Global Electricity Active Power Prediction

AOUIDANE Imed Eddine

```
library(tidyverse)
library(TSstudio)
library(xts)
library(forecast)
library(readr)
library(lubridate)
library(dygraphs)
library(imputeTS)
Sys.setlocale("LC_ALL",locale = "eng")
```

```
## [1] "LC_COLLATE=English_United Kingdom.1252;LC_CTYPE=English_United Kingdom.1252;LC_MONETARY=English_
```

Data Description

Attribute Information

- **date**: Date in format `dd/mm/yyyy`
- **time**: Time in format `hh:mm:ss`
- **global_active_power**: Household global minute-averaged active power (in kilowatt)
- **global_reactive_power**: Household global minute-averaged reactive power (in kilowatt)
- **voltage**: Minute-averaged voltage (in volt)
- **global_intensity**: Household global minute-averaged current intensity (in ampere)
- **sub_metering_1**: Energy sub-metering No. 1 (in watt-hour of active energy), corresponding to the kitchen.
- **sub_metering_2**: Energy sub-metering No. 2 (in watt-hour of active energy), corresponding to the laundry room.
- **sub_metering_3**: Energy sub-metering No. 3 (in watt-hour of active energy), corresponding to an electric water-heater and an air-conditioner.

Notes

1. The expression `(global_active_power*1000/60 - sub_metering_1 - sub_metering_2 - sub_metering_3)` represents the active energy consumed every minute (in watt-hour) by electrical equipment not measured by the sub-meterings.
2. The dataset contains missing values (approximately 1.25% of the rows). Missing values are represented by the absence of values between two consecutive semi-colon attribute separators. For example, missing values are observed on April 28, 2007.

Original Source

- Georges Hébrail, Senior Researcher, EDF R&D, Clamart, France

- Alice Bérard, TELECOM ParisTech Master of Engineering Internship at EDF R&D, Clamart, France

```
read.csv("C:\\Users\\dell\\Downloads\\pc\\fac\\4eme\\time series analysis\\individual+household+electri
head(data)
```

```
##         Date     Time Global_active_power Global_reactive_power Voltage
## 1 16/12/2006 17:24:00               4.216                 0.418 234.840
## 2 16/12/2006 17:25:00               5.360                 0.436 233.630
## 3 16/12/2006 17:26:00               5.374                 0.498 233.290
## 4 16/12/2006 17:27:00               5.388                 0.502 233.740
## 5 16/12/2006 17:28:00               3.666                 0.528 235.680
## 6 16/12/2006 17:29:00               3.520                 0.522 235.020
##   Global_intensity Sub_metering_1 Sub_metering_2 Sub_metering_3
## 1           18.400          0.000          1.000             17
## 2           23.000          0.000          1.000             16
## 3           23.000          0.000          2.000             17
## 4           23.000          0.000          1.000             17
## 5           15.800          0.000          1.000             17
## 6           15.000          0.000          2.000             17
```

```
dim(data)
```

```
## [1] 2075259       9
```

```
data$Date <- as.Date(data$Date,format = "%d/%m/%Y")
data <- data %>%
  mutate(Date_time = lubridate::ymd(Date) + lubridate::hms(Time)) %>%
  mutate_if(is.character,as.numeric) %>%
  mutate(apparent_power = sqrt(Global_active_power^2 + Global_reactive_power^2)) %>%
  select(-Date,-Time,-Global_reactive_power) %>%
  select(Date_time,apparent_power,everything())
```

```
## Warning: There were 7 warnings in `mutate()`.
## The first warning was:
## i In argument: `Time = .Primitive("as.double")(Time)`.
## Caused by warning:
## ! NAs introduits lors de la conversion automatique
## i Run `dplyr::last_dplyr_warnings()` to see the 6 remaining warnings.
```

```
head(data)
```

```
##             Date_time apparent_power Global_active_power Voltage
## 1 2006-12-16 17:24:00       4.236671               4.216  234.84
## 2 2006-12-16 17:25:00       5.377704               5.360  233.63
## 3 2006-12-16 17:26:00       5.397025               5.374  233.29
## 4 2006-12-16 17:27:00       5.411335               5.388  233.74
## 5 2006-12-16 17:28:00       3.703828               3.666  235.68
## 6 2006-12-16 17:29:00       3.558495               3.520  235.02
##   Global_intensity Sub_metering_1 Sub_metering_2 Sub_metering_3
## 1             18.4             0              1             17
## 2             23.0             0              1             16
```

```
## 3             23.0              0              2             17
## 4             23.0              0              1             17
## 5             15.8              0              1             17
## 6             15.0              0              2             17
```

Calculating the power factor

```
data$power_factor <- data$Global_active_power/data$apparent_power
```

```
data %>%
  map(~sum(is.na(.)))
```

```
## $Date_time
## [1] 0
##
## $apparent_power
## [1] 25979
##
## $Global_active_power
## [1] 25979
##
## $Voltage
## [1] 25979
##
## $Global_intensity
## [1] 25979
##
## $Sub_metering_1
## [1] 25979
##
## $Sub_metering_2
## [1] 25979
##
## $Sub_metering_3
## [1] 25979
##
## $power_factor
## [1] 25979
```

```
xts_data <- xts(data[,-1],order.by = data$Date_time)
head(xts_data)
```

```
## Warning: object timezone (UTC) is different from system timezone ()
##   NOTE: set 'options(xts_check_TZ = FALSE)'to disable this warning
##     This note is displayed once per session
```

```
##                     apparent_power Global_active_power Voltage Global_intensity
## 2006-12-16 17:24:00       4.236671               4.216  234.84             18.4
## 2006-12-16 17:25:00       5.377704               5.360  233.63             23.0
```

```
## 2006-12-16 17:26:00       5.397025                5.374  233.29             23.0
## 2006-12-16 17:27:00       5.411335                5.388  233.74             23.0
## 2006-12-16 17:28:00       3.703828                3.666  235.68             15.8
## 2006-12-16 17:29:00       3.558495                3.520  235.02             15.0
##                      Sub_metering_1 Sub_metering_2 Sub_metering_3 power_factor
## 2006-12-16 17:24:00              0              1             17    0.9951210
## 2006-12-16 17:25:00              0              1             16    0.9967080
## 2006-12-16 17:26:00              0              2             17    0.9957337
## 2006-12-16 17:27:00              0              1             17    0.9956877
## 2006-12-16 17:28:00              0              1             17    0.9897868
## 2006-12-16 17:29:00              0              2             17    0.9891823
```

```
ts_info(xts_data)
```

```
##  The xts_data series is a xts object with 8 variables and 2075259 observations
##  Frequency: 1 mins
##  Start time: 2006-12-16 17:24:00
##  End time: 2010-11-26 21:02:00
```

```
skimr::skim(xts_data)
```

Table 1: Data summary

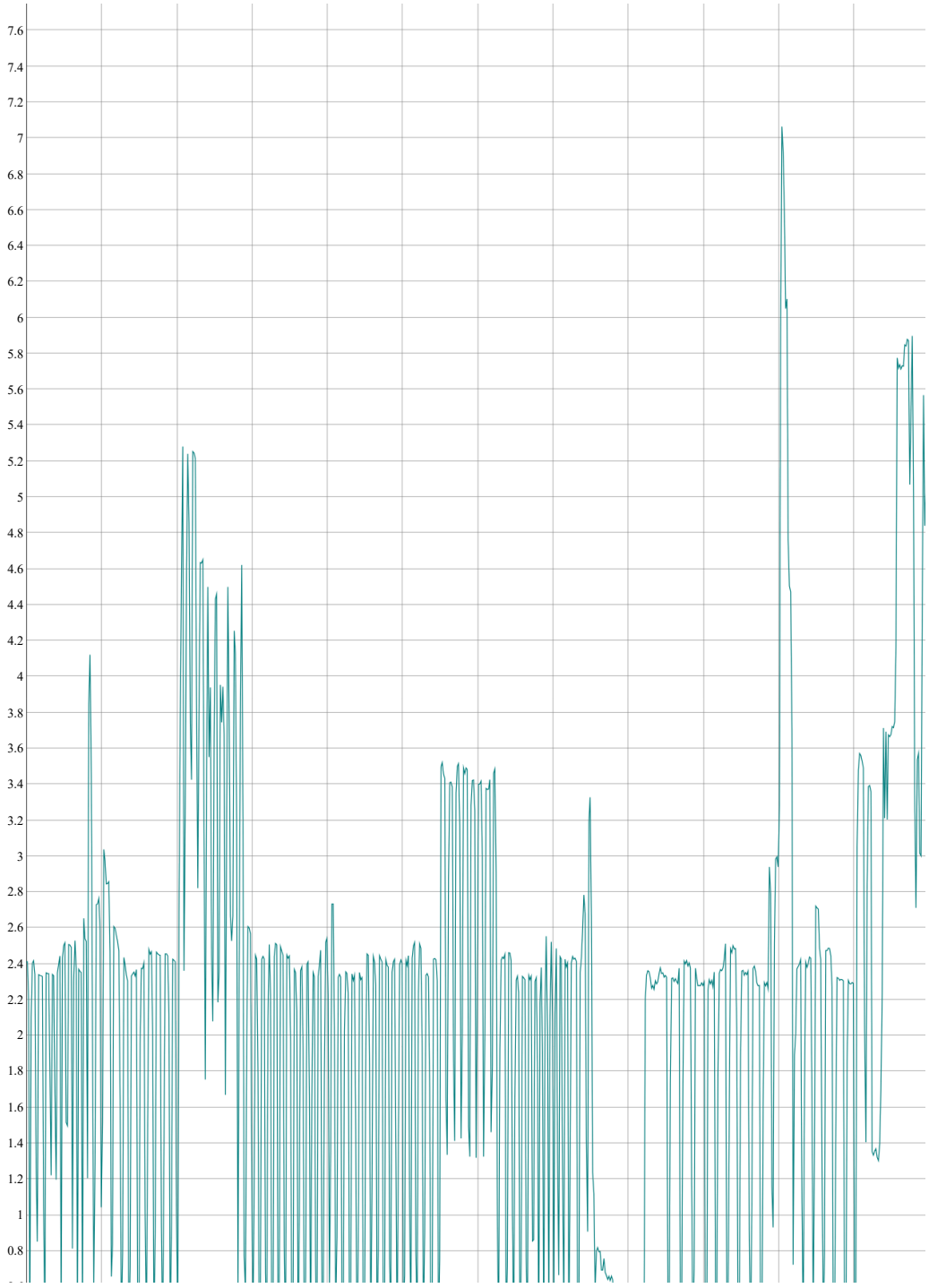
Name	xts_data
Number of rows	2075259
Number of columns	8
Column type frequency:	
numeric	8
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
apparent_power	25979	0.99	1.11	1.05	0.08	0.33	0.63	1.54	11.12	
Global_active_power	25979	0.99	1.09	1.06	0.08	0.31	0.60	1.53	11.12	
Voltage	25979	0.99	240.84	3.24	223.20	238.99	241.01	242.89	254.15	
Global_intensity	25979	0.99	4.63	4.44	0.20	1.40	2.60	6.40	48.40	
Sub_metering_1	25979	0.99	1.12	6.15	0.00	0.00	0.00	0.00	88.00	
Sub_metering_2	25979	0.99	1.30	5.82	0.00	0.00	0.00	1.00	80.00	
Sub_metering_3	25979	0.99	6.46	8.44	0.00	0.00	1.00	17.00	31.00	
power_factor	25979	0.99	0.96	0.06	0.56	0.95	0.99	1.00	1.00	

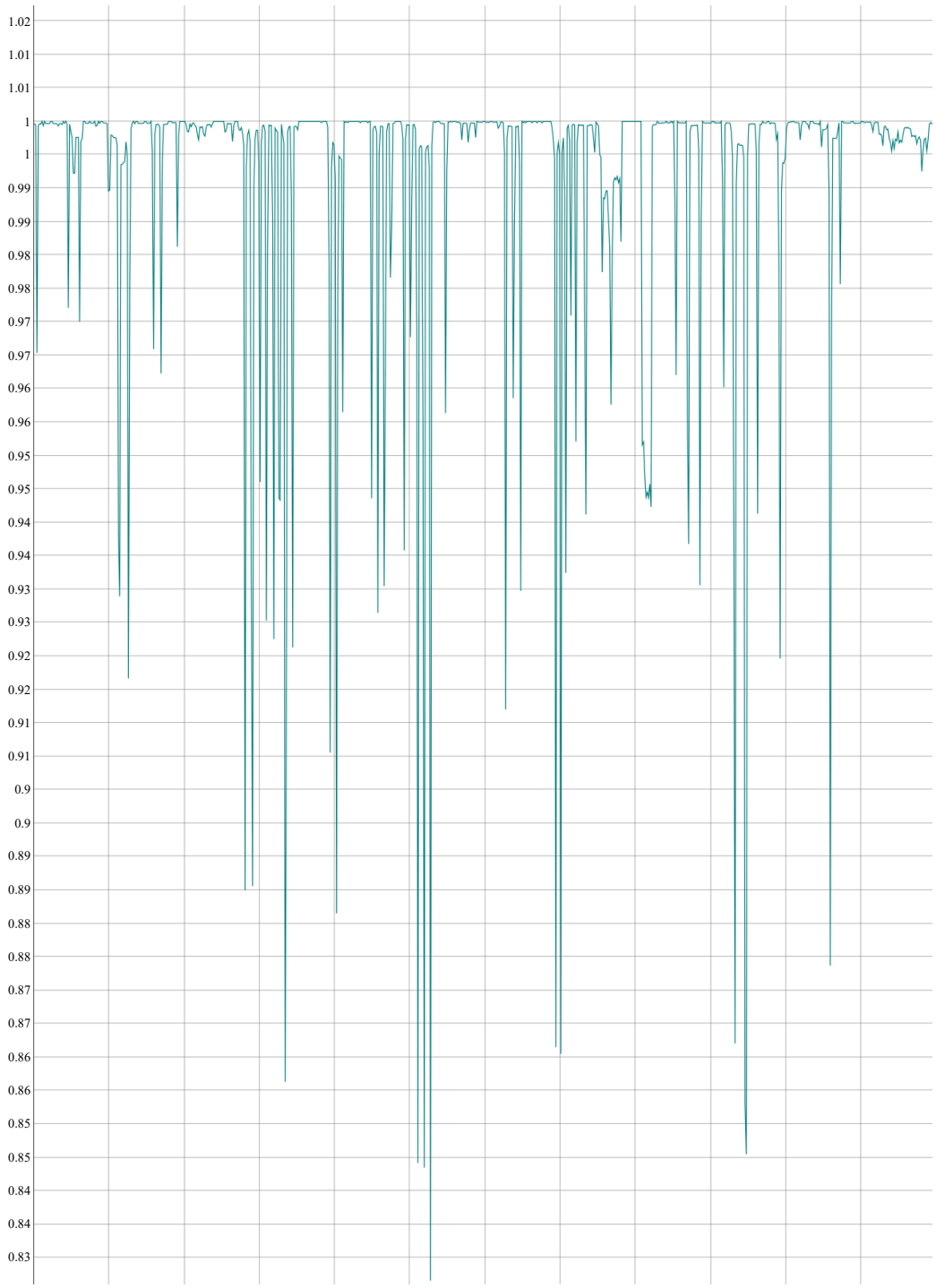
Ploting for a single day

```
dt <- seq.POSIXt(from = as.POSIXct("2006-12-17 00:00:00"),to = as.POSIXct("2006-12-18 00:00:00"),by = "m
dygraph(xts_data[dt,2]) %>%
  dyRangeSelector()
```



- We can see that the Global active power experiences a positive trend after 08:00 AM. and a negative one after 22:00 PM. indicating high power demand in the day ## Plotting the power factor

```
dygraph(xts_data[dt,8]) %>%
  dyRangeSelector()
```



- The power factor shows alot of noise during the night, while in the day its almost stable which means the total power used in the circuit is high.

NA's Interpolation

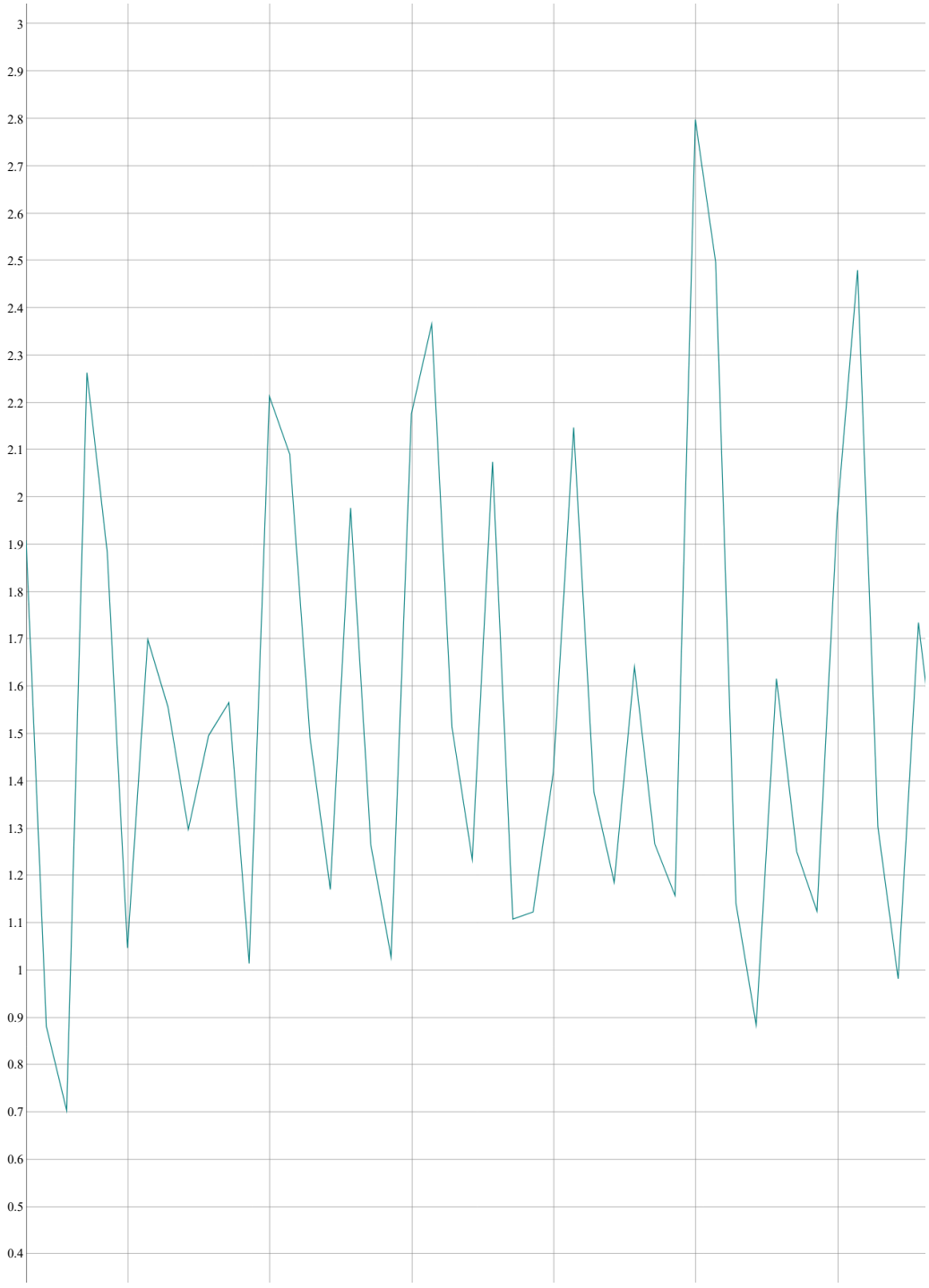
- From the statistical resume we saw earlier we can see that there's alot of missing values, which we need to fix before doing furthur analysis , droping the missing values isn't a solution so we have to fix it. one of the most efficient ways for a large dataset such as ours is na.approx which performs linear interpolation efficiently and it's optimized for large datasets, making it a good choice when you have many missing values.

```
xts_data <- na.approx(xts_data)
```

Aggregated data

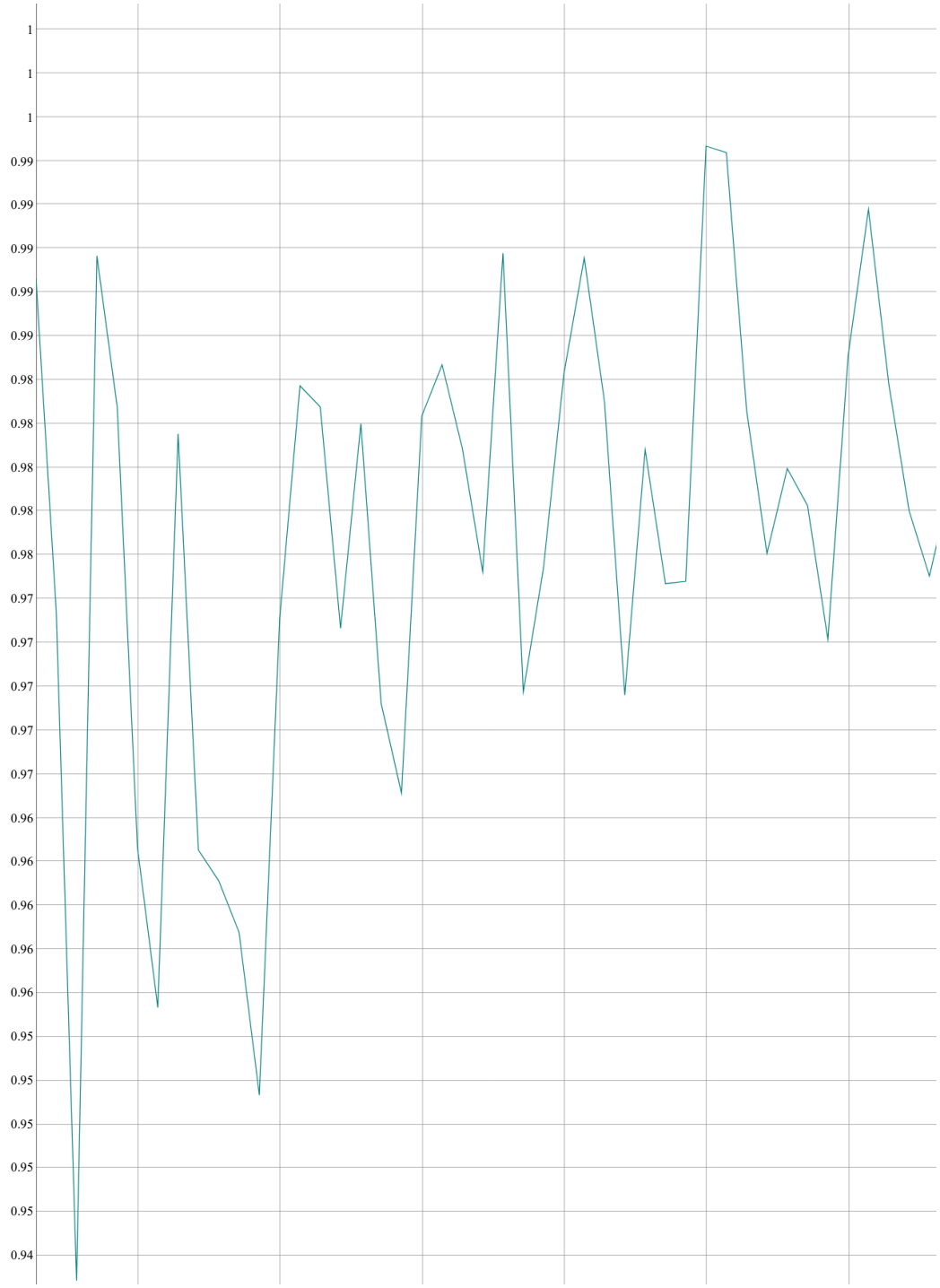
```
agg_data_daily <- apply.daily(xts_data,FUN = mean)
agg_data_monthly <- apply.monthly(xts_data,FUN = mean)
```

```
# Plotting the first trimester of 2007
dygraph(agg_data_daily[time(agg_data_daily) > "2007-01-01 23:59:00 UTC" &
                         time(agg_data_daily) < "2007-04-01 23:59:00 UTC",
                       2]) %>%
  dyRangeSelector()
```



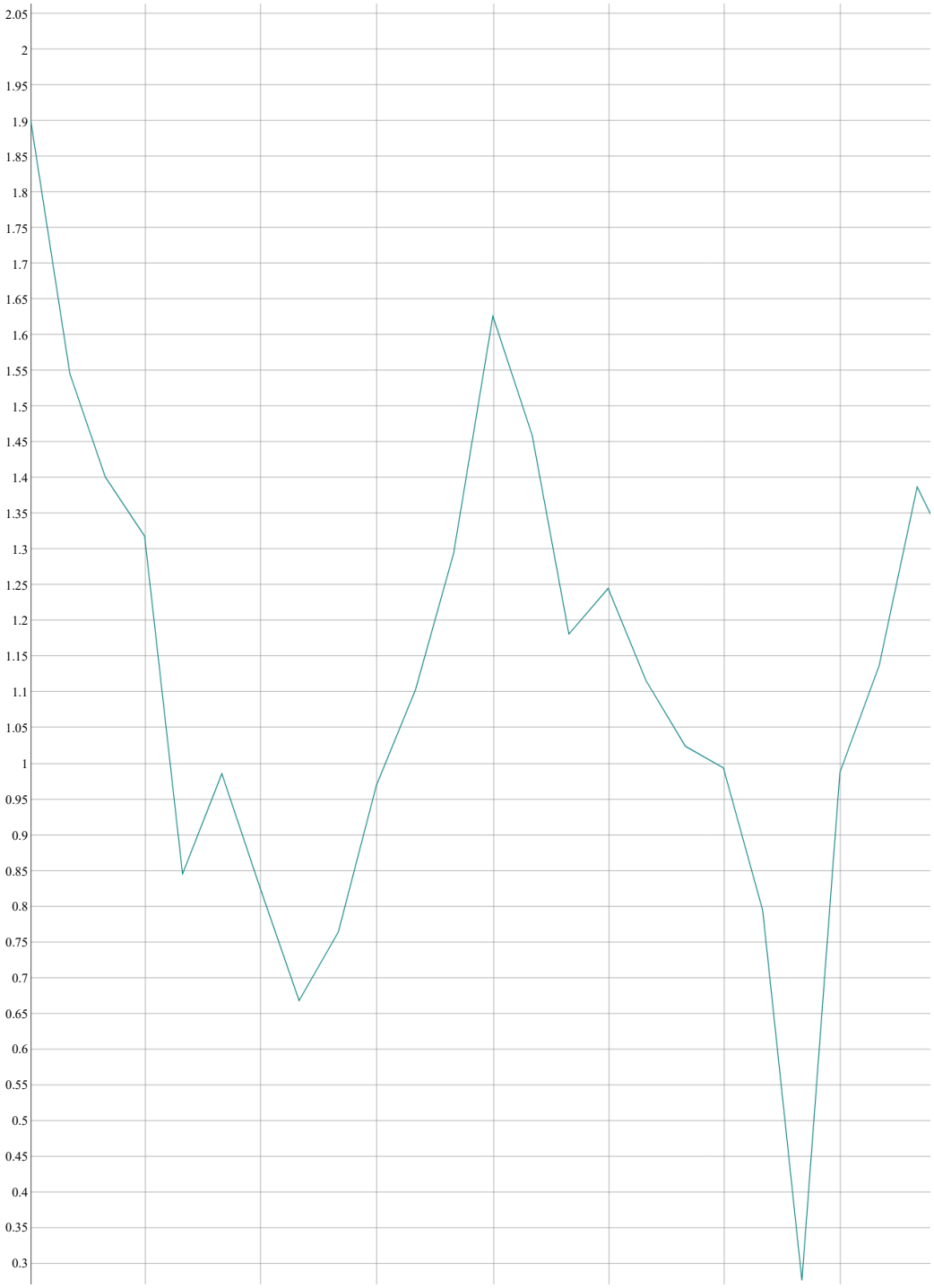
- We can see that there's a sort of seasonality with some fluctuations, we can consider it as an anomali

```
# Power factor for the trimester
dygraph(agg_data_daily[time(agg_data_daily) > "2007-01-01 23:59:00 UTC" &
                         time(agg_data_daily) < "2007-04-01 23:59:00 UTC",
                       8]) %>%
  dyRangeSelector()
```



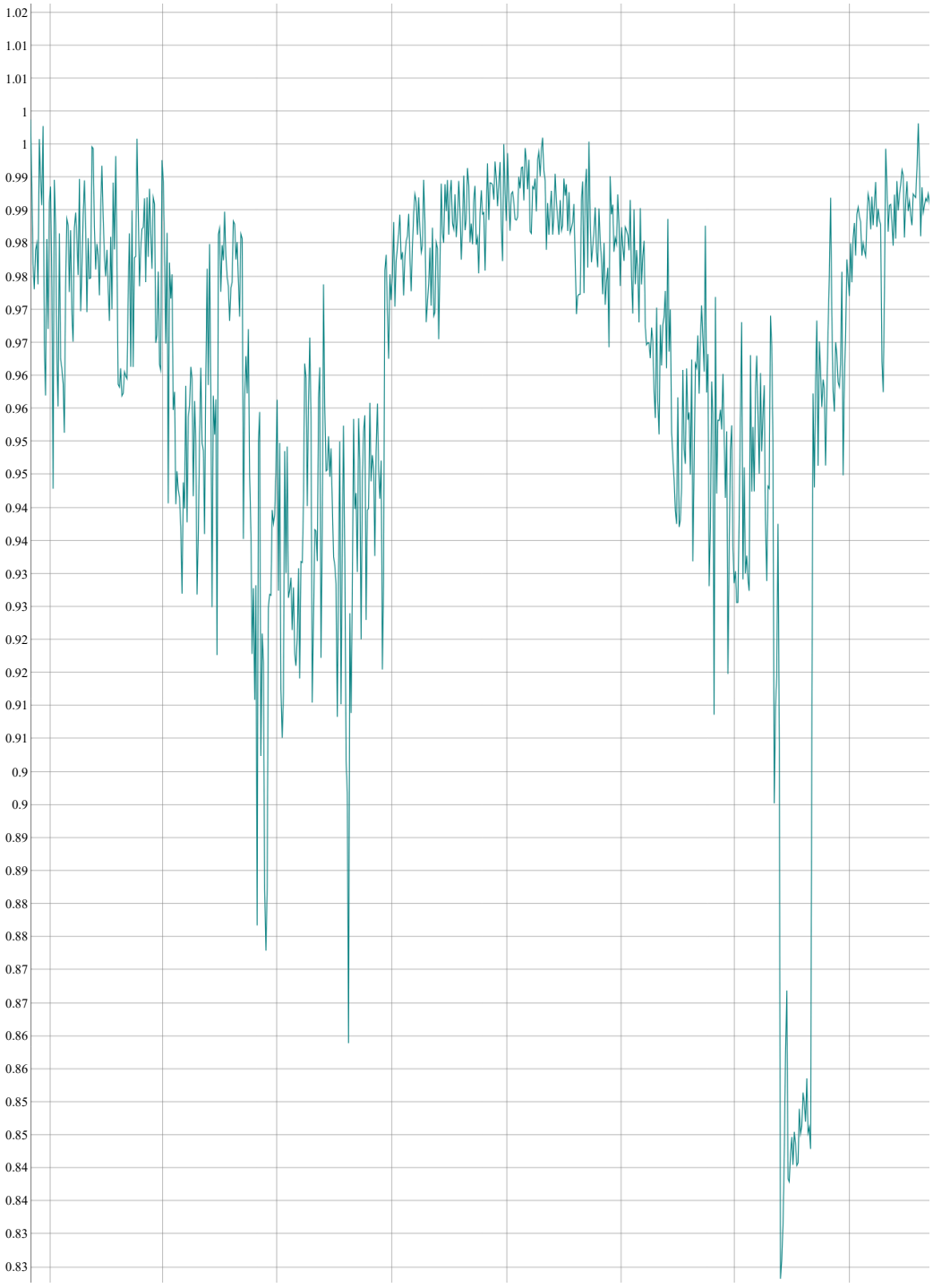

```
# plotting monthly consumption
dygraph(agg_data_monthly[,2]) %>%
```

```
dyRangeSelector()
```



- Same thing as the trimester

```
# Monthly Power factor
dygraph(agg_data_daily[,8]) %>%
  dyRangeSelector()
```


1.02
1.01
1.01
1
1
0.99
0.99
0.98
0.98
0.97
0.97
0.96
0.96
0.95
0.95
0.94
0.94
0.93
0.93
0.92
0.92
0.91
0.91
0.9
0.9
0.89
0.89
0.88
0.88
0.87
0.87
0.86
0.86
0.85
0.85
0.84
0.84
0.83
0.83

```r
head(agg_data_monthly)
```

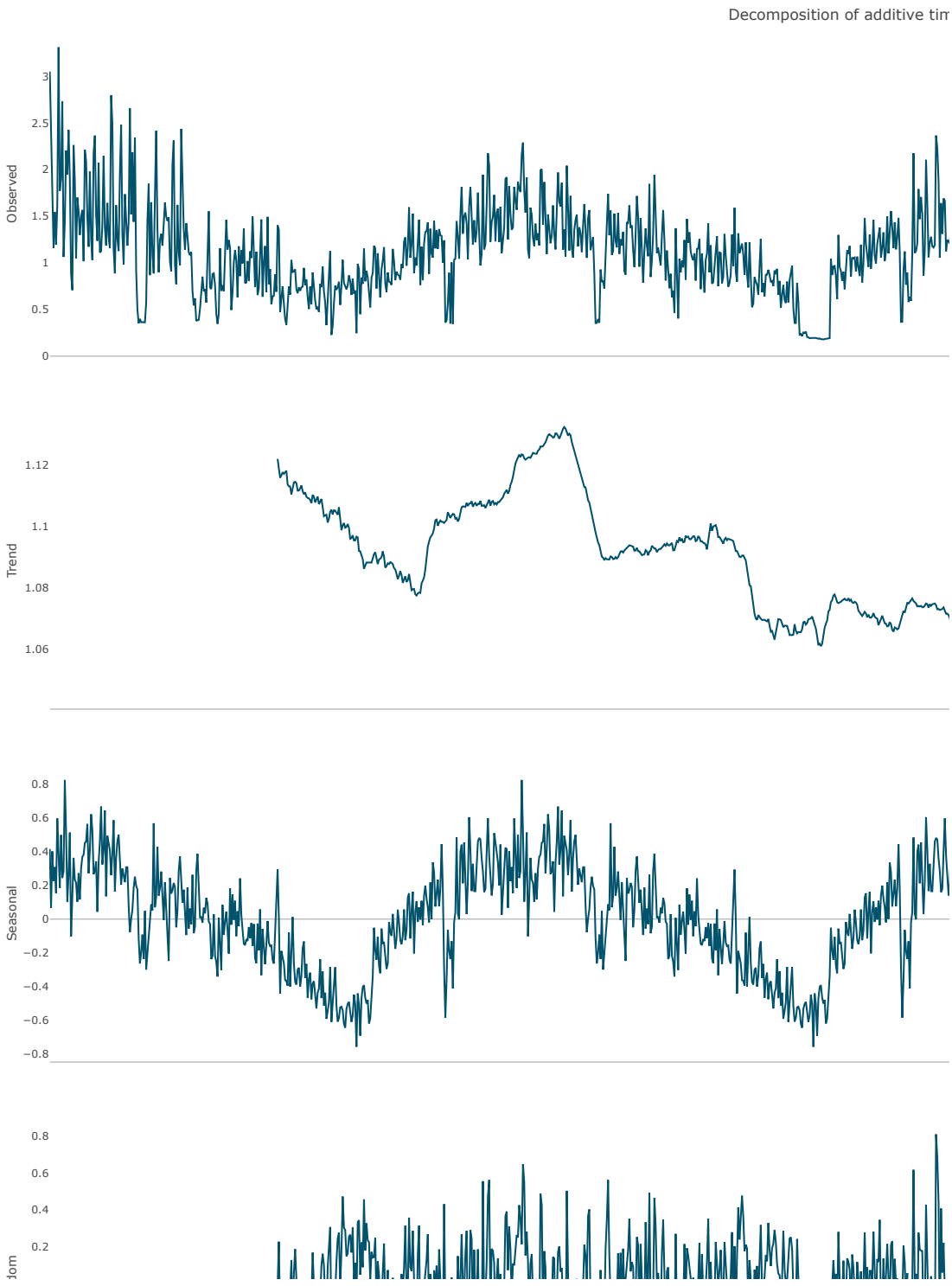
```
## Warning: object timezone (UTC) is different from system timezone ()
```

```
##                     apparent_power Global_active_power  Voltage
## 2006-12-31 23:59:00      1.9143050           1.9015642 241.4408
## 2007-01-31 23:59:00      1.5612363           1.5460864 240.9048
## 2007-02-28 23:59:00      1.4142381           1.4012110 240.5192
## 2007-03-31 23:59:00      1.3316349           1.3186093 240.5135
## 2007-04-30 23:59:00      0.8660198           0.8455831 239.0557
## 2007-05-31 23:59:00      1.0019143           0.9858618 235.1784
##                     Global_intensity Sub_metering_1 Sub_metering_2
## 2006-12-31 23:59:00         8.031087      1.2518185      2.2167212
## 2007-01-31 23:59:00         6.547142      1.2641801      1.7758625
## 2007-02-28 23:59:00         5.915104      1.1801587      1.6031746
## 2007-03-31 23:59:00         5.572905      1.3613127      2.3468190
## 2007-04-30 23:59:00         3.633866      0.9740278      0.8892824
## 2007-05-31 23:59:00         4.297464      1.6966174      1.6158602
##                     Sub_metering_3 power_factor
## 2006-12-31 23:59:00       7.409802    0.9800570
## 2007-01-31 23:59:00       7.383748    0.9743313
## 2007-02-28 23:59:00       6.704067    0.9766100
## 2007-03-31 23:59:00       6.504503    0.9762203
## 2007-04-30 23:59:00       4.386644    0.9532596
## 2007-05-31 23:59:00       5.139964    0.9661986
```

Seasonal decomposition

```r
# Daily data
ts_data_daily <- ts(agg_data_daily[, "Global_active_power"], frequency = 365)
ts_data_daily_decomposed <- decompose(ts_data_daily)
ts_decompose(ts_data_daily,type = "all")
```

```
## Warning in ts_decompose(ts_data_daily, type = "all"): The value of 'type' is
## not valide, using the default option - 'additive'
```




```r
# Monthly data
ts_data_monthly <- ts(agg_data_monthly[, "Global_active_power"], frequency = 12)
```

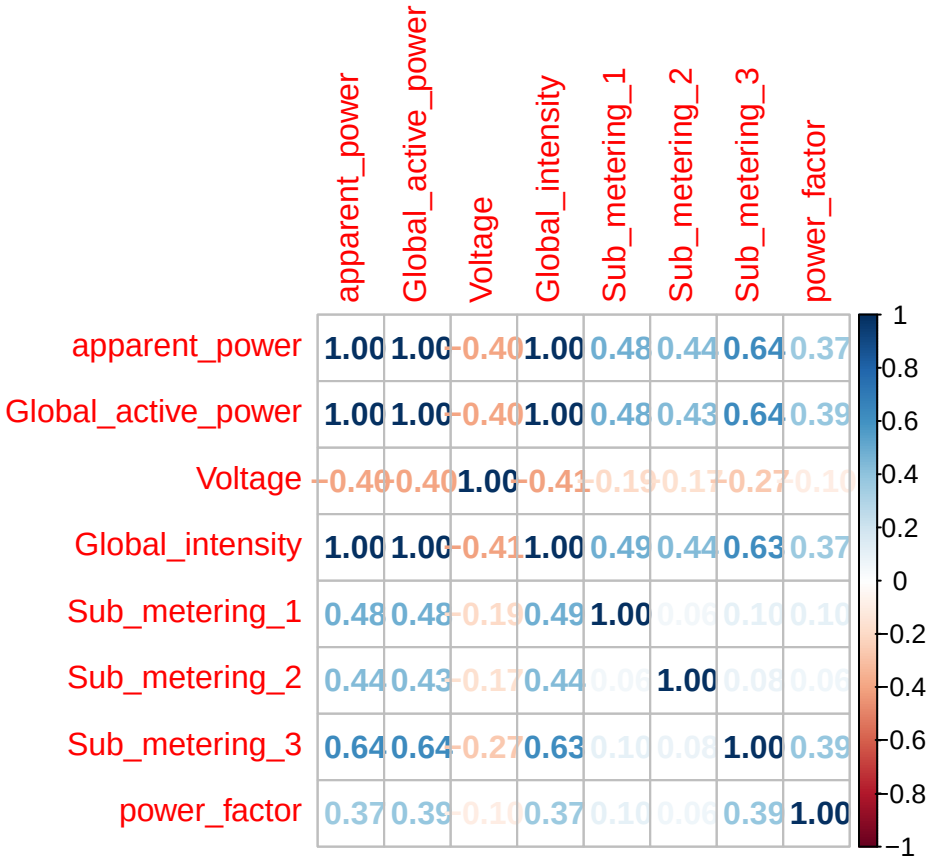
```r
ts_decompose(ts_data_monthly,type = "all")
```

```
## Warning in ts_decompose(ts_data_monthly, type = "all"): The value of 'type' is
## not valide, using the default option - 'additive'
```


Decomposition of additive time
Observed
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
Trend
1.12
1.11
1.1
1.09
1.08
1.07
1.06
Seasonal
0.2
0
−0.2
−0.4
0.2
0.1
0

Correlation

```
corrplot::corrplot(cor(xts_data),method = "number")
```



- Based on the correlation analysis, we observed that global active power is highly correlated with both apparent power and global intensity, suggesting potential redundancy. To avoid multicollinearity, we can consider dropping apparent power and global intensity for future modeling steps. Additionally, voltage shows a negative correlation with the power variables, providing useful variability, while the sub-metering variables and power factor have a significant correlation with the target variable (Global active power) so they can be used to predict it.

Feature Engineering

- Using tsfeatures package to extract various features from our time series data.

```
daily_modeling_data <- agg_data_daily[,- c(1,4)]
head(daily_modeling_data)
```

```
## Warning: object timezone (UTC) is different from system timezone ()
```

```
##                     Global_active_power  Voltage Sub_metering_1 Sub_metering_2
## 2006-12-16 23:59:00            3.053475 236.2438      0.0000000       1.378788
## 2006-12-17 23:59:00            2.354486 240.0870      1.4118056       2.907639
```

```
## 2006-12-18 23:59:00            1.530435 241.2317      0.7381944       1.820139
## 2006-12-19 23:59:00            1.157079 241.9993      0.5826389       5.279167
## 2006-12-20 23:59:00            1.545658 242.3081      0.0000000       1.838889
## 2006-12-21 23:59:00            1.192440 241.0419      1.2256944       1.821528
##                     Sub_metering_3 power_factor
## 2006-12-16 23:59:00      12.439394    0.9987684
## 2006-12-17 23:59:00       9.264583    0.9884013
## 2006-12-18 23:59:00       9.734722    0.9768335
## 2006-12-19 23:59:00       4.303472    0.9730608
## 2006-12-20 23:59:00       9.765972    0.9789830
## 2006-12-21 23:59:00       7.236806    0.9800750
```

```
tsfeatures::tsfeatures(daily_modeling_data)
```

```
## # A tibble: 6 x 16
##   frequency nperiods seasonal_period  trend    spike linearity curvature  e_acf1
##       <dbl>    <dbl>           <dbl>  <dbl>    <dbl>     <dbl>     <dbl>   <dbl>
## 1   0.00556        0         0.00556 0.425   6.32e-7     -3.73    2.12    0.326
## 2   0.00556        0         0.00556 0.523   7.01e-7      14.9    -6.77    0.757
## 3   0.00556        0         0.00556 0.0506  2.87e-6     -3.29    0.0294  0.0800
## 4   0.00556        0         0.00556 0.0410  2.13e-6     -5.61    2.08   -0.0819
## 5   0.00556        0         0.00556 0.291   5.86e-7      6.42   -0.842   0.255
## 6   0.00556        0         0.00556 0.625   5.62e-7      1.08    1.85    0.712
## # i 8 more variables: e_acf10 <dbl>, entropy <dbl>, x_acf1 <dbl>,
## #   x_acf10 <dbl>, diff1_acf1 <dbl>, diff1_acf10 <dbl>, diff2_acf1 <dbl>,
## #   diff2_acf10 <dbl>
```

```
index(daily_modeling_data) <- as.POSIXct(index(daily_modeling_data))
```

Adding additional features

```
daily_modeling_data$week_day <- as.factor(weekdays(index(daily_modeling_data)))
daily_modeling_data$month <- month(index(daily_modeling_data))
daily_modeling_data$seasonal <- ts_data_daily_decomposed$seasonal
daily_modeling_data$trend <- ts_data_daily_decomposed$trend
daily_modeling_data$year <- as.numeric(year(index(daily_modeling_data)))
head(daily_modeling_data)
```

```
## Warning: object timezone (UTC) is different from system timezone ()
```

```
##                     Global_active_power  Voltage Sub_metering_1 Sub_metering_2
## 2006-12-16 23:59:00            3.053475 236.2438      0.0000000       1.378788
## 2006-12-17 23:59:00            2.354486 240.0870      1.4118056       2.907639
## 2006-12-18 23:59:00            1.530435 241.2317      0.7381944       1.820139
## 2006-12-19 23:59:00            1.157079 241.9993      0.5826389       5.279167
## 2006-12-20 23:59:00            1.545658 242.3081      0.0000000       1.838889
## 2006-12-21 23:59:00            1.192440 241.0419      1.2256944       1.821528
##                     Sub_metering_3 power_factor week_day month  seasonal trend
## 2006-12-16 23:59:00      12.439394    0.9987684        3    12 0.4161188    NA
## 2006-12-17 23:59:00       9.264583    0.9884013        4    12 0.0673147    NA
```

```
## 2006-12-18 23:59:00       9.734722    0.9768335        2    12 0.4024407    NA
## 2006-12-19 23:59:00       4.303472    0.9730608        6    12 0.2258670    NA
## 2006-12-20 23:59:00       9.765972    0.9789830        7    12 0.3086377    NA
## 2006-12-21 23:59:00       7.236806    0.9800750        5    12 0.1540841    NA
##                      year
## 2006-12-16 23:59:00 2006
## 2006-12-17 23:59:00 2006
## 2006-12-18 23:59:00 2006
## 2006-12-19 23:59:00 2006
## 2006-12-20 23:59:00 2006
## 2006-12-21 23:59:00 2006
```

Modeling

- We will model only the daily data due to the lack small ammount of data we have in the monthly dataset

Converting the data into a dataframe

```
daily_modeling_data_as_df <- data.frame(daily_modeling_data)
daily_modeling_data_as_df$date <- time(daily_modeling_data)
daily_modeling_data_as_df <- daily_modeling_data_as_df %>%
  select(date,year,month,week_day,Global_active_power, everything()) %>%
  mutate(date = as.Date(date))
head(daily_modeling_data_as_df)
```

```
##                          date year month week_day Global_active_power  Voltage
## 2006-12-16 23:59:00 2006-12-16 2006    12        3            3.053475 236.2438
## 2006-12-17 23:59:00 2006-12-17 2006    12        4            2.354486 240.0870
## 2006-12-18 23:59:00 2006-12-18 2006    12        2            1.530435 241.2317
## 2006-12-19 23:59:00 2006-12-19 2006    12        6            1.157079 241.9993
## 2006-12-20 23:59:00 2006-12-20 2006    12        7            1.545658 242.3081
## 2006-12-21 23:59:00 2006-12-21 2006    12        5            1.192440 241.0419
##                     Sub_metering_1 Sub_metering_2 Sub_metering_3 power_factor
## 2006-12-16 23:59:00      0.0000000       1.378788      12.439394    0.9987684
## 2006-12-17 23:59:00      1.4118056       2.907639       9.264583    0.9884013
## 2006-12-18 23:59:00      0.7381944       1.820139       9.734722    0.9768335
## 2006-12-19 23:59:00      0.5826389       5.279167       4.303472    0.9730608
## 2006-12-20 23:59:00      0.0000000       1.838889       9.765972    0.9789830
## 2006-12-21 23:59:00      1.2256944       1.821528       7.236806    0.9800750
##                      seasonal trend
## 2006-12-16 23:59:00 0.4161188    NA
## 2006-12-17 23:59:00 0.0673147    NA
## 2006-12-18 23:59:00 0.4024407    NA
## 2006-12-19 23:59:00 0.2258670    NA
## 2006-12-20 23:59:00 0.3086377    NA
## 2006-12-21 23:59:00 0.1540841    NA
```

```r
dim(daily_modeling_data_as_df)
```

```
## [1] 1442   12
```

```r
daily_modeling_data_as_df %>%
  map(~sum(is.na(.)))
```

```
## $date
## [1] 0
##
## $year
## [1] 0
##
## $month
## [1] 0
##
## $week_day
## [1] 0
##
## $Global_active_power
## [1] 0
##
## $Voltage
## [1] 0
##
## $Sub_metering_1
## [1] 0
##
## $Sub_metering_2
## [1] 0
##
## $Sub_metering_3
## [1] 0
##
## $power_factor
## [1] 0
##
## $seasonal
## [1] 0
##
## $trend
## [1] 364
```

- Adding the trend as a feature resulted of having missing values at the beggining and at the end of the series due to the way of calculating the trend through moving average.

```r
cleaned_data <- daily_modeling_data_as_df %>%
  na.omit()
cleaned_data %>%
  map(~sum(is.na(.)))
```

```
## $date
## [1] 0
##
## $year
## [1] 0
##
## $month
## [1] 0
##
## $week_day
## [1] 0
##
## $Global_active_power
## [1] 0
##
## $Voltage
## [1] 0
##
## $Sub_metering_1
## [1] 0
##
## $Sub_metering_2
## [1] 0
##
## $Sub_metering_3
## [1] 0
##
## $power_factor
## [1] 0
##
## $seasonal
## [1] 0
##
## $trend
## [1] 0
```

```
head(cleaned_data)
```

```
##                          date year month week_day Global_active_power  Voltage
## 2007-06-16 23:59:00 2007-06-16 2007     6        3           1.4012236 239.5017
## 2007-06-17 23:59:00 2007-06-17 2007     6        4           1.3521458 240.2192
## 2007-06-18 23:59:00 2007-06-18 2007     6        2           0.4711625 240.7762
## 2007-06-19 23:59:00 2007-06-19 2007     6        6           0.5870181 240.2671
## 2007-06-20 23:59:00 2007-06-20 2007     6        7           0.7460319 240.1562
## 2007-06-21 23:59:00 2007-06-21 2007     6        5           0.5603792 240.5749
##                     Sub_metering_1 Sub_metering_2 Sub_metering_3 power_factor
## 2007-06-16 23:59:00      2.2840278      3.1236111       7.713889    0.9499591
## 2007-06-17 23:59:00      2.8638889      3.4986111       6.065278    0.9544926
## 2007-06-18 23:59:00      0.0000000      0.3673611       2.945139    0.9023998
## 2007-06-19 23:59:00      0.6756944      0.3729167       3.045833    0.9209419
## 2007-06-20 23:59:00      1.2229167      1.7972222       3.545833    0.9165838
## 2007-06-21 23:59:00      0.6472222      1.6666667       2.577083    0.8821845
##                          seasonal    trend
## 2007-06-16 23:59:00  0.294219855 1.122003
```

```
## 2007-06-17 23:59:00  0.007263443 1.118893
## 2007-06-18 23:59:00 -0.442246890 1.115864
## 2007-06-19 23:59:00 -0.188919744 1.116662
## 2007-06-20 23:59:00 -0.224937129 1.117670
## 2007-06-21 23:59:00 -0.261748548 1.117145
```

Modeling with tidymodels

```
library(tidymodels)
```

Splitting the data

- Because our data is a time series we will use initial_time_split function so it would make a time based split.

```
splitted_data <- initial_time_split(cleaned_data)
train_data <- training(splitted_data)
test_data <- testing(splitted_data)
```

```
dim(train_data)
```

```
## [1] 808  12
```

```
dim(test_data)
```

```
## [1] 270  12
```

Setting the models

- Instead of processing the data we can set a workflow for each model ### Random forest

```
randomf_model <- rand_forest() %>%
  set_engine("ranger") %>%
  set_mode("regression")


randomf_workflow <- workflow() %>%
  add_recipe(recipe(Global_active_power ~ ., data = train_data) %>%
               step_normalize(all_numeric_predictors()) %>%
               step_dummy(all_factor_predictors())) %>%
  add_model(randomf_model)
```

XGboost model

```
library(xgboost)
```

```
## Warning: le package 'xgboost' a été compilé avec la version R 4.3.3

##
## Attachement du package : 'xgboost'

## L'objet suivant est masqué depuis 'package:dplyr':
##
##     slice
```

```
xgboost_model <- boost_tree(trees = 1000,
                            tree_depth = 6,
                            learn_rate = 0.1,
                            loss_reduction = 0.01,
                            sample_size = 0.8,
                            mtry = 2) %>%
  set_engine("xgboost") %>%
  set_mode("regression")

xgboost_workflow <- workflow() %>%
  add_recipe(recipe(Global_active_power ~ ., data = train_data) %>%
               step_rm(date) %>%
               step_normalize(all_numeric_predictors()) %>%
               step_dummy(all_factor_predictors())) %>%
  add_model(xgboost_model)
```

Fitting the models

Random forest models

```
randomf_fit <- fit(randomf_workflow, data = train_data)
randomf_predictions <- predict(randomf_fit, test_data)
```

XGboost model

```
xgboost_fit <- fit(xgboost_workflow, data = train_data)
xgboost_predictions <- predict(xgboost_fit, test_data)
```

Evaluating The Models

```
randomf_rmse <- randomf_predictions %>%
  bind_cols(test_data) %>%
  rmse(truth = Global_active_power, estimate = .pred)

xgboost_rmse <- xgboost_predictions %>%
  bind_cols(test_data) %>%
  rmse(truth = Global_active_power, estimate = .pred)

randomf_rmse
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.128
```

```
xgboost_rmse
```

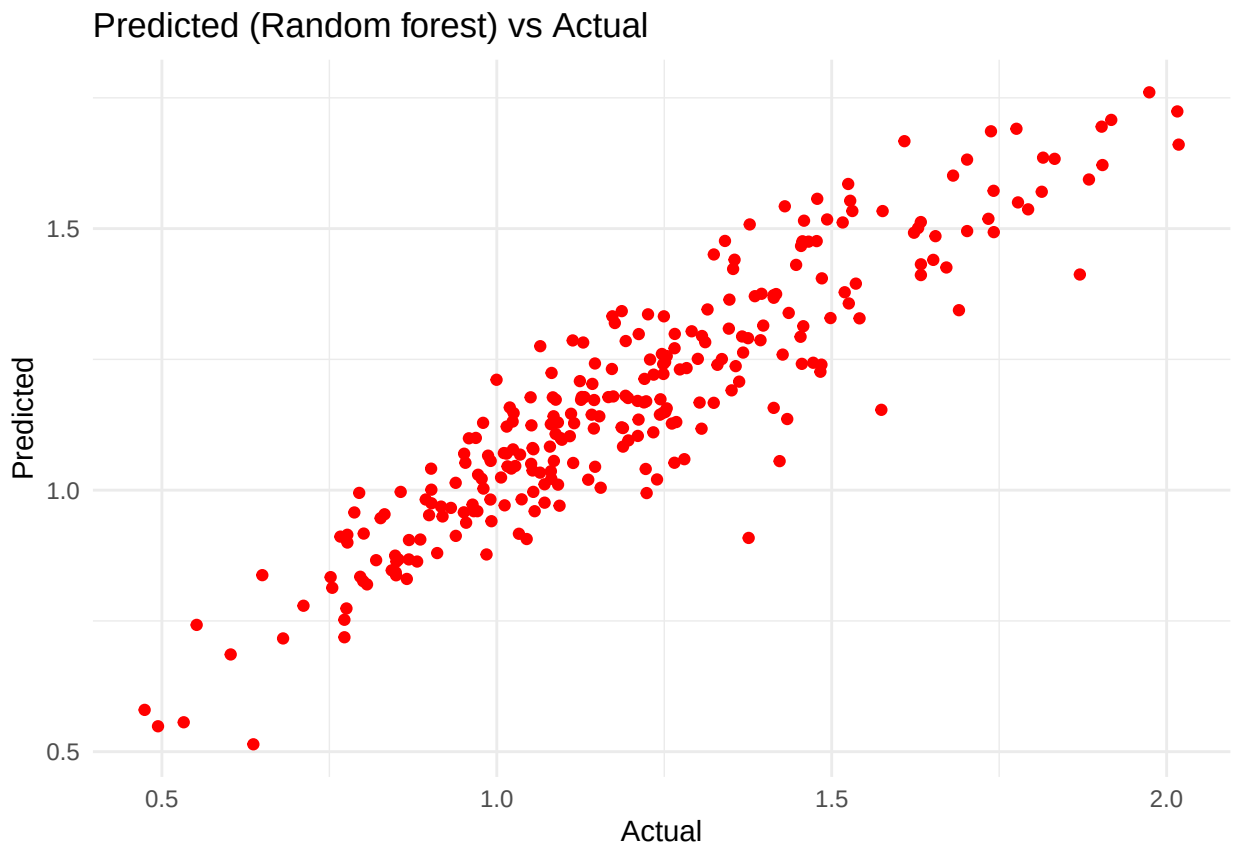
```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.116
```

Visualising the result

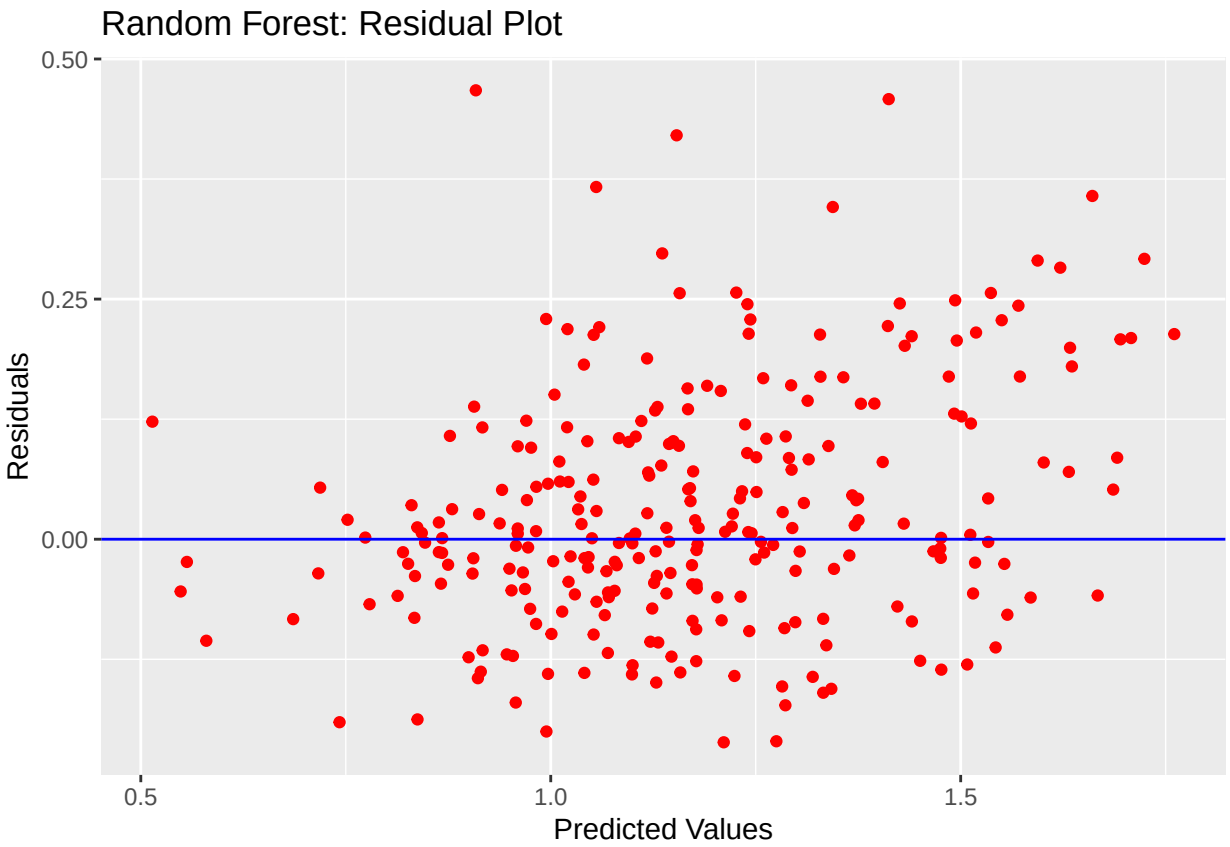
- Since the random forest model gave the best result we can proceed with it

```
results <- test_data %>%
  select(Global_active_power) %>%
  mutate(randomf_pred = randomf_predictions$.pred,
         xgboost_pred = xgboost_predictions$.pred,
         randomf_resid = Global_active_power - randomf_pred,
         xgboost_resid = Global_active_power - xgboost_pred)

results %>%
  ggplot(.,aes(x = Global_active_power , y = randomf_pred))+
  geom_point(color = "red")+
  theme_minimal()+
  labs(title = "Predicted (Random forest) vs Actual",x = "Actual",y = "Predicted")
```




```
ggplot(results, aes(x = randomf_pred, y = randomf_resid)) +
  geom_point(color = "red") +
  geom_hline(yintercept = 0, color = "blue") +
  labs(title = "Random Forest: Residual Plot",
       x = "Predicted Values",
       y = "Residuals")
```



- we can continue by applying some tuning or other techniques and then deploying the model with vetiver and other packages. thanks for reading to this point, if you have an advice or you spotted a mistake please be comfortable sharing it so we can all get better together .