
  theme(legend.position = "right") +
  labs(y="Degrees C", title="Maximum Daily Temperatures: City of Melbourne, Australia")
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

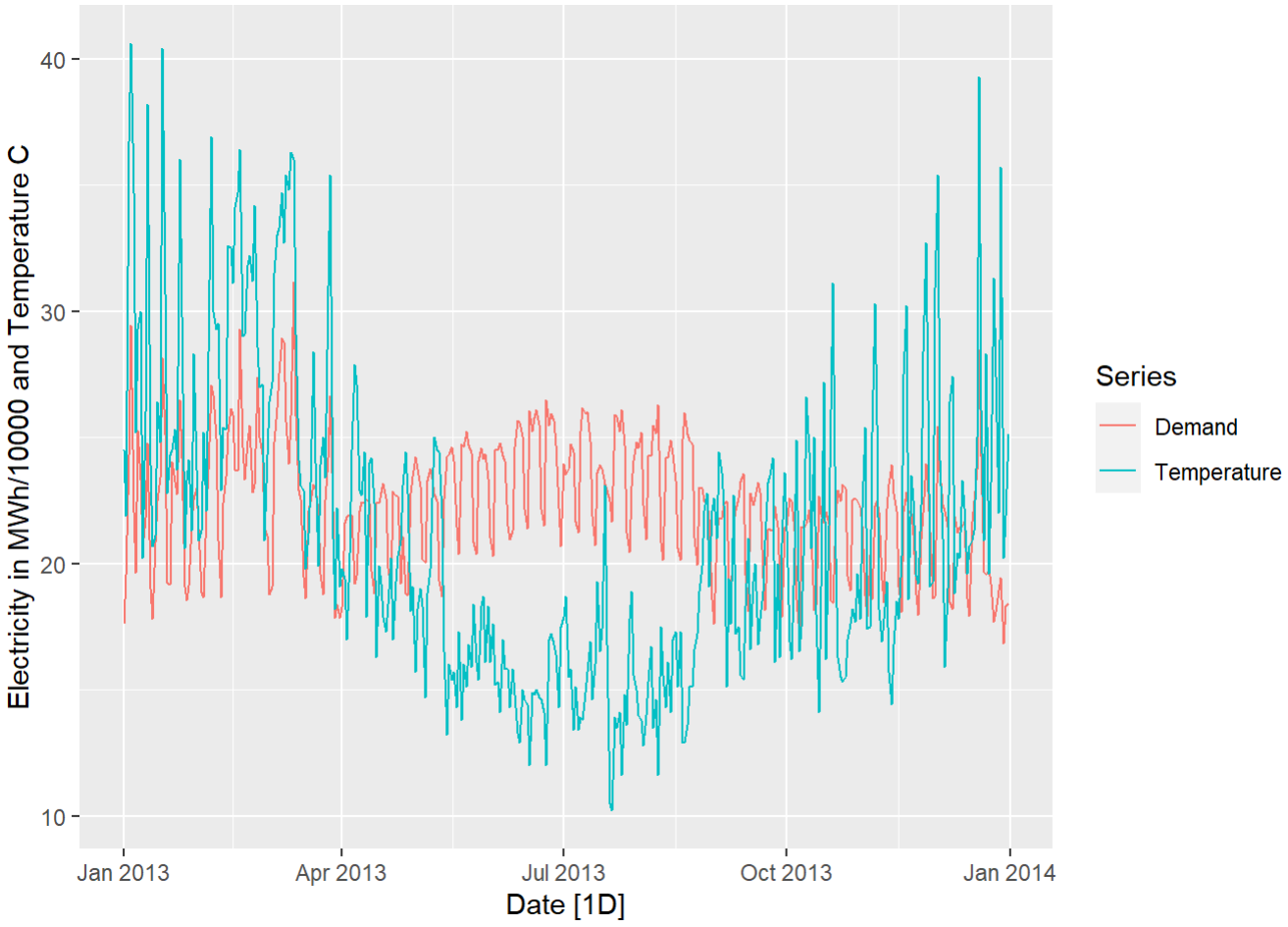
```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
grid.arrange(p1,p2)
```



- A different view:

```
v2 %>%
  pivot_longer(c(Demand, Temperature), names_to="Series") %>%
  autoplot(value) +
  labs(y = "Electricity in MWh/10000 and Temperature C")
```

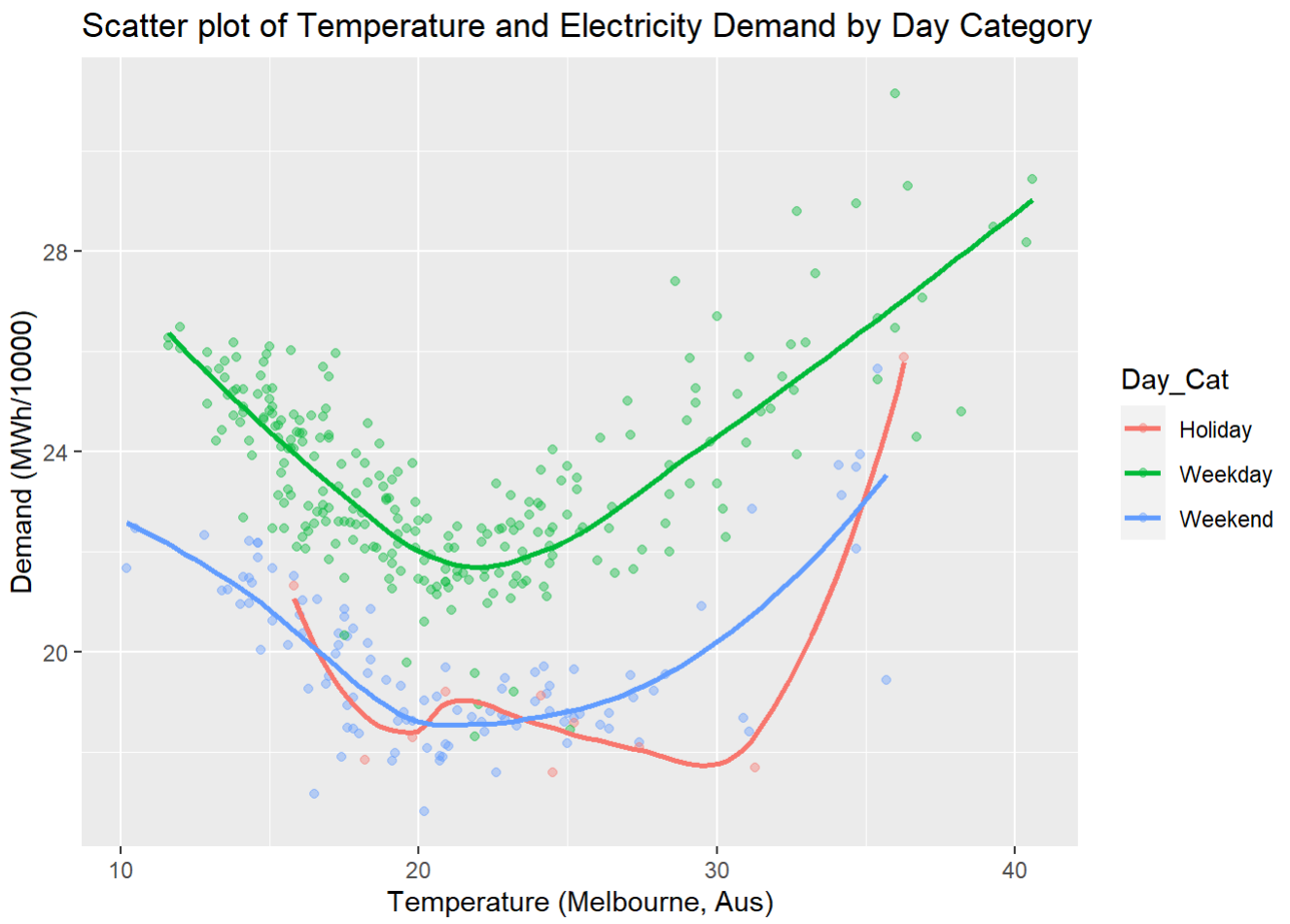


- Here in the Scatter plot we can see the relationship between demand and temperature is non linear: meaning the demand increases as temperature increases but demand also increases for lower temperatures. A linear forecasting model would not be a good fit in this case.

```
# Scatterplot of Demand Changes with Temperature dependent on Day Category:

v2 %>%
  ggplot(aes(x = Temperature, y = Demand, color = Day_Cat)) +
  labs(y = "Demand (MWh/10000)",
       x = "Temperature (Melbourne, Aus)",
       title = 'Scatter plot of Temperature and Electricity Demand by Day Category') +
  geom_point(alpha = 0.4) +
  geom_smooth(method = "loess", se = FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Step 3. Define a model (Specify):

- Since the relationship between Temperature and Demand is non-linear, we will use a quadratic regression model ARIMA. We will use the Day_Cat variable to indicate if working or non work day:

```
v2quad <- v2 %>%
  model(ARIMA((Demand) ~ Temperature + I(Temperature^2) +
                (Day_Cat == "Weekday")))
```

```
report(v2quad)
```

```
## Series: Demand
## Model: LM w/ ARIMA(1,1,2)(1,0,2)[7] errors
## Transformation: (Demand)
##
## Coefficients:
##           ar1     ma1      ma2     sar1    sma1    sma2  Temperature
##       -0.8920  0.6149  -0.3189  -0.6591  0.6950  0.1547      -0.7376
## s.e.   0.0621  0.0863   0.0650   0.2502  0.2592  0.0555       0.0537
##       I(Temperature^2)  Day_Cat == "Weekday"TRUE
##                 0.0173                    3.4176
## s.e.            0.0010                    0.0864
##
## sigma^2 estimated as 0.6071:  log likelihood=-421.43
## AIC=862.87   AICc=863.49   BIC=901.84
```

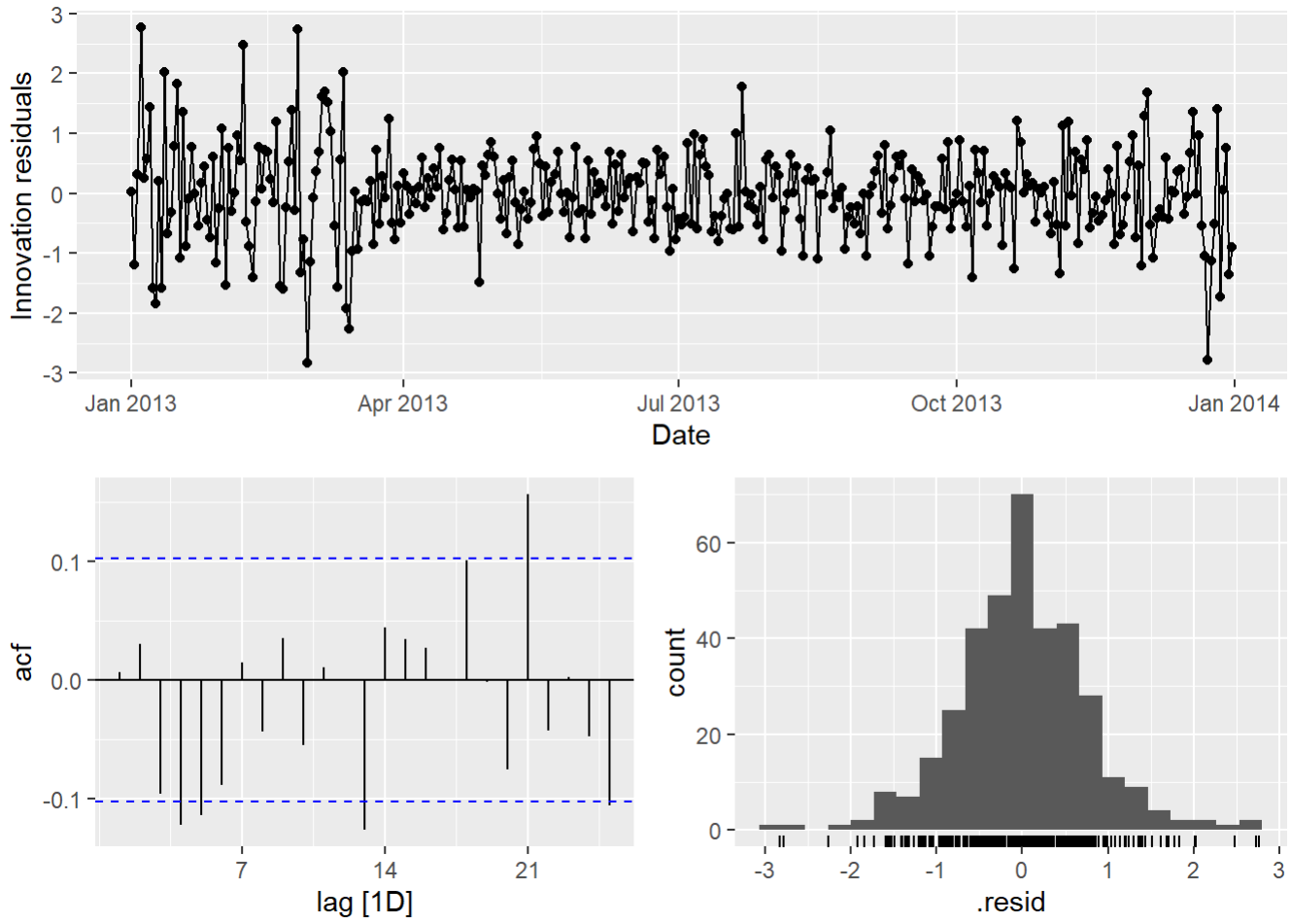
Step 4: Check for accuracy:

```
v2quad %>% accuracy()
```

.model <chr>
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))

1 row | 1-2 of 10 columns

```
v2quad %>% gg_tsresiduals()
```



- The residuals are not homoscedastic with larger variances in early and later months of each year. There is some autocorrelation in the residuals as well as some slight skew which may affect our prediction interval.

Step 5: Forecast.

-Fist, let's structure a tsibble for 2014 observed values of daily demand and maximum Temperature. This will allow us to view the daily Demand and maximum temperature for those days for comparison with the forecast:

```
v2014 <- vic_elec %>%
  filter(year(Time) == 2014) %>%
 index_by(Date = as_date(Time)) %>%
  summarise(
    Demand = sum(Demand/10000),
    Temperature = max(Temperature),
    Holiday = any(Holiday)
  ) %>%
  mutate(Day_Cat = case_when(
    Holiday ~ "Holiday",
    wday(Date) %in% 2:6 ~ "Weekday",
    TRUE ~ "Weekend"
  ))
head(v2014, 7L)
```

Date <date>	Demand <dbl>	Temperature <dbl>	Holiday <lgl>	Day_Cat <chr>
2014-01-01	17.51850	26.0	TRUE	Holiday
2014-01-02	18.83506	23.0	FALSE	Weekday
2014-01-03	18.90856	22.2	FALSE	Weekday
2014-01-04	17.37976	20.3	FALSE	Weekend
2014-01-05	16.97328	26.1	FALSE	Weekend
2014-01-06	19.52411	19.6	FALSE	Weekday
2014-01-07	19.97705	20.0	FALSE	Weekday

7 rows

- Next, lets build a new set using the acutal observed temperatures from the first 7 days of 2014:

```
v3 <- new_data(v2, 7) %>%
  mutate(
    Temperature = c(26,23,22.2,20.3,26.1,19.6,20),
    Holiday = c(TRUE, rep(FALSE, 6)),
    Day_Cat = case_when(
      Holiday ~ "Holiday",
      wday(Date) %in% 2:6 ~ "Weekday",
      TRUE ~ "Weekend"
    )
  )
head(v3, 7L)
```

Date <date>	Temperature <dbl>	Holiday <lgl>	Day_Cat <chr>
2014-01-01	26.0	TRUE	Holiday
2014-01-02	23.0	FALSE	Weekday

Date <date>	Temperature <dbl>	Holiday <lgl>	Day_Cat <chr>
2014-01-03	22.2	FALSE	Weekday
2014-01-04	20.3	FALSE	Weekend
2014-01-05	26.1	FALSE	Weekend
2014-01-06	19.6	FALSE	Weekday
2014-01-07	20.0	FALSE	Weekday

7 rows

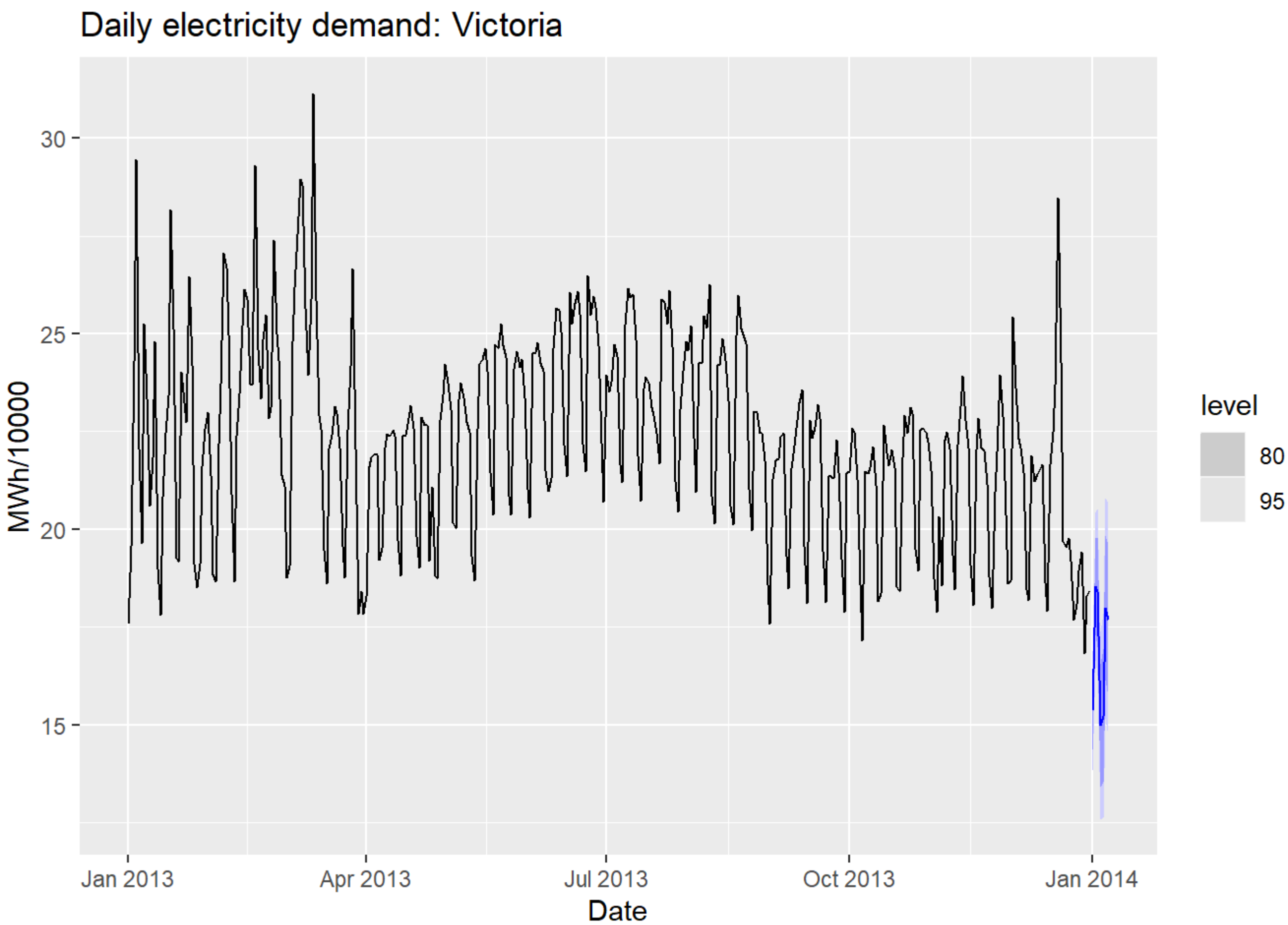
- Next we run the forecast:

```
forecast1 <- forecast(v2quad, v3)
forecast1
```

.model <chr>
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))

7 rows | 1-3 of 7 columns

```
forecast(v2quad, v3) %>%
  autoplot(v2) +
  labs(title="Daily electricity demand: Victoria",
       y="MWh/10000")
```




```
forecast1
```

.model <chr>
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))
ARIMA((Demand) ~ Temperature + I(Temperature^2) + (Day_Cat == \n "Weekday"))

7 rows | 1-3 of 7 columns

```
head(v2014, 7L)
```

Date <date>	Demand <dbl>	Temperature <dbl>	Holiday <lgl>	Day_Cat <chr>

Date <date>	Demand <dbl>	Temperature <dbl>	Holiday <lgl>	Day_Cat <chr>
2014-01-01	17.51850	26.0	TRUE	Holiday
2014-01-02	18.83506	23.0	FALSE	Weekday
2014-01-03	18.90856	22.2	FALSE	Weekday
2014-01-04	17.37976	20.3	FALSE	Weekend
2014-01-05	16.97328	26.1	FALSE	Weekend
2014-01-06	19.52411	19.6	FALSE	Weekday
2014-01-07	19.97705	20.0	FALSE	Weekday

7 rows

- The forecast values are close the the actual observed values for the first 7 days of 2014 though on average the forecast values come out lower than actual. Further correction to the model may be needed or a different model type may be better suited.

Next, lets forecast looking at the time period of a week:

```
# lets adjust the scale of the Demand variable so it is easier to see the variation with Tempera
ture in plots; also lets look at the time series in just one week; Also, let's pick a week witho
ut a holiday so the data is not affected by holidays.  December 16 - 22, 2013 will meet that cri
teria. We will use the data to perform regression modeling on a weekly look:

vweek <-  v %>%  mutate(Demand1= (Demand/100)) %>%
  filter_index('2013-12-16 00:00:00' ~ '2013-12-22 00:00:00') %>%
  mutate(Day_Cat = case_when(
      Holiday ~ "Holiday",
      wday(Date) %in% 2:6 ~ "Weekday",
      TRUE ~ "Weekend")
    )
head(vweek, 7L)
```

Time <dttm>	Demand <dbl>	Temperature <dbl>	Date <date>	Holiday <lgl>	Demand1 <dbl>	Day_Cat <chr>
2013-12-16 00:00:00	4066.209	15.4	2013-12-16	FALSE	40.66209	Weekday
2013-12-16 00:30:00	4183.947	15.6	2013-12-16	FALSE	41.83947	Weekday
2013-12-16 01:00:00	3932.095	15.5	2013-12-16	FALSE	39.32095	Weekday
2013-12-16 01:30:00	3724.803	15.4	2013-12-16	FALSE	37.24803	Weekday
2013-12-16 02:00:00	3576.172	15.4	2013-12-16	FALSE	35.76172	Weekday
2013-12-16 02:30:00	3444.816	15.5	2013-12-16	FALSE	34.44816	Weekday

Time <dttm>	Demand <dbl>	Temperature <dbl>	Date <date>	Holiday <lgl>	Demand1 <dbl>	Day_Cat <chr>
2013-12-16 03:00:00	3323.209	15.5	2013-12-16	FALSE	33.23209	Weekday

7 rows

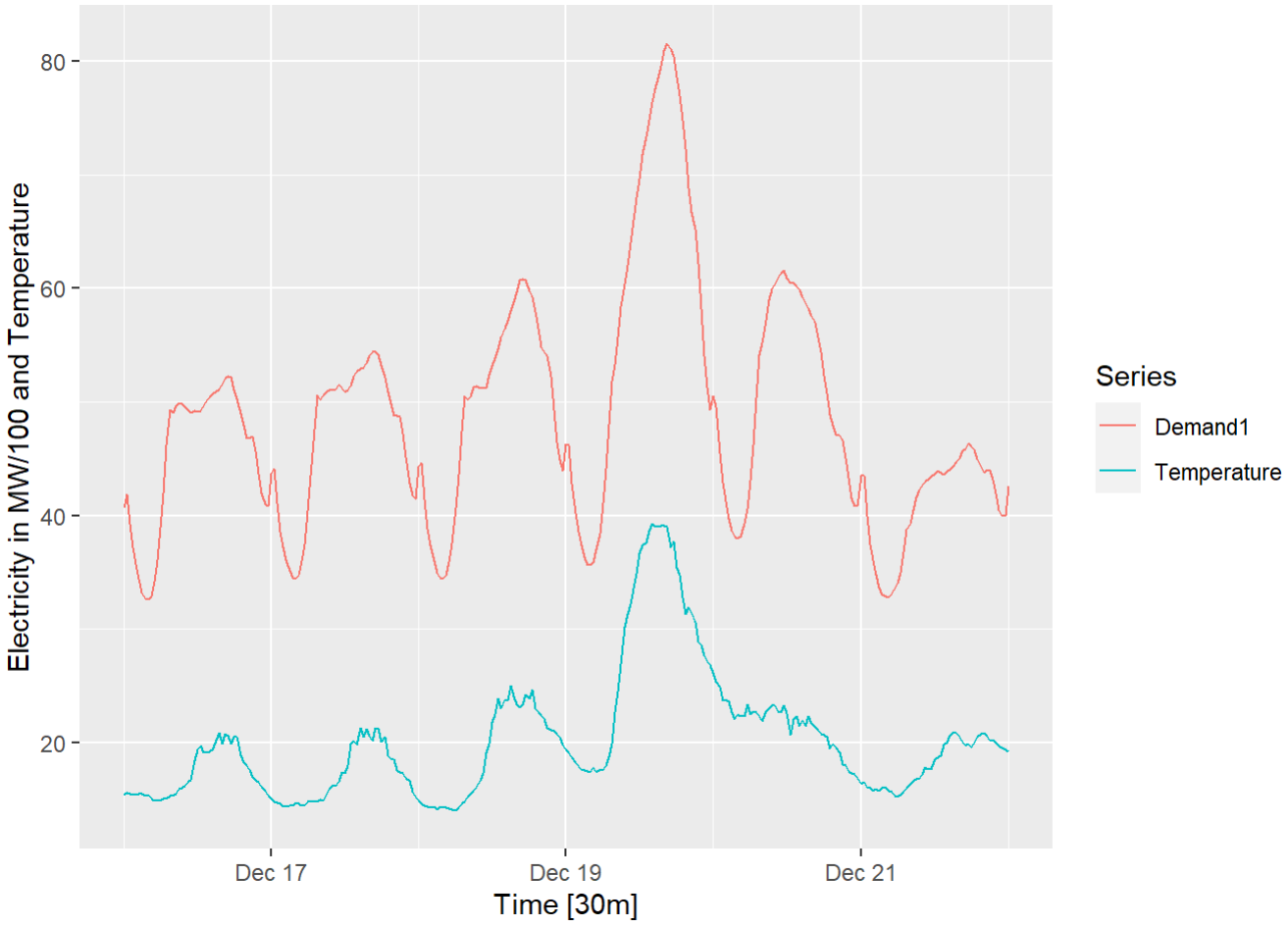
```
tail(vweek)
```

Time <dttm>	Demand <dbl>	Temperature <dbl>	Date <date>	Holiday <lgl>	Demand1 <dbl>	Day_Cat <chr>
2013-12-21 21:30:00	4319.701	20.3	2013-12-21	FALSE	43.19701	Weekend
2013-12-21 22:00:00	4197.135	20.0	2013-12-21	FALSE	41.97135	Weekend
2013-12-21 22:30:00	4052.962	19.7	2013-12-21	FALSE	40.52962	Weekend
2013-12-21 23:00:00	3996.186	19.6	2013-12-21	FALSE	39.96186	Weekend
2013-12-21 23:30:00	3993.594	19.4	2013-12-21	FALSE	39.93594	Weekend
2013-12-22 00:00:00	4260.169	19.2	2013-12-22	FALSE	42.60169	Weekend

6 rows

```
# here's a plot of Demand with Temperature:

vweek %>%
  pivot_longer(c(Demand1, Temperature), names_to="Series") %>%
  autoplot(value) +
  labs(y = "Electricity in MW/100 and Temperature")
```

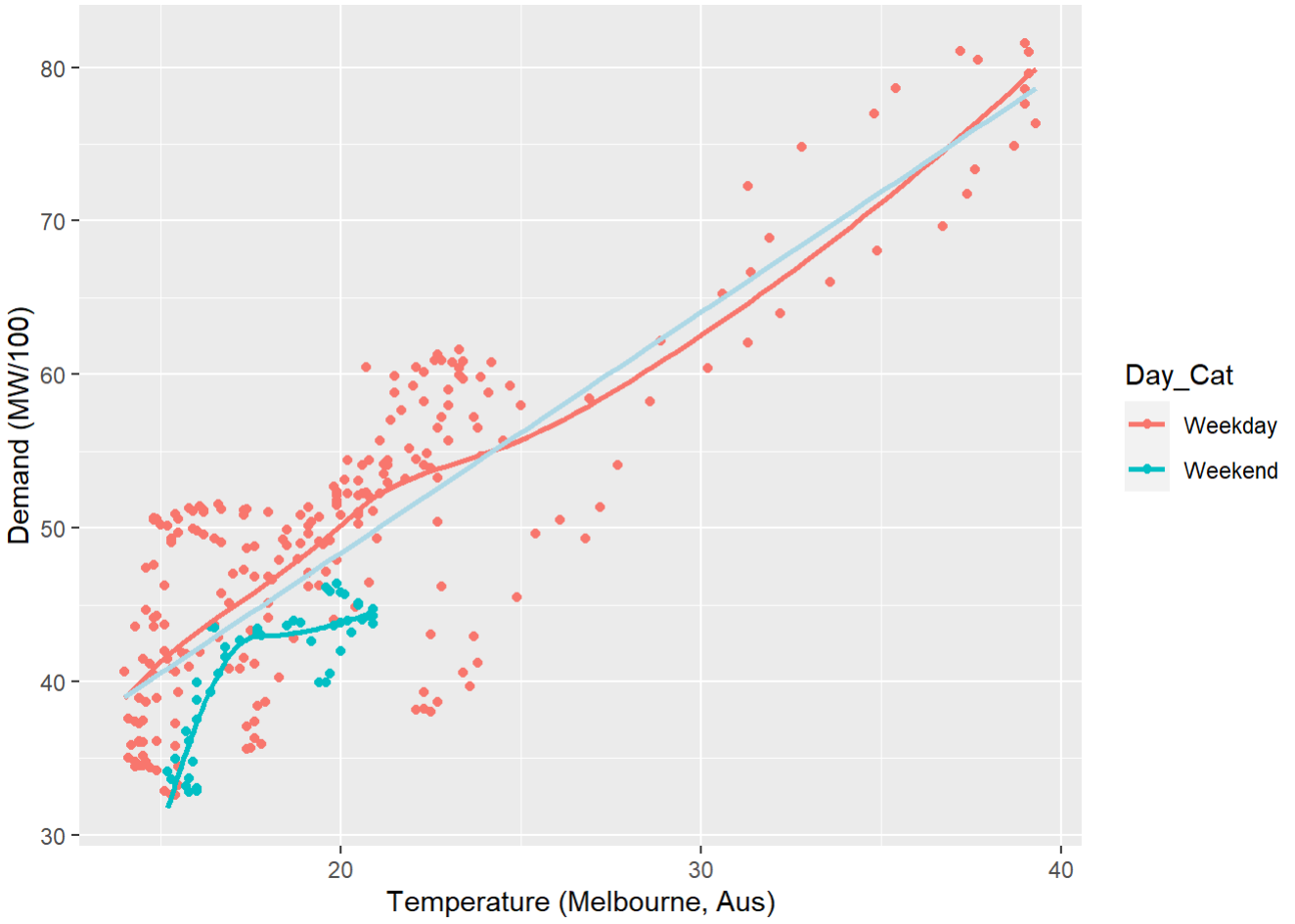


- below is a scatterplot of Temperature and Demand/100. It shows some correlation in that Demand is increased with higher temperatures; Demand decreases as Temperature decreases to a point when Demand rises again with further decrease in Temperature. This rise is probably explained by increase in heating with lower Temperatures. We can see a linear regression line plotted as well.

```
# Scatterplot of Demand Changes with Temperature:

vweek %>%
  ggplot(aes(x = Temperature, y = Demand1, color = Day_Cat)) +
  labs(y = "Demand (MW/100)",
       x = "Temperature (Melbourne, Aus)") +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE) +
  geom_smooth(method = 'lm', se = FALSE, color = 'lightblue')
```

```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```

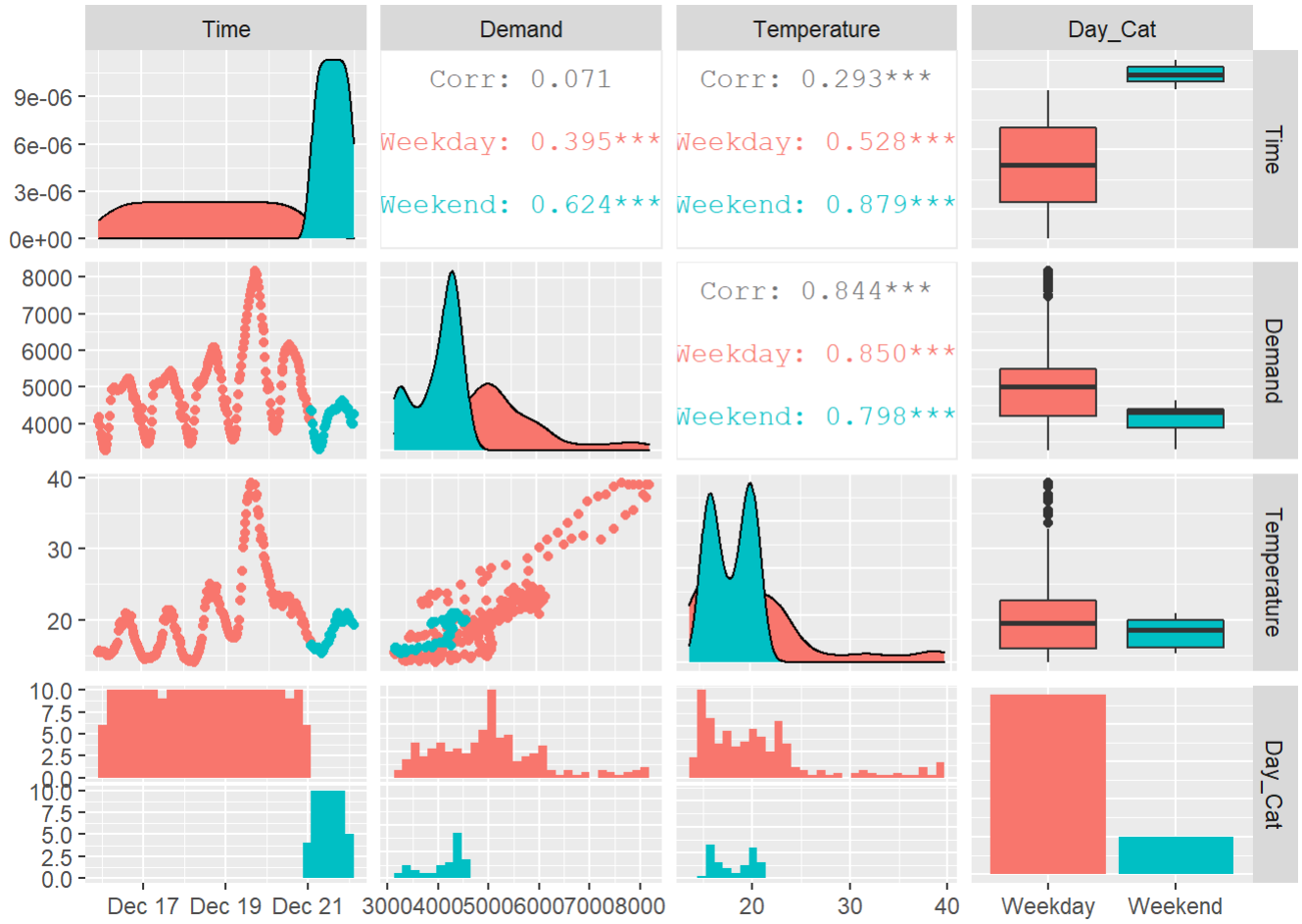


- in the plot below, we can see the Demand is generally higher with weekdays and higher temperatures during this weekly period:

```
vweek %>%
  GGally::ggpairs(columns = c(1:3,7), mapping = aes(color = Day_Cat))
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Fitting a simple linear regression model:

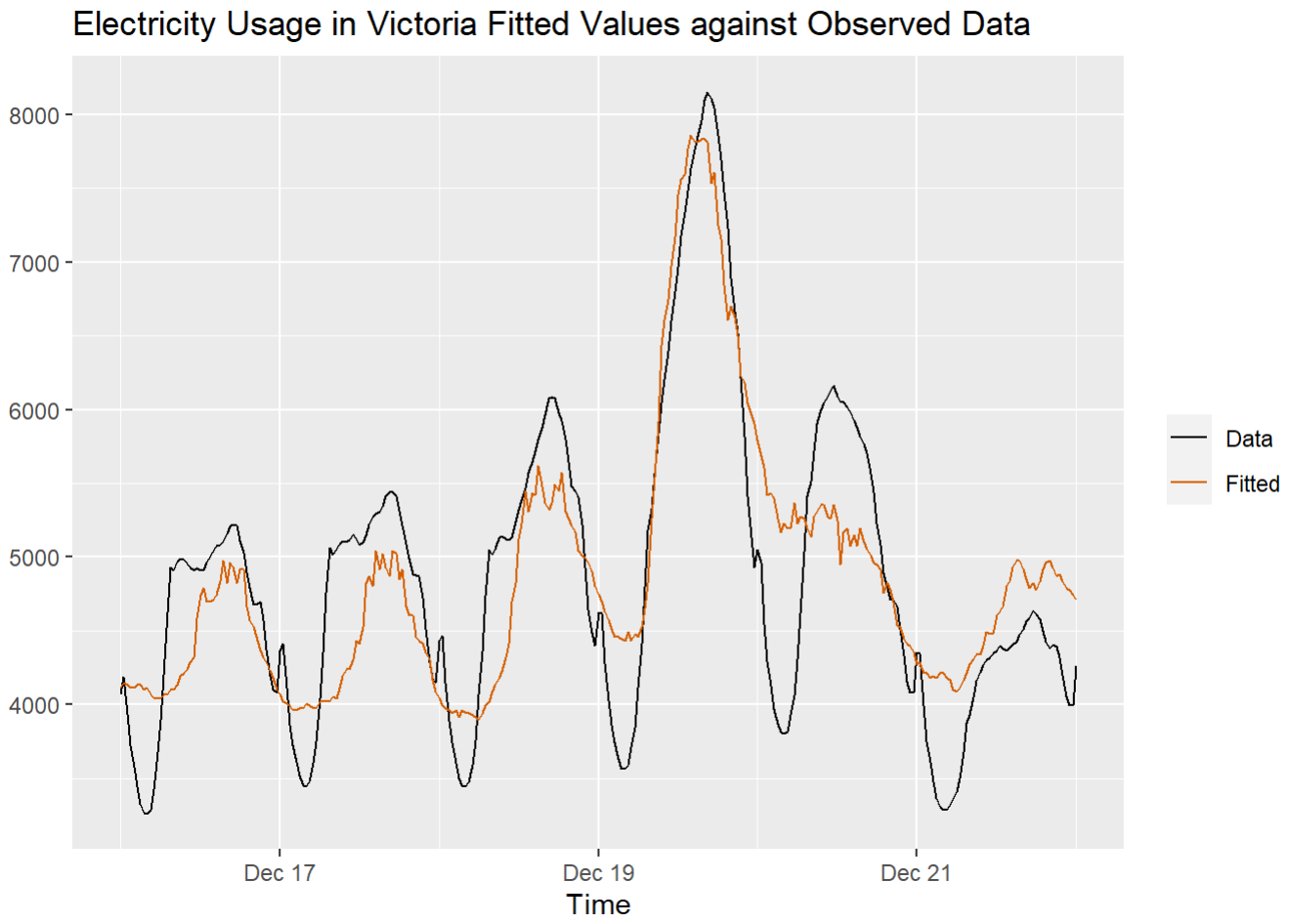
```
fitvweek <- vweek %>%
  model(TSLM(Demand ~ Temperature)) %>%
  report()
```

```
## Series: Demand
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1431.07  -407.93    35.76   398.64  1099.98
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1703.176    123.371   13.80   <2e-16 ***
## Temperature  156.731      5.884   26.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 564.3 on 287 degrees of freedom
## Multiple R-squared: 0.712,   Adjusted R-squared: 0.711
## F-statistic: 709.5 on 1 and 287 DF, p-value: < 2.22e-16
```

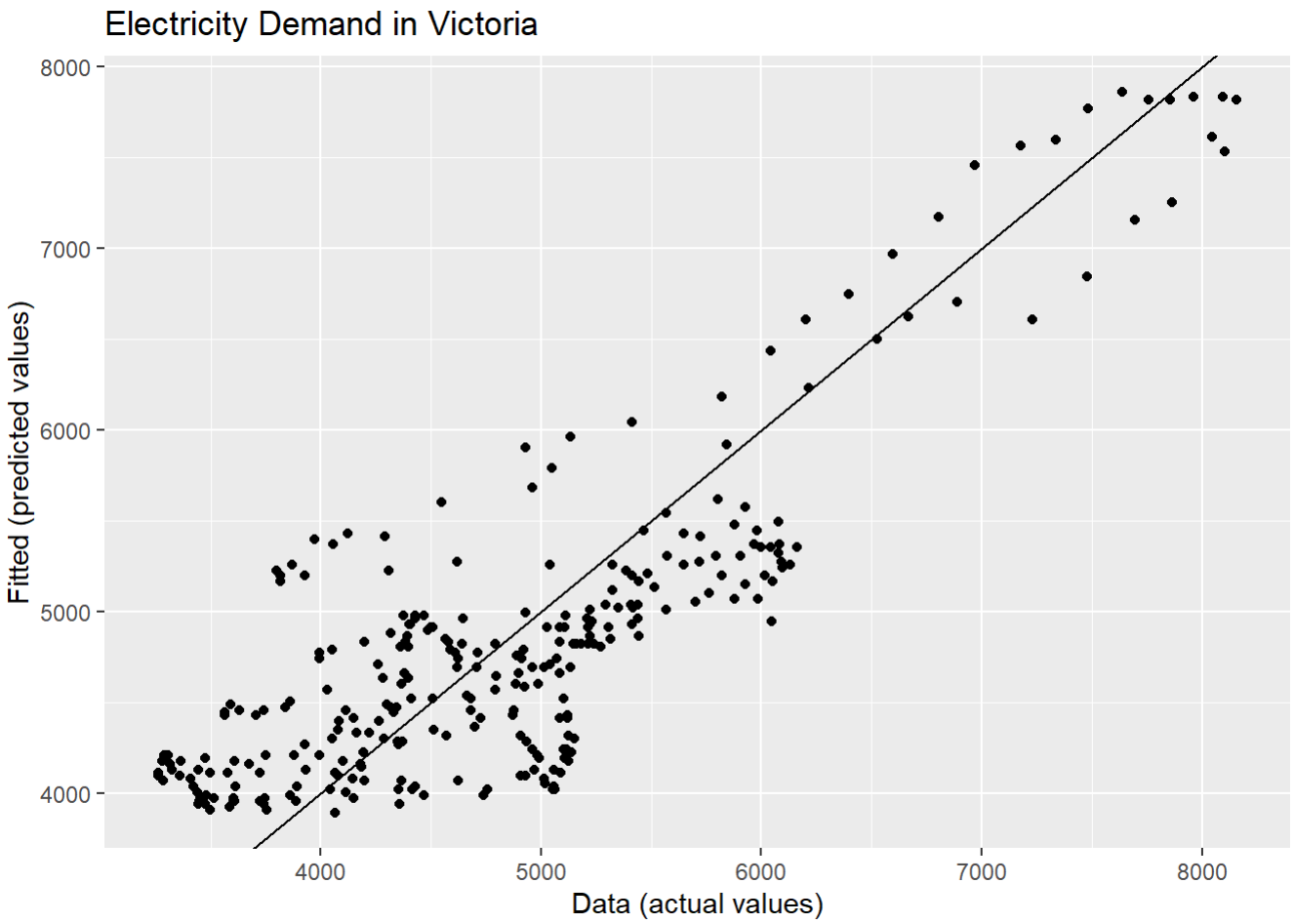
Checking accuracy:

We can see the overfit generally follows the observed data but there is some significant departure:

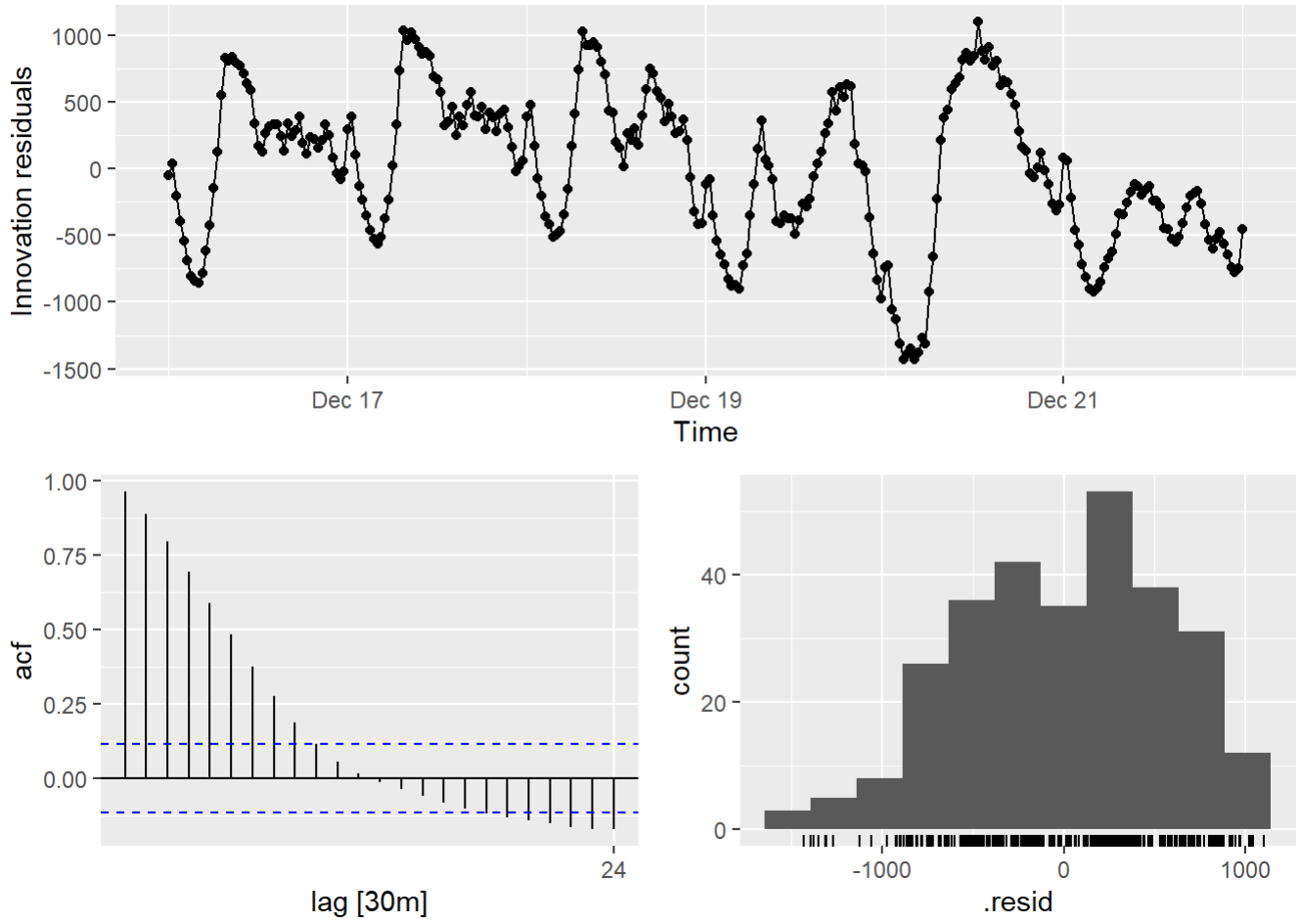
```
augment(fitvweek) %>%
  ggplot(aes(x = Time)) +
  geom_line(aes(y = Demand, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  labs(y = NULL,
    title = "Electricity Usage in Victoria Fitted Values against Observed Data"
  ) +
  scale_colour_manual(values=c(Data="black",Fitted="#D55E00")) +
  guides(colour = guide_legend(title = NULL))
```




```
augment(fitvweek) %>%
  ggplot(aes(x = Demand, y = .fitted)) +
  geom_point() +
  labs(
    y = "Fitted (predicted values)",
    x = "Data (actual values)",
    title = "Electricity Demand in Victoria"
  ) +
  geom_abline(intercept = 0, slope = 1)
```




```
fitvweek %>% gg_tsresiduals()
```




```
fitvweek %>% accuracy()
```

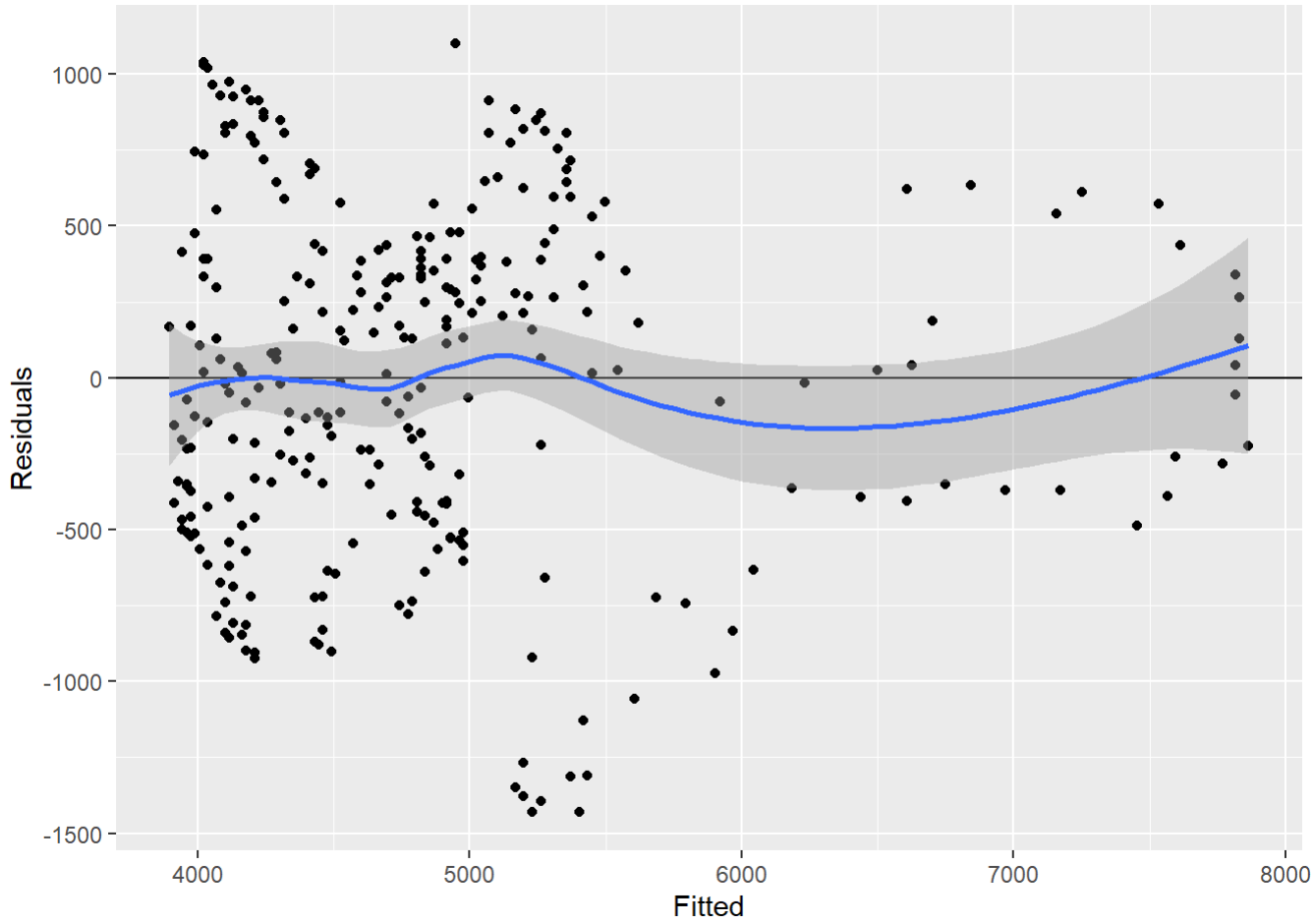
.model	.type	ME	RMSE	MAE	MPE	MAP
<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<db
TSLM(Demand ~ Temperature)	Training	-9.308039e-14	562.3385	468.6004	-1.511917	10.239

1 row | 1-8 of 10 columns

The above shows the MAPE is about 10 percent error

```
augment(fitvweek) %>%
  ggplot(aes(x = .fitted, y = .resid)) +
  geom_point() + labs(x = "Fitted", y = "Residuals")+
  geom_abline(intercept = 0, slope = 0)+
  geom_smooth(method = 'loess')
```

```
## `geom_smooth()` using formula 'y ~ x'
```

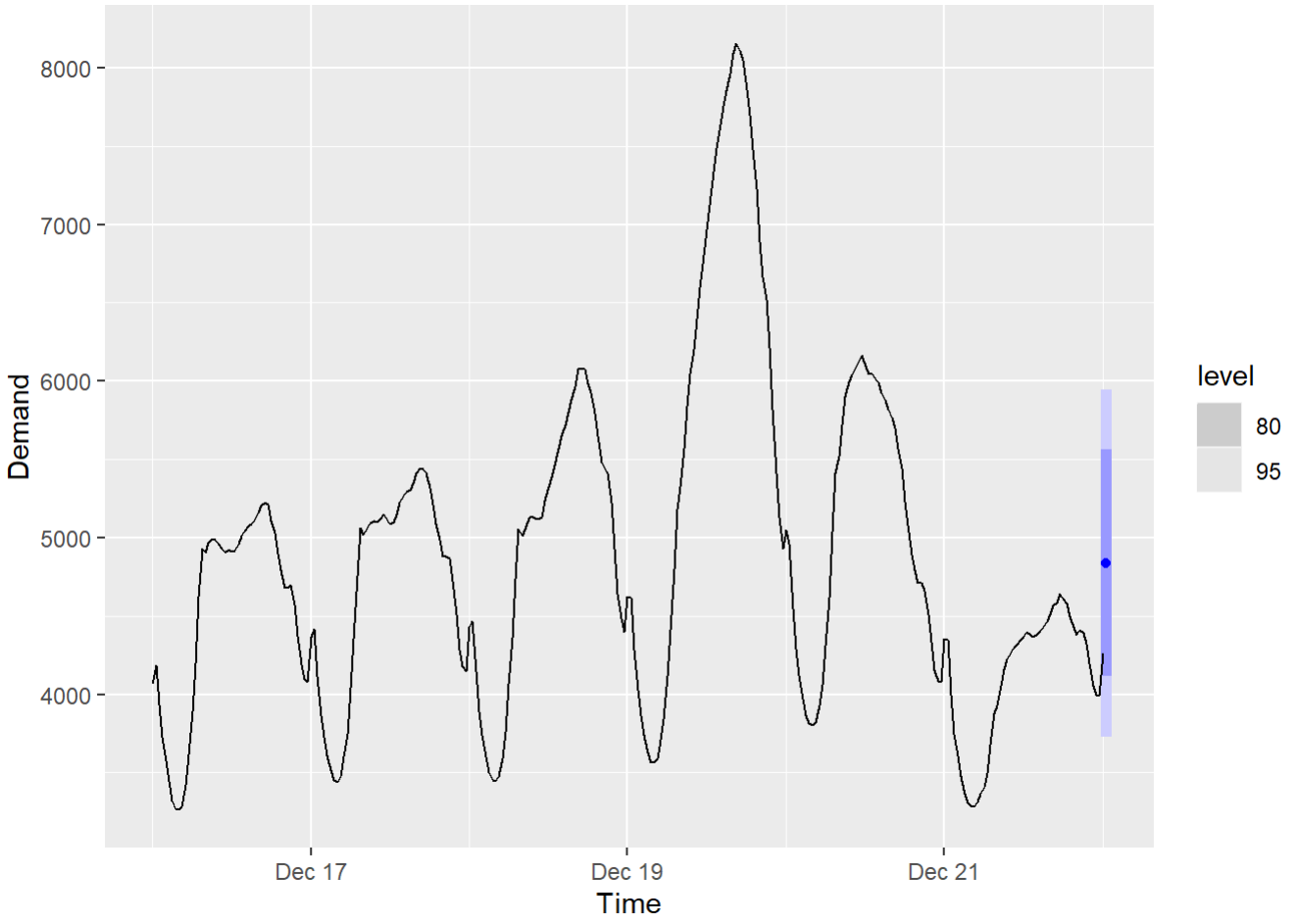


- we see above there is heteroscedasticity in the residuals so the model is skewed.

Forecast

- lets forecast the next 30 minute period if the temperature is 20 degrees:

```
vweek %>%
  model(TSLM(Demand ~ Temperature)) %>%
  forecast(
    new_data(vweek, 1) %>%
      mutate(Temperature = 20)
  ) %>%
  autoplot(vweek)
```


8000
7000
6000
5000
4000
Demand
Dec 17
Dec 19
Dec 21
Time
level
80
95