Selvalakshmi_TS

Selvalakshmi

2024-08-17

```
# Install and load the necessary packages
install.packages("forecast")
```

```
## Installing package into 'C:/Users/selva/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
```

```
## package 'forecast' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\selva\AppData\Local\Temp\RtmpmQIzjw\downloaded_packages
```

```
install.packages("tidyr")
```

```
## Installing package into 'C:/Users/selva/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
```

```
## package 'tidyr' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\selva\AppData\Local\Temp\RtmpmQIzjw\downloaded_packages
```

```
install.packages("readxl")
```

```
## Installing package into 'C:/Users/selva/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
```

```
## package 'readxl' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\selva\AppData\Local\Temp\RtmpmQIzjw\downloaded_packages
```

```
install.packages("rmarkdown")
```

```
## Installing package into 'C:/Users/selva/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
```

```
##
##   There is a binary version available but the source version is later:
##           binary source needs_compilation
## rmarkdown   2.27   2.28             FALSE
```

```
## installing the source package 'rmarkdown'
```

```
#install.packages("tinytex")
#tinytex::install_tinytex()
library(fpp2)
```

```
## Warning: package 'fpp2' was built under R version 4.3.3
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## -- Attaching packages --------------------------------------------- fpp2 2.5 --
```

```
## v ggplot2   3.5.1      v fma       2.5
## v forecast  8.23.0     v expsmooth 2.3
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Warning: package 'forecast' was built under R version 4.3.3
```

```
## Warning: package 'fma' was built under R version 4.3.3
```

```
## Warning: package 'expsmooth' was built under R version 4.3.3
```

```
##
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3
```

```
## Warning: package 'tidyr' was built under R version 4.3.3
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(forecast)
library(ggplot2)
library(xts)
```

```
## Warning: package 'xts' was built under R version 4.3.3
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 4.3.3
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
##
## ############################ Warning from 'xts' package ##########################
## #                                                                                #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                               #
## #                                                                                #
```

```
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                             #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                             #
## ###############################################################################
##
## Attaching package: 'xts'
##
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
library(dplyr)
library(tidyr)
library(openxlsx)
```

```
## Warning: package 'openxlsx' was built under R version 4.3.3
```

```
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.3.3
```

```
library(stats)
```

```
options(repos = c(CRAN = "https://cran.r-project.org"))
```

```
install.packages("forecast")
```

```
## Warning: package 'forecast' is in use and will not be installed
```

```
#loading data set
setwd("C:/Users/selva/Documents/DSTI/TimeSeries/Exam/")
my_data = read_excel("2023-11-Elec-train.xlsx")
View(my_data)
```

```
names(my_data)
```

```
## [1] "Timestamp"  "Power (kW)" "Temp (C°)"
```

```
#rename columns
names(my_data) <- c("Timestamp", "power", "temp")
names(my_data)
```

```
## [1] "Timestamp" "power"     "temp"
```

```
#Explore data
str(my_data)
```

```
## tibble [4,987 x 3] (S3: tbl_df/tbl/data.frame)
##  $ Timestamp: chr [1:4987] "40179.052083333336" "1/1/2010 1:30" "1/1/2010 1:45" "1/1/2010 2:00" ...
##  $ power    : num [1:4987] 165 152 147 154 154 ...
##  $ temp     : num [1:4987] 10.6 10.6 10.6 10.6 10.6 ...
```

```
summary(my_data)
```

```
##   Timestamp             power            temp
##  Length:4987        Min.   :  0.0   Min.   : 3.889
```

```
##  Class :character   1st Qu.:162.9   1st Qu.: 9.444
##  Mode  :character   Median :253.0   Median :11.111
##                     Mean   :230.8   Mean   :10.946
##                     3rd Qu.:277.3   3rd Qu.:12.778
##                     Max.   :355.1   Max.   :19.444
##                     NA's   :96
```

```
head(my_data)
```

```
## # A tibble: 6 x 3
##   Timestamp          power  temp
##   <chr>              <dbl> <dbl>
## 1 40179.052083333336  165.  10.6
## 2 1/1/2010 1:30       152.  10.6
## 3 1/1/2010 1:45       147.  10.6
## 4 1/1/2010 2:00       154.  10.6
## 5 1/1/2010 2:15       154.  10.6
## 6 1/1/2010 2:30       159   10.6
```

```
dim(my_data)
```

```
## [1] 4987    3
```

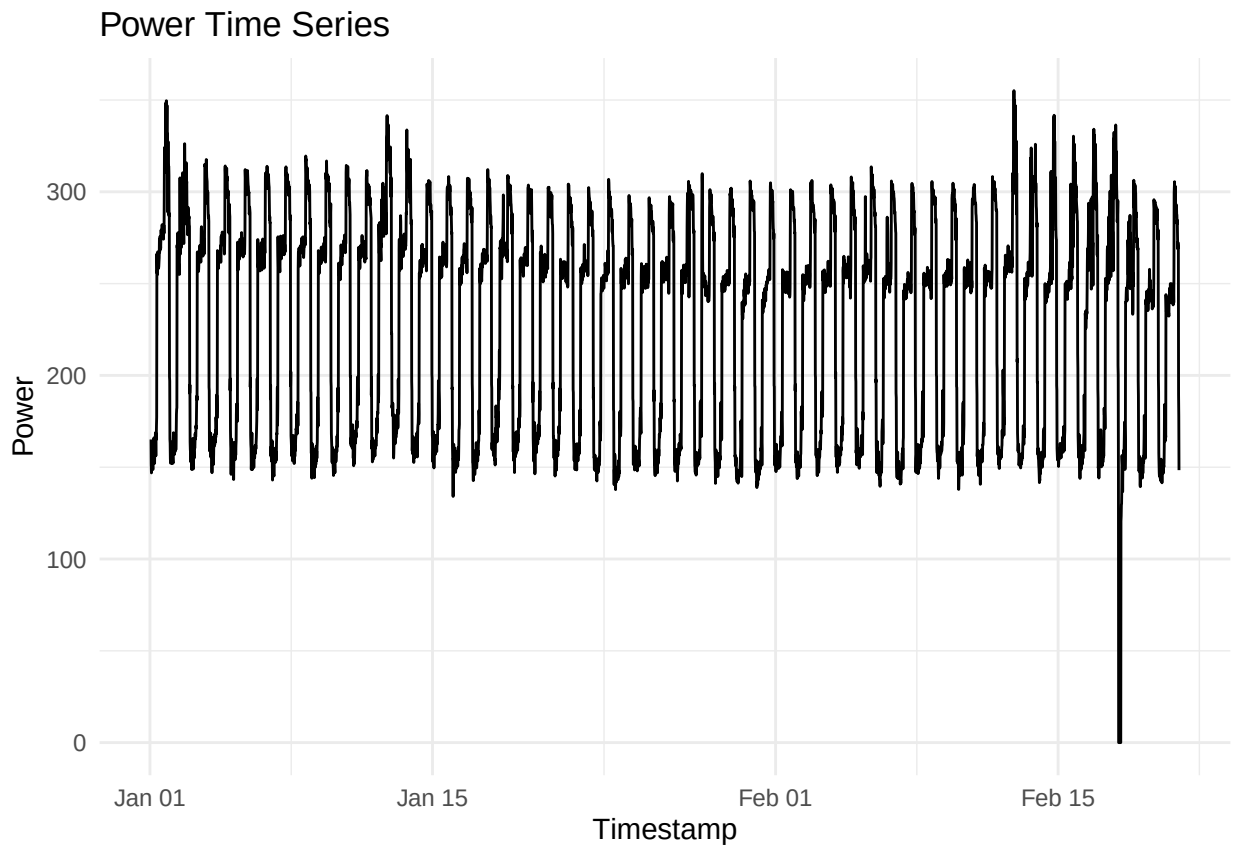
```
#Data wrangling
#Conversion field timeStamp in date format
# Define start and end dates
start.date <- as.POSIXct("2010-01-01 01:15", format = "%Y-%m-%d %H:%M")
end.date <- as.POSIXct("2010-02-21 23:45", format = "%Y-%m-%d %H:%M")

# Generate the sequence of dates
my_date <- seq(start.date, end.date, by = "15 min")

# Assign the sequence to your data frame
my_data$Timestamp <- my_date

# Check the structure of your data frame
str(my_data)
```

```
## tibble [4,987 x 3] (S3: tbl_df/tbl/data.frame)
##  $ Timestamp: POSIXct[1:4987], format: "2010-01-01 01:15:00" "2010-01-01 01:30:00" ...
##  $ power    : num [1:4987] 165 152 147 154 154 ...
##  $ temp     : num [1:4987] 10.6 10.6 10.6 10.6 10.6 ...
```

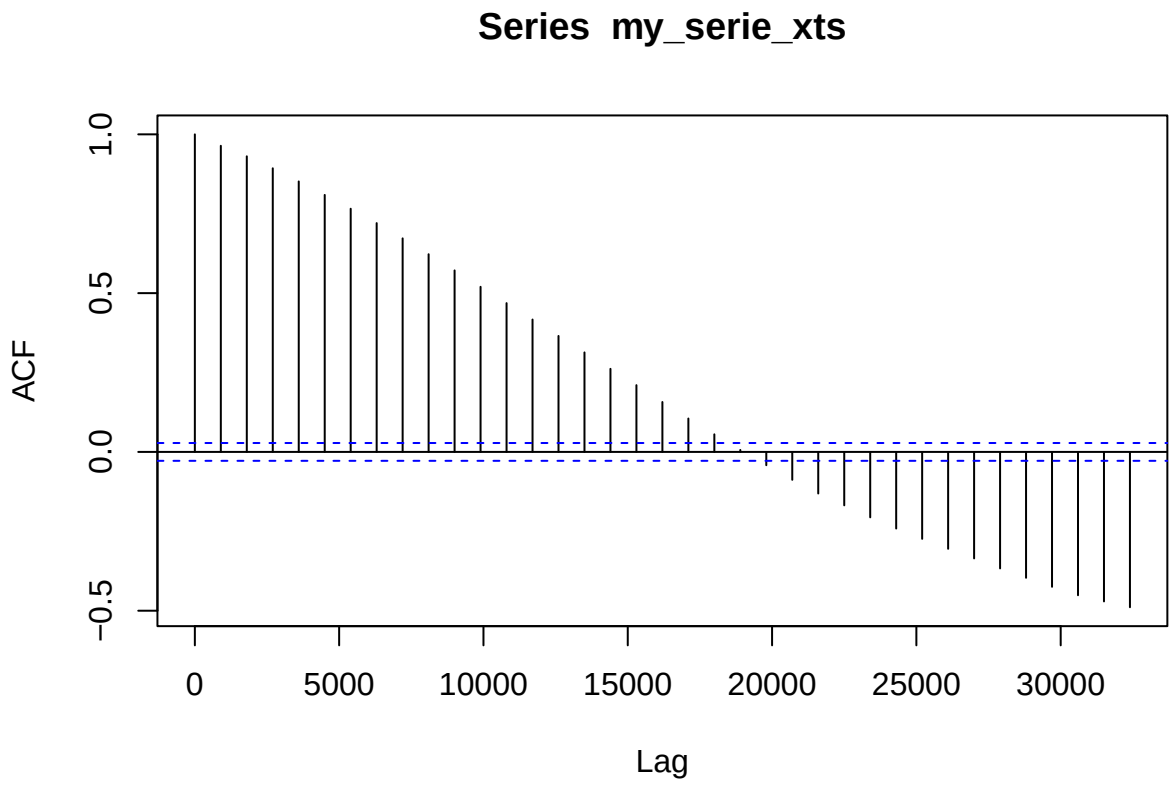
```
#Data splitting between :
#data1 used to train and test forecasting
#data2 used for forecasting

# Remove rows with any NA values
data1 <- my_data %>% drop_na()

# Filter rows where 'power' is NA
data2 <- my_data %>% filter(is.na(power))
```

```
#Time serie creation and visualization. Serie seems to have a period of 24 hours

# Create the time series object
my_serie <- ts(data1$power, start = min(data1$Timestamp), frequency = 24 * 60 / 15)
```

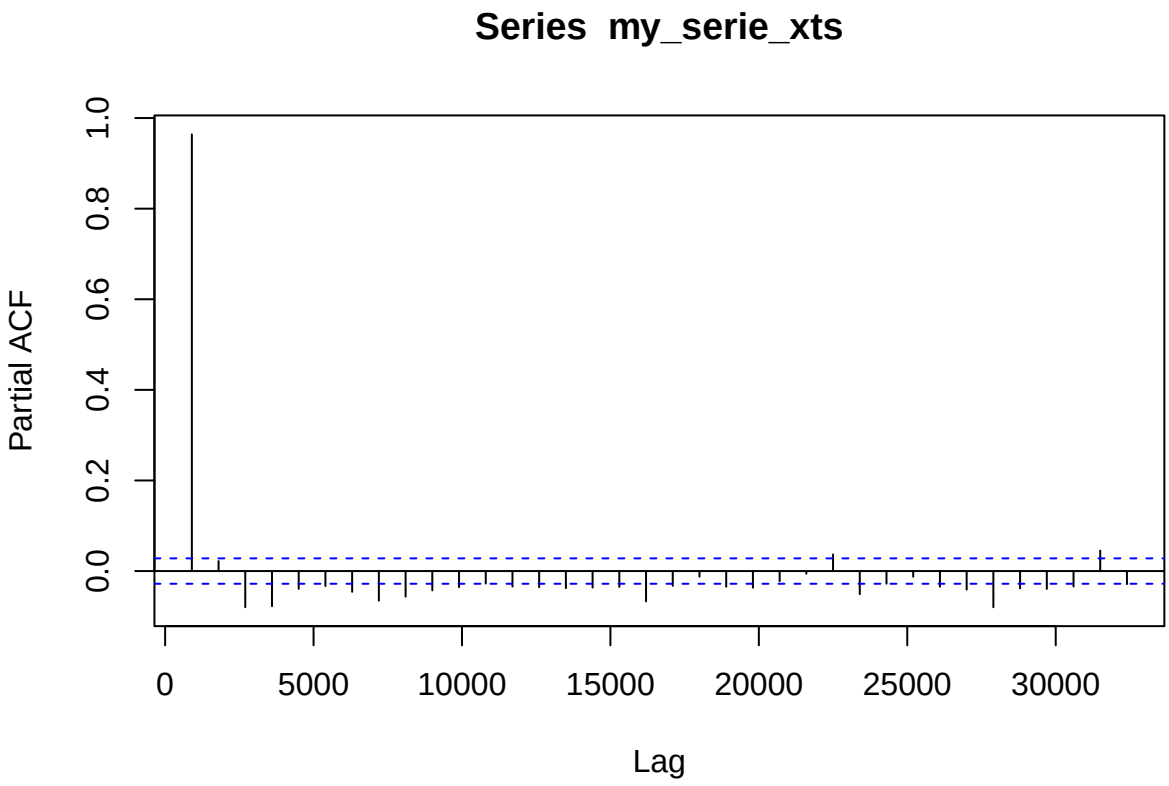
```r
# Create the xts object for better visualization
my_serie_xts <- xts(data1$power, order.by = data1$Timestamp)

# Plot the xts object using ggplot2 for enhanced visualization
autoplot(my_serie_xts) +
  labs(title = "Power Time Series",
       x = "Timestamp",
       y = "Power") +
  theme_minimal()
```

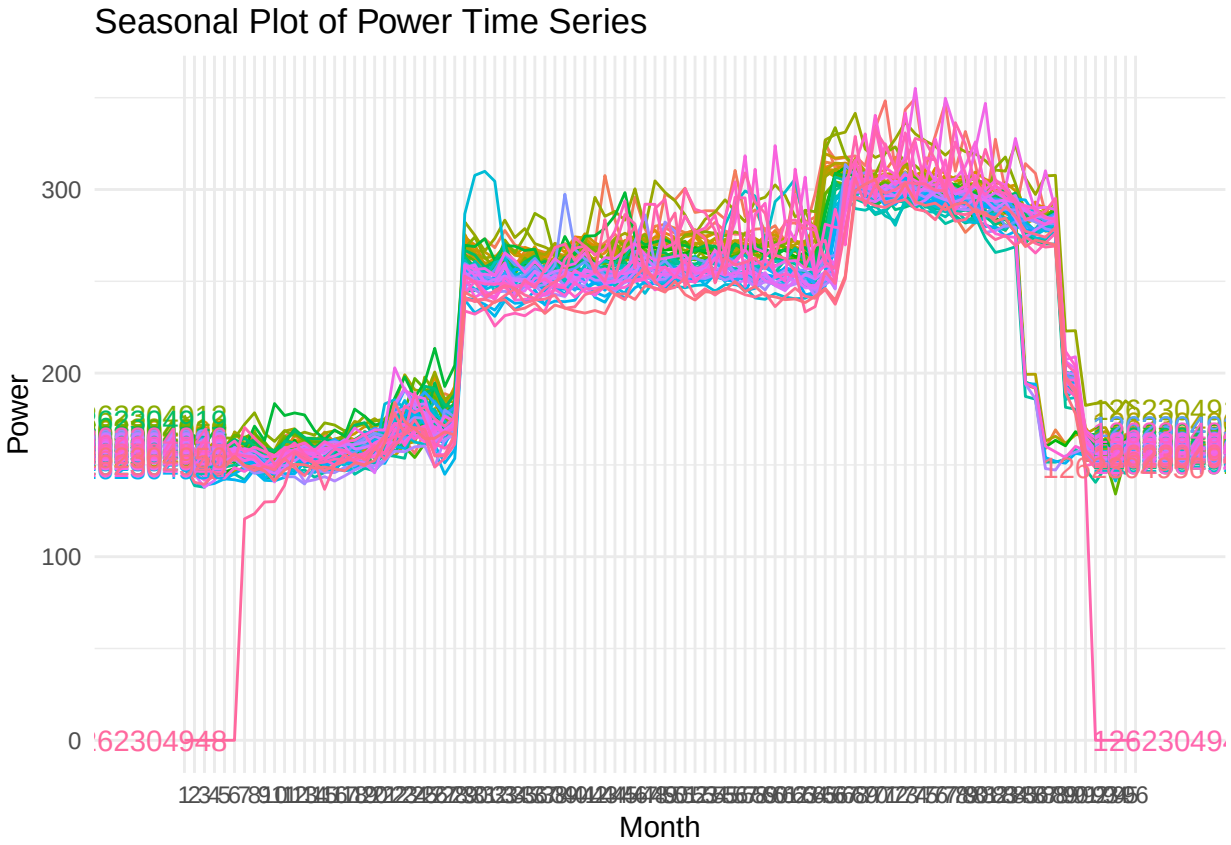



```r
acf(my_serie_xts)
```




```
pacf(my_serie_xts)
```


Series my_serie_xts
Partial ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5000
10000
15000
20000
25000
30000
Lag




```
#Some statistic for time serie
mean(my_serie)
```

```
## [1] 230.7694
```

```
var(my_serie)
```

```
## [1] 3422.982
```

```
# Calculate the ACF without plotting
my_acf <- acf(my_serie, type = "cor", plot = FALSE)

# Extract the first 10 ACF values
acf_values <- my_acf$acf[1:10, 1, 1]

# Print the ACF values
print(acf_values)
```

```
##  [1] 1.0000000 0.9640109 0.9308839 0.8932234 0.8518503 0.8093465 0.7658229
##  [8] 0.7205684 0.6725523 0.6226128
```

It appears that there is no significant long-term (linear) trend in the data. However, a clear seasonal pattern is evident, as shown by the seasonal plot and the autocorrelation table. We estimate a 24-hour periodicity for the series. Notably, there are periods of peak consumption between 16:00 and 23:00, and periods of lower consumption from 23:00 to 06:00. Additionally, we observed some cyclical patterns that can be further analyzed using spectral analysis with Fast Fourier Transform (FFT).

For model evaluation, we will split the data into training and test sets. Approximately 80% of the data will be used for training to estimate model parameters, while the most recent 20% will be reserved for testing to

assess forecast accuracy.

```
# Define the number of observations for the test set
test_size <- 901

# Split the time series
my_serie_test <- tail(my_serie, test_size)
my_serie_train <- head(my_serie, length(my_serie) - test_size)

# Print the lengths of the training and test sets to verify
cat("Training set length:", length(my_serie_train), "\n")
```

```
## Training set length: 3990
```

```
cat("Test set length:", length(my_serie_test), "\n")
```

```
## Test set length: 901
```

Building and evaluating models We will use several models and compare them to select the best one

```
# Holt's linear trend method without damping
fit_holt <- holt(my_serie_train, h = 901, damped = FALSE)

# Holt's linear trend method with damping
fit_holt_damped <- holt(my_serie_train, h = 901, damped = TRUE)

# Auto ARIMA model
fit_arima <- auto.arima(my_serie_train)
forecast_arima <- forecast(fit_arima, h = 901)

# Holt-Winters method with additive seasonality
fit_hw <- HoltWinters(my_serie_train, seasonal = "additive")
forecast_hw <- predict(fit_hw, n.ahead = 901)

# Print summaries of the fitted models
summary(fit_holt)
```

```
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
## holt(y = my_serie_train, h = 901, damped = FALSE)
##
##   Smoothing parameters:
##     alpha = 0.9805
##     beta  = 1e-04
##
##   Initial states:
##     l = 156.3712
##     b = 0.02
##
##   sigma:  14.8793
##
##      AIC     AICc      BIC
```

```
## 54635.06 54635.08 54666.52
##
## Error measures:
##                        ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.002106401 14.87188 6.843205 -0.2285353 3.165913 0.8651569
##                       ACF1
## Training set -0.001962508
##
## Forecasts:
##                 Point Forecast        Lo 80    Hi 80         Lo 95     Hi 95
## 1262304941.562       255.1777  236.1090783 274.2464  226.01474642  284.3407
## 1262304941.573       255.1986  228.4921781 281.9051  214.35464484  296.0426
## 1262304941.583       255.2195  222.6169654 287.8220  205.35822427  305.0808
## 1262304941.594       255.2404  217.6546430 292.8261  197.75794914  312.7228
## 1262304941.604       255.2613  213.2788690 297.2437  191.05472217  319.4678
## 1262304941.615       255.2822  209.3210054 301.2433  184.99063416  325.5737
## 1262304941.625       255.3031  205.6804258 304.9257  179.41179007  331.1943
## 1262304941.635       255.3239  202.2913717 308.3565  174.21762102  336.4303
## 1262304941.646       255.3448  199.1080702 311.5816  169.33812353  341.3516
## 1262304941.656       255.3657  196.0971416 314.6343  164.72224763  346.0092
## 1262304941.667       255.3866  193.2333559 317.5399  160.33140724  350.4418
## 1262304941.677       255.4075  190.4970925 320.3179  156.13559543  354.6794
## 1262304941.688       255.4284  187.8727358 322.9841  152.11093021  358.7459
## 1262304941.698       255.4493  185.3476165 325.5510  148.23803542  362.6605
## 1262304941.708       255.4702  182.9112873 328.0291  144.50093345  366.4394
## 1262304941.719       255.4911  180.5550132 330.4271  140.88626518  370.0959
## 1262304941.729       255.5120  178.2714025 332.7525  137.38272588  373.6412
## 1262304941.740       255.5328  176.0541341 335.0115  133.98064849  377.0850
## 1262304941.750       255.5537  173.8977526 337.2097  130.67168958  380.4358
## 1262304941.760       255.5746  171.7975107 339.3517  127.44858873  383.7006
## 1262304941.771       255.5955  169.7492469 341.4418  124.30498147  386.8860
## 1262304941.781       255.6164  167.7492889 343.4835  121.23525171  389.9975
## 1262304941.792       255.6373  165.7943771 345.4802  118.23441410  393.0402
## 1262304941.802       255.6582  163.8816021 347.4347  115.29801909  396.0183
## 1262304941.812       255.6791  162.0083542 349.3498  112.42207566  398.9361
## 1262304941.823       255.7000  160.1722822 351.2276  109.60298778  401.7969
## 1262304941.833       255.7208  158.3712586 353.0704  106.83750177  404.6042
## 1262304941.844       255.7417  156.6033509 354.8801  104.12266234  407.3608
## 1262304941.854       255.7626  154.8667979 356.6584  101.45577566  410.0695
## 1262304941.865       255.7835  153.1599886 358.4070   98.83437804  412.7326
## 1262304941.875       255.8044  151.4814452 360.1274   96.25620940  415.3526
## 1262304941.885       255.8253  149.8298081 361.8208   93.71919037  417.9314
## 1262304941.896       255.8462  148.2038230 363.4885   91.22140278  420.4710
## 1262304941.906       255.8671  146.6023301 365.1318   88.76107263  422.9731
## 1262304941.917       255.8880  145.0242540 366.7517   86.33655542  425.4394
## 1262304941.927       255.9088  143.4685956 368.3491   83.94632323  427.8714
## 1262304941.938       255.9297  141.9344249 369.9250   81.58895353  430.2705
## 1262304941.948       255.9506  140.4208740 371.4804   79.26311927  432.6381
## 1262304941.958       255.9715  138.9271321 373.0159   76.96758011  434.9754
## 1262304941.969       255.9924  137.4524398 374.5324   74.70117472  437.2836
## 1262304941.979       256.0133  135.9960848 376.0305   72.46281391  439.5638
## 1262304941.990       256.0342  134.5573981 377.5110   70.25147451  441.8169
## 1262304942.000       256.0551  133.1357504 378.9744   68.06619389  444.0439
## 1262304942.010       256.0760  131.7305485 380.4214   65.90606507  446.2459
```

```
## 1262304950.458       273.0170 -297.6558188 843.6899 -599.75178297 1145.7859
## 1262304950.469       273.0379 -297.9968208 844.0727 -600.28435857 1146.3602
## 1262304950.479       273.0588 -298.3376614 844.4553 -600.81668738 1146.9343
## 1262304950.490       273.0797 -298.6783410 844.8378 -601.34876989 1147.5082
## 1262304950.500       273.1006 -299.0188598 845.2201 -601.88060657 1148.0818
## 1262304950.510       273.1215 -299.3592183 845.6022 -602.41219790 1148.6552
## 1262304950.521       273.1424 -299.6994166 845.9842 -602.94354436 1149.2283
## 1262304950.531       273.1633 -300.0394551 846.3660 -603.47464641 1149.8012
## 1262304950.542       273.1842 -300.3793341 846.7476 -604.00550452 1150.3738
## 1262304950.552       273.2050 -300.7190539 847.1291 -604.53611918 1150.9462
## 1262304950.562       273.2259 -301.0586148 847.5105 -605.06649083 1151.5184
## 1262304950.573       273.2468 -301.3980172 847.8917 -605.59661996 1152.0903
## 1262304950.583       273.2677 -301.7372612 848.2727 -606.12650703 1152.6619
## 1262304950.594       273.2886 -302.0763473 848.6535 -606.65615250 1153.2334
## 1262304950.604       273.3095 -302.4152758 849.0343 -607.18555683 1153.8045
## 1262304950.615       273.3304 -302.7540468 849.4148 -607.71472049 1154.3755
## 1262304950.625       273.3513 -303.0926608 849.7952 -608.24364393 1154.9462
## 1262304950.635       273.3722 -303.4311180 850.1754 -608.77232762 1155.5166
## 1262304950.646       273.3930 -303.7694188 850.5555 -609.30077201 1156.0869
## 1262304950.656       273.4139 -304.1075634 850.9354 -609.82897756 1156.6568
## 1262304950.667       273.4348 -304.4455521 851.3152 -610.35694472 1157.2266
## 1262304950.677       273.4557 -304.7833852 851.6948 -610.88467395 1157.7961
## 1262304950.688       273.4766 -305.1210630 852.0743 -611.41216570 1158.3654
## 1262304950.698       273.4975 -305.4585859 852.4536 -611.93942041 1158.9344
## 1262304950.708       273.5184 -305.7959541 852.8327 -612.46643855 1159.5032
## 1262304950.719       273.5393 -306.1331679 853.2117 -612.99322055 1160.0718
## 1262304950.729       273.5602 -306.4702275 853.5905 -613.51976687 1160.6401
## 1262304950.740       273.5810 -306.8071334 853.9692 -614.04607795 1161.2082
## 1262304950.750       273.6019 -307.1438857 854.3478 -614.57215423 1161.7760
## 1262304950.760       273.6228 -307.4804848 854.7261 -615.09799615 1162.3436
## 1262304950.771       273.6437 -307.8169310 855.1044 -615.62360417 1162.9110
## 1262304950.781       273.6646 -308.1532244 855.4824 -616.14897872 1163.4782
## 1262304950.792       273.6855 -308.4893656 855.8604 -616.67412023 1164.0451
## 1262304950.802       273.7064 -308.8253546 856.2381 -617.19902916 1164.6118
## 1262304950.812       273.7273 -309.1611918 856.6157 -617.72370592 1165.1783
## 1262304950.823       273.7482 -309.4968776 856.9932 -618.24815097 1165.7445
## 1262304950.833       273.7691 -309.8324120 857.3705 -618.77236473 1166.3105
## 1262304950.844       273.7899 -310.1677956 857.7477 -619.29634763 1166.8762
## 1262304950.854       273.8108 -310.5030285 858.1247 -619.82010012 1167.4418
## 1262304950.865       273.8317 -310.8381110 858.5015 -620.34362261 1168.0071
## 1262304950.875       273.8526 -311.1730434 858.8783 -620.86691554 1168.5721
## 1262304950.885       273.8735 -311.5078259 859.2548 -621.38997934 1169.1370
## 1262304950.896       273.8944 -311.8424590 859.6312 -621.91281443 1169.7016
## 1262304950.906       273.9153 -312.1769427 860.0075 -622.43542124 1170.2660
## 1262304950.917       273.9362 -312.5112775 860.3836 -622.95780019 1170.8301
## 1262304950.927       273.9571 -312.8454636 860.7596 -623.47995171 1171.3941
## 1262304950.938       273.9779 -313.1795012 861.1354 -624.00187622 1171.9578
```

```
summary(fit_holt_damped)
```

```
##
## Forecast method: Damped Holt's method
##
## Model Information:
## Damped Holt's method
```

```
##
## Call:
## holt(y = my_serie_train, h = 901, damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.9077
##     beta  = 0.0862
##     phi   = 0.8018
##
##   Initial states:
##     l = 167.3992
##     b = -8.351
##
##   sigma:  14.8135
##
##      AIC     AICc      BIC
## 54600.67 54600.69 54638.41
##
## Error measures:
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.02432144 14.80421 6.927864 -0.1445804 3.190256 0.8758601
##                      ACF1
## Training set -0.00288654
##
## Forecasts:
##                 Point Forecast        Lo 80    Hi 80       Lo 95     Hi 95
## 1262304941.562       255.0893  236.1050396 274.0736  226.055380  284.1232
## 1262304941.573       255.1781  228.6399279 281.7164  214.591443  295.7649
## 1262304941.583       255.2494  222.2604928 288.2383  204.797229  305.7015
## 1262304941.594       255.3065  216.4990652 294.1139  195.955652  314.6573
## 1262304941.604       255.3523  211.1724172 299.5322  187.785007  322.9196
## 1262304941.615       255.3890  206.1864588 304.5915  180.140207  330.6378
## 1262304941.625       255.4184  201.4839167 309.3530  172.932708  337.9042
## 1262304941.635       255.4420  197.0255047 313.8586  166.101663  344.7824
## 1262304941.646       255.4610  192.7818888 318.1400  159.601596  351.3203
## 1262304941.656       255.4761  188.7298353 322.2224  153.396485  357.5558
## 1262304941.667       255.4883  184.8501865 326.1264  147.456636  363.5199
## 1262304941.677       255.4980  181.1267115 329.8694  141.756911  369.2392
## 1262304941.688       255.5059  177.5454049 333.4663  136.275636  374.7361
## 1262304941.698       255.5121  174.0940294 336.9302  130.993895  380.0303
## 1262304941.708       255.5171  170.7617989 340.2725  125.895028  385.1393
## 1262304941.719       255.5212  167.5391455 343.5032  120.964272  390.0781
## 1262304941.729       255.5244  164.4175423 346.6313  116.188482  394.8603
## 1262304941.740       255.5270  161.3893611 349.6646  111.555907  399.4981
## 1262304941.750       255.5291  158.4477571 352.6104  107.056013  404.0021
## 1262304941.760       255.5307  155.5865716 355.4749  102.679326  408.3821
## 1262304941.771       255.5321  152.8002503 358.2639   98.417309  412.6468
## 1262304941.781       255.5331  150.0837726 360.9825   94.262248  416.8040
## 1262304941.792       255.5340  147.4325910 363.6354   90.207162  420.8608
## 1262304941.802       255.5347  144.8425776 366.2268   86.245714  424.8237
## 1262304941.812       255.5352  142.3099784 368.7605   82.372146  428.6983
## 1262304941.823       255.5357  139.8313722 371.2400   78.581210  432.4902
## 1262304941.833       255.5360  137.4036354 373.6684   74.868120  436.2040
## 1262304941.844       255.5363  135.0239101 376.0487   71.228494  439.8441
```

```
## 1262304950.854       255.5375 -456.1647110 967.2397 -832.917088 1343.9920
## 1262304950.865       255.5375 -456.5641945 967.6391 -833.528045 1344.6030
## 1262304950.875       255.5375 -456.9634540 968.0384 -834.138660 1345.2136
## 1262304950.885       255.5375 -457.3624899 968.4374 -834.748933 1345.8239
## 1262304950.896       255.5375 -457.7613026 968.8362 -835.358864 1346.4338
## 1262304950.906       255.5375 -458.1598925 969.2348 -835.968455 1347.0434
## 1262304950.917       255.5375 -458.5582598 969.6332 -836.577705 1347.6526
## 1262304950.927       255.5375 -458.9564051 970.0313 -837.186615 1348.2616
## 1262304950.938       255.5375 -459.3543286 970.4293 -837.795187 1348.8701
```

```
summary(fit_arima)
```

```
## Series: my_serie_train
## ARIMA(1,0,0)(0,1,0)[96]
##
## Coefficients:
##          ar1
##       0.7745
## s.e.  0.0101
##
## sigma^2 = 97.33:  log likelihood = -14438.8
## AIC=28881.6   AICc=28881.6   BIC=28894.13
##
## Training set error measures:
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.07270899 9.744813 5.712995 -0.1197716 2.641239 0.7222694
##                      ACF1
## Training set -0.01386996
```

```
summary(fit_hw)
```

```
##              Length Class  Mode
## fitted       15576  mts    numeric
## x             3990  ts     numeric
## alpha            1  -none- numeric
## beta             1  -none- numeric
## gamma            1  -none- numeric
## coefficients    98  -none- numeric
## seasonal         1  -none- character
## SSE              1  -none- numeric
## call             3  -none- call
```

We encounter issue with freq > 24 for seasonal hw damped We use seasonal HolWinters as workaround

RMSE

```
# Calculate the RMSE for the Holt model
rmse_holt <- sqrt(mean((fit_holt$mean - my_serie_test)^2))

# Print the RMSE with a descriptive message
cat('Holt RMSE:', rmse_holt, '\n')
```

```
## Holt RMSE: 73.25496
```

```
# Calculate the RMSE for the Damped Holt model
rmse_damped_holt <- sqrt(mean((fit_holt_damped$mean - my_serie_test)^2))

# Print the RMSE with a descriptive message
```

```r
cat('Damped Holt RMSE:', rmse_damped_holt, '\n')
```

```
## Damped Holt RMSE: 68.8236
```

```r
# Calculate the RMSE for the auto.arima model
rmse_arima <- sqrt(mean((forecast_arima$mean - my_serie_test)^2))

# Print the RMSE with a descriptive message
cat('auto.arima RMSE:', rmse_arima, '\n')
```

```
## auto.arima RMSE: 23.47388
```

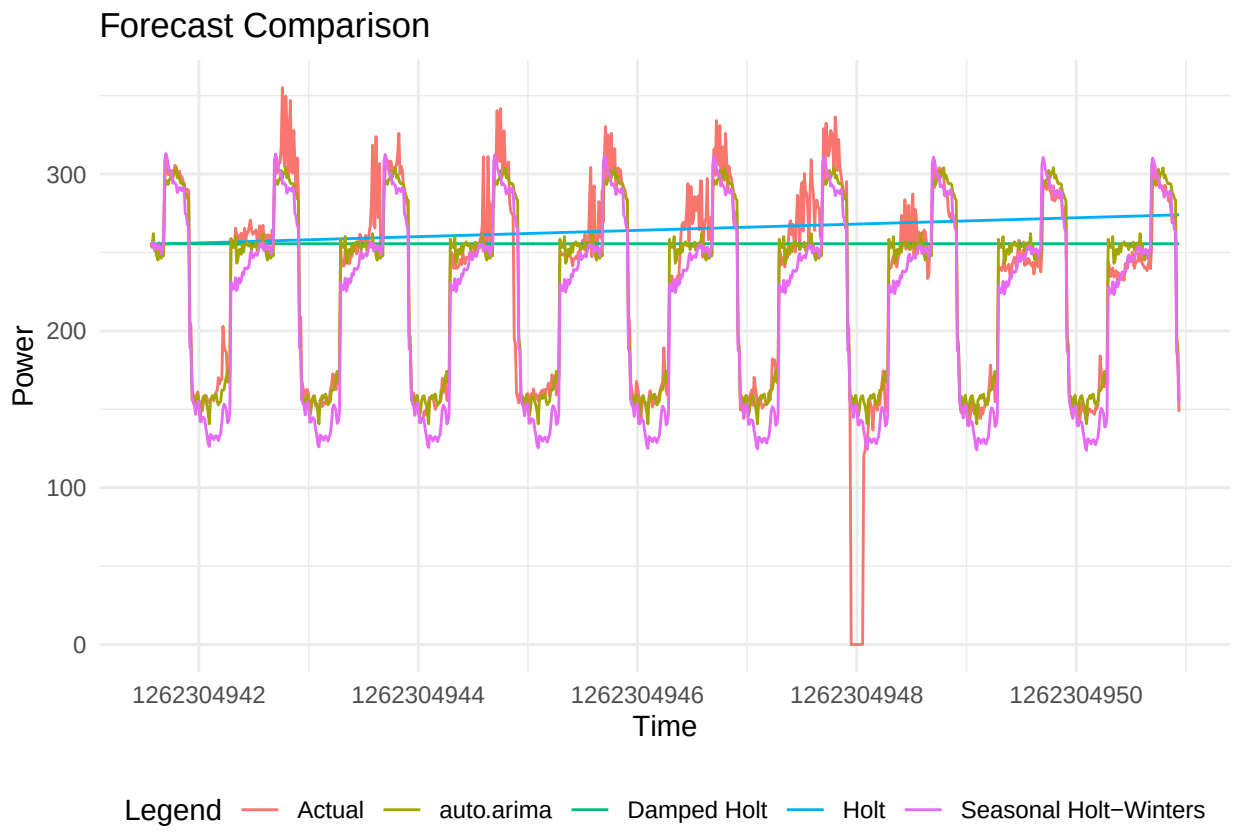
```r
# Calculate the RMSE for the seasonal Holt-Winters model
rmse_hw <- sqrt(mean((forecast_hw - my_serie_test)^2))

# Print the RMSE with a descriptive message
cat('Seasonal Holt-Winters RMSE:', rmse_hw, '\n')
```

```
## Seasonal Holt-Winters RMSE: 25.97643
```

We notice that the better model is auto.arima with perform a RMSE=23.47388. It could better to use cross validation for better model selection

```r
# Create the plot with customizations
autoplot(my_serie_test, series = "Actual") +
  autolayer(fit_holt$mean, series = "Holt", PI = FALSE) +
  autolayer(fit_holt_damped$mean, series = "Damped Holt", PI = FALSE) +
  autolayer(forecast_arima$mean, series = "auto.arima", PI = FALSE) +
  autolayer(forecast_hw, series = "Seasonal Holt-Winters", PI = FALSE) +
  labs(title = "Forecast Comparison",
       x = "Time",
       y = "Power",
       color = "Legend") +
  theme_minimal() +
  theme(legend.position = "bottom")
```

```
## Warning in ggplot2::geom_line(ggplot2::aes(x = .data[["timeVal"]], y = .data[["seriesVal"]], : Ignor
## Ignoring unknown parameters: `PI`
## Ignoring unknown parameters: `PI`
## Ignoring unknown parameters: `PI`
```



Forecasting

Let's forecast 96 data using auto.arima Now we will use all data1 set as training ()

```
# Fit the auto.arima model
fit_arima_full <- auto.arima(my_serie)

# Forecast using the fitted model
forecast_arima_full <- forecast(fit_arima_full, h = 96)

# Check the model summary
summary(fit_arima_full)
```

```
## Series: my_serie
## ARIMA(5,0,1)(0,1,0)[96]
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5     ma1
##       0.3966  0.2911  0.1917  -0.2141  0.0633  0.3312
## s.e.  0.1173  0.0830  0.0186   0.0283  0.0303  0.1174
##
## sigma^2 = 137.4:  log likelihood = -18604.56
## AIC=37223.11   AICc=37223.14   BIC=37268.44
##
## Training set error measures:
##                       ME    RMSE      MAE  MPE MAPE     MASE          ACF1
## Training set -0.08744663 11.6002 6.643672 -Inf  Inf 0.712952 -0.0003429888
```

```r
# Extract the forecasted mean values
forecasted_means <- forecast_arima_full$mean
print(forecasted_means)
```

```
## Time Series:
## Start = c(1262304950, 92)
## End = c(1262304951, 91)
## Frequency = 96
##  [1] 145.2011 140.3435 140.9941 148.9116 151.1115 152.0155 141.9249 144.5476
##  [9] 141.4772 147.6041 153.6907 148.9995 148.2129 146.3382 141.1884 146.8614
## [17] 150.1116 149.9427 148.0809 148.7022 147.7216 146.7394 151.0499 150.0609
## [25] 152.3689 161.6747 184.0803 173.5841 169.6873 164.1900 164.4919 167.9936
## [33] 166.2949 243.9959 241.0968 234.5974 238.9979 239.9984 234.1987 235.3990
## [41] 238.2992 232.4993 237.0995 235.3996 233.7997 232.6997 234.0998 232.2998
## [49] 242.8999 243.1999 240.0999 247.5999 246.3999 244.1000 250.0000 242.8000
## [57] 239.8000 244.3000 243.9000 245.2000 249.9000 244.4000 247.2000 242.4000
## [65] 241.5000 240.0000 240.4000 238.4000 241.8000 246.1000 239.7000 252.8000
## [73] 299.2000 302.9000 305.5000 292.6000 294.5000 303.1000 293.4000 291.3000
## [81] 292.6000 288.9000 291.9000 292.9000 287.9000 287.4000 282.6000 284.5000
## [89] 276.2000 268.9000 272.5000 271.2000 269.8000 189.6000 177.9000 148.4000
```

Forecasting with covariates

Here we will use dynamic regression by using variable temperature

```r
# Fit the auto.arima model with external regressors
fit_arima_xreg <- auto.arima(my_serie, xreg = data1$temp)

# Forecast using the fitted model with external regressors
forecast_arima_xreg <- forecast(fit_arima_xreg, h = 96, xreg = data2$temp)

# Check the model summary
summary(fit_arima_xreg)
```

```
## Series: my_serie
## Regression with ARIMA(5,0,1)(0,1,0)[96] errors
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5     ma1    xreg
##       0.4105  0.2812  0.1895  -0.2176  0.0660  0.3161  0.3952
## s.e.  0.1204  0.0851  0.0188   0.0285  0.0306  0.1207  0.2653
##
## sigma^2 = 137.4:  log likelihood = -18603.45
## AIC=37222.89   AICc=37222.92   BIC=37274.69
##
## Training set error measures:
##                       ME     RMSE      MAE  MPE MAPE     MASE          ACF1
## Training set -0.08676412 11.59752 6.641026 -Inf  Inf 0.712668 -0.0004231387
```

```r
# Extract the forecasted mean values
forecasted_means_xreg <- forecast_arima_xreg$mean

# Print the forecasted mean values with a descriptive message
print(forecasted_means_xreg)
```

```
## Time Series:
```

```
## Start = c(1262304950, 92)
## End = c(1262304951, 91)
## Frequency = 96
##  [1] 145.2814 140.6166 141.3251 149.3395 151.5958 151.6516 141.6059 144.2444
##  [9] 141.1959 147.3390 153.4333 148.7539 147.9724 146.1027 140.9577 146.6325
## [17] 149.8856 149.7183 147.8575 148.4801 147.5000 147.6160 151.9271 150.9382
## [25] 153.2465 161.8939 184.2995 173.8034 169.9067 164.4095 164.7114 168.2131
## [33] 166.5144 243.7764 240.8772 234.3779 238.7784 239.3398 233.5401 234.7404
## [41] 237.6406 231.6212 236.2213 234.5214 232.9215 232.2607 233.6607 231.8608
## [49] 242.4608 242.5413 239.4413 246.9413 245.7413 243.6609 249.5609 242.3609
## [57] 239.3609 244.0804 243.6805 244.9805 249.6805 244.6195 247.4195 242.6195
## [65] 241.7195 240.4391 240.8391 238.8391 242.2391 246.5391 240.1391 253.2391
## [73] 299.6391 303.1195 305.7195 292.8195 294.7195 303.1000 293.4000 291.3000
## [81] 292.6000 288.9000 291.9000 292.9000 287.9000 287.1805 282.3805 284.2805
## [89] 275.9805 268.4609 272.0609 270.7609 269.3609 188.9414 177.2414 147.7414
```

```r
combined_forecasts <- data.frame(
  "Without Outdoor Temperature" = forecasted_means,
  "With Outdoor Temperature" = forecasted_means_xreg
)

write.xlsx(combined_forecasts, file = "selvalakshmi.xlsx")
```