Time Series - EXAM

Samd Guizani

2025-01-03

Assignment

- File 2023-11-Elec-train.xlsx contains electricity consumption (kW) and outdoor air temperature for a building., measured every 15 minutes, from 1/1/2010 1:15 to 2/20/2010 23:45.
- In addition, outdoor air temperature are available for 2/21/2010. The goal is to forecast electricity consumption (kW) for 2/21/2010.
- Two forecasts should be returned, in one Excel file entitled YourName.xlsx, with exactly two columns (one columns per forecast) and 96 rows:
    1. first one without using outdoor temperature
    2. the second one using outdoor temperature.

Working directory and imports

```
setwd("~/DSTI_MSc DS and AI/03-Advanced/03-Time Series/Exam")

library(readxl)
library(forecast)
```

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

```
library(funtimes) # trend tests
library(ggplot2)
library(imputeTS) # imputing missing data in a time series (interpolation)
library(randomForest)
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(xgboost)
library(e1071) # SVM
library(vars) # VAR model
```

```
## Loading required package: MASS
```

```
## Loading required package: strucchange
```

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

```
##
## Attaching package: 'zoo'
```

```
## The following object is masked from 'package:imputeTS':
##
##     na.locf
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
## Loading required package: urca
```

```
## Loading required package: lmtest
```

```
library(writexl)
```

Functions

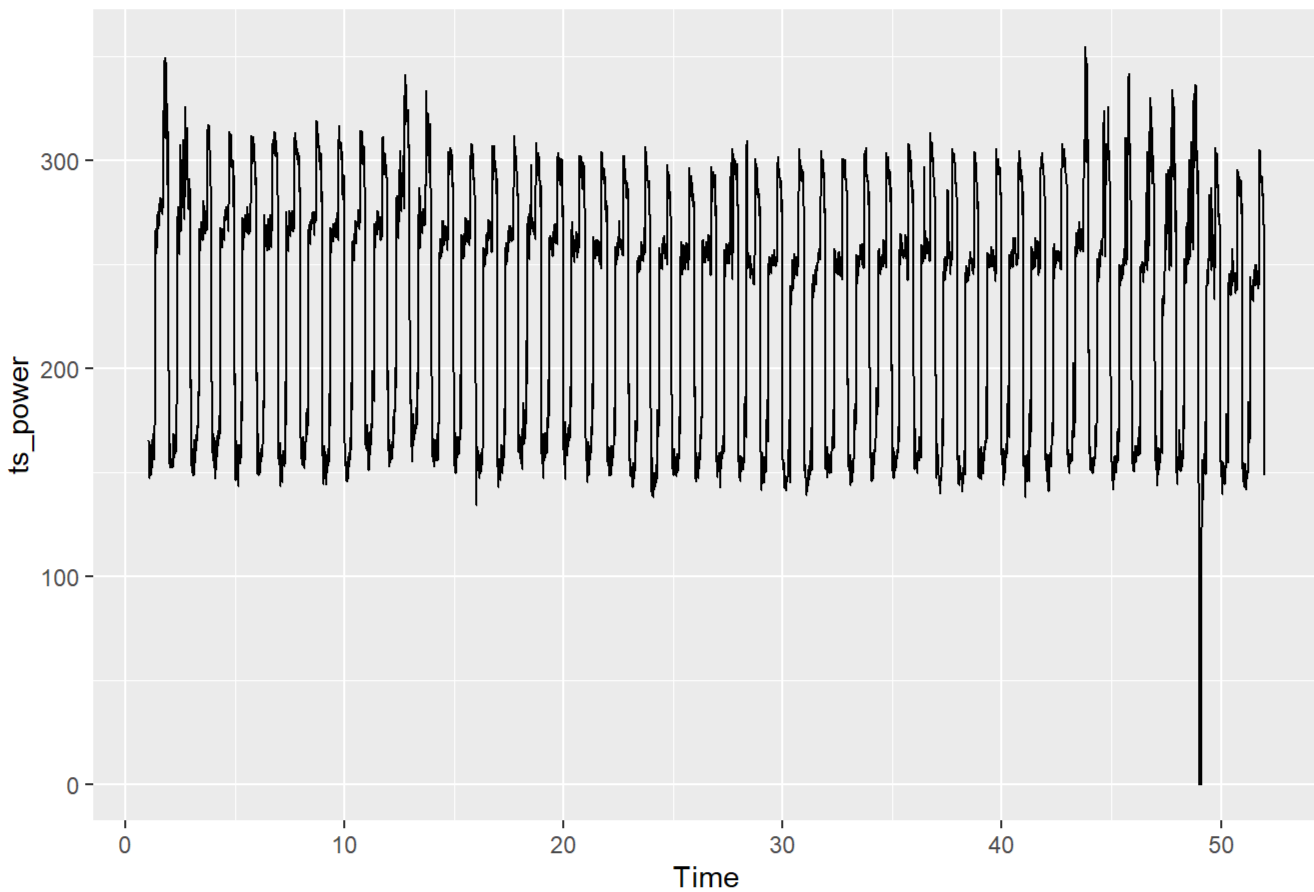
```r
## Forecast Time Series Using a Machine Learning Model

forecast_ML = function(fit, newdata, h)
  #' @description Time series forecast using a machine learning model (e.g., random forest or XG
Boost). Iteratively predicts future values based on previous predictions and updates the input d
ata matrix accordingly.
  #'
  #' @param fit A trained machine learning model (e.g., random forest, XGBoost) with a `predict`
method.
  #'
  #' @param newdata A matrix of shape (1 x n) used as input to make the initial prediction. The
matrix is updated iteratively for subsequent predictions.
  #'
  #' @param h An integer specifying the forecast horizon (number of future steps to predict).
  #'
  #' @return A numeric vector of length `h` containing the forecasted time series.
{
  prev = rep(NULL, h)
  for (t in 1:h) {
    prev[t] = predict(fit, newdata = newdata)
    newdata = matrix(c(newdata[-1], prev[t]), 1)
  }
  return(prev)
}


## Forecast Time Series Using a Machine Learning Model and covariates

forecast_ML_X = function(fit, newdata, h, xreg)
  #' @description Time series forecast using a machine learning model (e.g., random forest or XG
Boost). Iteratively predicts future values based on previous predictions and measured covariates
and updates the input data matrix accordingly.
  #'
  #' @param fit A trained machine learning model (e.g., random forest, XGBoost) with a `predict`
method.
  #'
  #' @param newdata A matrix of shape (1 x n) used as input to make the initial prediction. The
matrix is updated iteratively for subsequent predictions.
  #'
  #' @param h An integer specifying the forecast horizon (number of future steps to predict).
  #'
  #' @param xreg A matrix containing the observed covariates to be used for the prediction.
  #'
  #' @return A numeric vector of length `h` containing the forecasted time series.
{
  newdata = matrix(c(newdata, xreg[1]), 1)

  prev = rep(NULL, h)
  prev[1] = predict(fit, newdata = newdata)

  for (t in 2:h)
    {
```

```
    newdata = newdata[1,-1]
    newdata[h] = prev[t-1]
    newdata = matrix(c(newdata, xreg[t]), 1)
    prev[t] = predict(fit, newdata = newdata)
    }
  return(prev)
}


## Root Mean Squared Error

RMSE = function(y_act, y_prd, na.rm = FALSE)
  #' @description Computes Root Mean Squared Error (RMSE) between actual and predicted values.
  #'
  #' @param y_act Numeric vector. Actual observed values.
  #' @param y_prd Numeric vector. Predicted values.
  #' @param rm.na Bool.  Whether NA values should be stripped before the computation proceeds (D
efault = FALSE)
  #'
  #' @return Float. Representing RMSE.
{
  return(sqrt(mean((y_act - y_prd)^2, na.rm = na.rm)))
}
```

Load data and explore

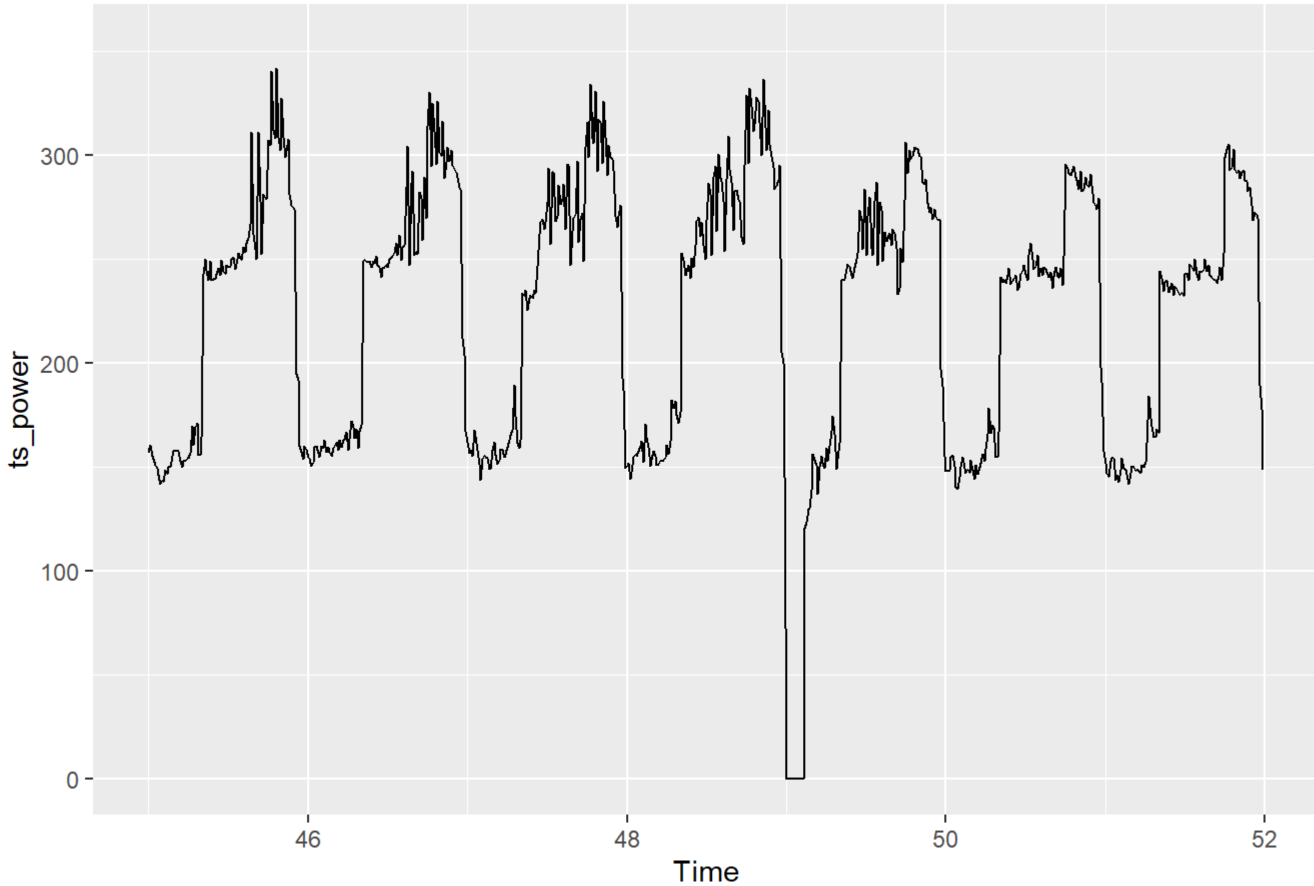
Plot time series and evaluate trends and seasonality patterns

```
data = read_excel('2023-11-Elec-train.xlsx')
data$Timestamp <- as.POSIXct(data[[1]], format = "%m/%d/%Y %H:%M")
data$Timestamp[1] <- as.POSIXct("1/1/2010 1:15", format = "%m/%d/%Y %H:%M") # fix import issue w
ith 1st timestamp.

ts_power = ts(data$`Power (kW)`[1:(dim(data)[1] - 96)], start = c(1,6), freq = 96) # last 96 obs
are NA, to be forecasted

autoplot(ts_power)
```

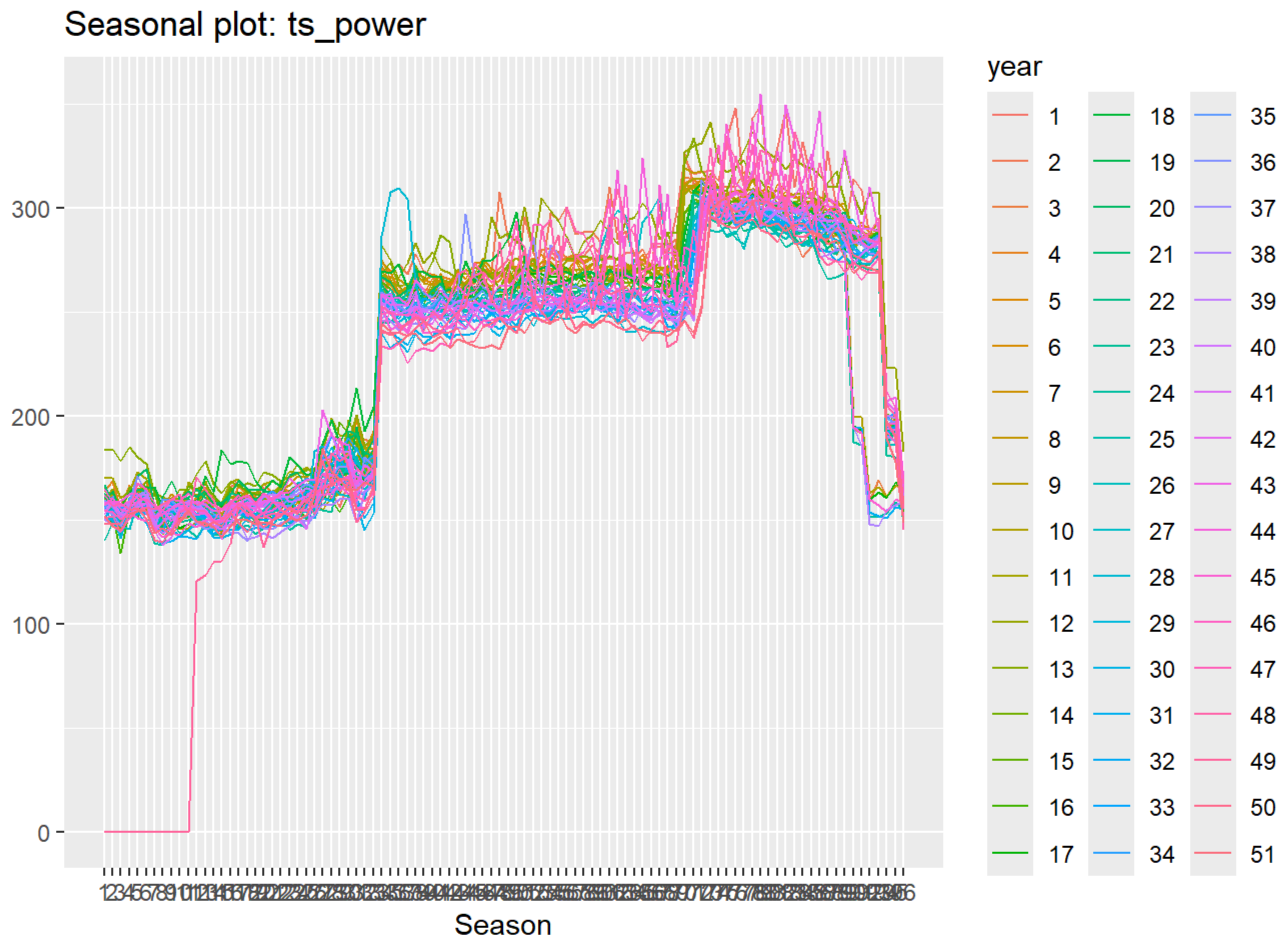



```
autoplot(ts_power) + xlim(c(45, 52)) # focus on unusual data
```

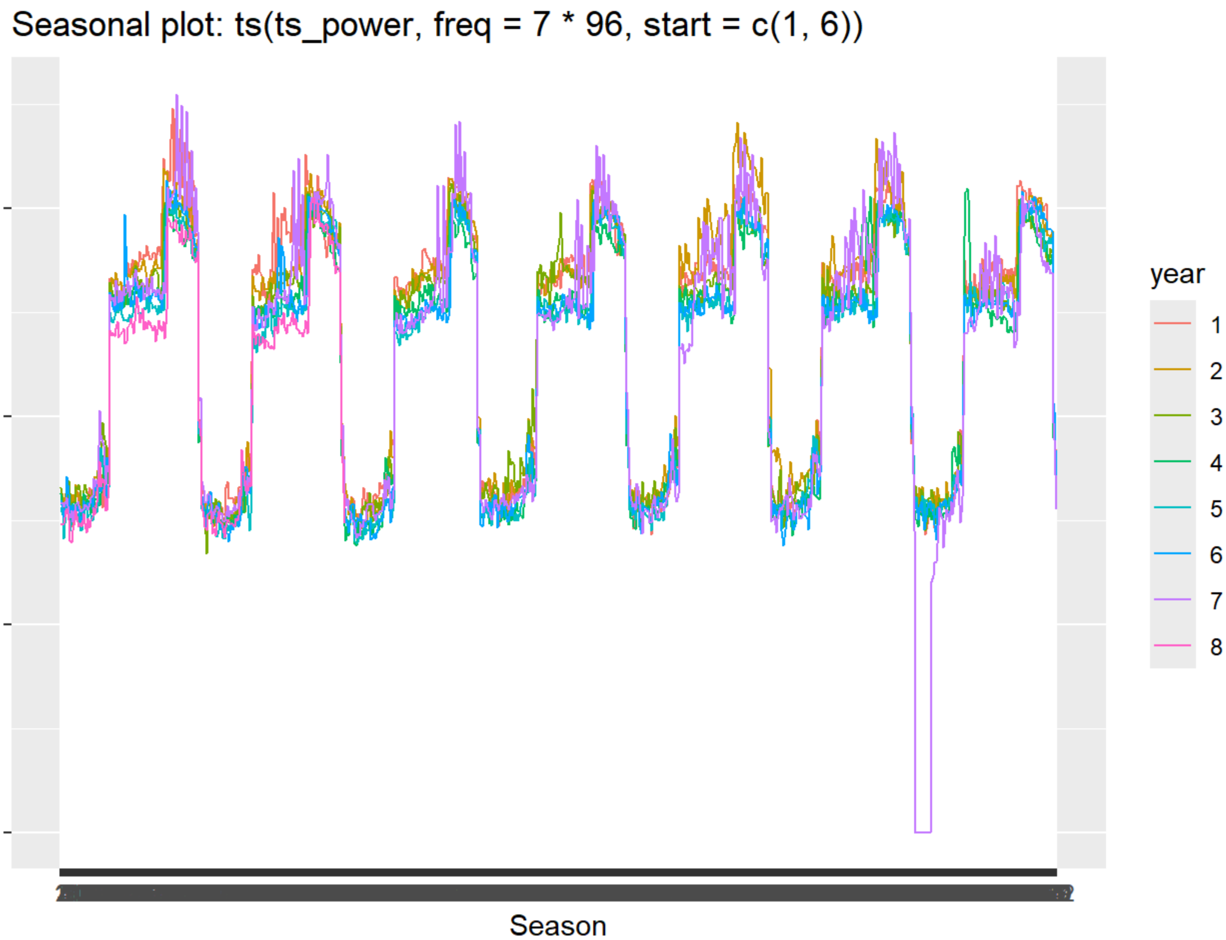
```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```



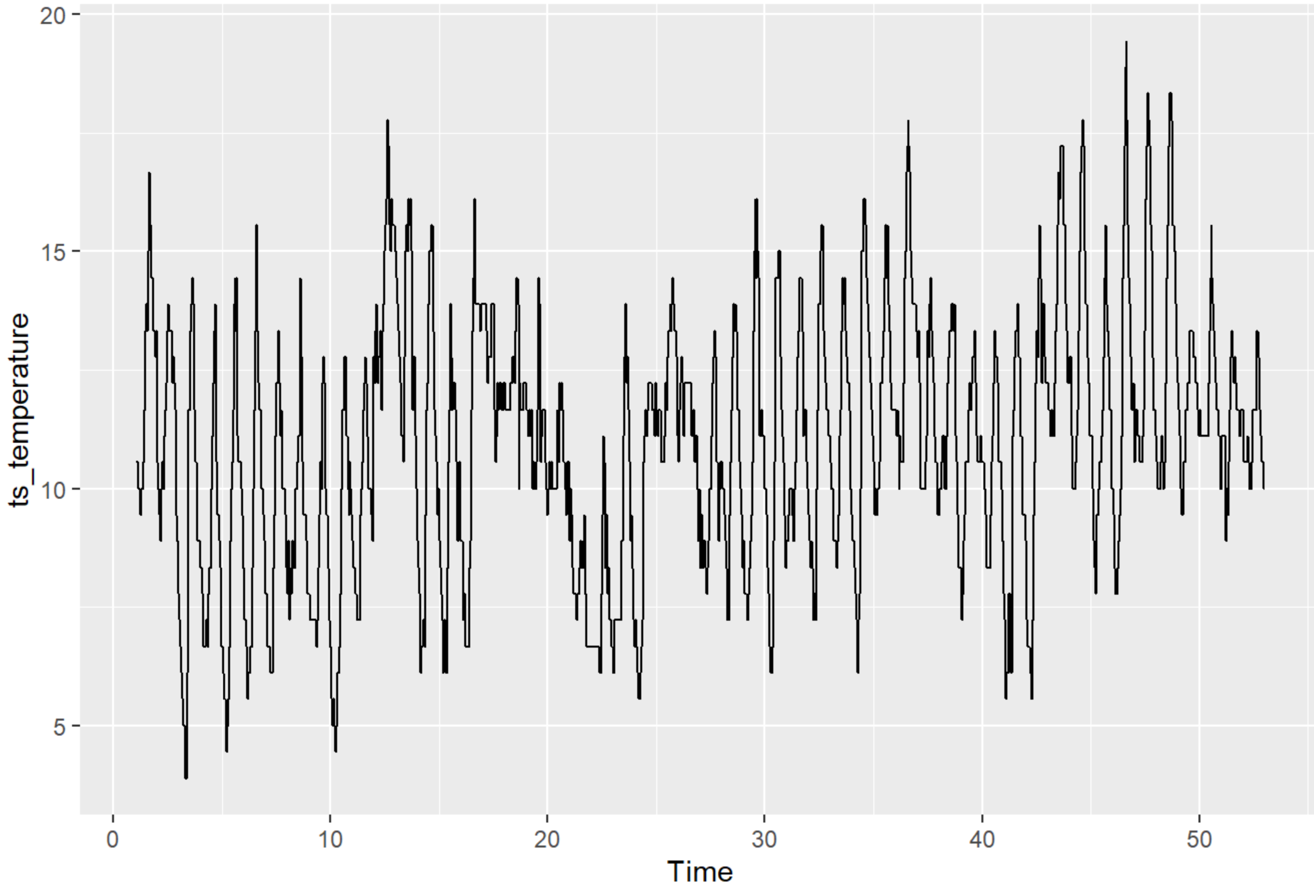

```
ggseasonplot(ts_power) # seasonal plot with daily period
```



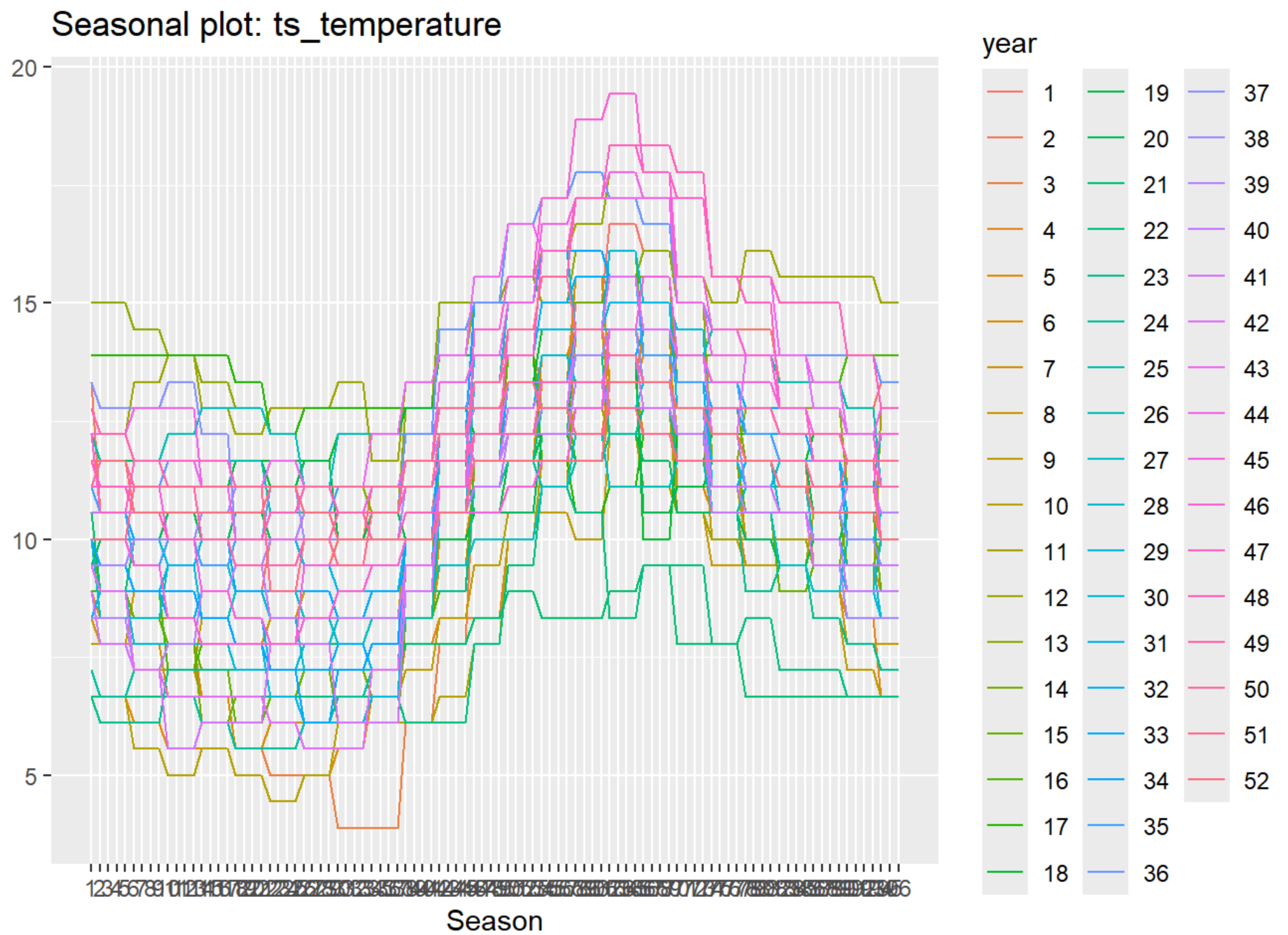

```r
ggseasonplot(ts(ts_power, freq = 7 * 96, start = c(1,6))) # seasonal plot with weekly period
```




```
ts_temperature = ts(data$`Temp (C°)`, start = c(1,6), freq = 96)
autoplot(ts_temperature)
```



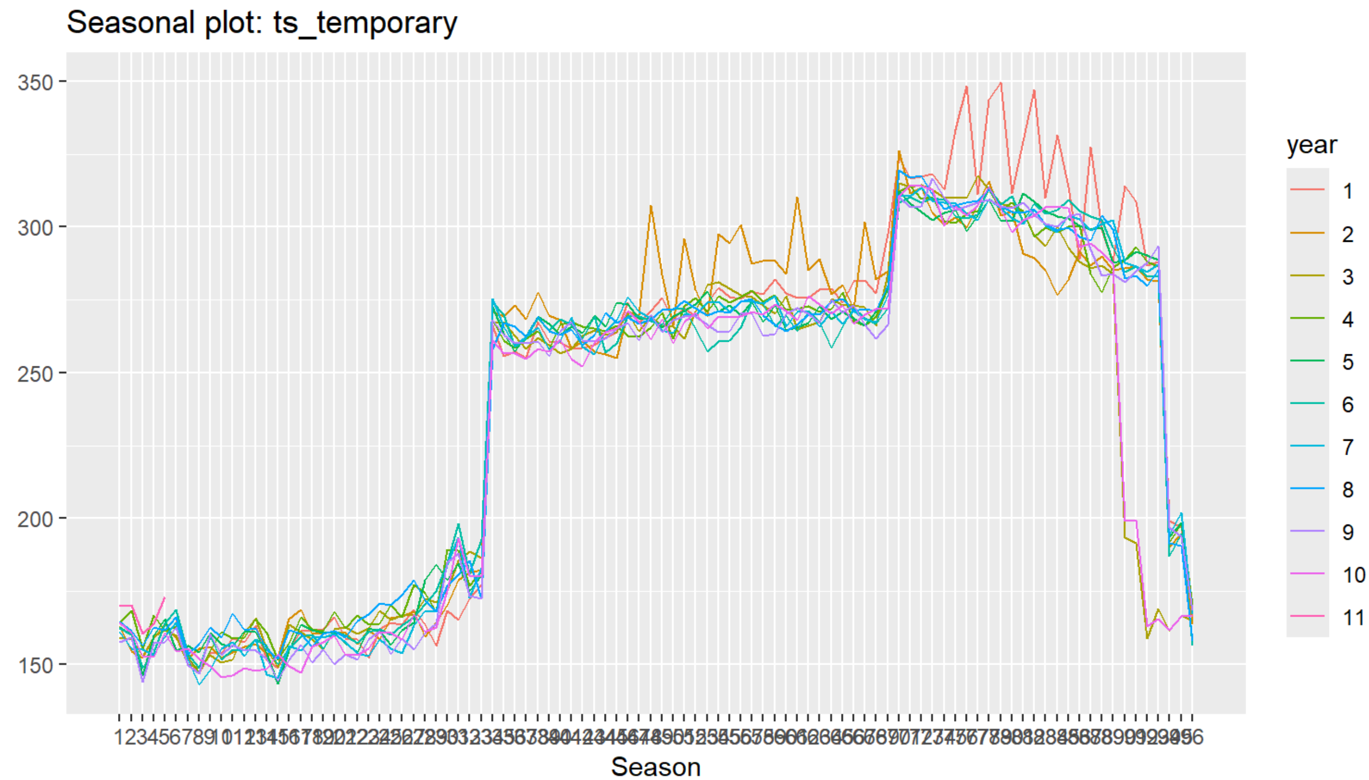

```
ggseasonplot(ts_temperature)
```



Notes:

- Power shows a daily and weekly periodic pattern. Possibly a slight decreasing trend. Variance seems constant over time. Unusual zero values on day 49 (i.e. 2/18/2010) and unusual peak of power consumption on day 28 (i.e. 1/28/2010).
- Temperature shows a daily periodic pattern and an increasing trend.

```
n_days = 10
ts_temporary = ts(data$`Power (kW)`[1:(96*n_days)], start = c(1,6), freq = 96)
ggseasonplot(ts_temporary) #+ xlim(0.75,1)
```



Notes:

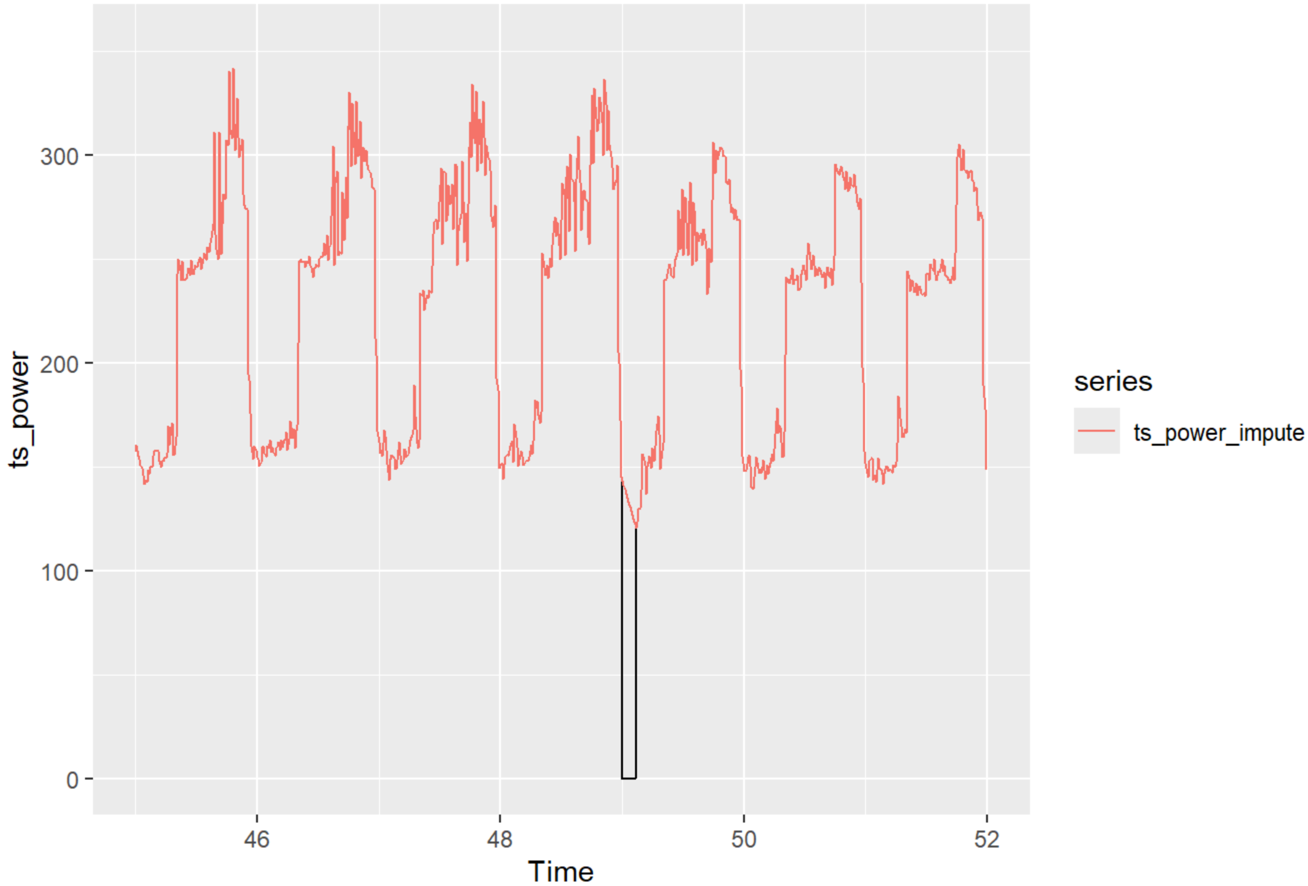
- Power daily pattern is comparable during 6 of the week days, but the 7th day has a specific pattern (earlier decrease on 01/03/2010, 01/10/2010, 01/17/2010, etc…)

Replace unusual values by interpolation

```
# focus on power values at zero
loc_0s = which(ts_power == 0)
ts_power_impute = ts_power
ts_power_impute[loc_0s] = NA
ts_power_impute = imputeTS::na_interpolation(ts_power_impute, option = 'linear')

autoplot(ts_power) +
  autolayer(ts_power_impute) +
  xlim(c(45, 52))
```

```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```

```
## Warning: Removed 4219 rows containing missing values or values outside the scale range
## (`geom_line()`).
```




```
ts_power_impute[loc_0s]
```

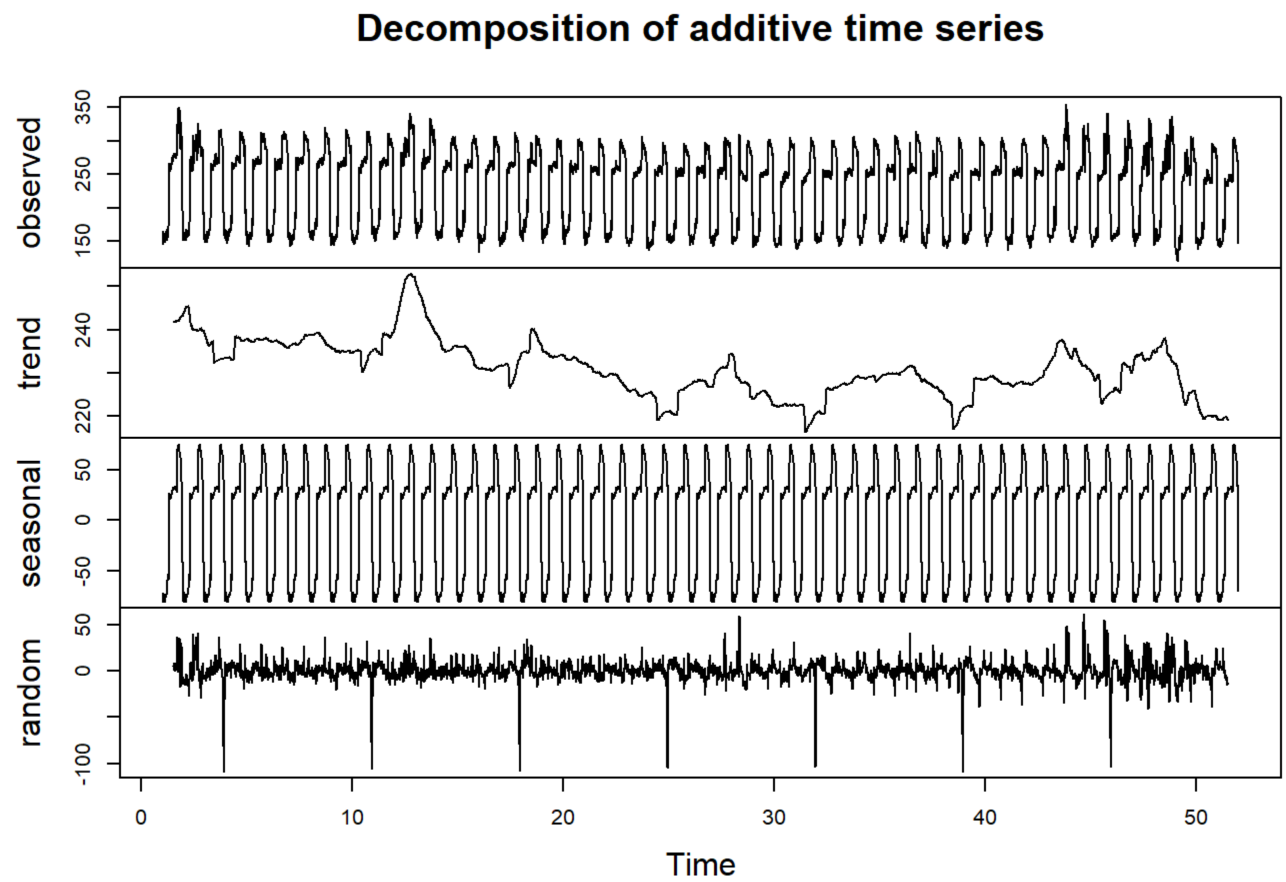
```
##  [1] 143.2417 141.1833 139.1250 137.0667 135.0083 132.9500 130.8917 128.8333
##  [9] 126.7750 124.7167 122.6583
```

Notes:

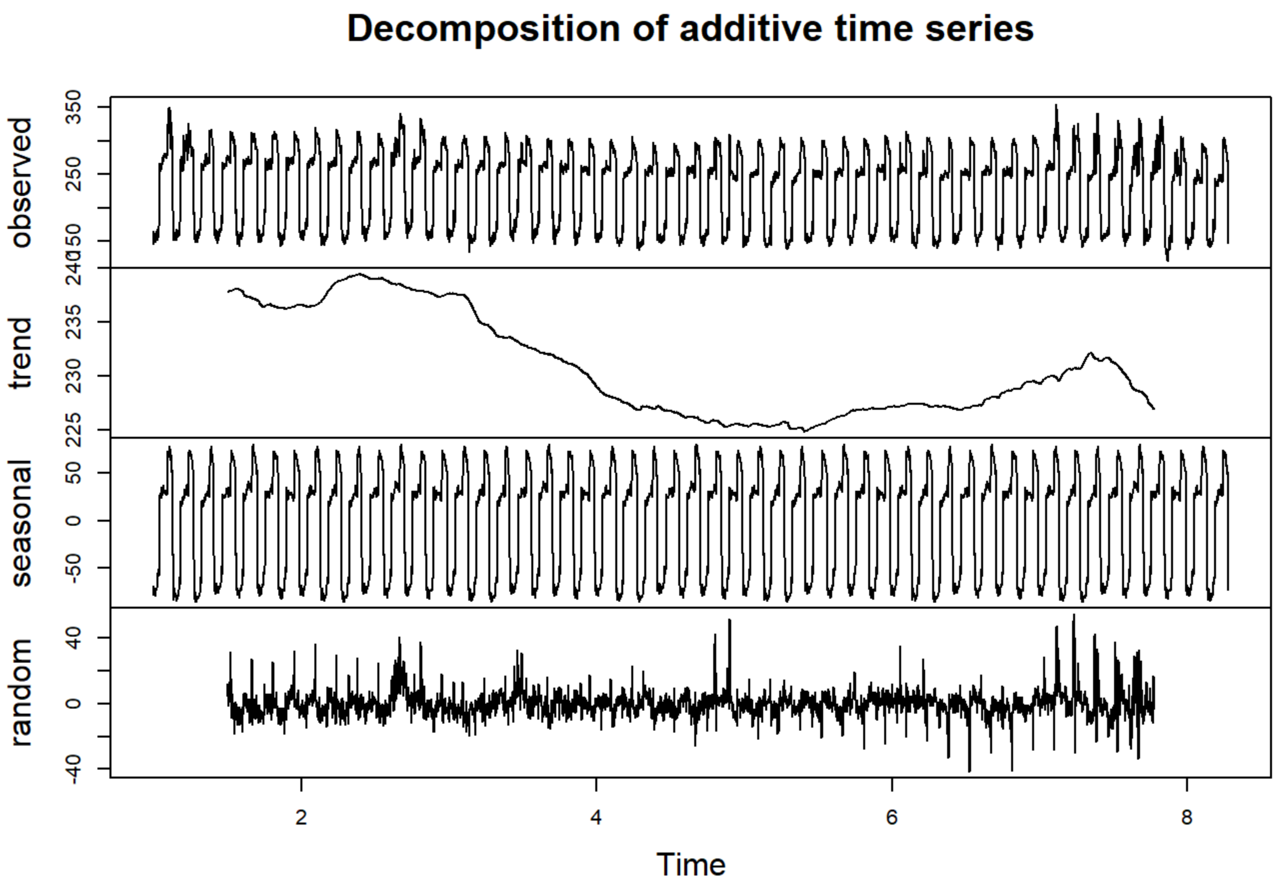
- Replacing 0 values by interpolated values seems reasonable.

Time series decomposition and differentiating

```
plot(decompose(ts_power_impute)) # daily period
```



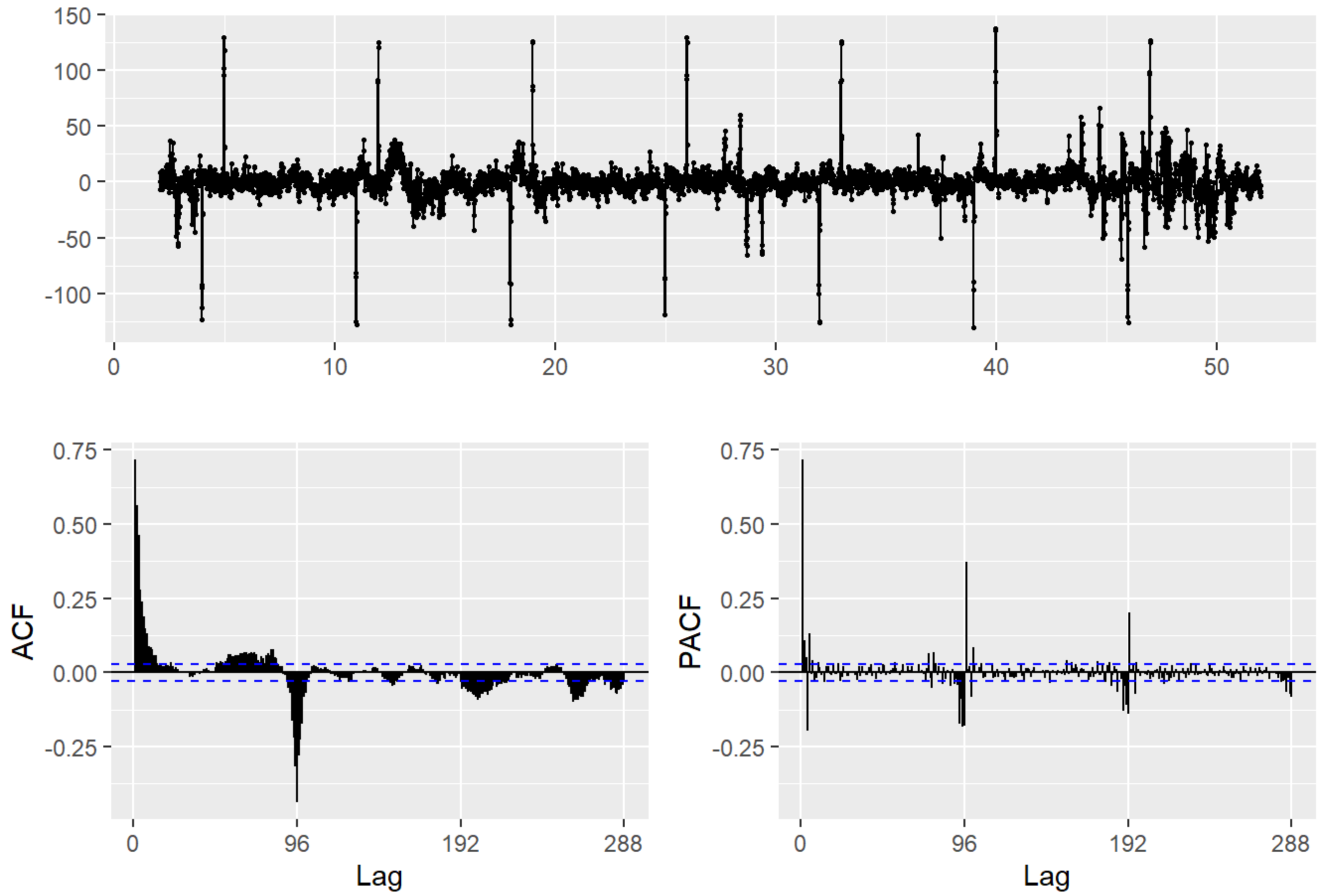

```
plot(decompose(ts(ts_power_impute, frequency = 7*96))) # weekly period
```



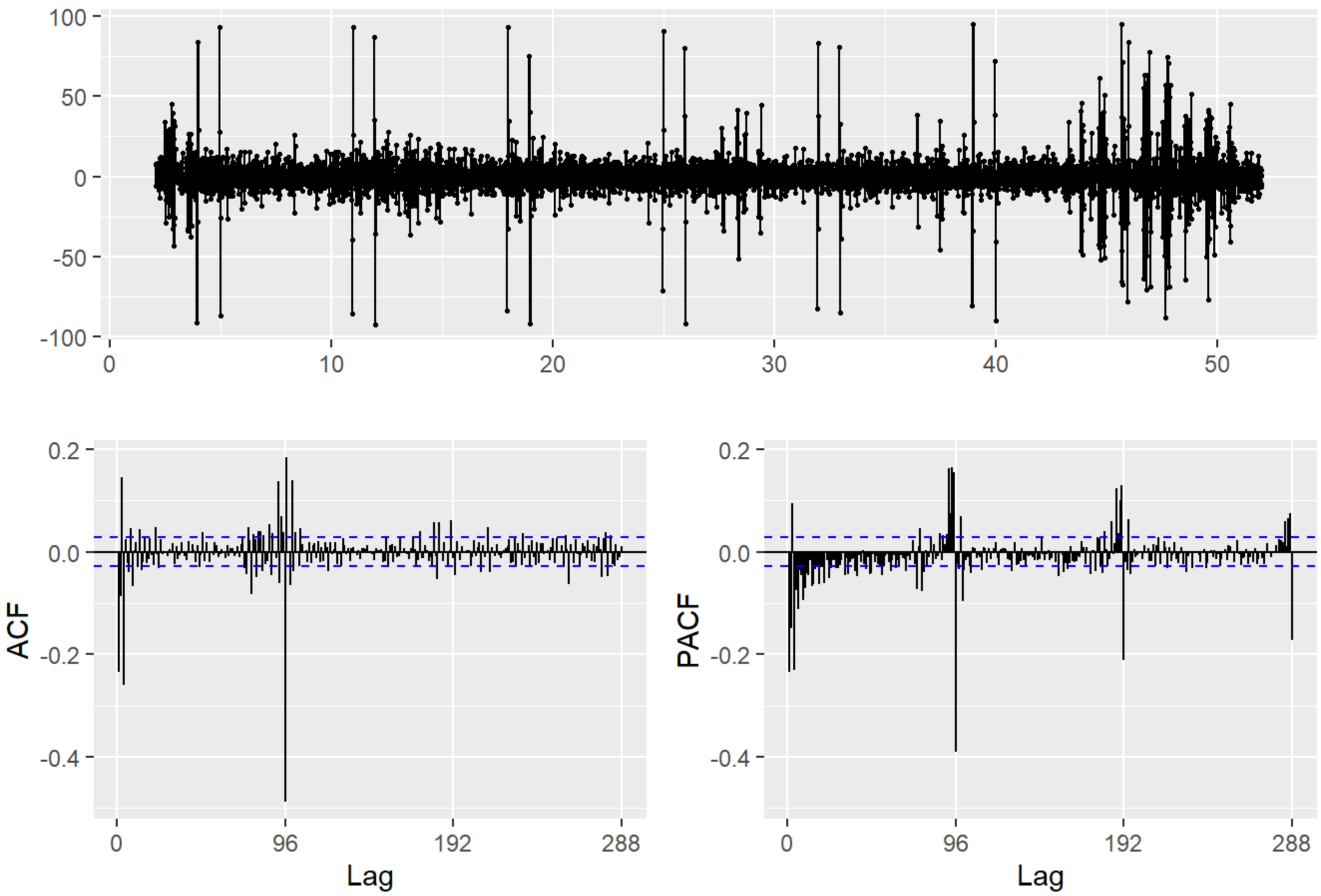
Notes:

- Decomposing based on **daily** period: still a seasonal pattern in the random series (period of 7 days, i.e. weekly) as well as in the trend component.
- Decomposing based on **weekly** period: trend component looks smooth with no seasonal pattern. Random component still shows daily pattern (-> information need to be modeled)

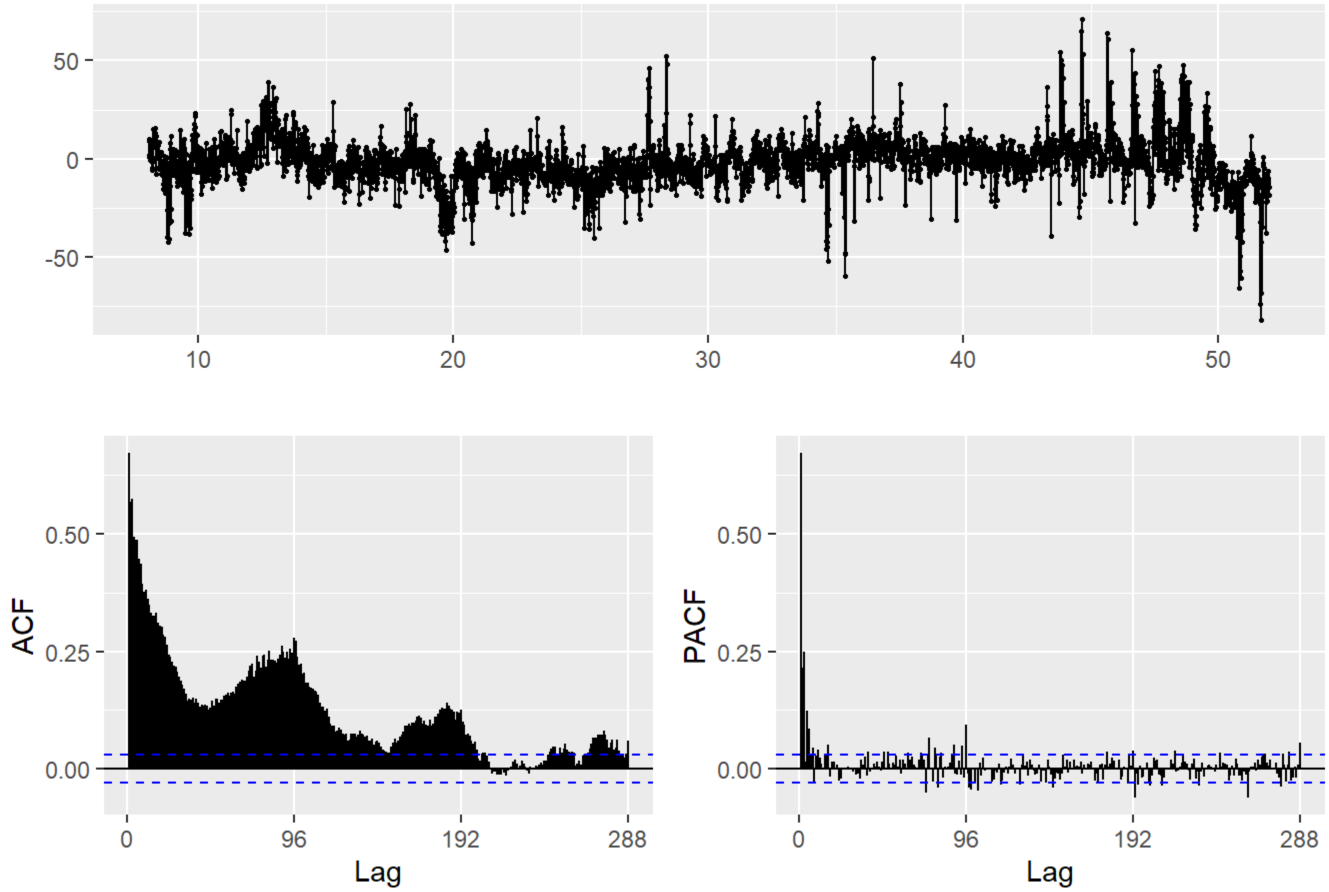
```
ggtsdisplay(diff(ts_power_impute, lag = 96, differences = 1)) # daily period
```



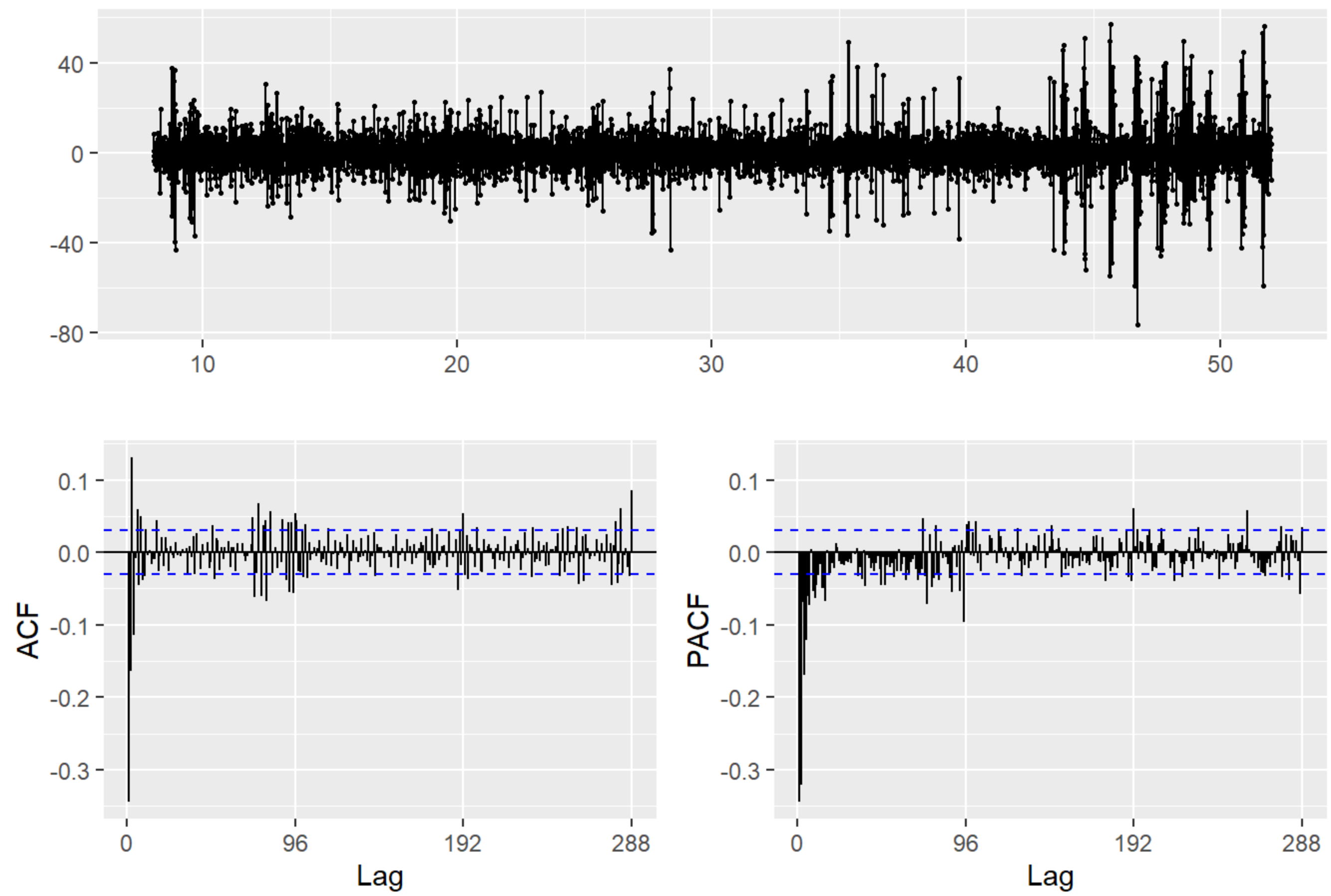

```
ggtsdisplay(diff(diff(ts_power_impute, lag = 96, differences = 1),
                 lag = 1,
                 differences = 1)) # daily period + diff with lag 1
```




```
ggtsdisplay(diff(ts_power_impute, lag = 7 * 96, differences = 1)) # weekly period
```




```
ggtsdisplay(diff(diff(ts_power_impute, lag = 7 * 96, differences = ),
                 lag = 1,
                 differences = 1)) # weekly period + diff with lag 1
```



Notes:

- Differentiating with a lag = 1 day period: still observe a weekly seasonal pattern (see time series plot)
- Differentiating twice (with a lag = 1 day period + lag = 1 for de-trending): still observe a weekly seasonal pattern (see time series plot)
- Differentiating with a lag = 1 week period: periodic pattern no longer observed, but a trend is still visible (see time series plot).
- Differentiating twice (with a lag = 1 week period + lag = 1 for de-trending): time series centered on 0, no visible trend. ACF/PACF show significant autocorrelation values (-> information to be modeled)

Modeling, without co-variates

```
# Converting ts_power_impute to daily period
y_daily = ts(ts_power_impute, start = c(1,6), frequency = 96)
y_daily_train = head(y_daily, length(y_daily) - 96)
y_daily_test = tail(y_daily, 96) # last day kept as test set

# Converting ts_power_impute to weekly period
y_weekly = ts(ts_power_impute, start = c(1,6), frequency = 7*96)
y_weekly_train = head(y_weekly, length(y_weekly) - 96)
y_weekly_test = tail(y_weekly, 96) # last day kept as test set
```

Holt-Winters, Daily period

```
# Code commented: Holt-Winters model failed to be fitted due to too large number of lags per per
iod (96).

# exec_t_start = Sys.time()
#
# fit = hw(y_daily_train, h=96, seasonal = "additive")
# fit |>  summary()
#
# ggtsdisplay(fit$residuals)
# checkresiduals(fit, plot = TRUE)
#
# exec_t_end = Sys.time()
# print(exec_t_end - exec_t_start)
```

Notes:

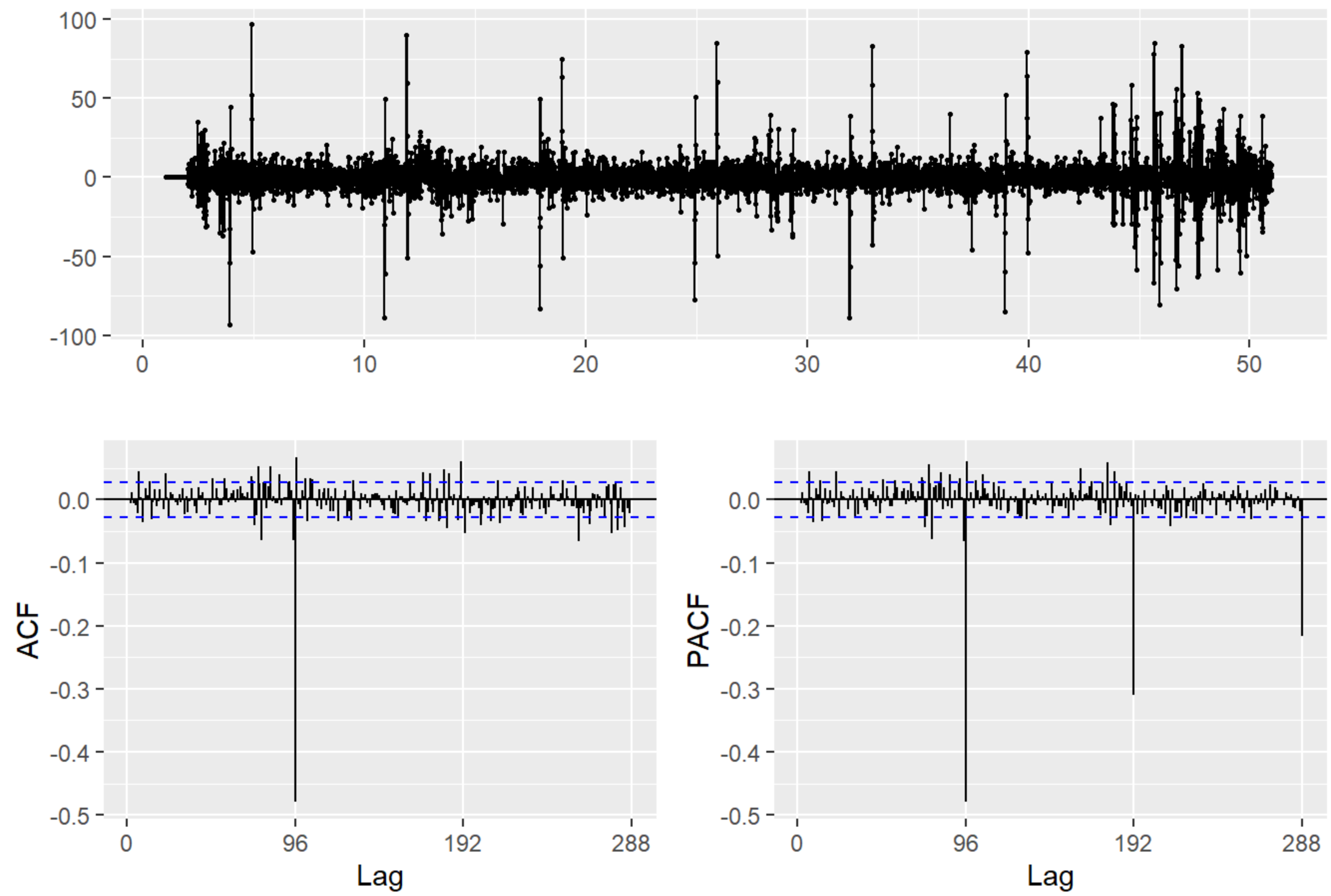
- Holt-Winters model fitting fails due to too high frequency (96).

SARIMA (auto), Daily period

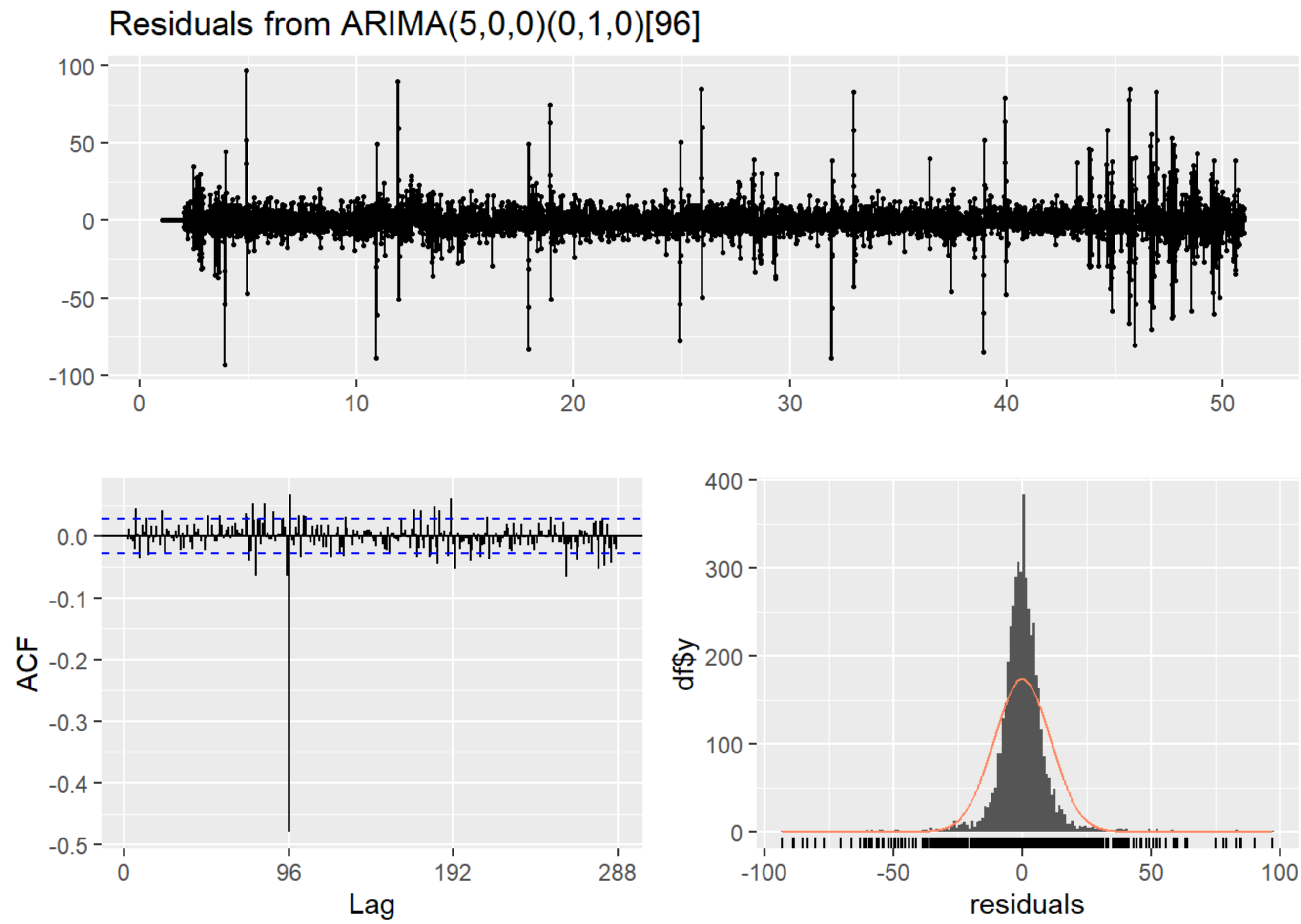
```
# Auto SARIMA, daily period
exec_t_start = Sys.time()

fit = auto.arima(y_daily_train)
fit |>  summary()
```

```
## Series: y_daily_train
## ARIMA(5,0,0)(0,1,0)[96]
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5
##       0.6705  0.0671  0.1623  -0.2823  0.1330
## s.e.  0.0145  0.0170  0.0168   0.0170  0.0145
##
## sigma^2 = 122.7:  log likelihood = -17966.78
## AIC=35945.56   AICc=35945.58   BIC=35984.29
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.1036317 10.96142 6.457121 -0.1529683 2.921813 0.7344067
##                      ACF1
## Training set 0.0005224987
```

```
ggtsdisplay(fit$residuals)
```




```
checkresiduals(fit, plot = TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,0,0)(0,1,0)[96]
## Q* = 1540.2, df = 187, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 192
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 1.057395 mins
```

```
# saveRDS(fit, file = "ARIMA_auto_(5,0,0)(0,1,0)[96].rds")
```

Notes:

- ACF shows significant autocorrelation at 96 (= 1 day period) and PACF shows exponentially decreasing autocorrelation for daily periods -> try adding seasonal MA (Q = 1)
- Some autocorrelation values are significant within the 1st period on ACF and PACF - > try changing the order p and q

Cross-validation

```
# Code commented: very long computation

# # Forcasting function to cross-validate
# Arima_ <- function(x, h) {
#   forecast(Arima(x,
#                  order=c(5,0,0),
#                  seasonal = c(0,1,0)
#                  ))
# }
#
# # Crossvalidation execution
# exec_t_start = Sys.time()
#
# e <- tsCV(y_daily, Arima_, h=96, window = 4795)
#
# exec_t_end = Sys.time()
# print(exec_t_end - exec_t_start)
#
# print(paste0("Cross-validation RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
```

Notes:

Cross-validation:

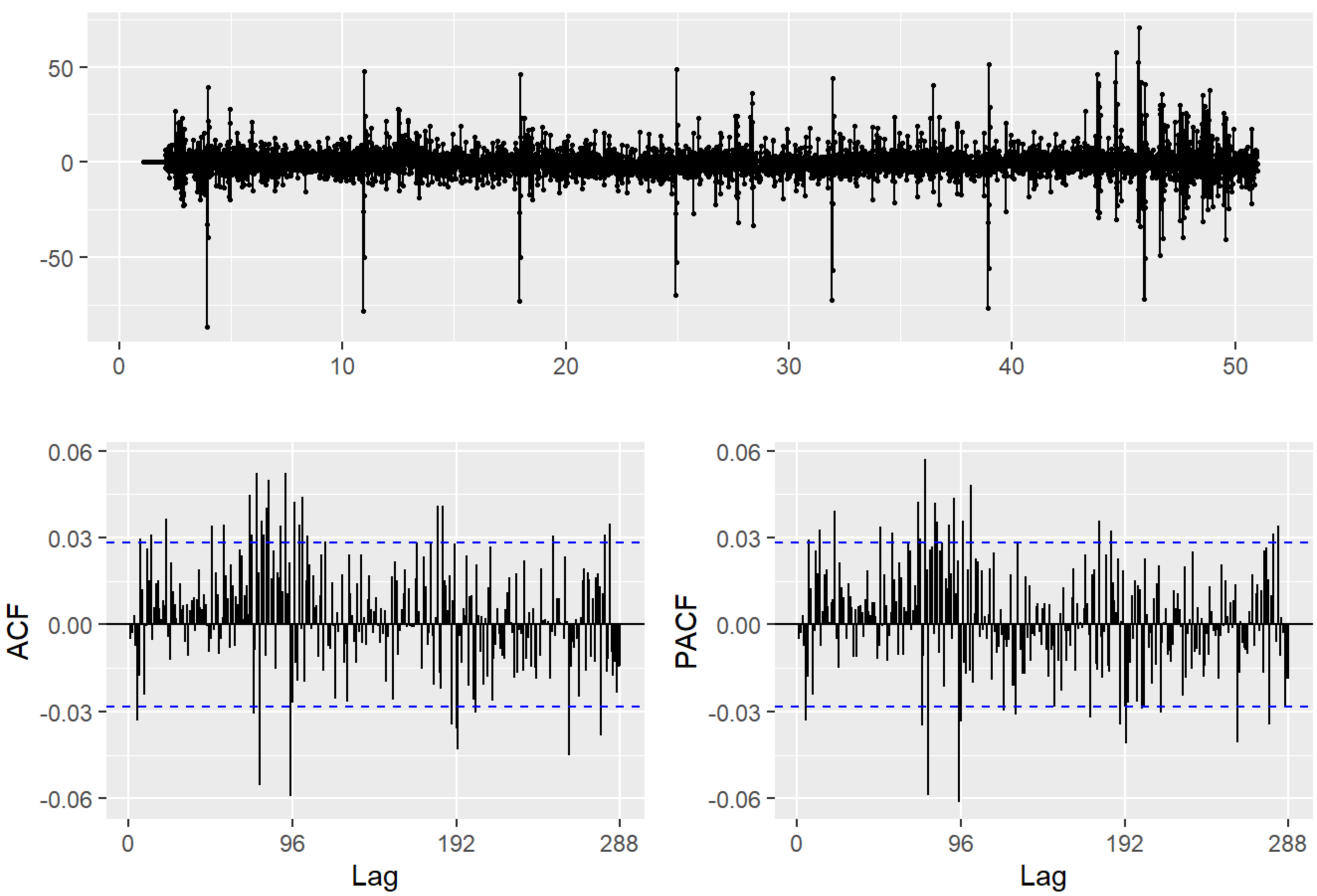
```
Time difference of 47.4164 mins
[1] "Cross-validation RMSE: 6.38355803349186"
```

SARIMA (manual), Daily period

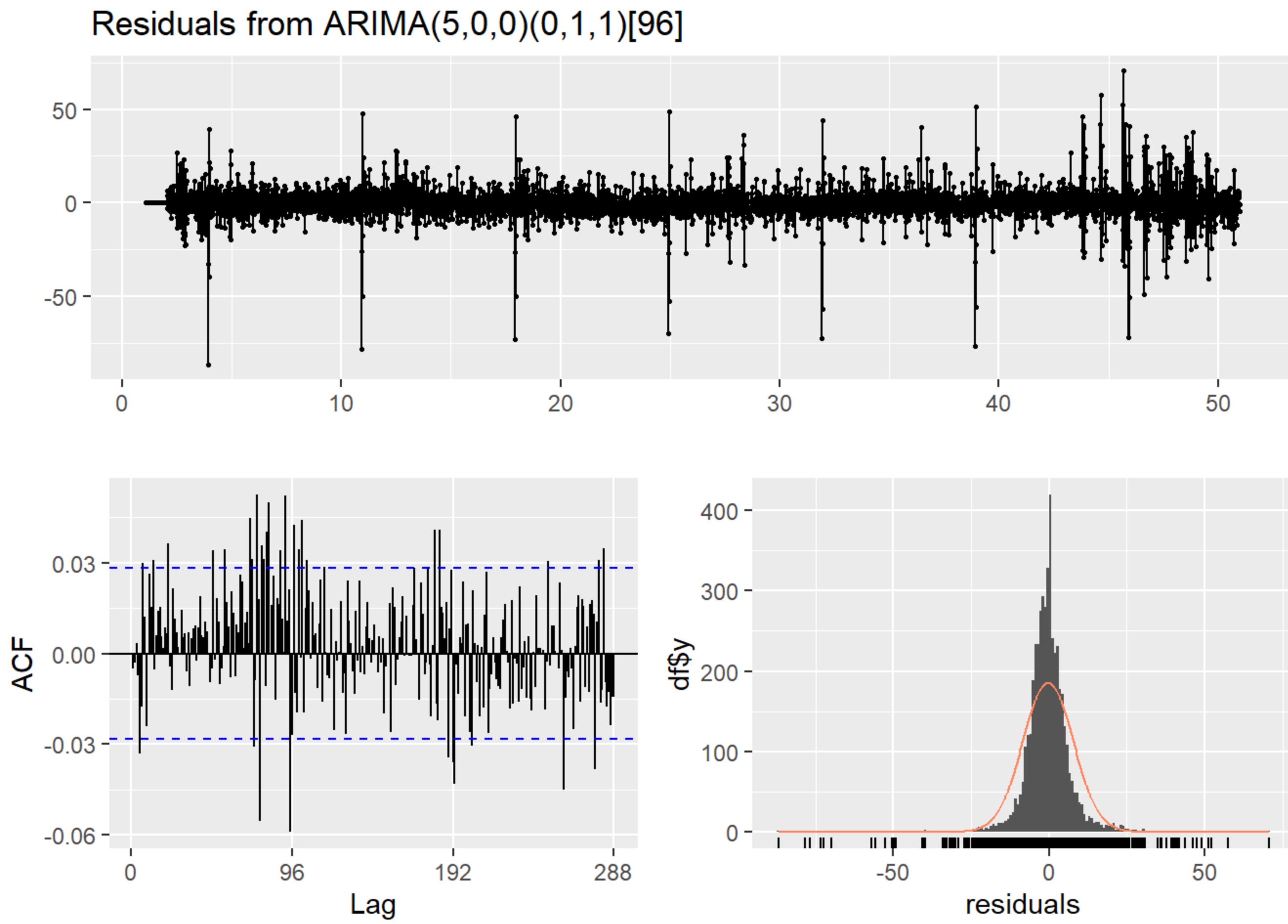
```
# SARIMA, daily period
exec_t_start = Sys.time()

fit = Arima(y_daily_train, order = c(5,0,0), seasonal = c(0,1,1))
fit |>  summary()
```

```
## Series: y_daily_train
## ARIMA(5,0,0)(0,1,1)[96]
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5     sma1
##       0.6729  0.0688  0.1632  -0.2361  0.1268  -0.8755
## s.e.  0.0145  0.0172  0.0170   0.0172  0.0145   0.0076
##
## sigma^2 = 67.66:  log likelihood = -16636.84
## AIC=33287.67   AICc=33287.7   BIC=33332.86
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.34689 8.137587 4.989581 -0.2607279 2.257775 0.5674946
##                      ACF1
## Training set -0.005121753
```

```
ggtsdisplay(fit$residuals)
```




```
checkresiduals(fit, plot = TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,0,0)(0,1,1)[96]
## Q* = 347.46, df = 186, p-value = 7.013e-12
##
## Model df: 6.   Total lags used: 192
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 3.555382 mins
```

```
# saveRDS(fit, file = "ARIMA_man_(5,0,0)(0,1,1)[96].rds")
```

Cross-validation

```
# Code commented: very long computation

# # Cross validation using tsCV(), ref: https://pkg.robjhyndman.com/forecast/reference/tsCV.html
#
# # Forcasting function to cross-validate
# Arima_ <- function(x, h) {
#   forecast(Arima(x,
#                  order=c(5,0,0),
#                  seasonal = c(0,1,1)
#                  ))
# }
#
# # Crossvalidation execution
# exec_t_start = Sys.time()
#
# e <- tsCV(y_daily, Arima_, h=96, window = 4795)
#
# exec_t_end = Sys.time()
# print(exec_t_end - exec_t_start)
#
# print(paste0("Cross-validation RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
```

Notes:

Cross-validation:

```
Time difference of 5.734455 hours
[1] "Cross-validation RMSE: 12.5587524976877"
```

NNetAR, Daily period

```
exec_t_start = Sys.time()

fit = nnetar(y_daily_train)
fit |>  summary()
```

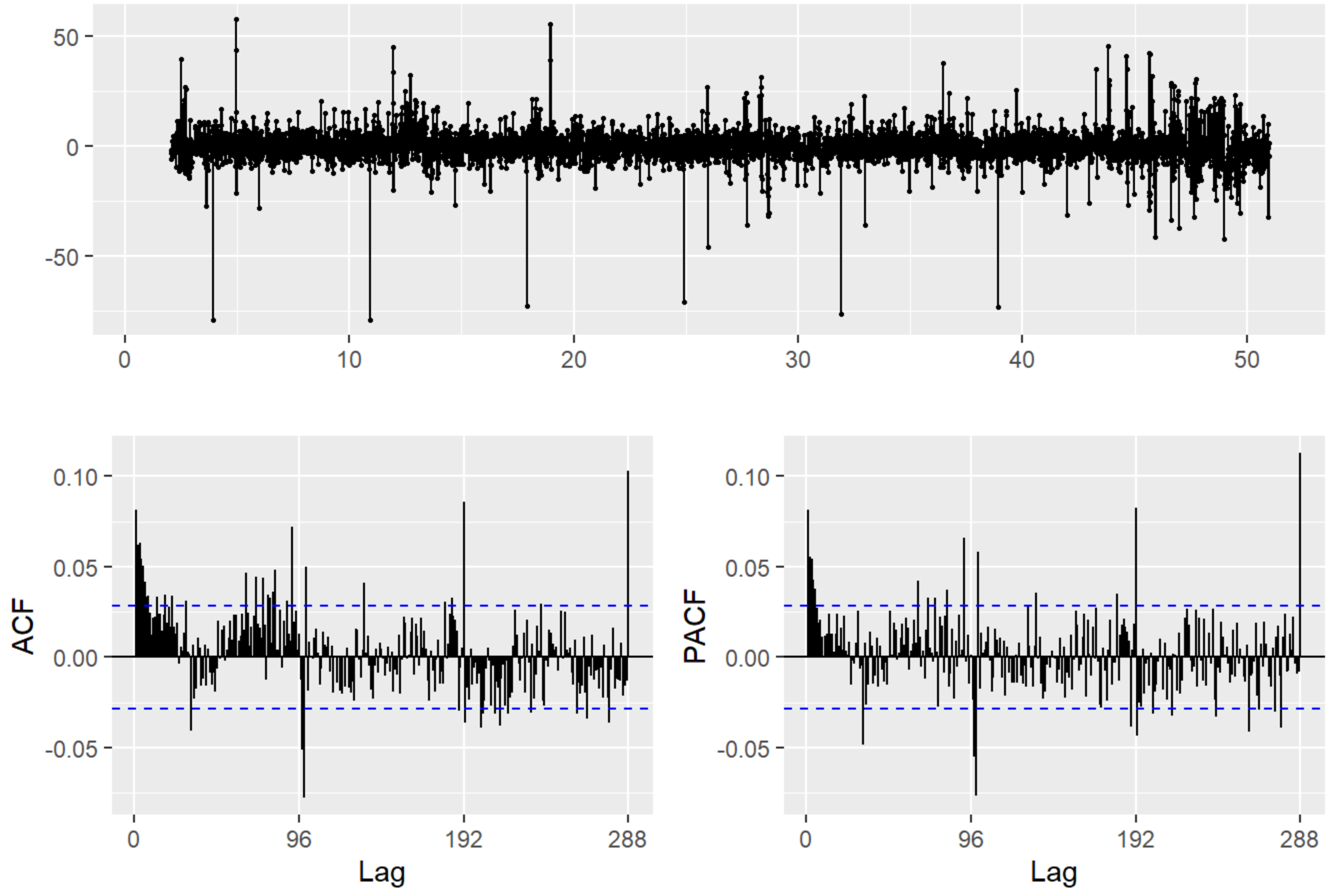
```
##           Length Class        Mode     
## x         4795   ts           numeric  
## m            1   -none-       numeric  
## p            1   -none-       numeric  
## P            1   -none-       numeric  
## scalex       2   -none-       list     
## size         1   -none-       numeric  
## subset    4795   -none-       numeric  
## model       20   nnetarmodels list     
## nnetargs     0   -none-       list     
## fitted    4795   ts           numeric  
## residuals 4795   ts           numeric  
## lags        26   -none-       numeric  
## series       1   -none-       character
## method       1   -none-       character
## call         2   -none-       call
```

```
e = fit$residuals
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
```

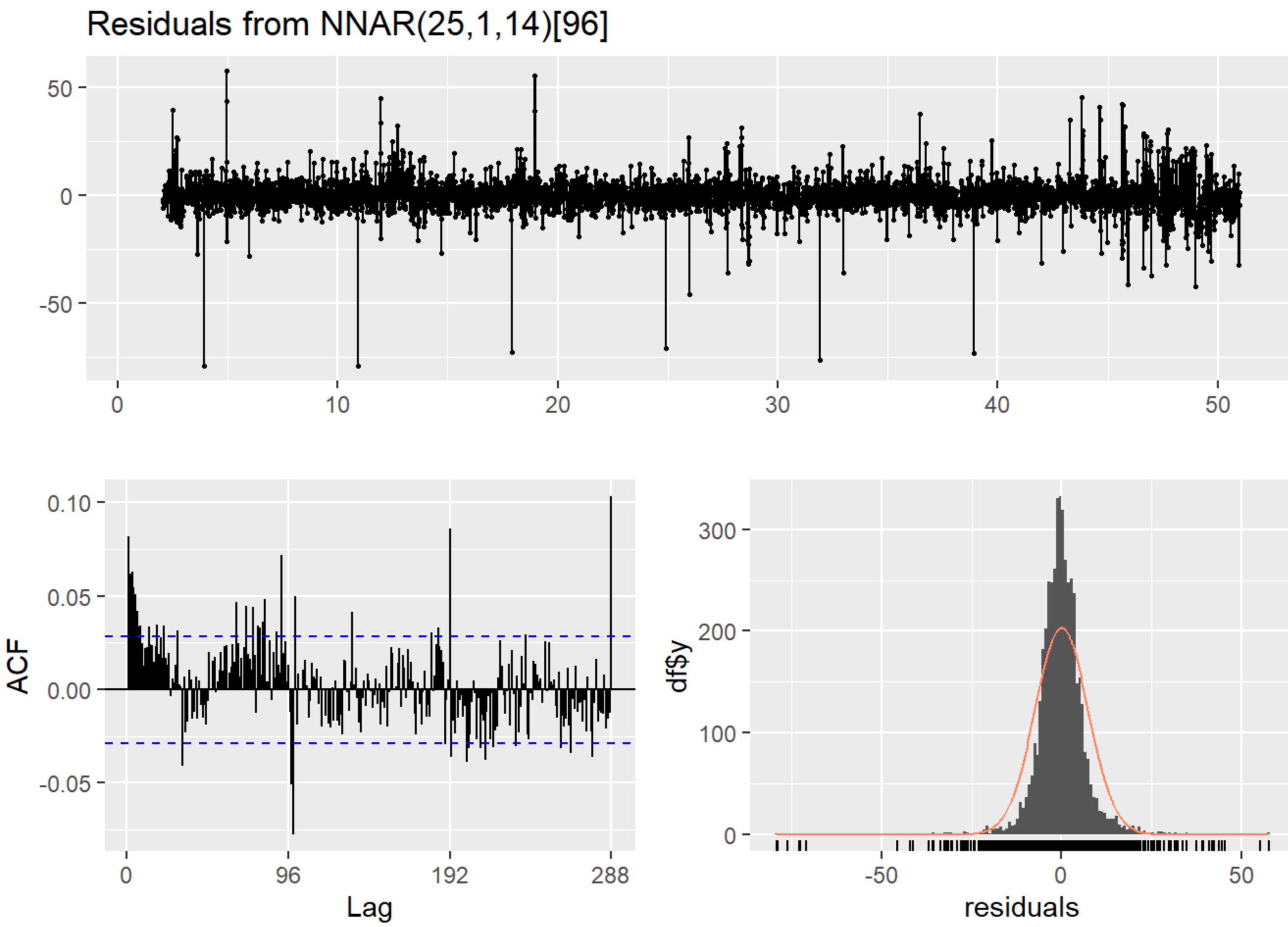
```
## [1] "Train RMSE: 7.36936636232128"
```

```
ggtsdisplay(fit$residuals)
```

```
## Warning: Removed 96 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
checkresiduals(fit, plot = TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from NNAR(25,1,14)[96]
## Q* = 495.37, df = 192, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 192
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 46.29163 secs
```

```
# saveRDS(fit, file = "NNetAR_daily.rds")
```

SARIMA (auto), Weekly period

```
# Code commented: long fitting time, model performance not great

# # Auto ARIMA, weekly period
# exec_t_start = Sys.time()
#
# fit = auto.arima(y_weekly_train)
# fit |>  summary()
#
# ggtsdisplay(fit$residuals)
# checkresiduals(fit, plot = TRUE)
#
# exec_t_end = Sys.time()
# print(exec_t_end - exec_t_start)
```

```
# saveRDS(fit, file = "ARIMA_auto_(5,1,2)(0,1,0)[672].rds")
```

SARIMA (manual), Weekly period

```
# Code commented: fail to be fitted due to too large number of lags

# # ARIMA, weekly period
# exec_t_start = Sys.time()
#
# fit = Arima(y_weekly_train, order = c(5,1,2), seasonal = c(0,1,1))
# fit |>  summary()
#
# ggtsdisplay(fit$residuals)
# checkresiduals(fit, plot = TRUE)
#
# exec_t_end = Sys.time()
# print(exec_t_end - exec_t_start)
```

```
# saveRDS(fit, file = "ARIMA_auto_(5,1,2)(0,1,1)[672].rds")
```

NetAR, Weekly period

```
exec_t_start = Sys.time()

fit = nnetar(y_weekly_train)
fit |>  summary()
```

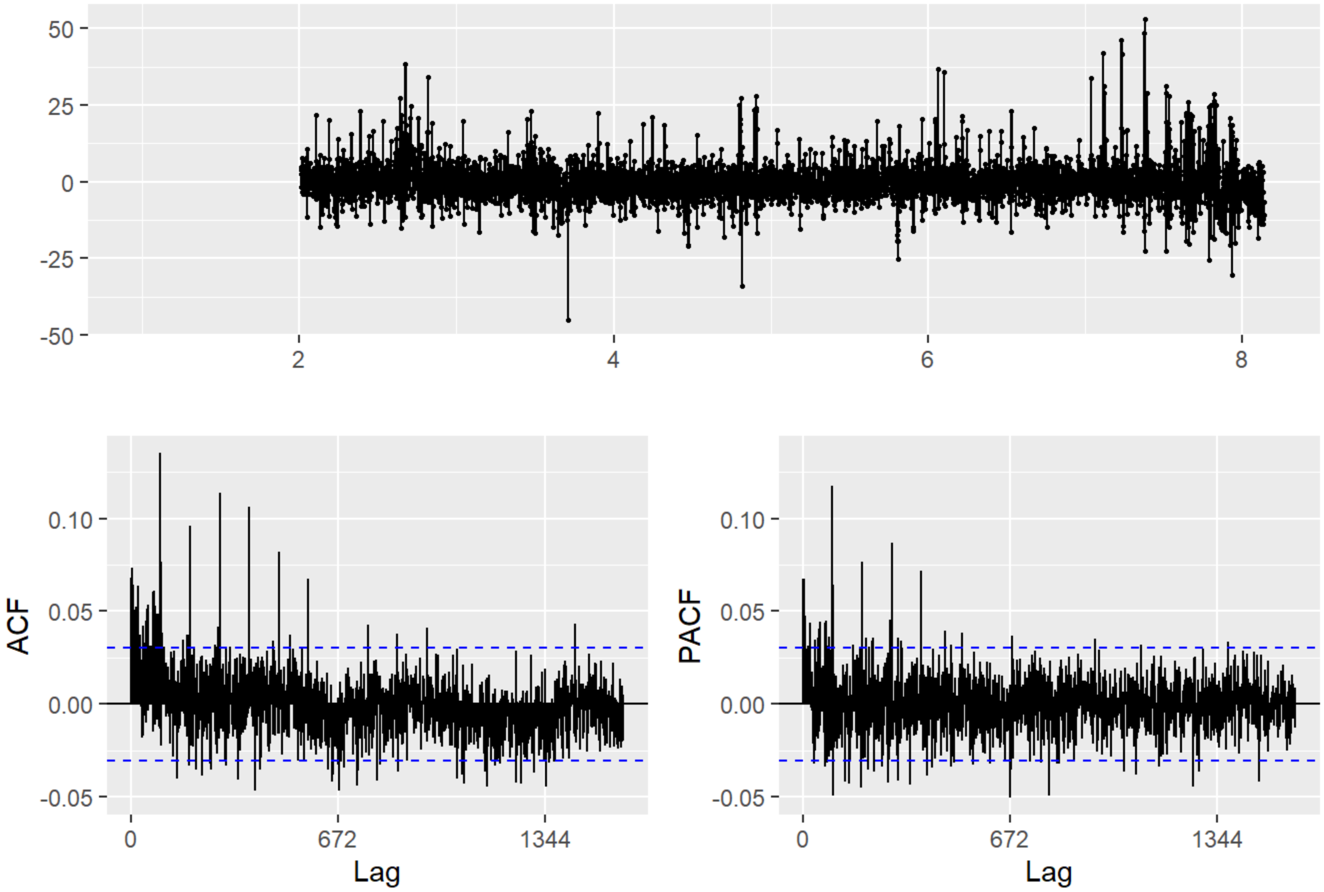
```
##           Length Class        Mode
## x         4795   ts           numeric
## m            1   -none-       numeric
## p            1   -none-       numeric
## P            1   -none-       numeric
## scalex       2   -none-       list
## size         1   -none-       numeric
## subset    4795   -none-       numeric
## model       20   nnetarmodels list
## nnetargs     0   -none-       list
## fitted    4795   ts           numeric
## residuals 4795   ts           numeric
## lags        18   -none-       numeric
## series       1   -none-       character
## method       1   -none-       character
## call         2   -none-       call
```

```
e = fit$residuals
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
```

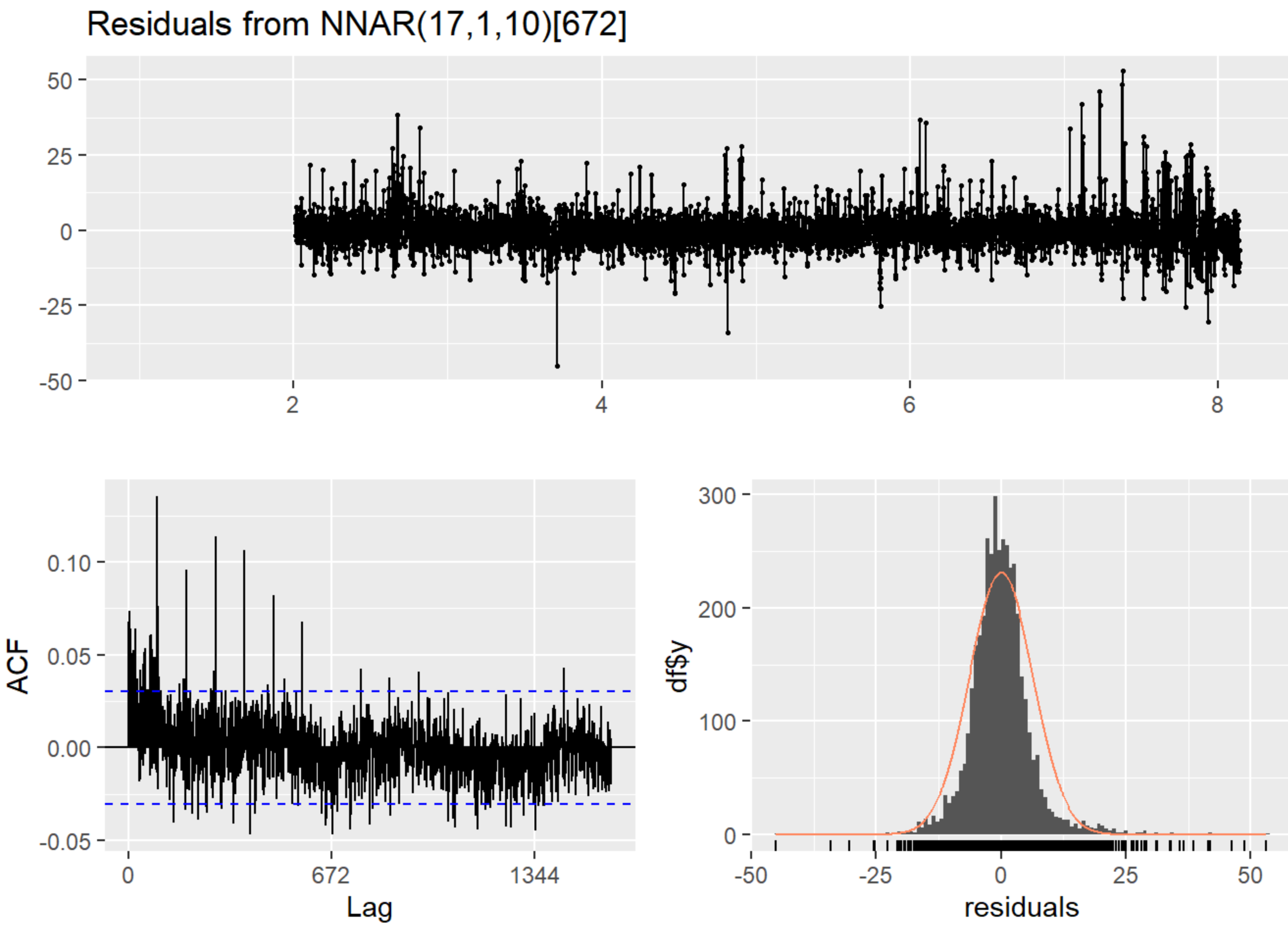
```
## [1] "Train RMSE: 6.35882115176639"
```

```
ggtsdisplay(fit$residuals)
```

```
## Warning: Removed 672 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



```
checkresiduals(fit, plot = TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from NNAR(17,1,10)[672]
## Q* = 1743.6, df = 959, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 959
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 26.27995 secs
```

```
# saveRDS(fit, file = "NNetAR_weekly.rds")
```

ML data prep

```
# next observation based on last day
df_daily = as.vector(y_daily_train)[1:(96+1)]
for (i in 1:(length(y_daily_train)-(96+1)))
{
  df_daily = rbind(df_daily, as.vector(y_daily_train)[(i+1):(i+96+1)])
}

# next observation based on last week
df_weekly = as.vector(y_weekly_train)[1:(7*96+1)]
for (i in 1:(length(y_weekly_train)-(7*96+1)))
{
  df_weekly = rbind(df_weekly, as.vector(y_weekly_train)[(i+1):(i+7*96+1)])
}

# next 96 observations based on 2 last week
df_2weeks = as.vector(y_weekly_train)[1:(2*7*96+96)]
for (i in 1:(length(y_weekly_train)-(2*7*96+96)))
{
  df_2weeks = rbind(df_2weeks, as.vector(y_weekly_train)[(i+1):(i+2*7*96+96)])
}
```

ML - Random Forest, Daily period

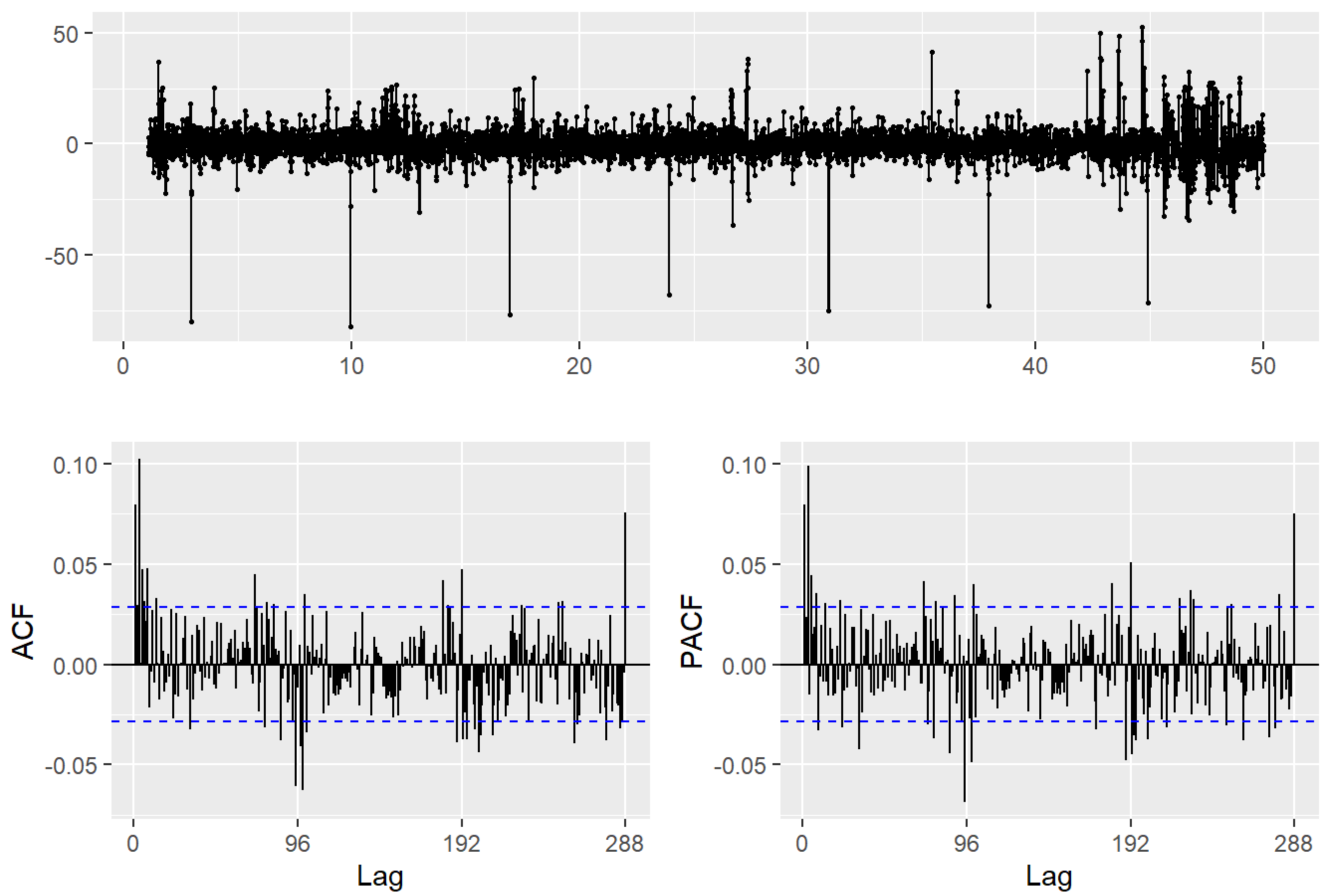
```
exec_t_start = Sys.time()

fit = randomForest(x = df_daily[,-(96+1)], y = df_daily[, (96+1)])
fit |>  summary()
```

```
##                 Length Class  Mode
## call               3   -none- call
## type               1   -none- character
## predicted       4699   -none- numeric
## mse              500   -none- numeric
## rsq              500   -none- numeric
## oob.times       4699   -none- numeric
## importance        96   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y               4699   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
```

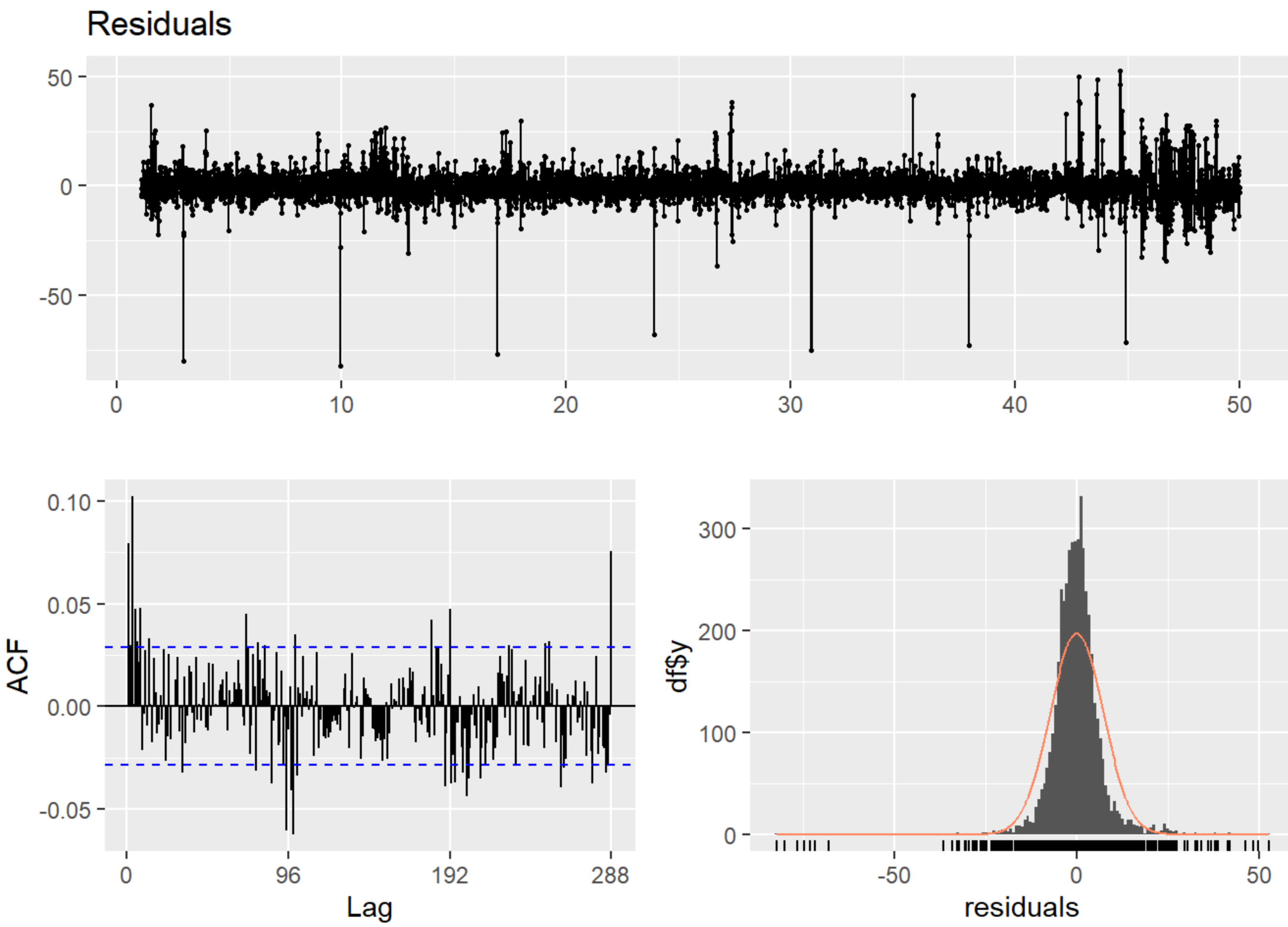
```
e = ts(fit$y - fit$predicted, start = c(1,6), frequency = 96)
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
```

```
## [1] "Train RMSE: 7.28924430975706"
```

```
ggtsdisplay(e)
```




```
checkresiduals(e, plot = TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 389.78, df = 192, p-value = 1.443e-15
##
## Model df: 0.   Total lags used: 192
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 1.442972 mins
```

```
# saveRDS(fit, file = "RF_daily.rds")
```

ML - XGBoost, Daily period

```
exec_t_start = Sys.time()

fit = xgboost(data = df_daily[,-(96+1)], label = df_daily[, (96+1)],
              max_depth = 10,
              eta = 0.5,
              nrounds = 100,
              objective = "reg:squarederror")
```

```
## [1]  train-rmse:119.391993
## [2]  train-rmse:60.372550
## [3]  train-rmse:30.994714
## [4]  train-rmse:16.557812
## [5]  train-rmse:9.467803
## [6]  train-rmse:6.085589
## [7]  train-rmse:4.583516
## [8]  train-rmse:3.692306
## [9]  train-rmse:3.293860
## [10] train-rmse:3.005140
## [11] train-rmse:2.776008
## [12] train-rmse:2.610726
## [13] train-rmse:2.392126
## [14] train-rmse:2.291819
## [15] train-rmse:2.189205
## [16] train-rmse:2.128939
## [17] train-rmse:2.011813
## [18] train-rmse:1.862358
## [19] train-rmse:1.778354
## [20] train-rmse:1.700839
## [21] train-rmse:1.606601
## [22] train-rmse:1.464531
## [23] train-rmse:1.326684
## [24] train-rmse:1.275947
## [25] train-rmse:1.211387
## [26] train-rmse:1.151056
## [27] train-rmse:1.055514
## [28] train-rmse:1.018223
## [29] train-rmse:0.890843
## [30] train-rmse:0.838673
## [31] train-rmse:0.810022
## [32] train-rmse:0.757357
## [33] train-rmse:0.711340
## [34] train-rmse:0.633122
## [35] train-rmse:0.616944
## [36] train-rmse:0.552809
## [37] train-rmse:0.538453
## [38] train-rmse:0.524320
## [39] train-rmse:0.499012
## [40] train-rmse:0.474222
## [41] train-rmse:0.456232
## [42] train-rmse:0.432571
## [43] train-rmse:0.405178
## [44] train-rmse:0.379571
## [45] train-rmse:0.337135
## [46] train-rmse:0.314113
## [47] train-rmse:0.298620
## [48] train-rmse:0.275058
## [49] train-rmse:0.253763
## [50] train-rmse:0.232977
## [51] train-rmse:0.220237
## [52] train-rmse:0.210566
```

```
## [53] train-rmse:0.191953
## [54] train-rmse:0.185008
## [55] train-rmse:0.170787
## [56] train-rmse:0.156534
## [57] train-rmse:0.139317
## [58] train-rmse:0.133946
## [59] train-rmse:0.129996
## [60] train-rmse:0.122209
## [61] train-rmse:0.115174
## [62] train-rmse:0.103148
## [63] train-rmse:0.099064
## [64] train-rmse:0.095086
## [65] train-rmse:0.087367
## [66] train-rmse:0.079384
## [67] train-rmse:0.071296
## [68] train-rmse:0.066791
## [69] train-rmse:0.063174
## [70] train-rmse:0.058606
## [71] train-rmse:0.055441
## [72] train-rmse:0.052071
## [73] train-rmse:0.049990
## [74] train-rmse:0.047387
## [75] train-rmse:0.044269
## [76] train-rmse:0.041390
## [77] train-rmse:0.037564
## [78] train-rmse:0.035504
## [79] train-rmse:0.033785
## [80] train-rmse:0.030566
## [81] train-rmse:0.028615
## [82] train-rmse:0.027422
## [83] train-rmse:0.026515
## [84] train-rmse:0.024614
## [85] train-rmse:0.023438
## [86] train-rmse:0.021405
## [87] train-rmse:0.019308
## [88] train-rmse:0.017997
## [89] train-rmse:0.016507
## [90] train-rmse:0.015865
## [91] train-rmse:0.015139
## [92] train-rmse:0.014362
## [93] train-rmse:0.013987
## [94] train-rmse:0.012579
## [95] train-rmse:0.011165
## [96] train-rmse:0.010469
## [97] train-rmse:0.009854
## [98] train-rmse:0.009299
## [99] train-rmse:0.008582
## [100]    train-rmse:0.008163
```

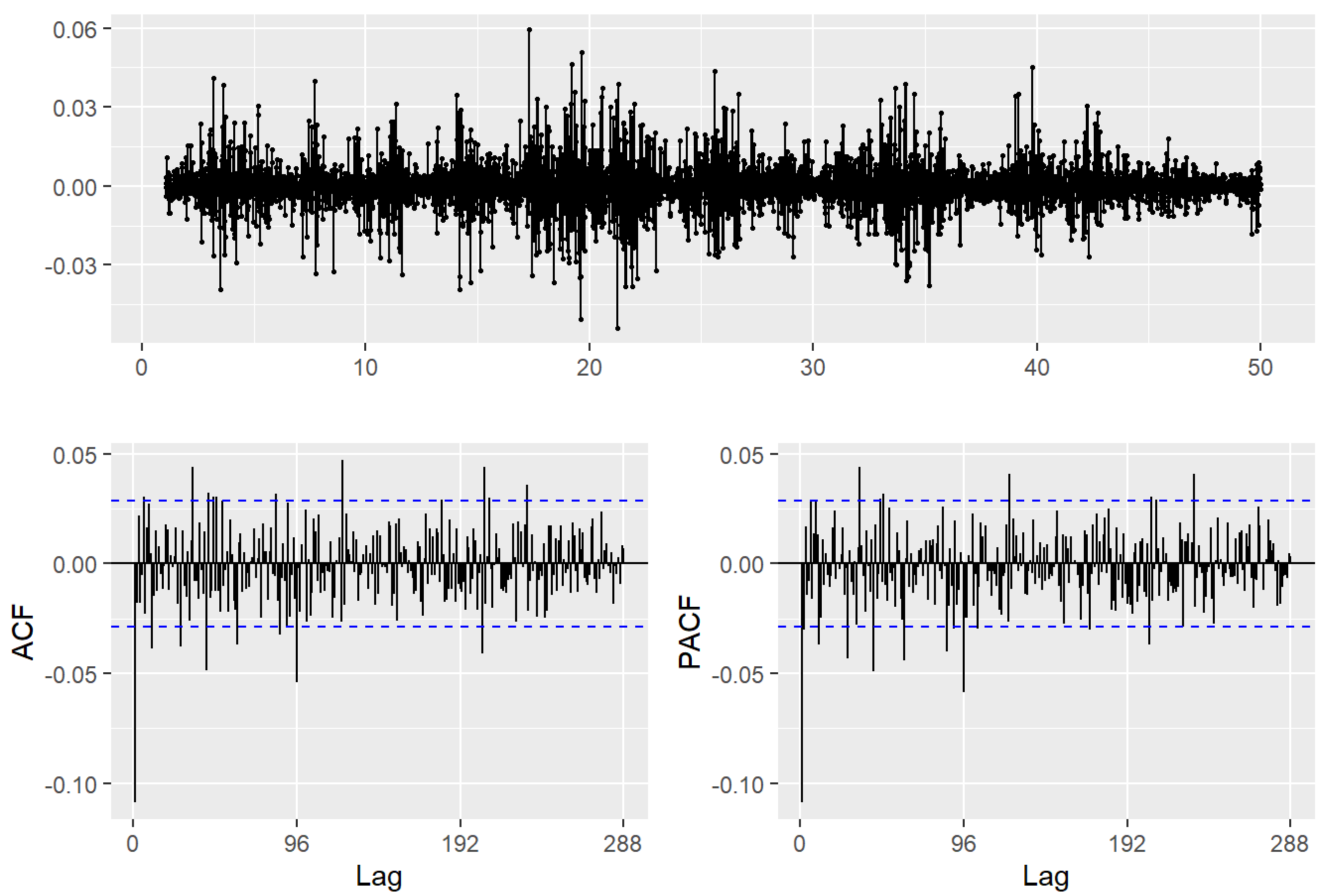
```
fit |>  summary()
```

```
##                  Length  Class              Mode
## handle                1 xgb.Booster.handle externalptr
## raw             1381676 -none-             raw
## niter                 1 -none-             numeric
## evaluation_log        2 data.table         list
## call                 16 -none-             call
## params                4 -none-             list
## callbacks             2 -none-             list
## nfeatures             1 -none-             numeric
```

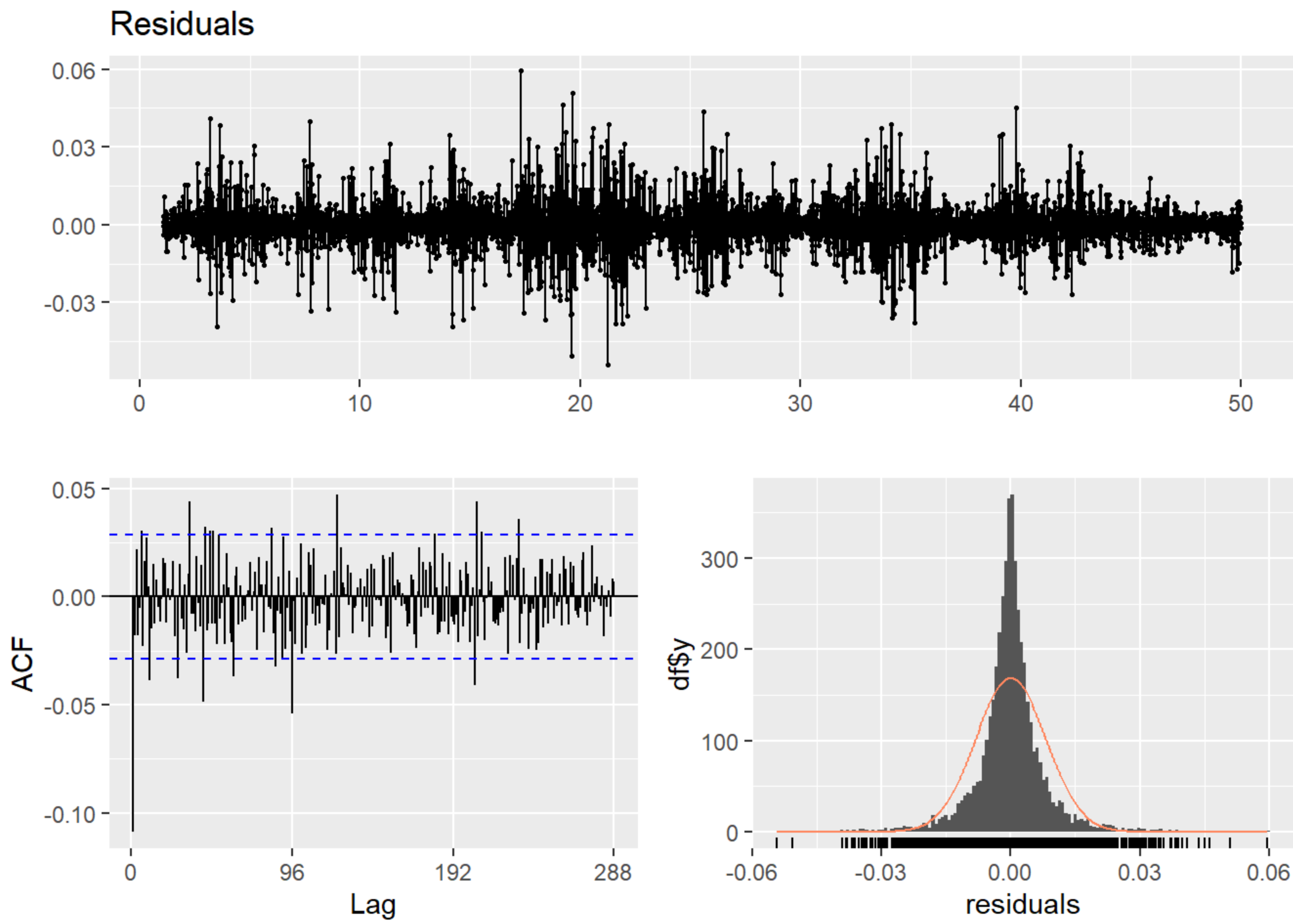
```
e = ts(df_daily[, (96+1)] - predict(fit, newdata = df_daily[,-(96+1)]), start = c(1,6), frequenc
y = 96)
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
```

```
## [1] "Train RMSE: 0.00816297752522528"
```

```
ggtsdisplay(e)
```




```
checkresiduals(e, plot = TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 316.33, df = 192, p-value = 3.986e-08
##
## Model df: 0.   Total lags used: 192
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 7.328984 secs
```

```
# saveRDS(fit, file = "XGBoost_daily.rds")
```

ML - Random Forest, Weekly period

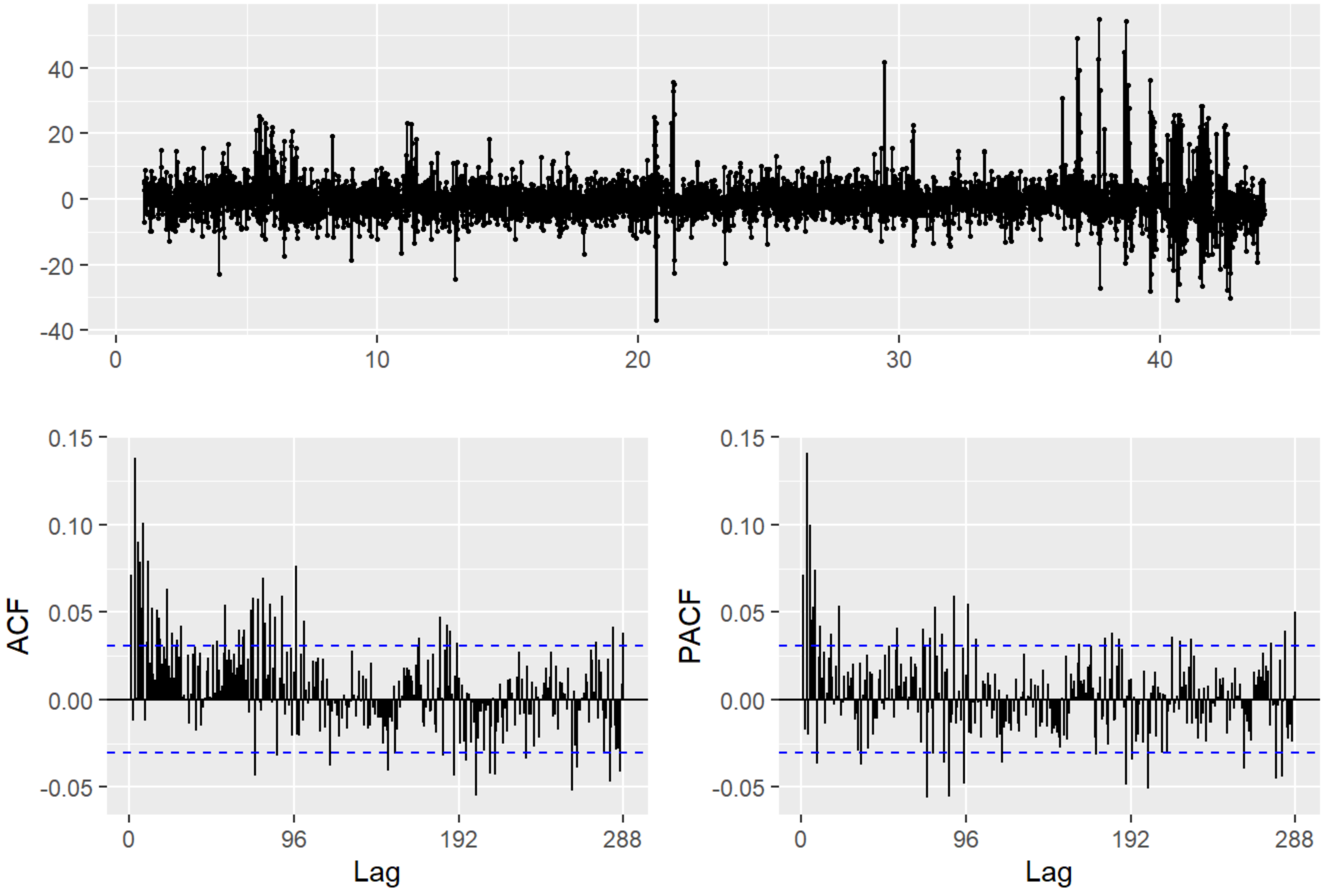
```
exec_t_start = Sys.time()

fit = randomForest(x = df_weekly[,-(7*96+1)], y = df_weekly[, (7*96+1)])
fit |>  summary()
```

```
##                 Length Class  Mode
## call              3   -none- call
## type              1   -none- character
## predicted      4123   -none- numeric
## mse             500   -none- numeric
## rsq             500   -none- numeric
## oob.times      4123   -none- numeric
## importance      672   -none- numeric
## importanceSD      0   -none- NULL
## localImportance   0   -none- NULL
## proximity         0   -none- NULL
## ntree             1   -none- numeric
## mtry              1   -none- numeric
## forest           11   -none- list
## coefs             0   -none- NULL
## y              4123   -none- numeric
## test              0   -none- NULL
## inbag             0   -none- NULL
```

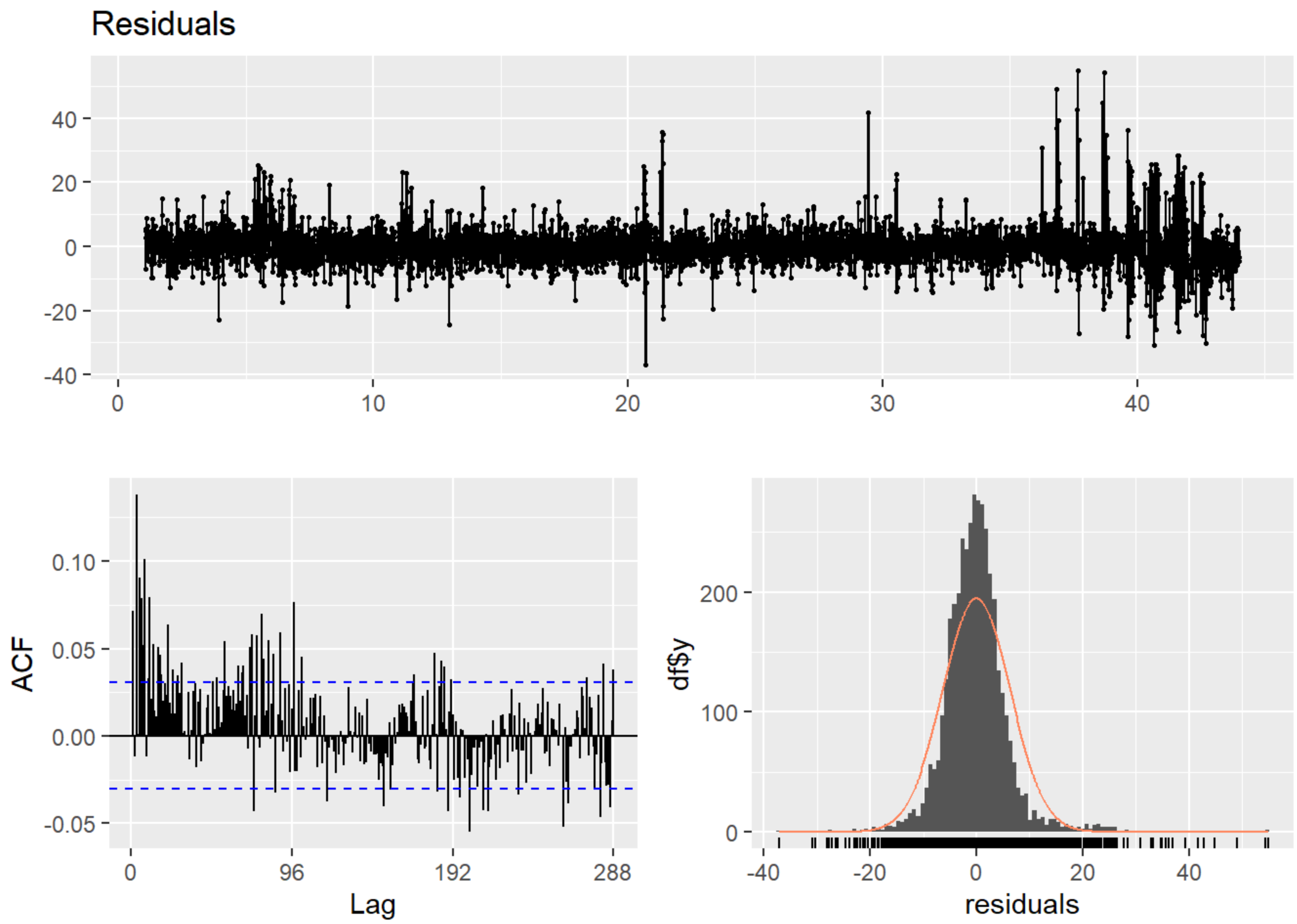
```
e = ts(fit$y - fit$predicted, start = c(1,6), frequency = 96)
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
```

```
## [1] "Train RMSE: 6.36450152436013"
```

```
ggtsdisplay(e)
```




```
checkresiduals(e, plot = TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 721.89, df = 192, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 192
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 7.877153 mins
```

```
# saveRDS(fit, file = "RF_weekly.rds")
```

ML - XGBoost, Weekly period

```
exec_t_start = Sys.time()

fit = xgboost(data = df_weekly[,-(7*96+1)], label = df_weekly[, (7*96+1)],
              max_depth = 10,
              eta = 0.5,
              nrounds = 100,
              objective = "reg:squarederror")
```

```
## [1]  train-rmse:118.891242
## [2]  train-rmse:60.019148
## [3]  train-rmse:30.771407
## [4]  train-rmse:16.348894
## [5]  train-rmse:9.246309
## [6]  train-rmse:5.909916
## [7]  train-rmse:4.046978
## [8]  train-rmse:3.216542
## [9]  train-rmse:2.717816
## [10] train-rmse:2.474393
## [11] train-rmse:2.248075
## [12] train-rmse:2.137974
## [13] train-rmse:2.049120
## [14] train-rmse:1.978210
## [15] train-rmse:1.910054
## [16] train-rmse:1.848426
## [17] train-rmse:1.798103
## [18] train-rmse:1.687037
## [19] train-rmse:1.603487
## [20] train-rmse:1.554446
## [21] train-rmse:1.482726
## [22] train-rmse:1.438925
## [23] train-rmse:1.396924
## [24] train-rmse:1.316335
## [25] train-rmse:1.217854
## [26] train-rmse:1.171994
## [27] train-rmse:1.136937
## [28] train-rmse:1.093460
## [29] train-rmse:1.081097
## [30] train-rmse:1.060568
## [31] train-rmse:1.025918
## [32] train-rmse:1.016761
## [33] train-rmse:0.983544
## [34] train-rmse:0.943015
## [35] train-rmse:0.910389
## [36] train-rmse:0.845086
## [37] train-rmse:0.834150
## [38] train-rmse:0.782884
## [39] train-rmse:0.767372
## [40] train-rmse:0.726262
## [41] train-rmse:0.698320
## [42] train-rmse:0.681956
## [43] train-rmse:0.642527
## [44] train-rmse:0.633433
## [45] train-rmse:0.598136
## [46] train-rmse:0.576124
## [47] train-rmse:0.561962
## [48] train-rmse:0.526319
## [49] train-rmse:0.497930
## [50] train-rmse:0.468705
## [51] train-rmse:0.444676
## [52] train-rmse:0.436936
```

```
## [53] train-rmse:0.427430
## [54] train-rmse:0.409290
## [55] train-rmse:0.397283
## [56] train-rmse:0.386850
## [57] train-rmse:0.370378
## [58] train-rmse:0.353291
## [59] train-rmse:0.347520
## [60] train-rmse:0.328985
## [61] train-rmse:0.324145
## [62] train-rmse:0.319833
## [63] train-rmse:0.307087
## [64] train-rmse:0.290952
## [65] train-rmse:0.274637
## [66] train-rmse:0.270203
## [67] train-rmse:0.262919
## [68] train-rmse:0.260111
## [69] train-rmse:0.249394
## [70] train-rmse:0.237734
## [71] train-rmse:0.227773
## [72] train-rmse:0.225499
## [73] train-rmse:0.216328
## [74] train-rmse:0.203853
## [75] train-rmse:0.194742
## [76] train-rmse:0.189131
## [77] train-rmse:0.177077
## [78] train-rmse:0.168198
## [79] train-rmse:0.164131
## [80] train-rmse:0.156567
## [81] train-rmse:0.147177
## [82] train-rmse:0.139430
## [83] train-rmse:0.136294
## [84] train-rmse:0.130313
## [85] train-rmse:0.128834
## [86] train-rmse:0.122085
## [87] train-rmse:0.115954
## [88] train-rmse:0.112879
## [89] train-rmse:0.110047
## [90] train-rmse:0.107453
## [91] train-rmse:0.102396
## [92] train-rmse:0.101596
## [93] train-rmse:0.098514
## [94] train-rmse:0.091126
## [95] train-rmse:0.088330
## [96] train-rmse:0.086501
## [97] train-rmse:0.081014
## [98] train-rmse:0.078205
## [99] train-rmse:0.074717
## [100]    train-rmse:0.073691
```

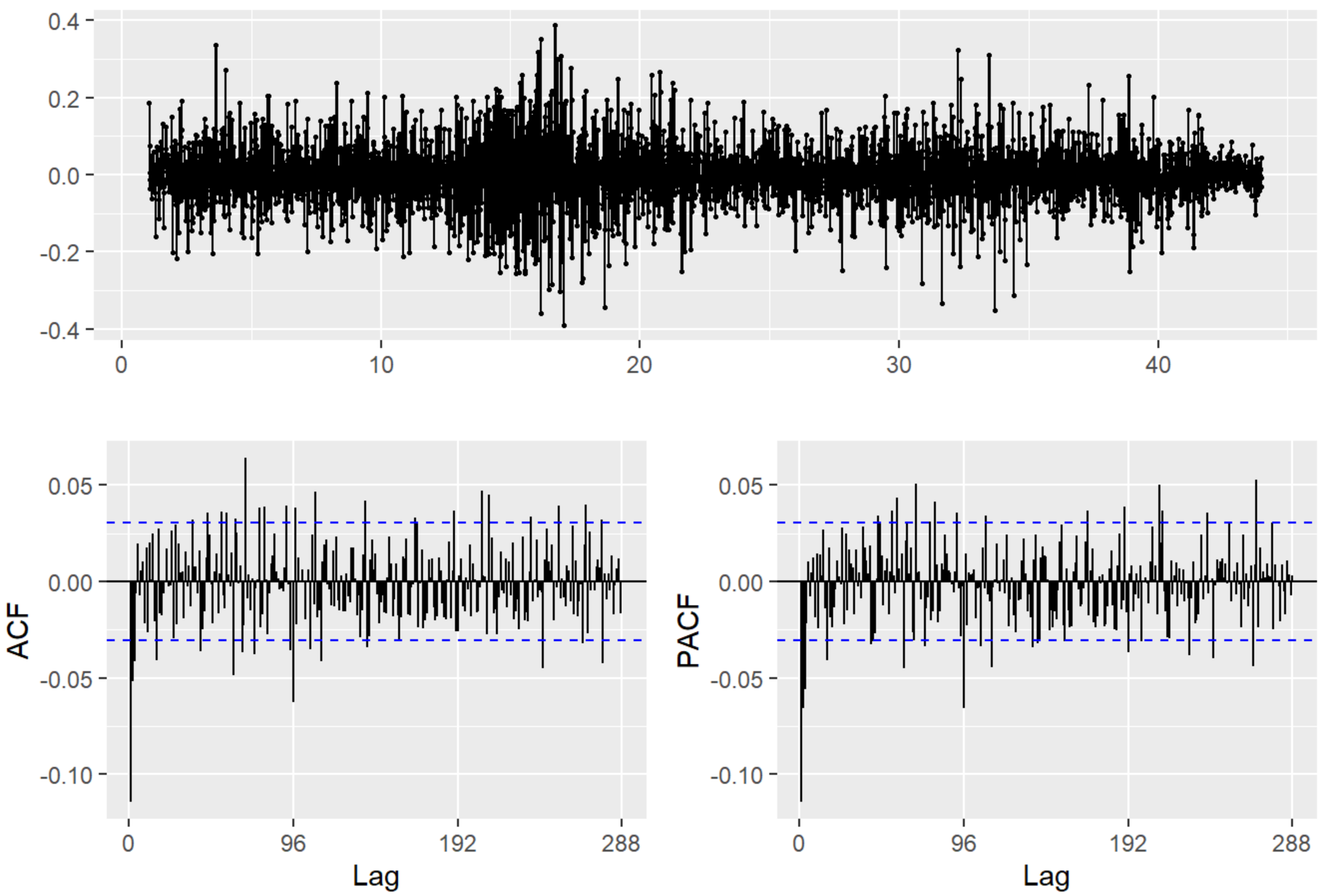
```
fit |>  summary()
```

```
##                   Length Class              Mode
## handle                 1 xgb.Booster.handle externalptr
## raw               641571 -none-             raw
## niter                  1 -none-             numeric
## evaluation_log         2 data.table         list
## call                  16 -none-             call
## params                 4 -none-             list
## callbacks              2 -none-             list
## nfeatures              1 -none-             numeric
```

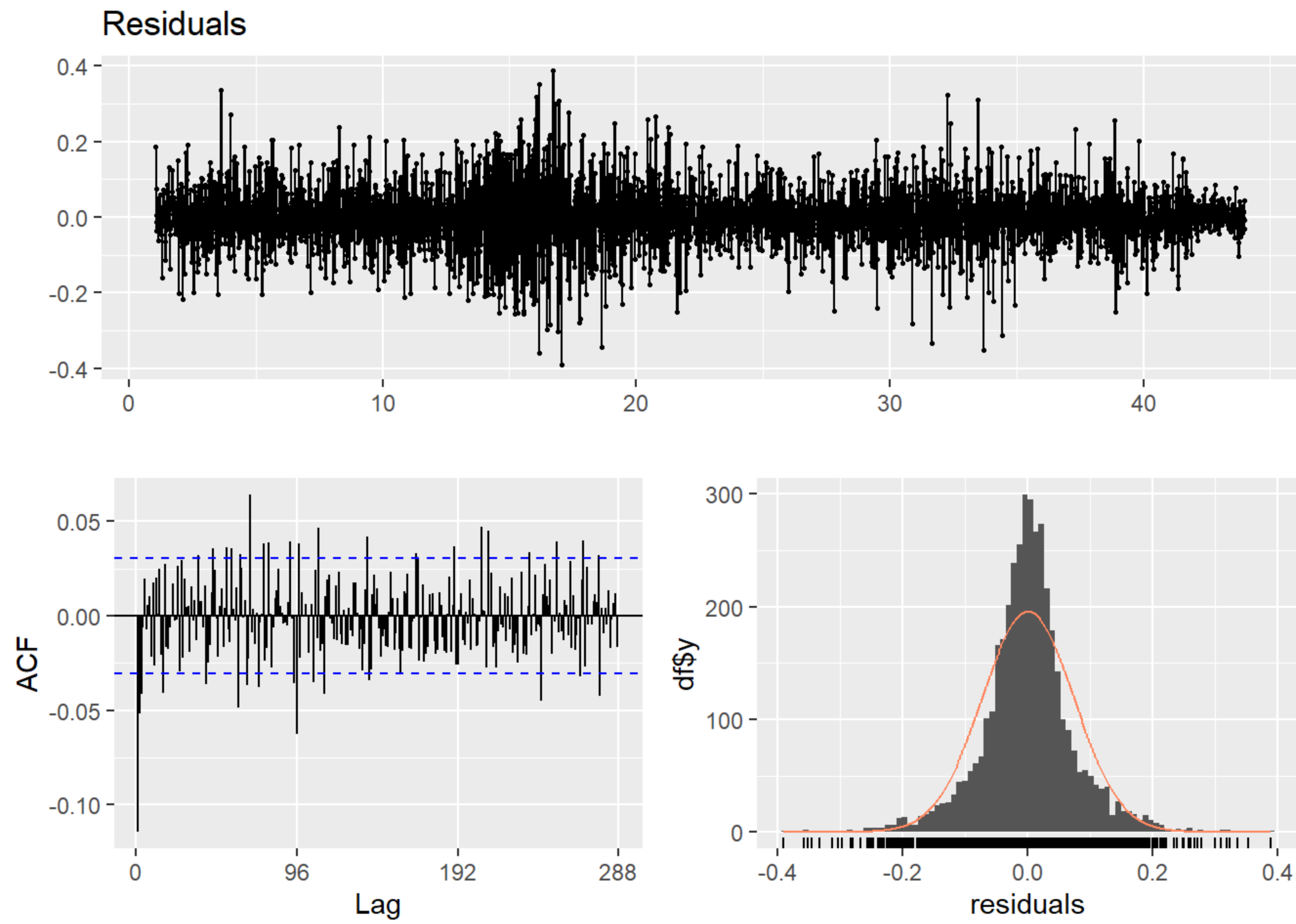
```
e = ts(df_weekly[, (7*96+1)] - predict(fit, newdata = df_weekly[,-(7*96+1)]), start = c(1,6), fr
equency = 96)
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
```

```
## [1] "Train RMSE: 0.0736913546926577"
```

```
ggtsdisplay(e)
```




```
checkresiduals(e, plot = TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 405.93, df = 192, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 192
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 15.59866 secs
```

```
# saveRDS(fit, file = "XGBoost_weeky.rds")
```

ML - PLS, 2 weeks history to forecast next day

```
library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
# Code commented: very long model fitting, model performance not great

# exec_t_start = Sys.time()
#
# fit = plsr(df_2weeks[,(2*7*96+1):(2*7*96+96)] ~ df_2weeks[,1:(2*7*96)],
#            scale = TRUE,
#            validation = "CV")
# fit |>  summary()
#
# # Cross-validation results
# validation_mse <- fit$validation$PRESS
# avg_mse <- colMeans(validation_mse)
#
# # Optimal components minimizing average MSE
# optimal_ncomp <- which.min(avg_mse)
#
# e = fit$residuals[,,optimal_ncomp]
# print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
# # ggtsdisplay(e)
# # checkresiduals(e, plot = TRUE)
#
# exec_t_end = Sys.time()
# print(exec_t_end - exec_t_start)
#
# # saveRDS(fit, file = "PLS_2weeks.rds")
```

Notes:

- PLS: Very long fitting time (approx. 6h). And produces a huge model object (+11 GB)
- Train RMSE = 8.41923 (optimal_ncomp = 144)

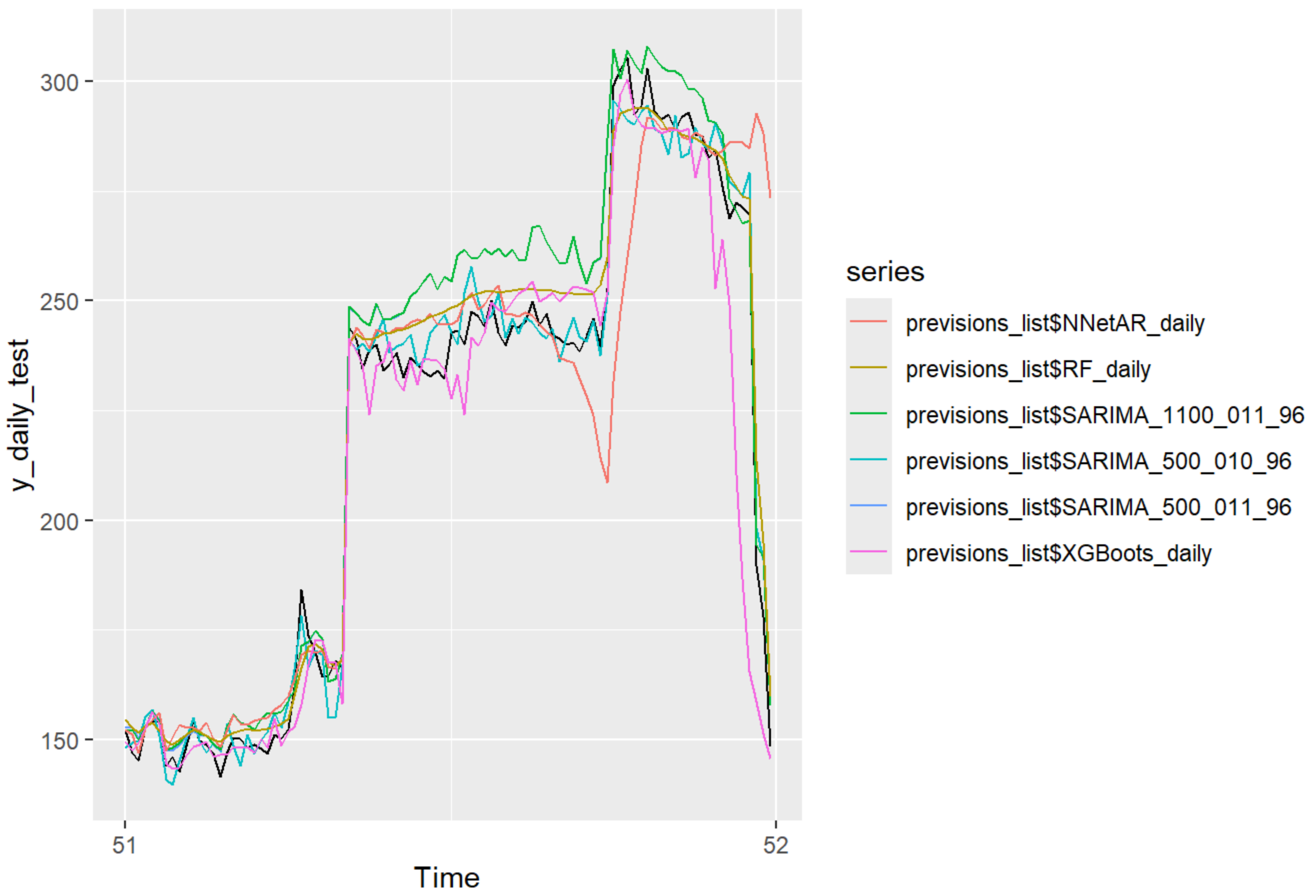
Model performance comparison on test set

Models based on daily period

```r
# Build a list of models
models_list = list()
models_list$SARIMA_500_010_96 = readRDS('ARIMA_auto_(5,0,0)(0,1,0)[96].rds')
models_list$SARIMA_500_011_96 = readRDS('ARIMA_man_(5,0,0)(0,1,1)[96].rds')
models_list$SARIMA_1100_011_96 = readRDS('ARIMA_man_(11,0,0)(0,1,1)[96].rds')
models_list$NNetAR_daily = readRDS('NNetAR_daily.rds')
models_list$RF_daily = readRDS('RF_daily.rds')
models_list$XGBoots_daily = readRDS('XGBoost_daily.rds')

# Make predictions with each model and store RMSE
previsions_list = list()
rmsep_list = list()
horizon = 96
freq = 96
newdata_ML = tail(y_daily_train, horizon)

for (name in names(models_list))
{
  cat(paste0("Forcasting model:", name, "\n"))

  if(grepl("RF", name) | grepl("XG", name)) # Use forecast_ML() with ML models
  {
    prevision = forecast_ML(models_list[[name]],
                            newdata = matrix(newdata_ML,1),
                            horizon)
    prevision = ts(prevision,
                   start = start(y_daily_test),
                   frequency = freq)
  }
  else # Use forecast() with ts models
  {
    prevision = forecast(models_list[[name]], h = horizon)
    prevision = prevision$mean
  }

  previsions_list[[name]] = prevision
  rmsep_list[[name]] = RMSE(y_daily_test,prevision)

  cat(paste0("Test set RMSE: ", rmsep_list[[name]], "\n\n"))
}
```

```
## Forcasting model:SARIMA_500_010_96
## Test set RMSE: 5.86369104634429
##
## Forcasting model:SARIMA_500_011_96
## Test set RMSE: 12.0202823766086
##
## Forcasting model:SARIMA_1100_011_96
## Test set RMSE: 12.0282964300985
##
## Forcasting model:NNetAR_daily
## Test set RMSE: 24.0839486974642
##
## Forcasting model:RF_daily
## Test set RMSE: 7.54168190610469
##
## Forcasting model:XGBoots_daily
## Test set RMSE: 17.3911854416541
```

```
# Plots
autoplot(y_daily_test) +
  autolayer(previsions_list$SARIMA_500_010_96) +
  autolayer(previsions_list$SARIMA_500_011_96) +
  autolayer(previsions_list$SARIMA_1100_011_96) +
  autolayer(previsions_list$NNetAR_daily) +
  autolayer(previsions_list$RF_daily) +
  autolayer(previsions_list$XGBoots_daily)
```



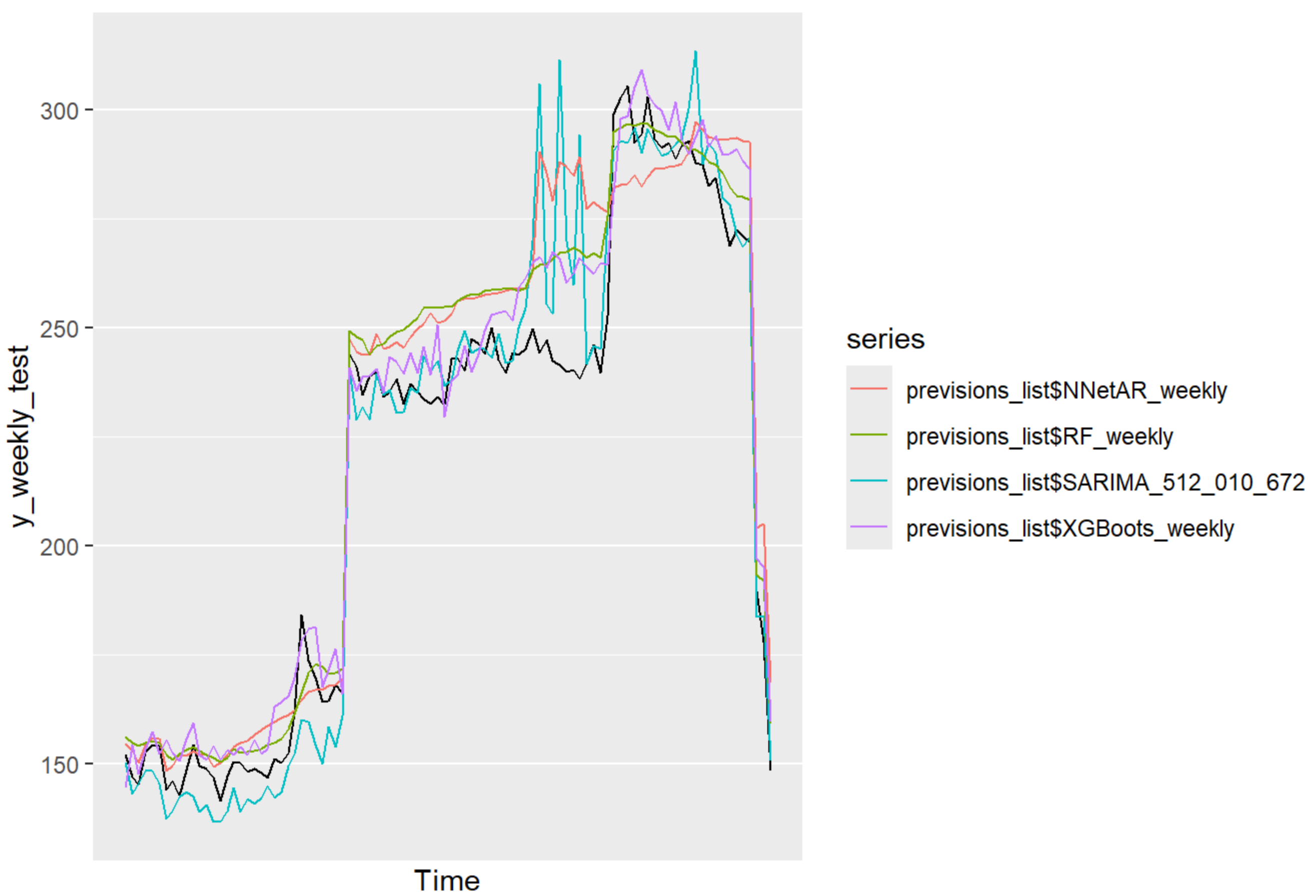
300
250
200
150
y_daily_test
51
52
Time
series
previsions_list$NNetAR_daily
previsions_list$RF_daily
previsions_list$SARIMA_1100_011_96
previsions_list$SARIMA_500_010_96
previsions_list$SARIMA_500_011_96
previsions_list$XGBoots_daily

Models based on weekly period

```
# Build a list of models
models_list = list()
models_list$SARIMA_512_010_672 = readRDS('ARIMA_auto_(5,1,2)(0,1,0)[672].rds')
models_list$NNetAR_weekly = readRDS('NNetAR_weekly.rds')
models_list$RF_weekly = readRDS('RF_weekly.rds')
models_list$XGBoots_weekly = readRDS('XGBoost_weeky.rds')

# Make predictions with each model and store RMSE
previsions_list = list()
rmsep_list = list()
horizon = 96
freq = 7 * 96
newdata_ML = tail(y_weekly_train, 7*horizon)

for (name in names(models_list))
{
  cat(paste0("Forcasting model:", name, "\n"))

  if(grepl("RF", name) | grepl("XG", name)) # Use forecast_ML() with ML models
  {
    prevision = forecast_ML(models_list[[name]],
                            newdata = matrix(newdata_ML,1),
                            horizon)
    prevision = ts(prevision,
                   start = start(y_weekly_test),
                   frequency = freq)
  }
  else # Use forecast() with ts models
  {
    prevision = forecast(models_list[[name]], h = horizon)
    prevision = prevision$mean
  }

  previsions_list[[name]] = prevision
  rmsep_list[[name]] = RMSE(y_weekly_test,prevision)

  cat(paste0("Test set RMSE: ", rmsep_list[[name]], "\n\n"))
}
```

```
## Forcasting model:SARIMA_512_010_672
## Test set RMSE: 14.2221948483852
##
## Forcasting model:NNetAR_weekly
## Test set RMSE: 17.7976830162591
##
## Forcasting model:RF_weekly
## Test set RMSE: 12.4254243102076
##
## Forcasting model:XGBoots_weekly
## Test set RMSE: 11.3326364580346
```

```
# Plots
autoplot(y_weekly_test) +
  autolayer(previsions_list$SARIMA_512_010_672) +
  autolayer(previsions_list$NNetAR_weekly) +
  autolayer(previsions_list$RF_weekly) +
  autolayer(previsions_list$XGBoots_weekly)
```



```
# Code commented: issue with prediction

# fit = readRDS("PLS_2weeks.rds")
#
# cat(paste0("Forcasting model:", "PLS_2weeks", "\n"))
#
# prevision = predict(fit,
#                     newdata = t(matrix(tail(y_weekly_train, 2*7*96), nrow = 1)),
#                     ncomp = 144)
# prevision = ts(prevision[1,,1], start = c(8,97), frequency = 7*96)
#
# RMSE_PLS_2weeks = RMSE(y_weekly_test,prevision)
# cat(paste0("Test set RMSE: ", RMSE_PLS_2weeks, "\n\n"))
#
# autoplot(y_weekly_test) +
#   autolayer(prevision)
```

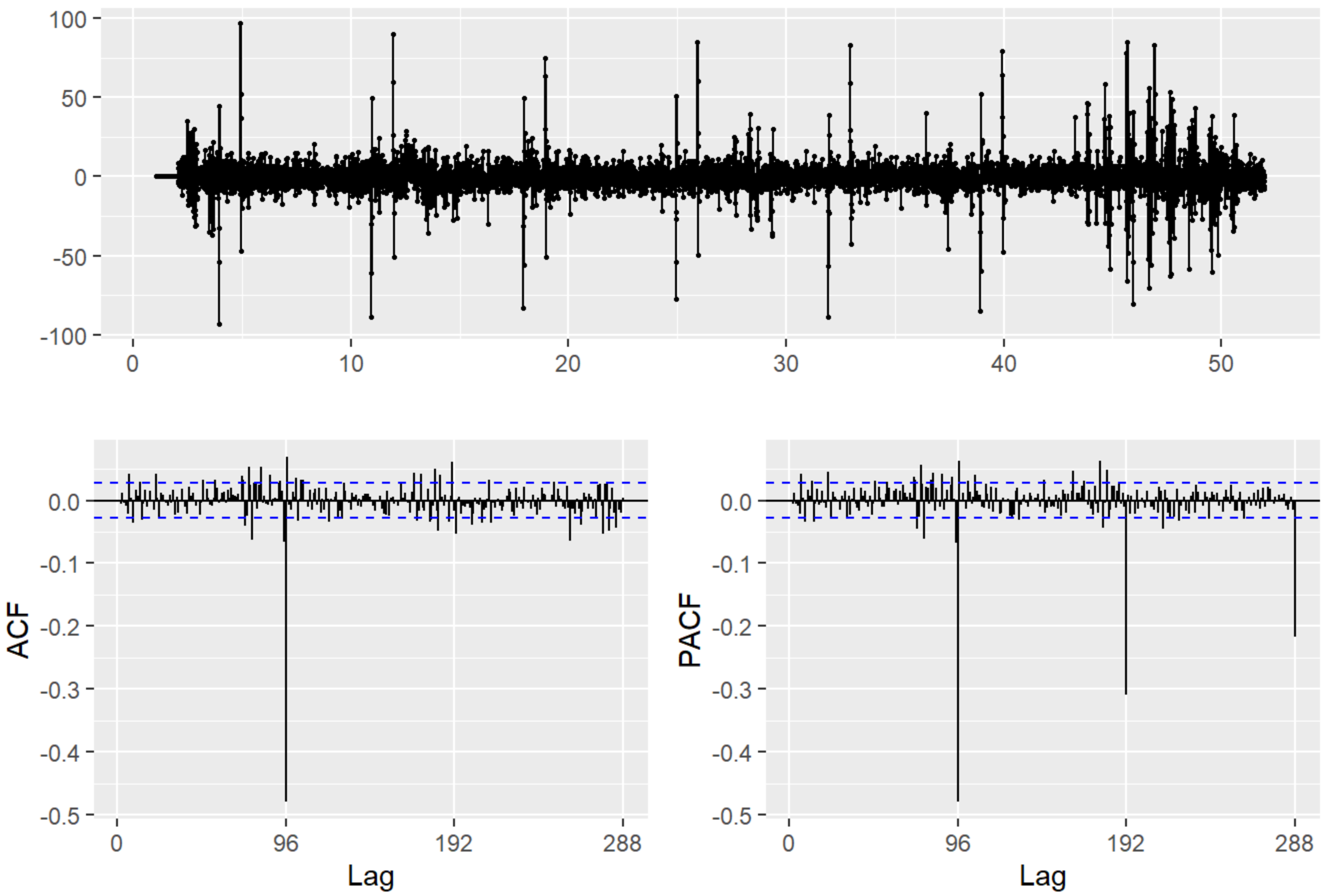
Retrain model on full Power time series and forecast unknown next 96 observations

SARIMA - Daily period

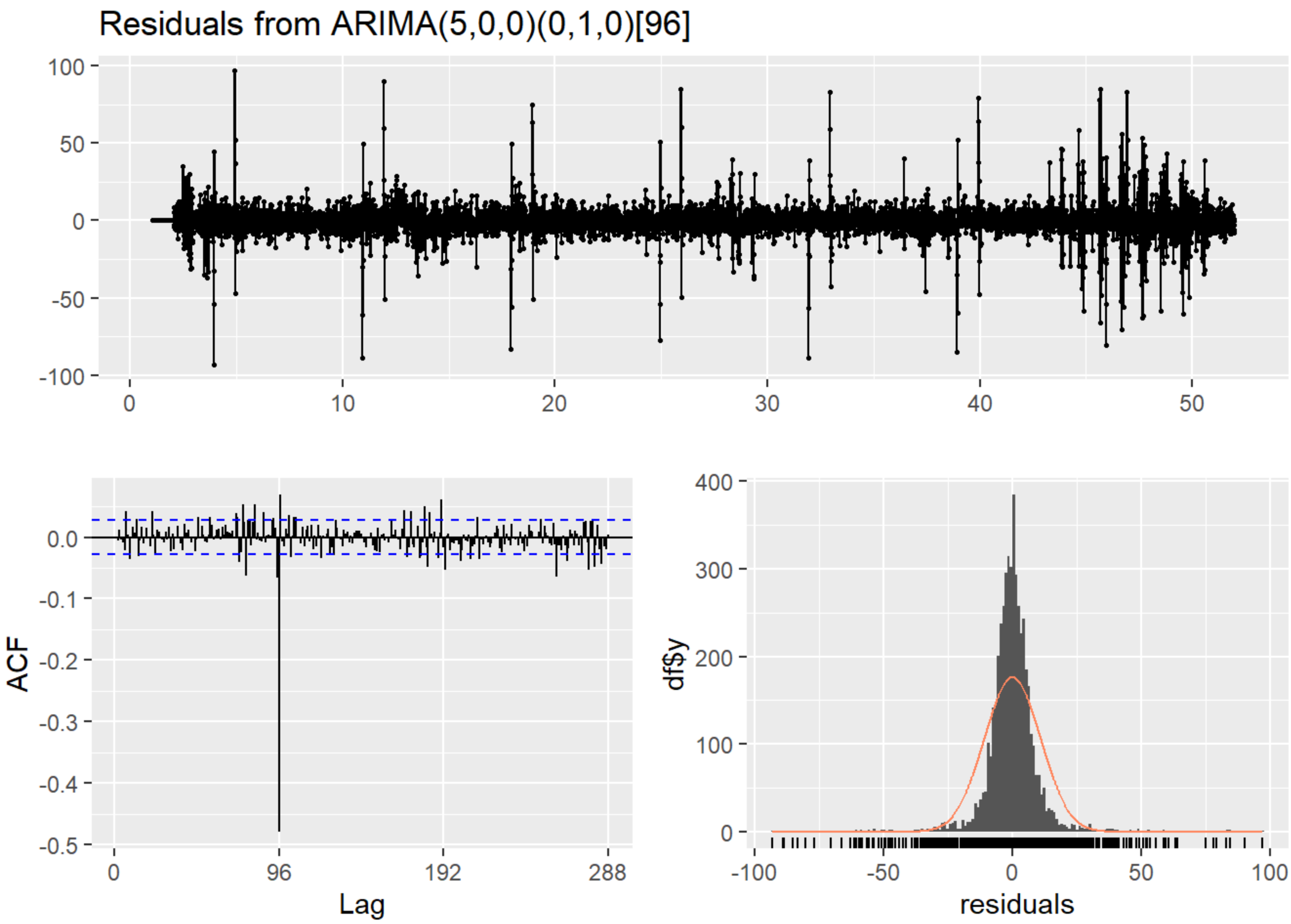
```
# SARIMA, daily period
exec_t_start = Sys.time()

fit = Arima(y_daily,
            order = c(5,0,0),
            seasonal = c(0,1,0))
fit |>  summary()
```

```
## Series: y_daily
## ARIMA(5,0,0)(0,1,0)[96]
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5
##       0.6699  0.0661  0.1630  -0.2808  0.1322
## s.e.  0.0143  0.0168  0.0167   0.0168  0.0143
##
## sigma^2 = 120.9:  log likelihood = -18297.01
## AIC=36606.01   AICc=36606.03   BIC=36644.86
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.1079554 10.87974 6.414877 -0.1535751 2.904919 0.7366183
##                      ACF1
## Training set 0.0005788584
```

```
ggtsdisplay(fit$residuals)
```




```
checkresiduals(fit, plot = TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,0,0)(0,1,0)[96]
## Q* = 1571.1, df = 187, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 192
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 32.74488 secs
```

```
saveRDS(fit, file = "Final_model_without_covariate_SARIMA_daily.rds")
```

```
# forecast 96 next values
horizon = 96

prevision_SARIMA = forecast(readRDS("Final_model_without_covariate_SARIMA_daily.rds"), h= horizo
n)$mean
```

Random Forest - Daily period

```
# next observation based on last day
df_daily = as.vector(y_daily)[1:(96+1)]
for (i in 1:(length(y_daily)-(96+1)))
{
  df_daily = rbind(df_daily, as.vector(y_daily)[(i+1):(i+96+1)])
}
```

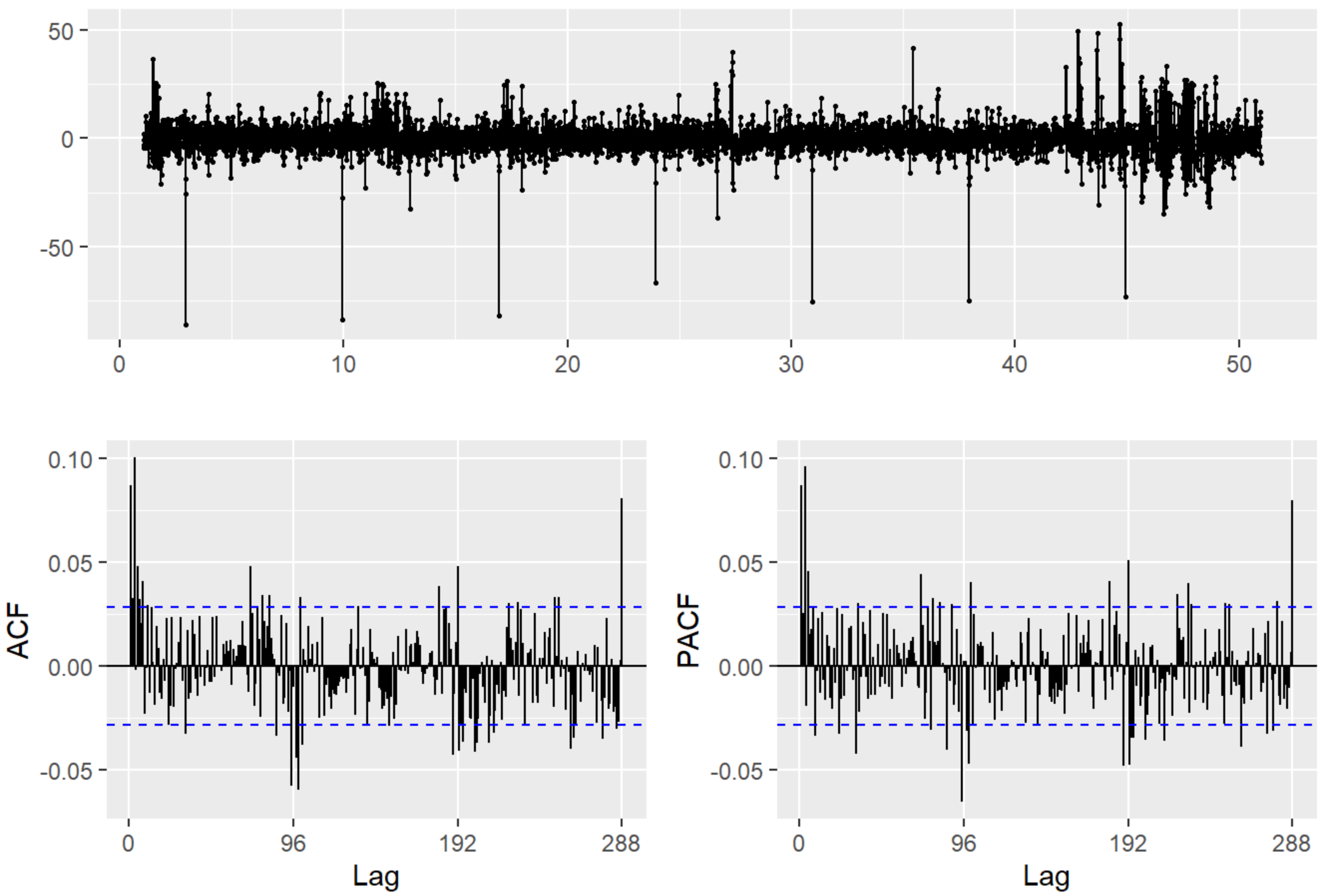
```
# train model
exec_t_start = Sys.time()

fit = randomForest(x = df_daily[,-(96+1)], y = df_daily[, (96+1)])
fit |>  summary()
```

```
##                 Length Class  Mode
## call               3   -none- call
## type               1   -none- character
## predicted       4795   -none- numeric
## mse              500   -none- numeric
## rsq              500   -none- numeric
## oob.times       4795   -none- numeric
## importance        96   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y               4795   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
```

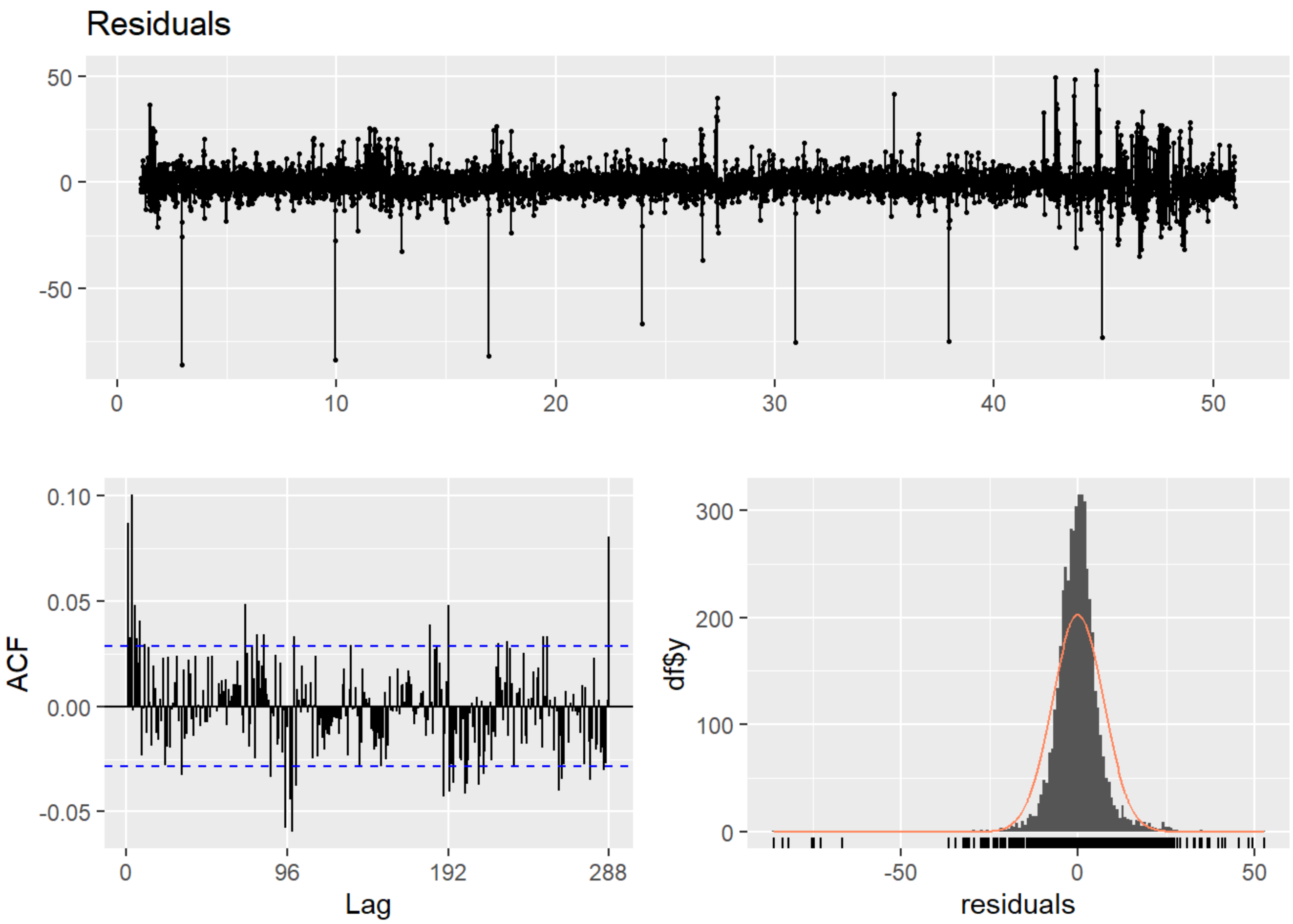
```
e = ts(fit$y - fit$predicted, start = c(1,6), frequency = 96)
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
```

```
## [1] "Train RMSE: 7.24713494926096"
```

```
ggtsdisplay(e)
```




```
checkresiduals(e, plot = TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 399, df = 192, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 192
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 1.445392 mins
```

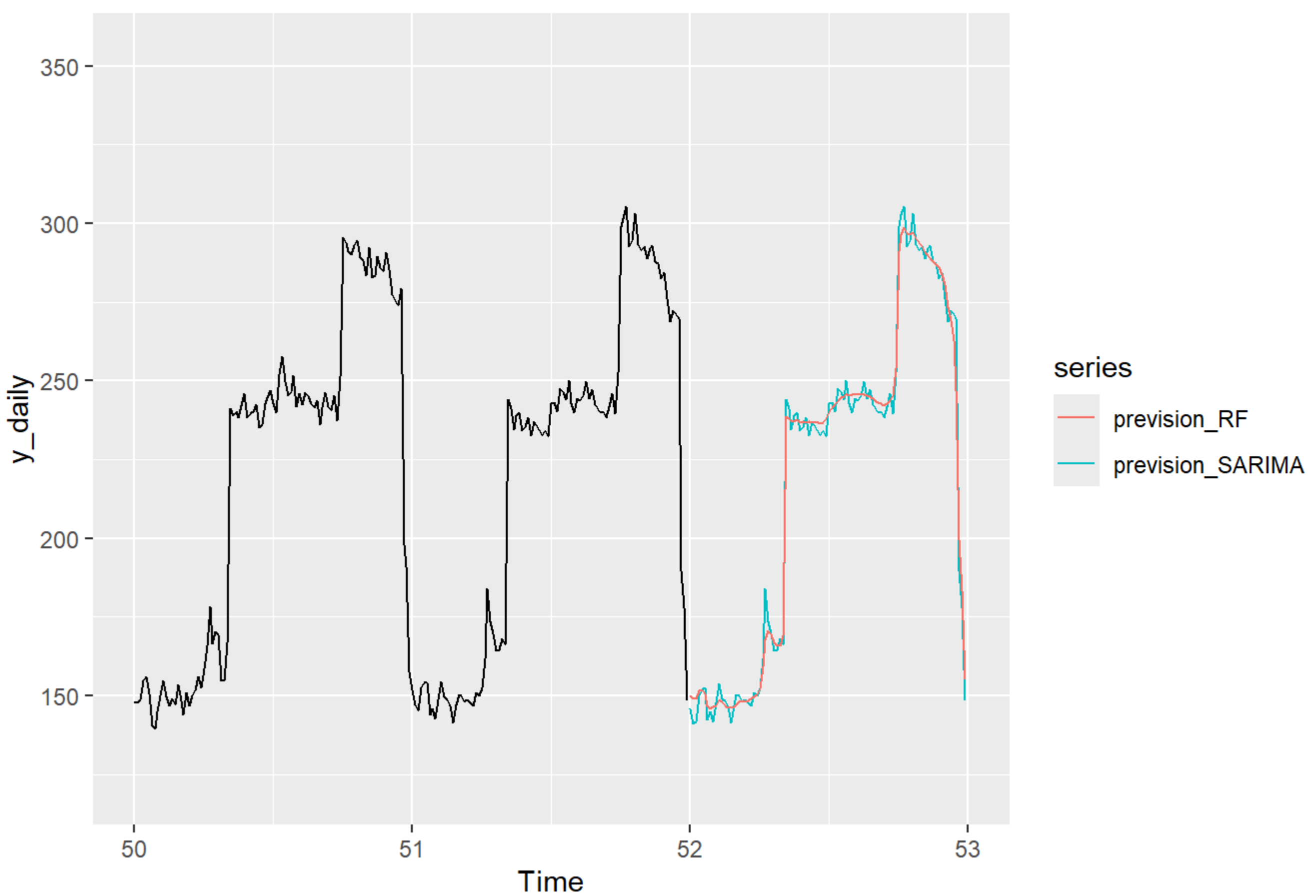
```
saveRDS(fit, file = "Final_model_without_covariate_RF_daily.rds")
```

```
# forecast 96 next values
horizon = 96
newdata_ML = tail(y_daily, horizon)

prevision_RF = ts(
  forecast_ML(readRDS("Final_model_without_covariate_RF_daily.rds"),
              newdata = matrix(newdata_ML,1),
              horizon),
  start = c(52,1),
  frequency = 96)
```

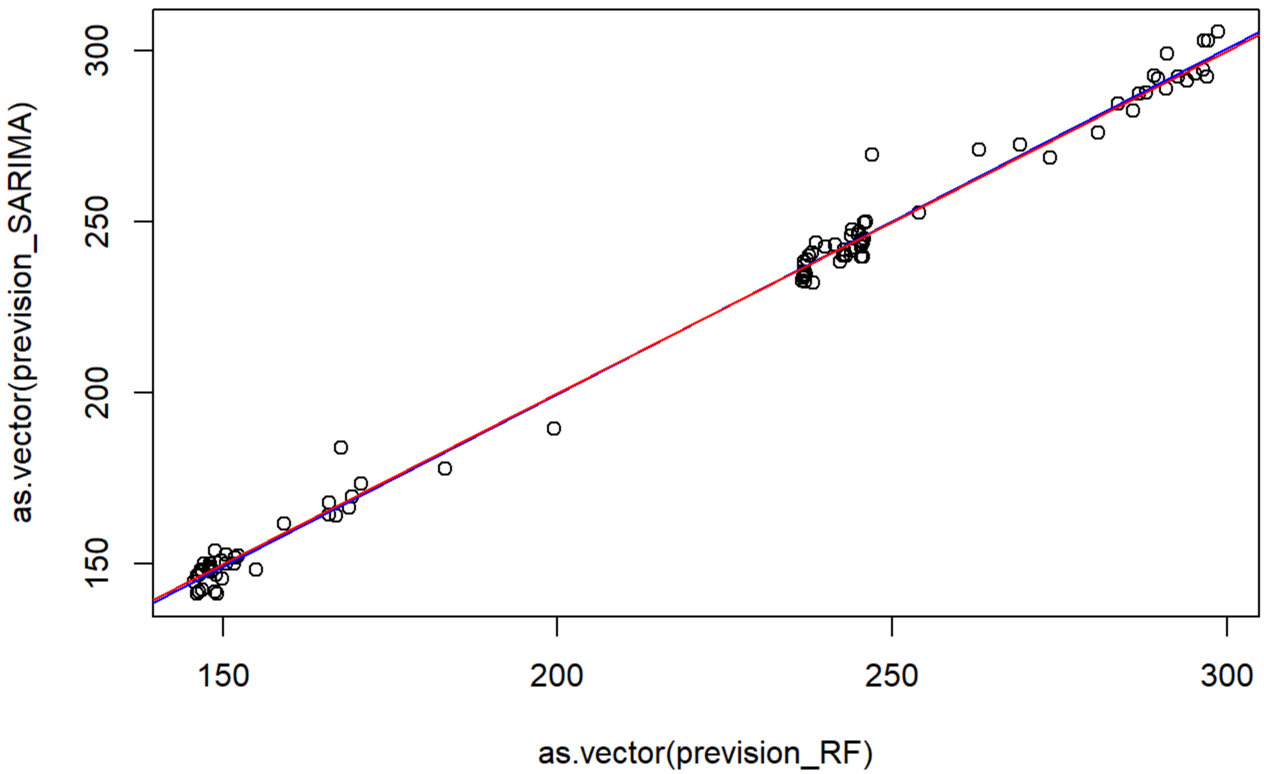
Forecast plots

```
# Plots
autoplot(y_daily) +
  autolayer(prevision_SARIMA) +
  autolayer(prevision_RF)+
  xlim(c(50,53))
```

```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```



```
plot(as.vector(prevision_RF), as.vector(prevision_SARIMA))
abline(lm(as.vector(prevision_SARIMA) ~ as.vector(prevision_RF)), col = "blue")
abline(a = 0 , b = 1, col="red")
```



Notes:

- RF and SARIMA(5,0,0)(0,1,0)[96] forecast are comparable.
- RF provides a “smoother” forecast, while SARIMA seems better are predicting the short term variability of the time series.

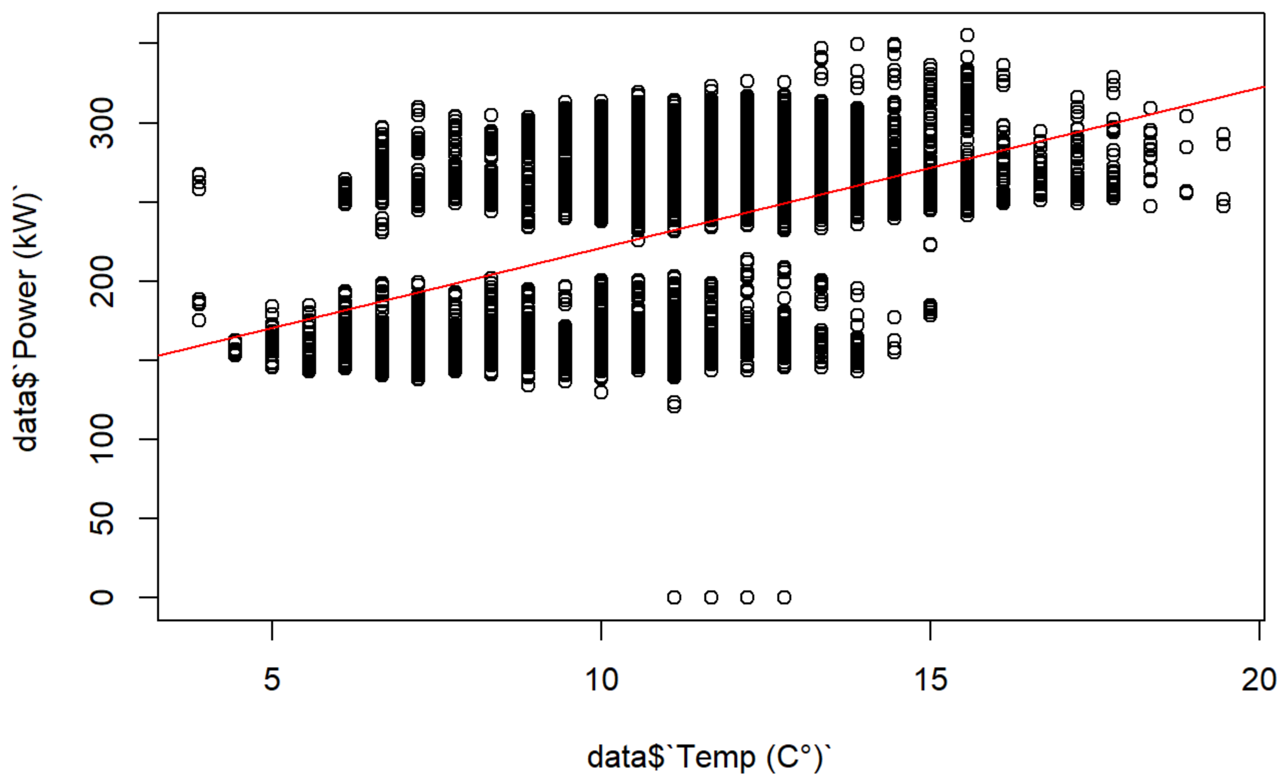
Save forecast results

```
df_results = data.frame(Forecast_no_covariate_SARIMA = prevision_SARIMA)
# df_results$Forecast_no_covariate_RF = prevision_RF
```

Modeling, with co-variates

Correlation power vs. temperature

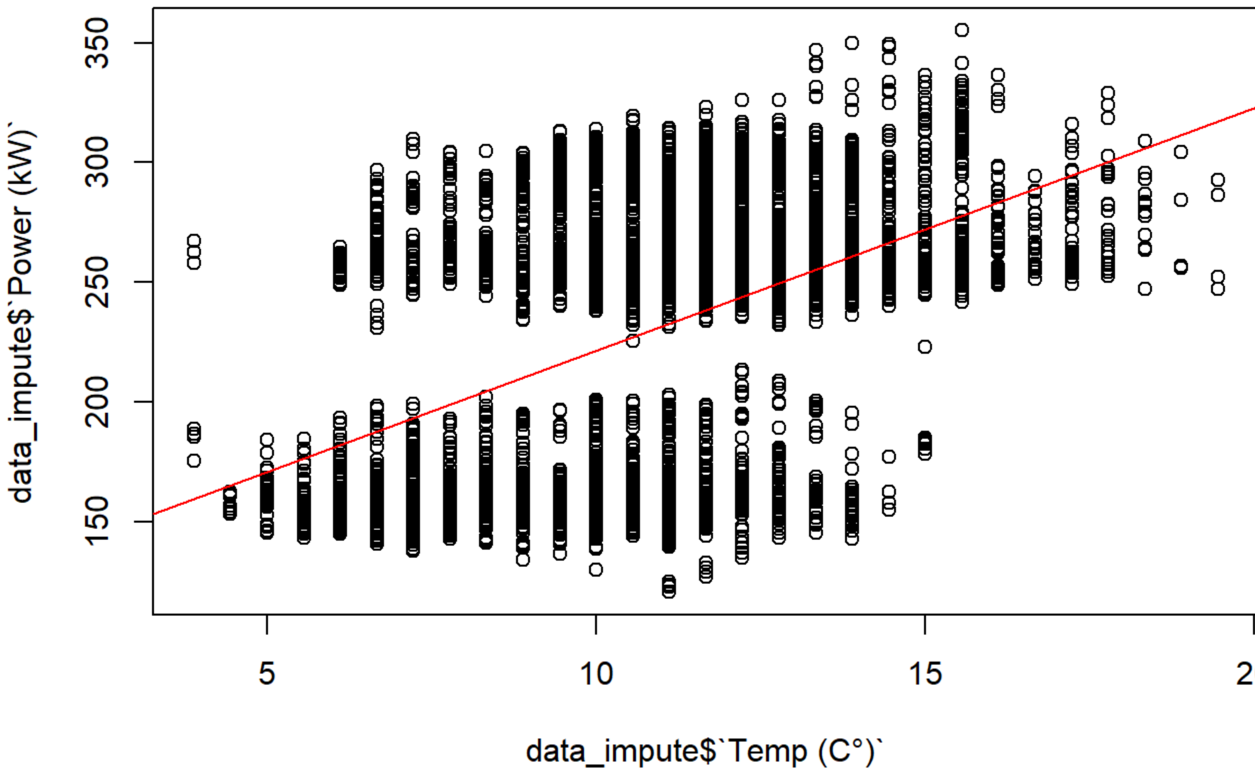
```
plot(data$`Temp (C°)`, data$`Power (kW)`)
abline(lm(data$`Power (kW)`~data$`Temp (C°)`), col="red")
```



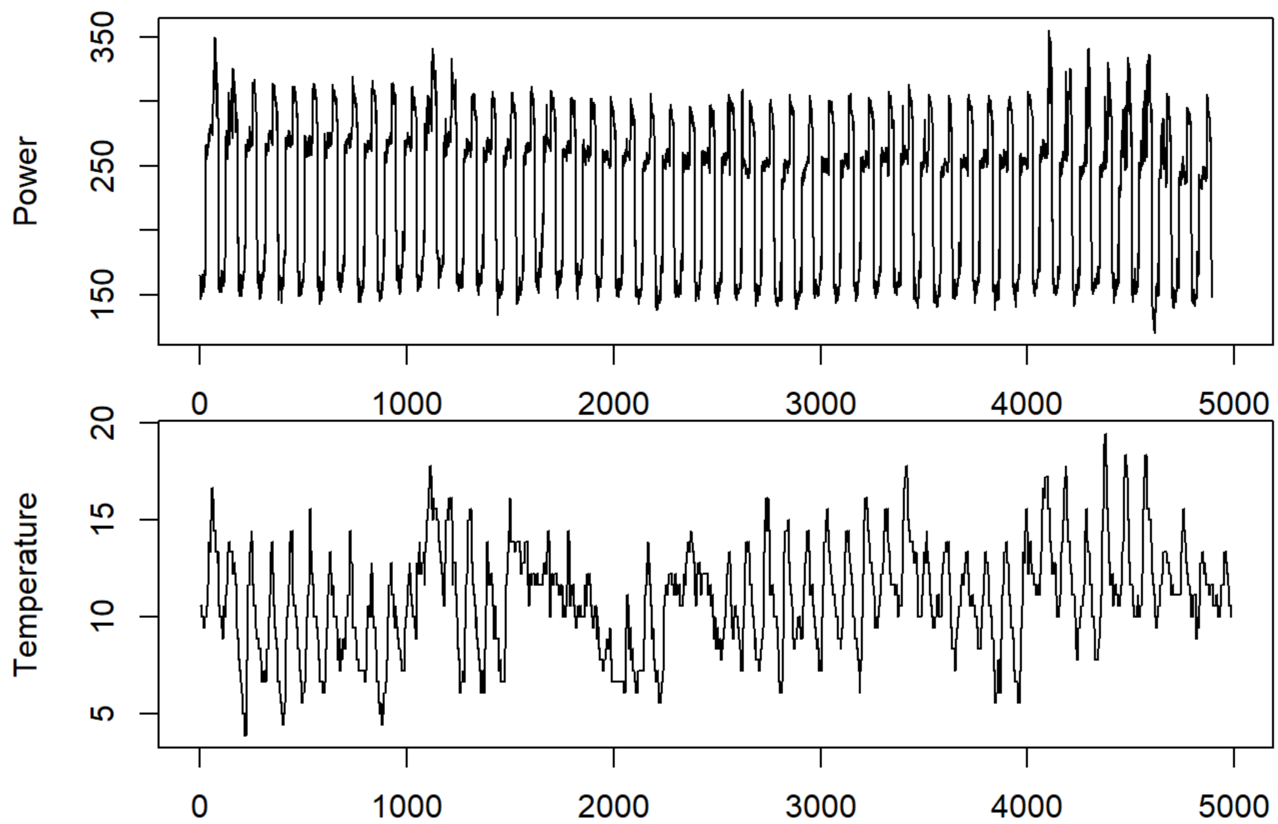

```
# replace "zero" values
data_impute = data
data_impute[1:length(ts_power_impute), 2] = ts_power_impute

print(paste0("Correlation coef. Power vs. Temp = ", cor(data_impute$`Power (kW)`, data_impute$`T
emp (C°)`, use = "complete.obs")))
```

```
## [1] "Correlation coef. Power vs. Temp = 0.4737903811482"
```

```
# Plots
plot(data_impute$`Temp (C°)`, data_impute$`Power (kW)`)
abline(lm(data_impute$`Power (kW)`~data_impute$`Temp (C°)`), col="red")
```




```
# Set up the layout for 2 rows and 1 column
par(mfrow = c(2, 1), mar = c(0, 4, 2, 2), oma = c(4, 0, 0, 0))

# Plot the first time series
ts.plot(data_impute$`Power (kW)`, xlab = "", ylab = "Power")

# Plot the second time series
par(mar = c(0, 4, 2, 2))  # Adjust margins for the second plot
ts.plot(data_impute$`Temp (C°)`, xlab = "", ylab = "Temperature")
```




```
# Reset Layout
par(mfrow = c(1, 1))
```

Notes:

- A correlation exists between Power and Temperature (higher Power is observed when Temperature increases).

Data preparation

```
# Convert to time series, daily period
data_impute = ts(data_impute[,2:3], start = c(1,6), frequency = 96)

# Split into train, test and forecast sets
data_train = window(data_impute, end = c(50,96))
data_test = window(data_impute, start = c(51,1), end = c(51,96))
data_forecast = window(data_impute, start = c(52,1))
```

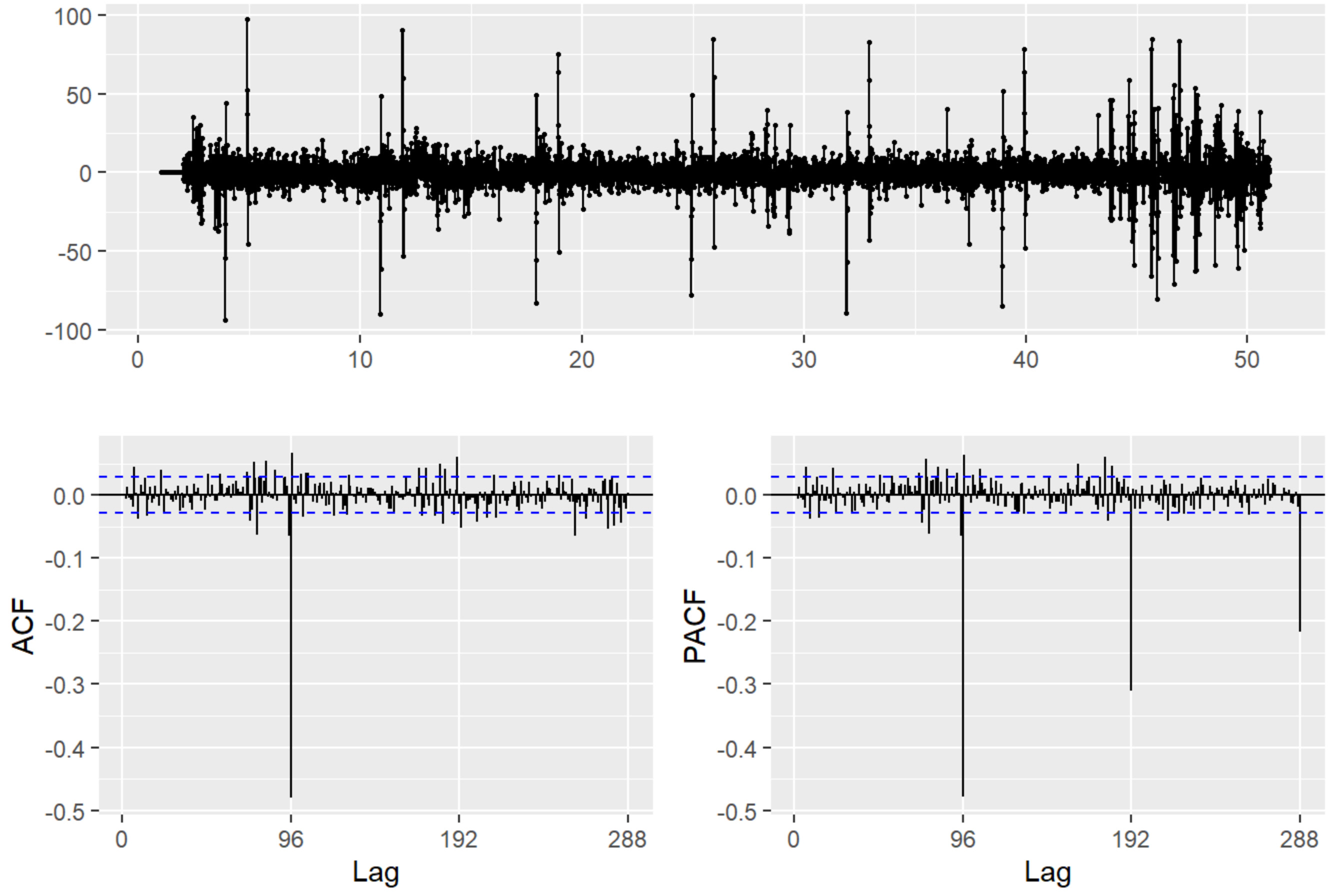
Dynamic Regression

Auto-ARIMA with covariate

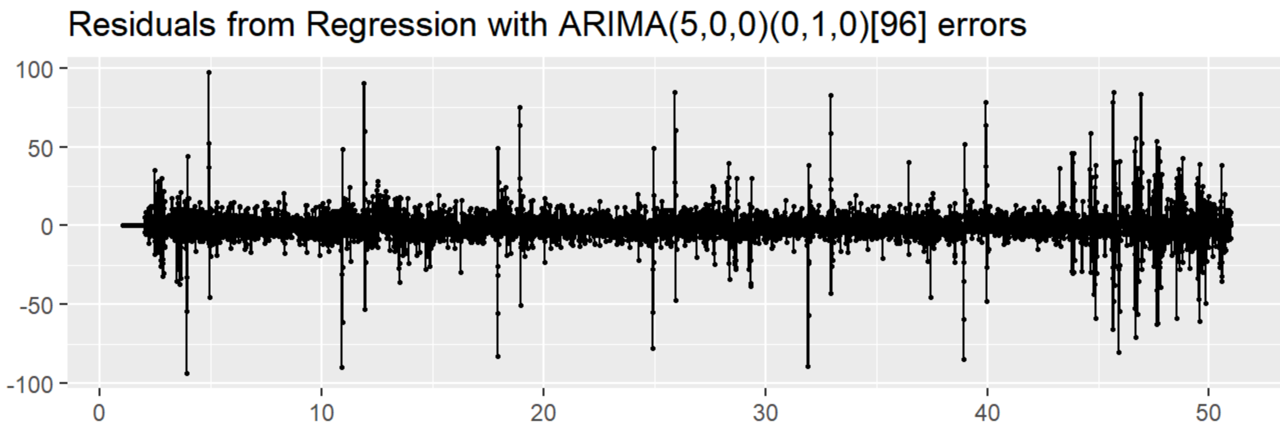
```
# Auto SARIMA, daily period, with covariate
exec_t_start = Sys.time()

fit = auto.arima(data_train[,1], xreg = data_train[,2])
fit |>  summary()
```

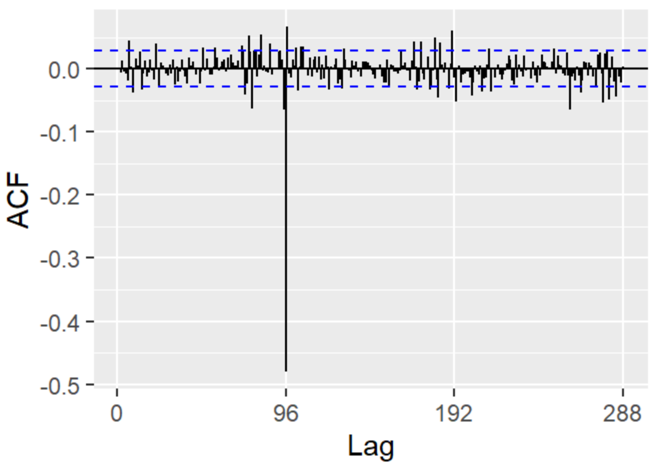
```
## Series: data_train[, 1]
## Regression with ARIMA(5,0,0)(0,1,0)[96] errors
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5    xreg
##       0.6676  0.0680  0.1606  -0.2828  0.1307  0.6002
## s.e.  0.0145  0.0169  0.0168   0.0169  0.0145  0.2279
##
## sigma^2 = 122.6:  log likelihood = -17963.38
## AIC=35940.76   AICc=35940.78   BIC=35985.94
##
## Training set error measures:
##                      ME     RMSE     MAE        MPE     MAPE     MASE
## Training set -0.1053383 10.95349 6.45461 -0.1548071 2.920209 0.734121
##                      ACF1
## Training set 0.0008102879
```

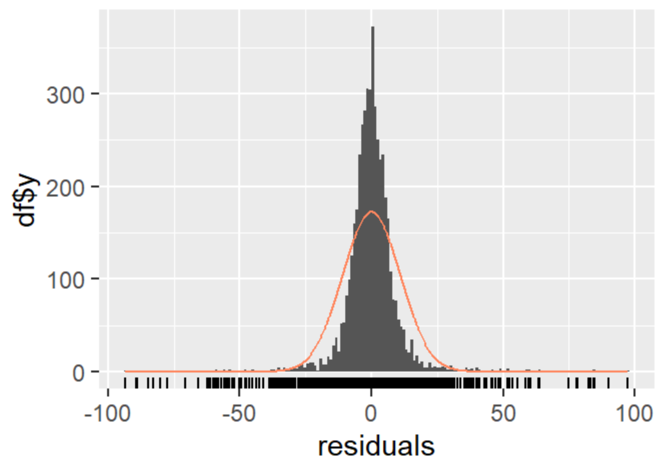
```
ggtsdisplay(fit$residuals)
```




```
checkresiduals(fit, plot = TRUE)
```








```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,0,0)(0,1,0)[96] errors
## Q* = 1542.2, df = 187, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 192
```

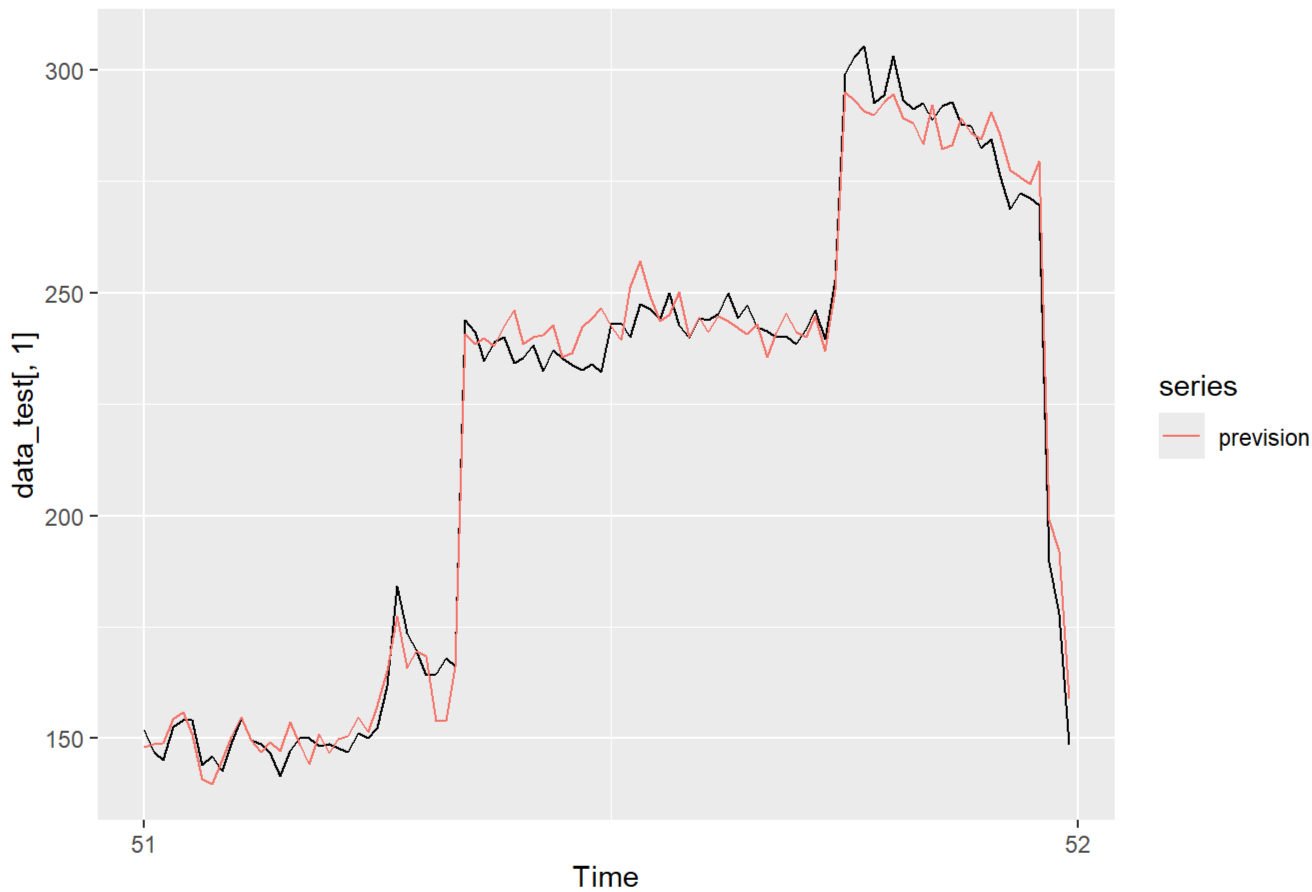
```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 1.303488 mins
```

```
# Prediction on test set
prevision = forecast(fit, h = 96, xreg = data_test[,2])$mean
cat(paste0("Test RMSE: ", RMSE(data_test[,1], prevision)))
```

```
## Test RMSE: 5.94262743321338
```

```
autoplot(data_test[,1]) +
  autolayer(prevision)
```



Notes:

- ACF shows significant autocorrelation at 96 (= 1 day period) and PACF shows exponentially decreasing autocorrelation for daily periods -> try adding seasonal MA (Q = 1)

```
# saveRDS(fit, file = "ARIMA_X_auto_(5,0,0)(0,1,0)[96].rds")
```

Cross-validation

```
# Code commented: very long computation

# # Cross validation using tsCV(), ref: https://pkg.robjhyndman.com/forecast/reference/tsCV.html
#
# # Forcasting function to cross-validate
# Arima_xreg <- function(x, h, xreg, newxreg) {
#   forecast(Arima(x,
#                  order=c(5,0,0),
#                  seasonal = c(0,1,0),
#                  xreg=xreg),
#            xreg=newxreg)
# }
#
# # Crossvalidation execution
# exec_t_start = Sys.time()
#
# e <- tsCV(data_impute[,1], Arima_xreg, h=96, xreg=data_impute[,2], window = 4795)
#
# exec_t_end = Sys.time()
# print(exec_t_end - exec_t_start)
#
# print(paste0("Cross-validation RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
```

Notes:

- Cross validation:

  ```
  Arima_xreg <- function(x, h, xreg, newxreg)
    {
    forecast(Arima(x,
                   order=c(5,0,0),
                   seasonal = c(0,1,0),
                   xreg=xreg), xreg=newxreg)
    }

  e <- tsCV(data_impute[,1], Arima_xreg, h=96, xreg=data_impute[,2], window = 4795)

  print(paste0("Cross-validation RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
  ```

- Results:

  ```
  Time difference of 3.731499 hours
  [1] "Cross-validation RMSE: 6.5453262433807"
  ```

Investigate linear modeling using `tslm()`

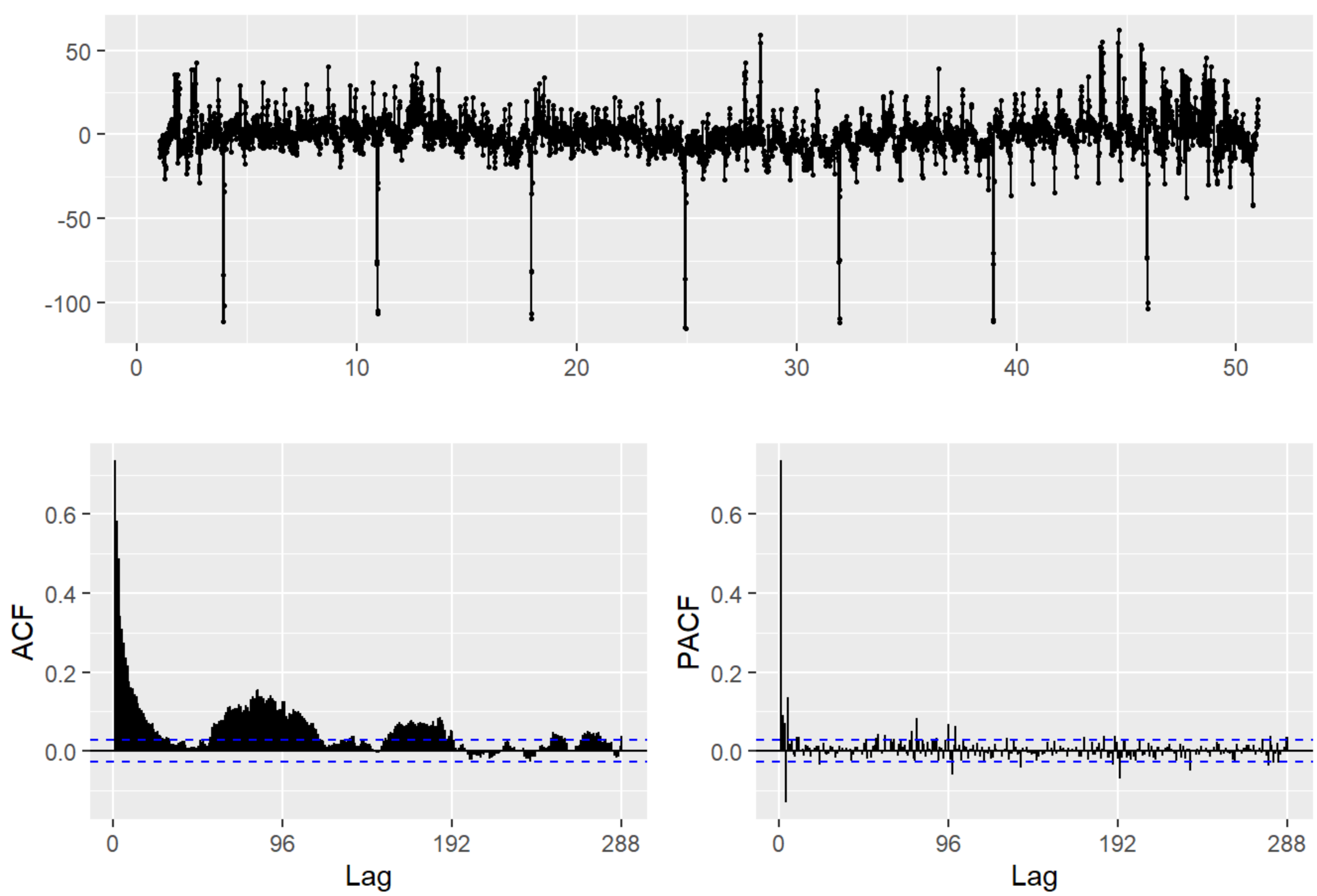
```
exec_t_start = Sys.time()

fit = tslm(data_train[,1] ~ data_train[,2] + trend + season, data = data_train)
fit |> summary()
```

```
##
## Call:
## tslm(formula = data_train[, 1] ~ data_train[, 2] + trend + season,
##     data = data_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -115.518   -5.356    0.001    4.999   62.334
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1.515e+02  2.004e+00  75.610  < 2e-16 ***
## data_train[, 2]  1.414e+00  9.515e-02  14.859  < 2e-16 ***
## trend           -3.446e-03  1.385e-04 -24.883  < 2e-16 ***
## season2         -4.376e-01  2.522e+00  -0.174  0.86226
## season3         -5.958e+00  2.522e+00  -2.362  0.01820 *
## season4         -3.820e-01  2.522e+00  -0.151  0.87960
## season5          2.732e+00  2.522e+00   1.083  0.27867
## season6          2.595e+00  2.510e+00   1.034  0.30129
## season7         -6.207e+00  2.510e+00  -2.473  0.01343 *
## season8         -6.511e+00  2.510e+00  -2.594  0.00951 **
## season9         -3.483e+00  2.510e+00  -1.388  0.16532
## season10        -3.474e+00  2.510e+00  -1.384  0.16651
## season11        -1.017e+00  2.510e+00  -0.405  0.68533
## season12        -1.773e+00  2.510e+00  -0.706  0.48005
## season13        -5.155e-01  2.510e+00  -0.205  0.83730
## season14        -4.476e+00  2.511e+00  -1.782  0.07478 .
## season15        -5.968e+00  2.511e+00  -2.377  0.01752 *
## season16        -3.247e-01  2.511e+00  -0.129  0.89712
## season17         1.615e+00  2.511e+00   0.643  0.52028
## season18         6.386e-01  2.513e+00   0.254  0.79939
## season19        -2.900e-01  2.513e+00  -0.115  0.90813
## season20         8.475e-01  2.513e+00   0.337  0.73592
## season21         9.889e-01  2.513e+00   0.394  0.69391
## season22         2.009e+00  2.513e+00   0.799  0.42409
## season23         2.717e+00  2.513e+00   1.081  0.27980
## season24         4.524e+00  2.513e+00   1.800  0.07192 .
## season25         4.574e+00  2.513e+00   1.820  0.06887 .
## season26         7.737e+00  2.513e+00   3.079  0.00209 **
## season27         1.532e+01  2.513e+00   6.095 1.18e-09 ***
## season28         1.687e+01  2.513e+00   6.711 2.16e-11 ***
## season29         1.767e+01  2.513e+00   7.031 2.34e-12 ***
## season30         2.187e+01  2.513e+00   8.701  < 2e-16 ***
## season31         1.975e+01  2.513e+00   7.857 4.85e-15 ***
## season32         1.563e+01  2.513e+00   6.219 5.45e-10 ***
## season33         1.962e+01  2.513e+00   7.806 7.22e-15 ***
## season34         1.027e+02  2.513e+00  40.863  < 2e-16 ***
## season35         1.005e+02  2.513e+00  40.007  < 2e-16 ***
## season36         9.808e+01  2.513e+00  39.035  < 2e-16 ***
## season37         9.746e+01  2.513e+00  38.789  < 2e-16 ***
## season38         1.003e+02  2.509e+00  39.990  < 2e-16 ***
## season39         9.538e+01  2.509e+00  38.010  < 2e-16 ***
```

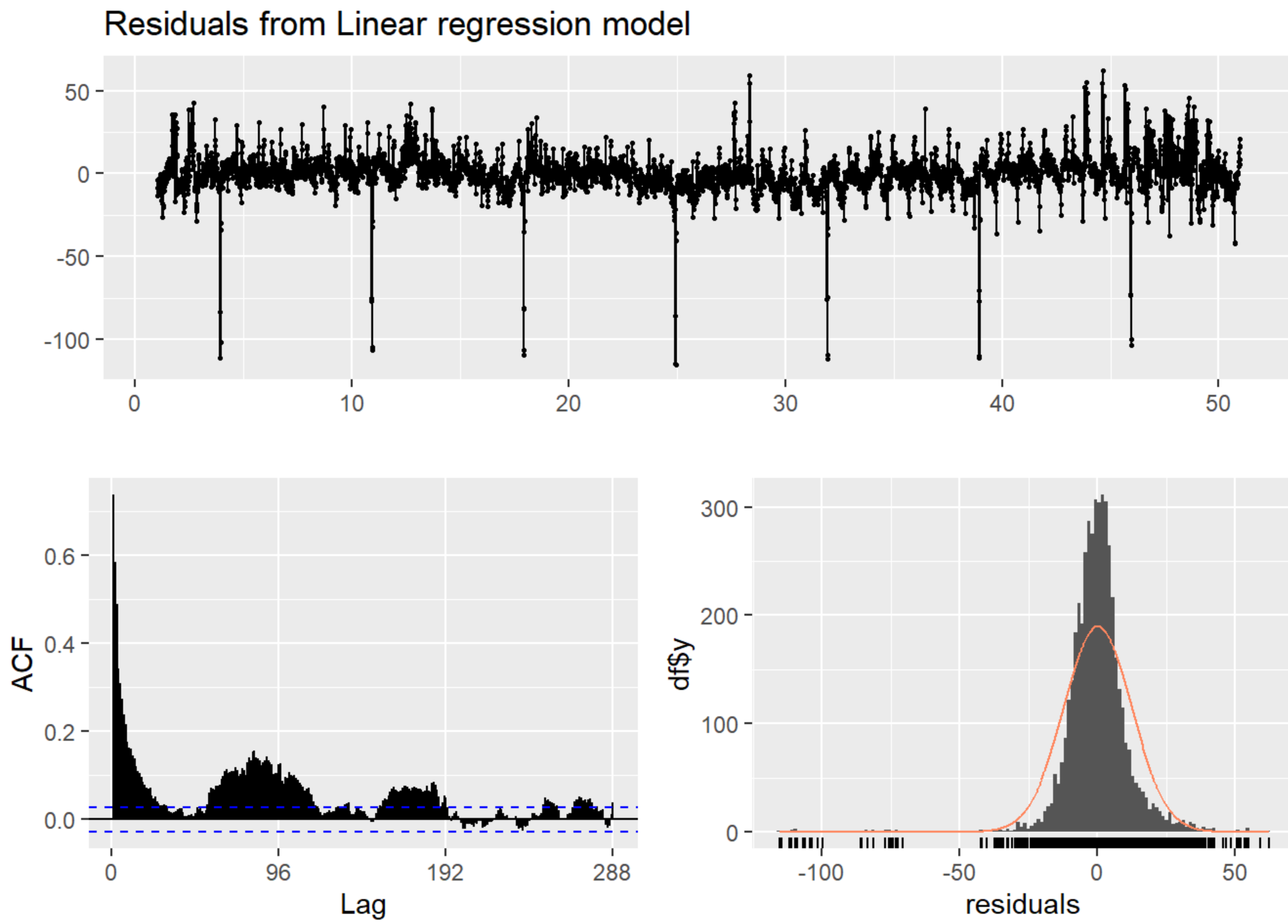
```
## season40          9.729e+01  2.509e+00  38.773  < 2e-16 ***
## season41          9.810e+01  2.509e+00  39.097  < 2e-16 ***
## season42          9.481e+01  2.510e+00  37.771  < 2e-16 ***
## season43          9.624e+01  2.510e+00  38.345  < 2e-16 ***
## season44          9.777e+01  2.510e+00  38.952  < 2e-16 ***
## season45          9.847e+01  2.510e+00  39.233  < 2e-16 ***
## season46          9.995e+01  2.515e+00  39.749  < 2e-16 ***
## season47          9.766e+01  2.515e+00  38.837  < 2e-16 ***
## season48          9.885e+01  2.515e+00  39.309  < 2e-16 ***
## season49          9.928e+01  2.515e+00  39.481  < 2e-16 ***
## season50          9.966e+01  2.522e+00  39.521  < 2e-16 ***
## season51          1.023e+02  2.522e+00  40.584  < 2e-16 ***
## season52          1.011e+02  2.522e+00  40.094  < 2e-16 ***
## season53          1.002e+02  2.522e+00  39.717  < 2e-16 ***
## season54          1.014e+02  2.528e+00  40.101  < 2e-16 ***
## season55          1.012e+02  2.528e+00  40.052  < 2e-16 ***
## season56          1.013e+02  2.528e+00  40.074  < 2e-16 ***
## season57          1.001e+02  2.528e+00  39.611  < 2e-16 ***
## season58          1.002e+02  2.534e+00  39.556  < 2e-16 ***
## season59          1.002e+02  2.534e+00  39.534  < 2e-16 ***
## season60          9.988e+01  2.534e+00  39.412  < 2e-16 ***
## season61          1.017e+02  2.534e+00  40.140  < 2e-16 ***
## season62          1.017e+02  2.537e+00  40.107  < 2e-16 ***
## season63          1.005e+02  2.537e+00  39.609  < 2e-16 ***
## season64          9.975e+01  2.537e+00  39.319  < 2e-16 ***
## season65          9.931e+01  2.537e+00  39.150  < 2e-16 ***
## season66          9.946e+01  2.531e+00  39.291  < 2e-16 ***
## season67          1.016e+02  2.531e+00  40.127  < 2e-16 ***
## season68          9.843e+01  2.531e+00  38.887  < 2e-16 ***
## season69          9.701e+01  2.531e+00  38.322  < 2e-16 ***
## season70          1.154e+02  2.522e+00  45.736  < 2e-16 ***
## season71          1.261e+02  2.522e+00  49.998  < 2e-16 ***
## season72          1.403e+02  2.522e+00  55.620  < 2e-16 ***
## season73          1.446e+02  2.522e+00  57.339  < 2e-16 ***
## season74          1.419e+02  2.515e+00  56.440  < 2e-16 ***
## season75          1.425e+02  2.515e+00  56.644  < 2e-16 ***
## season76          1.412e+02  2.515e+00  56.159  < 2e-16 ***
## season77          1.408e+02  2.515e+00  55.998  < 2e-16 ***
## season78          1.470e+02  2.513e+00  58.501  < 2e-16 ***
## season79          1.436e+02  2.513e+00  57.154  < 2e-16 ***
## season80          1.417e+02  2.513e+00  56.366  < 2e-16 ***
## season81          1.404e+02  2.513e+00  55.869  < 2e-16 ***
## season82          1.412e+02  2.511e+00  56.217  < 2e-16 ***
## season83          1.389e+02  2.511e+00  55.308  < 2e-16 ***
## season84          1.379e+02  2.511e+00  54.935  < 2e-16 ***
## season85          1.375e+02  2.511e+00  54.737  < 2e-16 ***
## season86          1.357e+02  2.510e+00  54.061  < 2e-16 ***
## season87          1.335e+02  2.510e+00  53.190  < 2e-16 ***
## season88          1.323e+02  2.510e+00  52.721  < 2e-16 ***
## season89          1.304e+02  2.510e+00  51.940  < 2e-16 ***
## season90          1.139e+02  2.509e+00  45.406  < 2e-16 ***
## season91          1.121e+02  2.509e+00  44.673  < 2e-16 ***
```

```
## season92         1.069e+02  2.509e+00  42.615  < 2e-16 ***
## season93         1.077e+02  2.509e+00  42.912  < 2e-16 ***
## season94         3.234e+01  2.509e+00  12.889  < 2e-16 ***
## season95         3.331e+01  2.509e+00  13.274  < 2e-16 ***
## season96         3.113e+00  2.509e+00   1.241  0.21485
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.48 on 4697 degrees of freedom
## Multiple R-squared:  0.9542, Adjusted R-squared:  0.9532
## F-statistic:  1008 on 97 and 4697 DF,  p-value: < 2.2e-16
```

```
ggtsdisplay(fit$residuals)
```




```
checkresiduals(fit, plot = TRUE)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 192
##
## data:  Residuals from Linear regression model
## LM test = 2857, df = 192, p-value < 2.2e-16
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

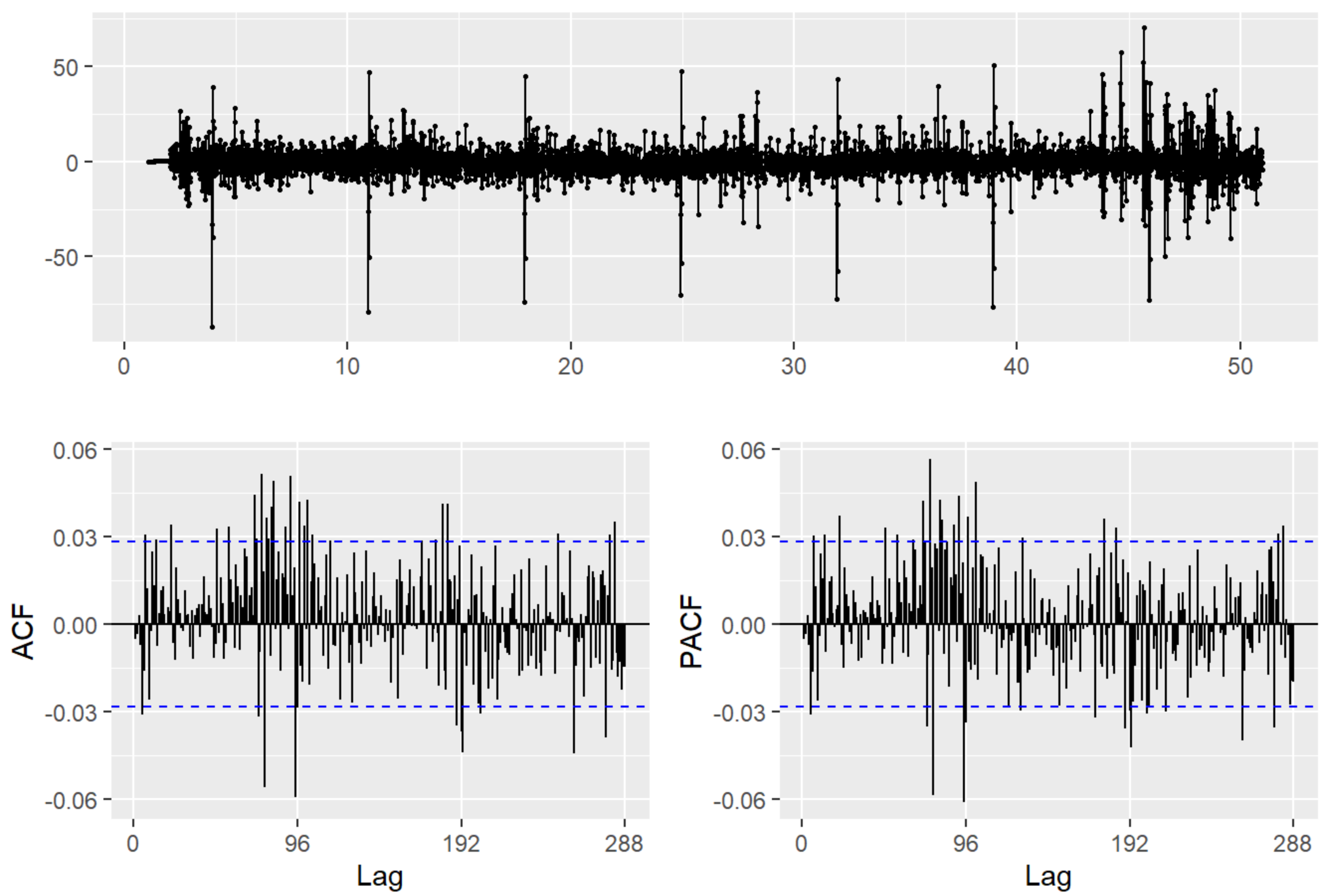
```
## Time difference of 2.198074 secs
```

Fit manually ARIMA with covariate

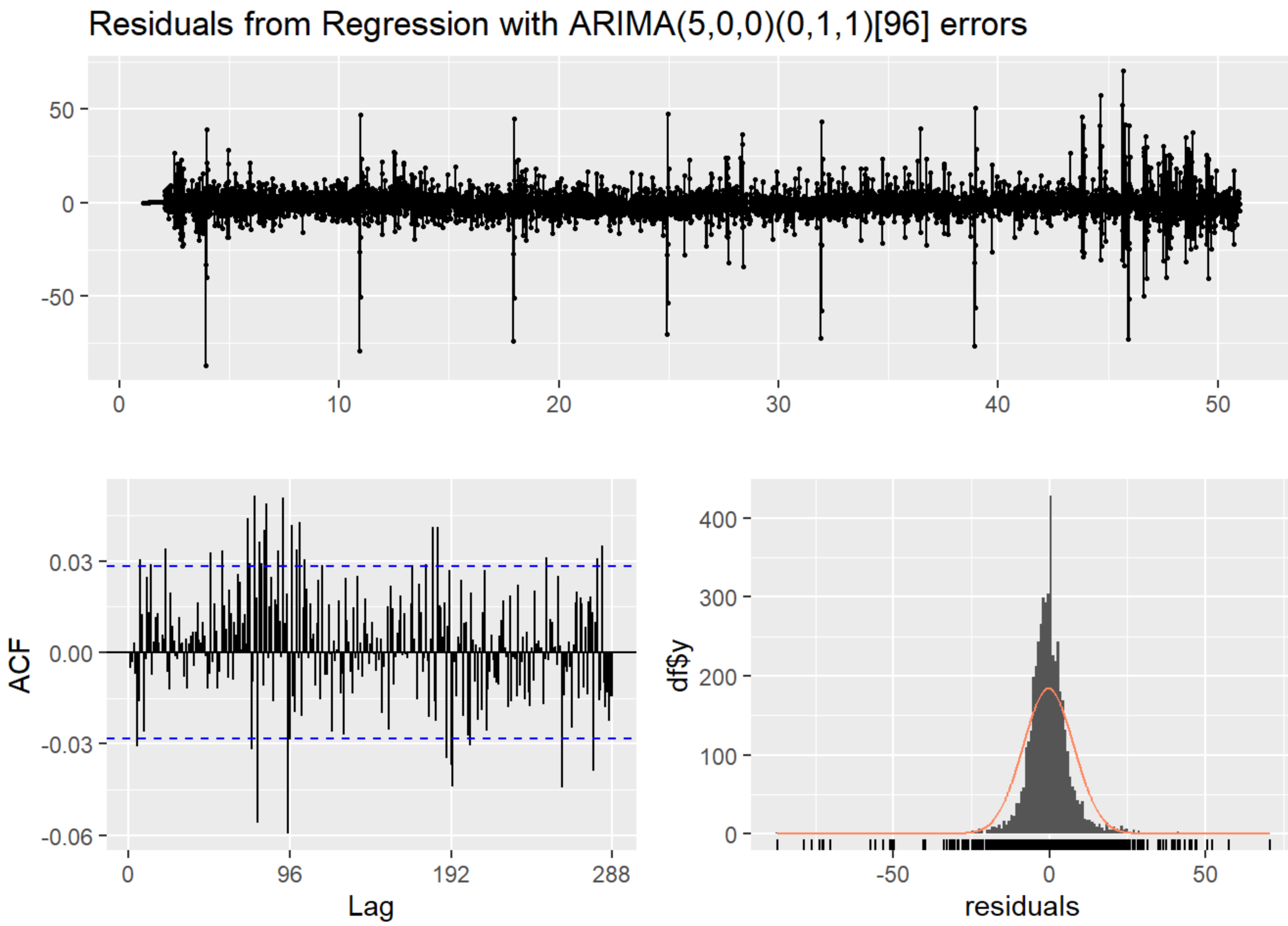
```
# SARIMA, daily period
exec_t_start = Sys.time()

fit = Arima(data_train[,1],
            xreg = data_train[,-1],
            order = c(5,0,0),
            seasonal = c(0,1,1))
fit |>  summary()
```

```
## Series: data_train[, 1]
## Regression with ARIMA(5,0,0)(0,1,1)[96] errors
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5     sma1    xreg
##       0.6692  0.0689  0.1616  -0.2370  0.1239  -0.8742  0.6670
## s.e.  0.0145  0.0171  0.0170   0.0171  0.0145   0.0076  0.2162
##
## sigma^2 = 67.56:  log likelihood = -16632.24
## AIC=33280.48   AICc=33280.51   BIC=33332.12
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE     MASE
## Training set -0.3843715 8.130442 4.992301 -0.2785698 2.259614 0.567804
##                      ACF1
## Training set -0.005122333
```

```
ggtsdisplay(fit$residuals)
```




```
checkresiduals(fit, plot = TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,0,0)(0,1,1)[96] errors
## Q* = 338.49, df = 186, p-value = 5.767e-11
##
## Model df: 6.   Total lags used: 192
```

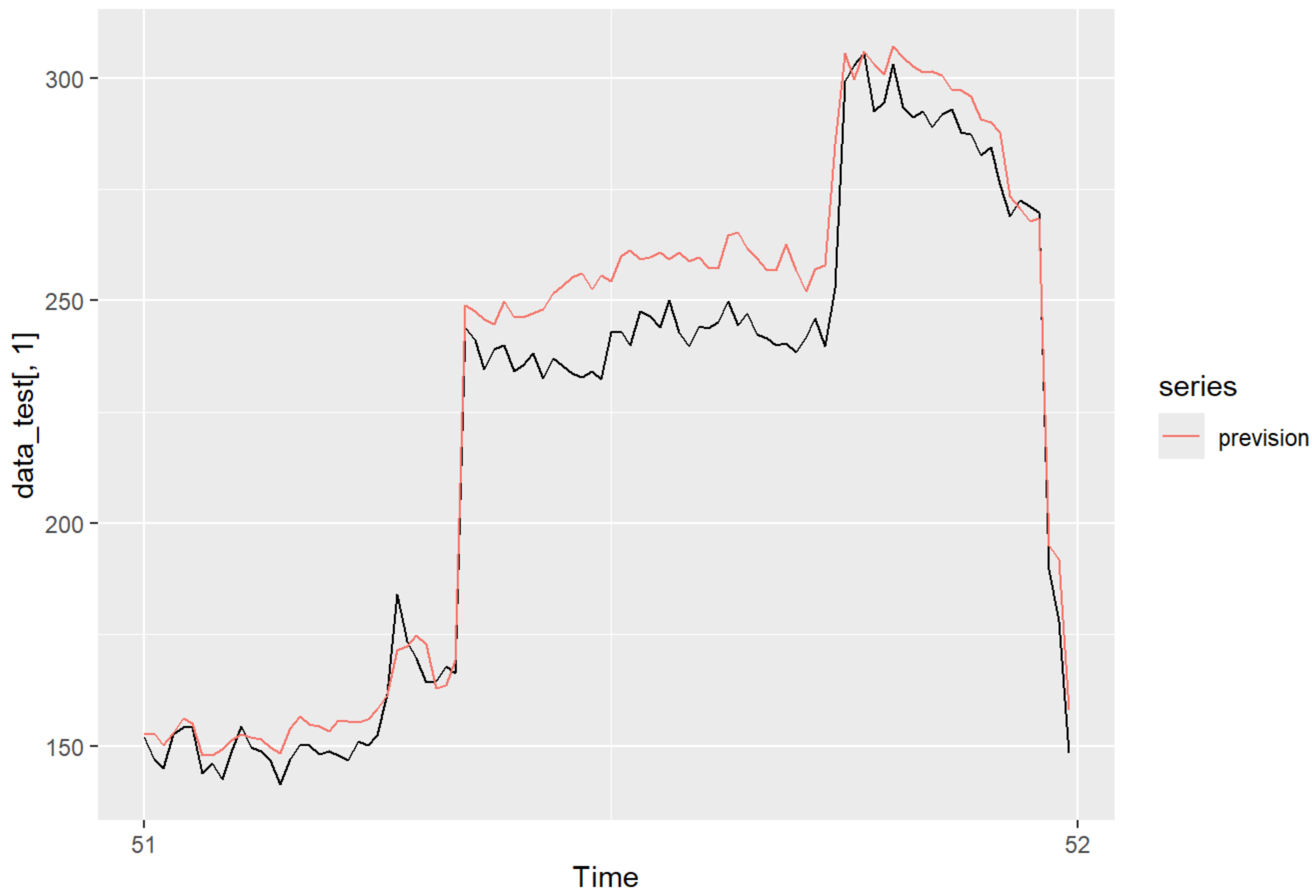
```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 5.972807 mins
```

```
# Prediction on test set
prevision = forecast(fit, h = 96, xreg = data_test[,2])$mean
cat(paste0("Test RMSE: ", RMSE(data_test[,1], prevision)))
```

```
## Test RMSE: 11.5012990199773
```

```
autoplot(data_test[,1]) +
  autolayer(prevision)
```




```
# saveRDS(fit, file = "ARIMA_X_auto_(5,0,0)(0,1,1)[96].rds")
```

ML modeling

ML Data prep

```
# next observation based on last day
df_daily_covariate = as.vector(data_train[1:(96+1),])
for (i in 1:(dim(data_train)[1]-(96+1)))
{
  df_daily_covariate = rbind(df_daily_covariate,
                             as.vector(data_train[(i+1):(i+96+1),]))
}
```

Random Forest

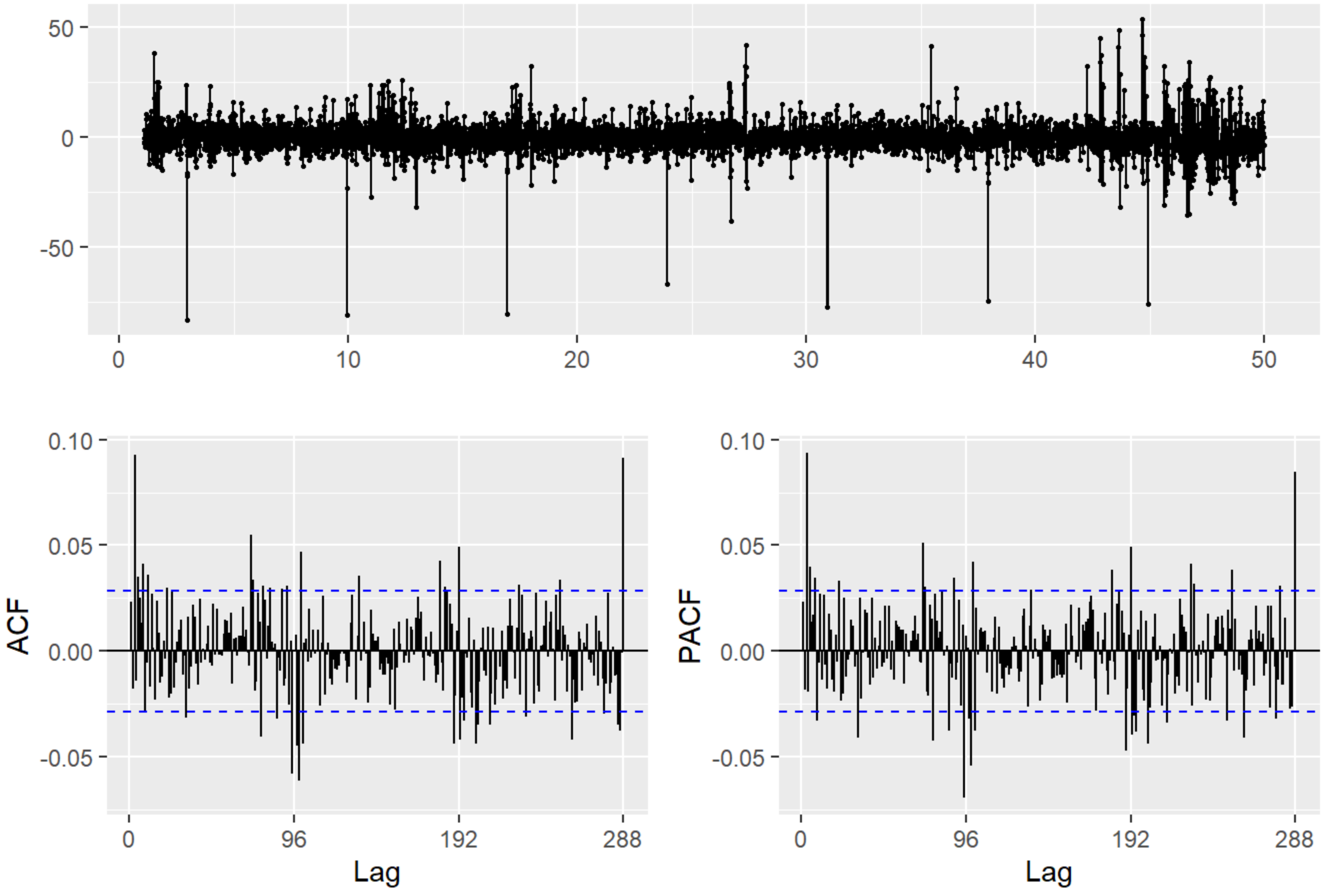
```
exec_t_start = Sys.time()

fit = randomForest(x = df_daily_covariate[,-(96+1)],
                   y = df_daily_covariate[,(96+1)])
fit |>  summary()
```

```
##                 Length Class  Mode
## call               3   -none- call
## type               1   -none- character
## predicted       4699   -none- numeric
## mse              500   -none- numeric
## rsq              500   -none- numeric
## oob.times       4699   -none- numeric
## importance       193   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y               4699   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
```

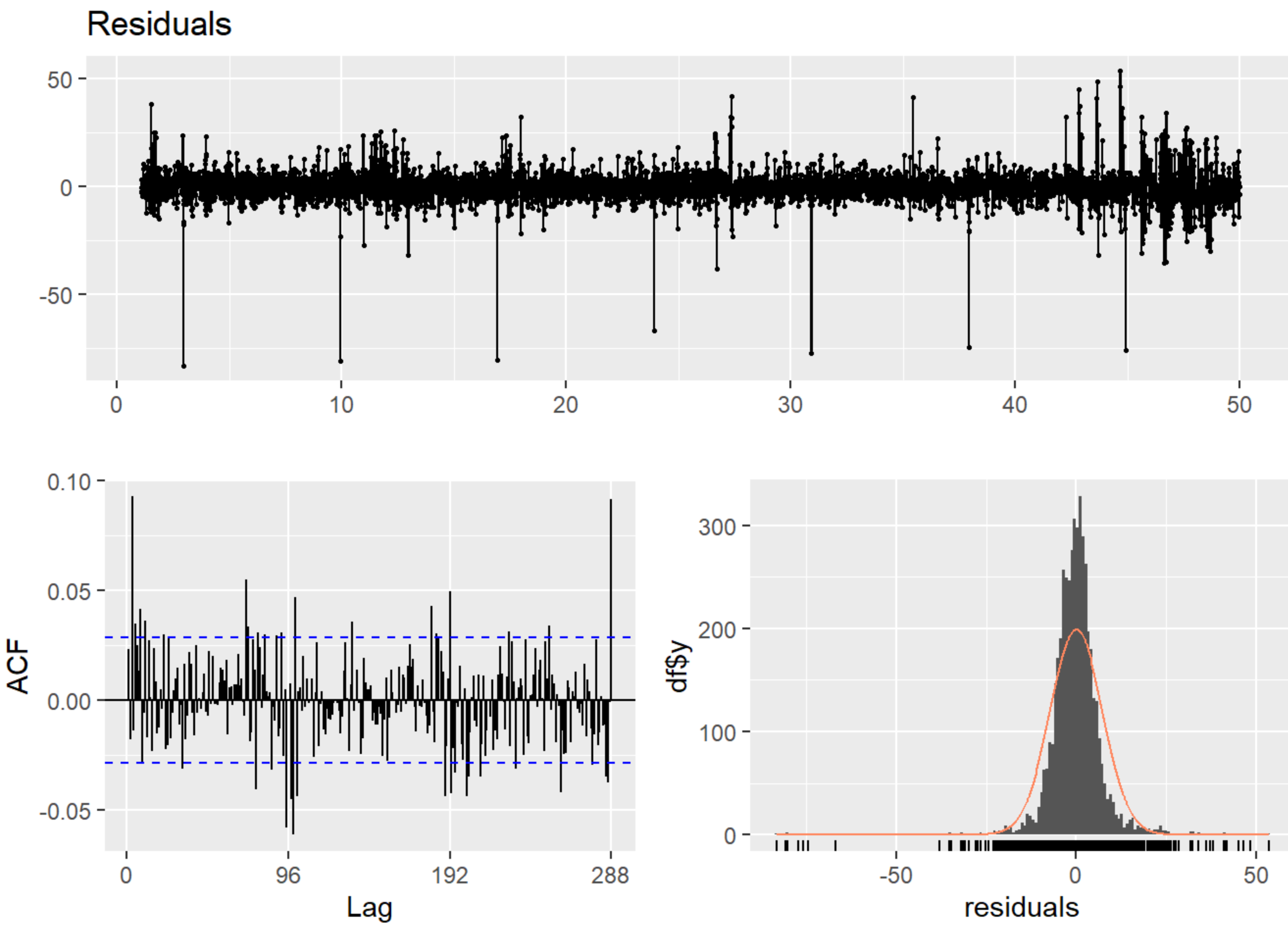
```
e = ts(fit$y - fit$predicted, start = c(1,6), frequency = 96)
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
```

```
## [1] "Train RMSE: 7.23889974531755"
```

```
ggtsdisplay(e)
```




```
checkresiduals(e, plot = TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 371.7, df = 192, p-value = 1.41e-13
##
## Model df: 0.   Total lags used: 192
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 2.562062 mins
```

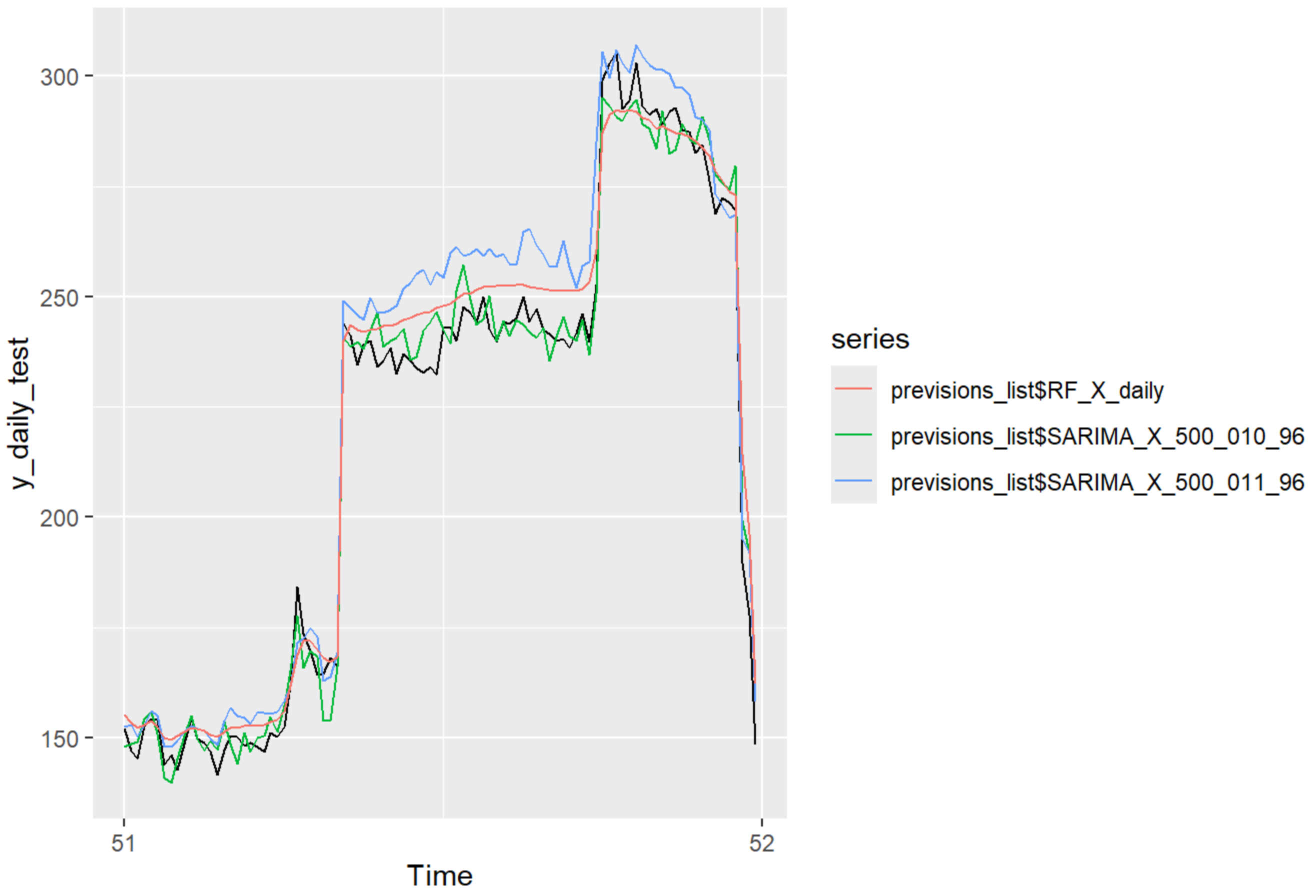
```
# saveRDS(fit, file = "RF_X_daily.rds")
```

Model performance comparison

```
# Build a list of models
models_list = list()
models_list$SARIMA_X_500_010_96 = readRDS('ARIMA_X_auto_(5,0,0)(0,1,0)[96].rds')
models_list$SARIMA_X_500_011_96 = readRDS('ARIMA_X_auto_(5,0,0)(0,1,1)[96].rds')
models_list$RF_X_daily = readRDS('RF_X_daily.rds')

# Make predictions with each model and store RMSE
previsions_list = list()
rmsep_list = list()
horizon = 96
freq = 96
newdata_ML = c(as.vector(tail(data_train, 96)))
xreg = as.vector(data_test[,2])

for (name in names(models_list))
{
  cat(paste0("Forcasting model:", name, "\n"))

  if(grepl("RF", name) | grepl("XG", name)) # Use forecast_ML_X() with ML models
  {
    prevision = forecast_ML_X(models_list[[name]],
                              newdata = matrix(newdata_ML,1),
                              horizon,
                              xreg = xreg)
    prevision = ts(prevision,
                   start = start(data_test),
                   frequency = freq)
  }
  else # Use forecast() with ts models
  {
    prevision = forecast(models_list[[name]], h = horizon, xreg = xreg)
    prevision = prevision$mean
  }

  previsions_list[[name]] = prevision
  rmsep_list[[name]] = RMSE(y_daily_test,prevision)

  cat(paste0("Test set RMSE: ", rmsep_list[[name]], "\n\n"))
}
```

```
## Forcasting model:SARIMA_X_500_010_96
## Test set RMSE: 5.94262743321338
##
## Forcasting model:SARIMA_X_500_011_96
## Test set RMSE: 11.5012990199773
##
## Forcasting model:RF_X_daily
## Test set RMSE: 7.74487203111378
```

```
# Plots
autoplot(y_daily_test) +
  autolayer(previsions_list$SARIMA_X_500_010_96) +
  autolayer(previsions_list$SARIMA_X_500_011_96) +
  autolayer(previsions_list$RF_X_daily)
```



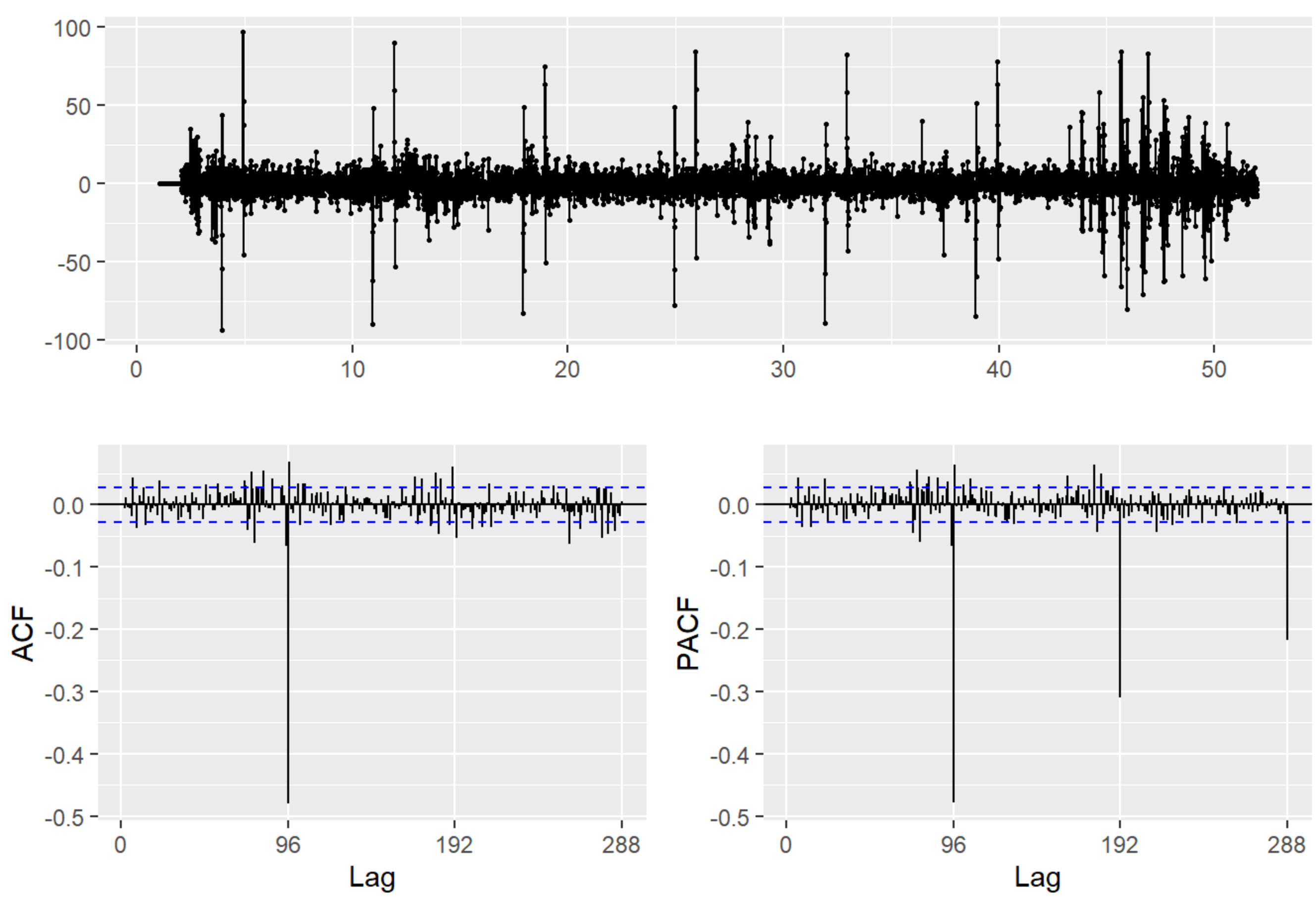
Retrain model on full Power time series and forecast unknown next 96 observations, using Temperature as covariate

SARIMA - Daily period

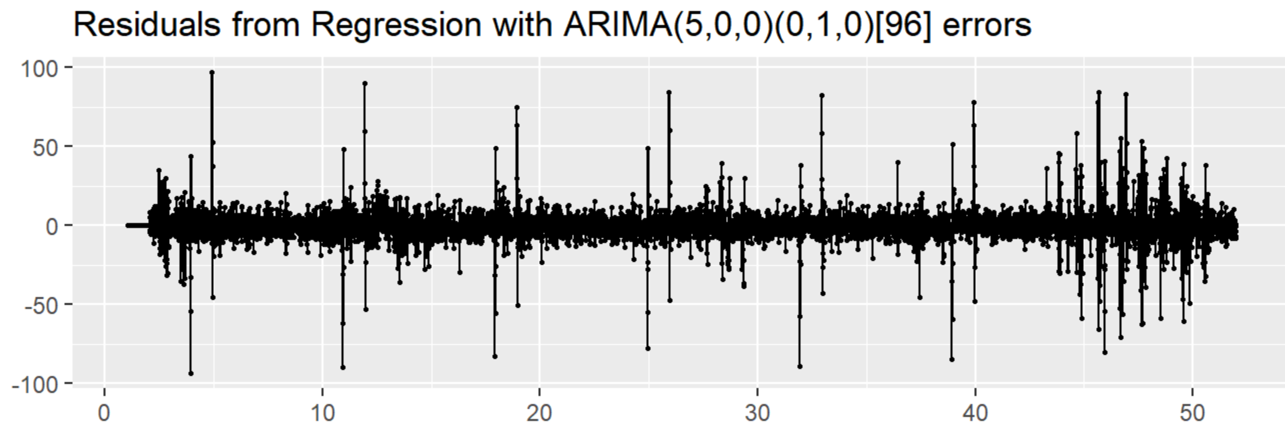
```
# SARIMA, daily period
exec_t_start = Sys.time()

fit = Arima(head(data_impute, dim(data_impute)[1]-96)[,1],
            xreg = head(data_impute, dim(data_impute)[1]-96)[,-1],
            order = c(5,0,0),
            seasonal = c(0,1,0))
fit |>  summary()
```

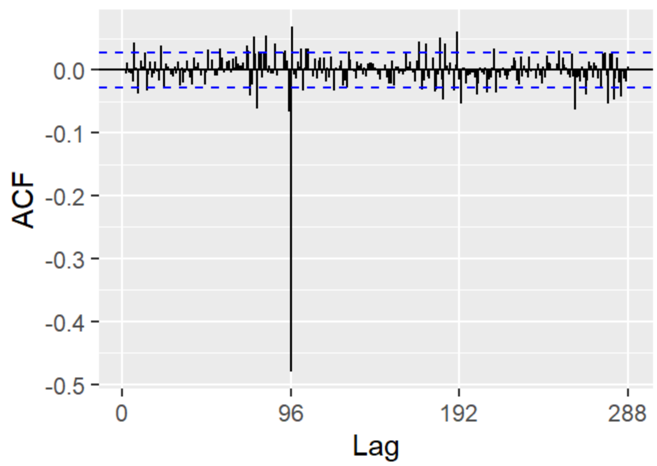
```
## Series: head(data_impute, dim(data_impute)[1] - 96)[, 1]
## Regression with ARIMA(5,0,0)(0,1,0)[96] errors
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5    xreg
##       0.6670  0.0670  0.1613  -0.2812  0.1299  0.5990
## s.e.  0.0143  0.0168  0.0166   0.0168  0.0143  0.2248
##
## sigma^2 = 120.7:  log likelihood = -18293.52
## AIC=36601.04   AICc=36601.06   BIC=36646.37
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.107984 10.87185 6.412602 -0.1545285 2.903275 0.7363569
##                      ACF1
## Training set 0.0008750915
```

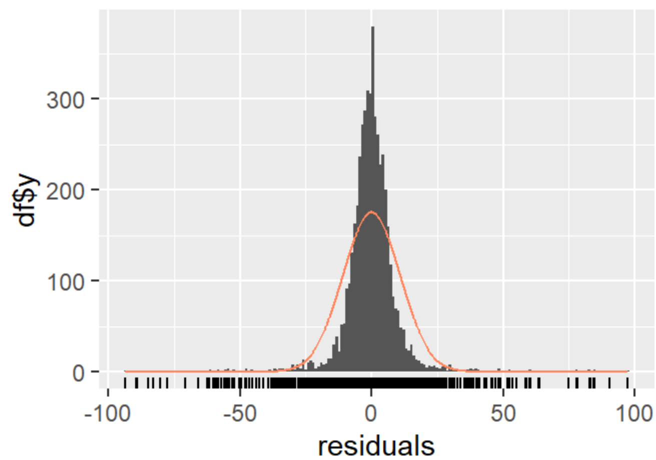
```
ggtsdisplay(fit$residuals)
```



```
checkresiduals(fit, plot = TRUE)
```








```
## 
##  Ljung-Box test
## 
## data:  Residuals from Regression with ARIMA(5,0,0)(0,1,0)[96] errors
## Q* = 1573.6, df = 187, p-value < 2.2e-16
## 
## Model df: 5.   Total lags used: 192
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 43.25694 secs
```

```
saveRDS(fit, file = "Final_model_with_covariate_SARIMA_daily.rds")
```

```
# forecast 96 next values
horizon = 96

prevision_SARIMA_X = forecast(readRDS("Final_model_with_covariate_SARIMA_daily.rds"), h= horizo
n, xreg = data_forecast[,-1])$mean
```

Random Forest - Daily period

```
# next observation based on last day
df_daily_covariate = as.vector(data_impute[1:(96+1),])

for (i in 1:(dim(data_impute)[1]-(96+1+96)))
{
  df_daily_covariate = rbind(df_daily_covariate,
                             as.vector(data_impute[(i+1):(i+96+1),]))
}
```

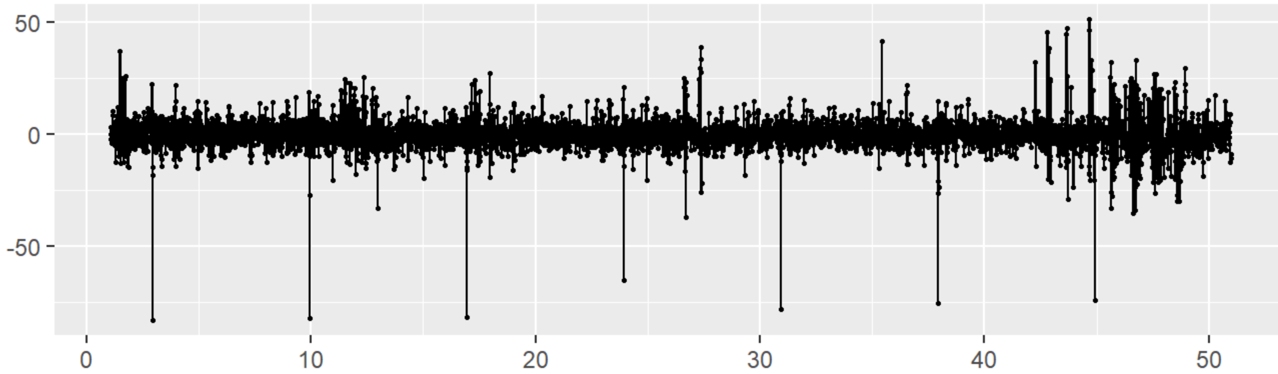
```
exec_t_start = Sys.time()

fit = randomForest(x = df_daily_covariate[,-(96+1)],
                   y = df_daily_covariate[,(96+1)])
fit |>  summary()
```

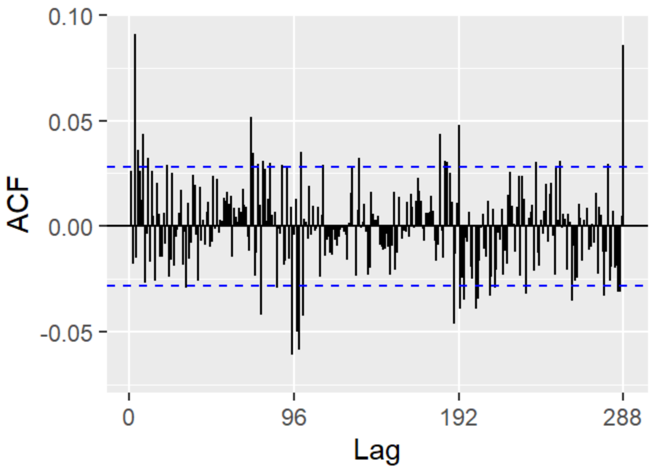
```
##                 Length Class  Mode
## call               3   -none- call
## type               1   -none- character
## predicted       4795   -none- numeric
## mse              500   -none- numeric
## rsq              500   -none- numeric
## oob.times       4795   -none- numeric
## importance       193   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y               4795   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
```

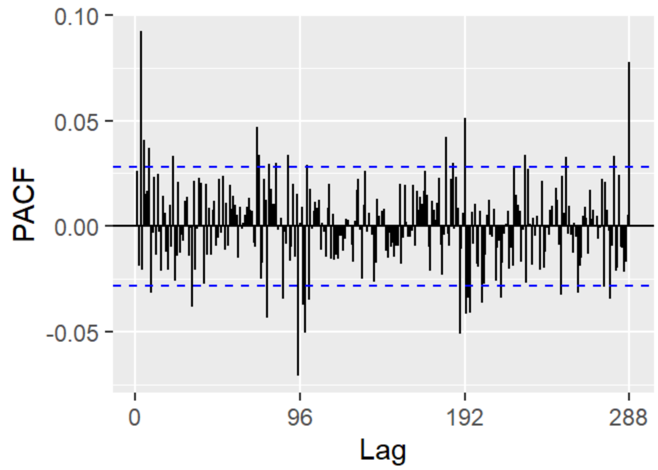
```
e = ts(fit$y - fit$predicted, start = c(1,6), frequency = 96)
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
```

```
## [1] "Train RMSE: 7.20633022987013"
```

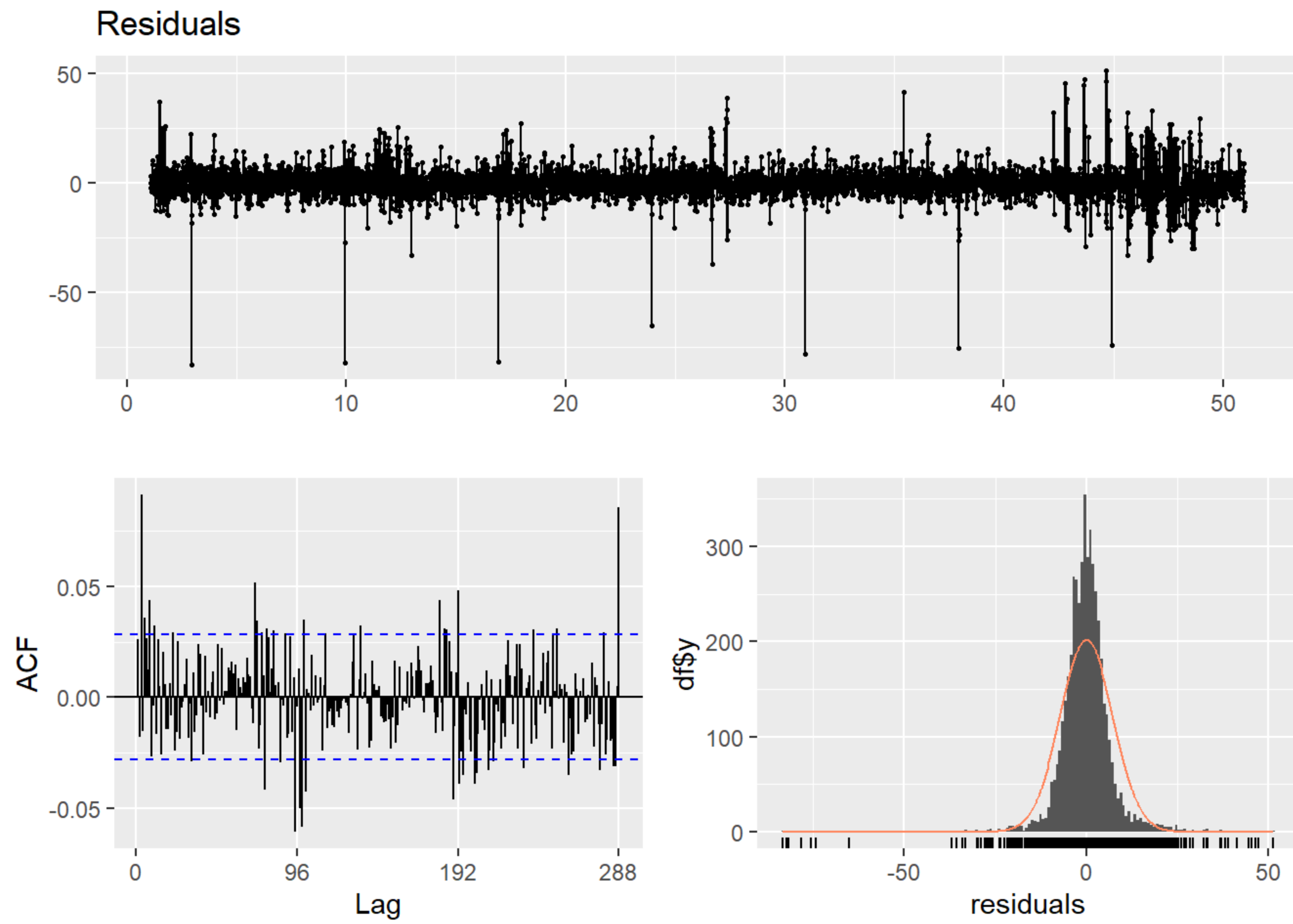
```
ggtsdisplay(e)
```








```
checkresiduals(e, plot = TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 372.09, df = 192, p-value = 1.278e-13
##
## Model df: 0.   Total lags used: 192
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```
## Time difference of 2.599786 mins
```

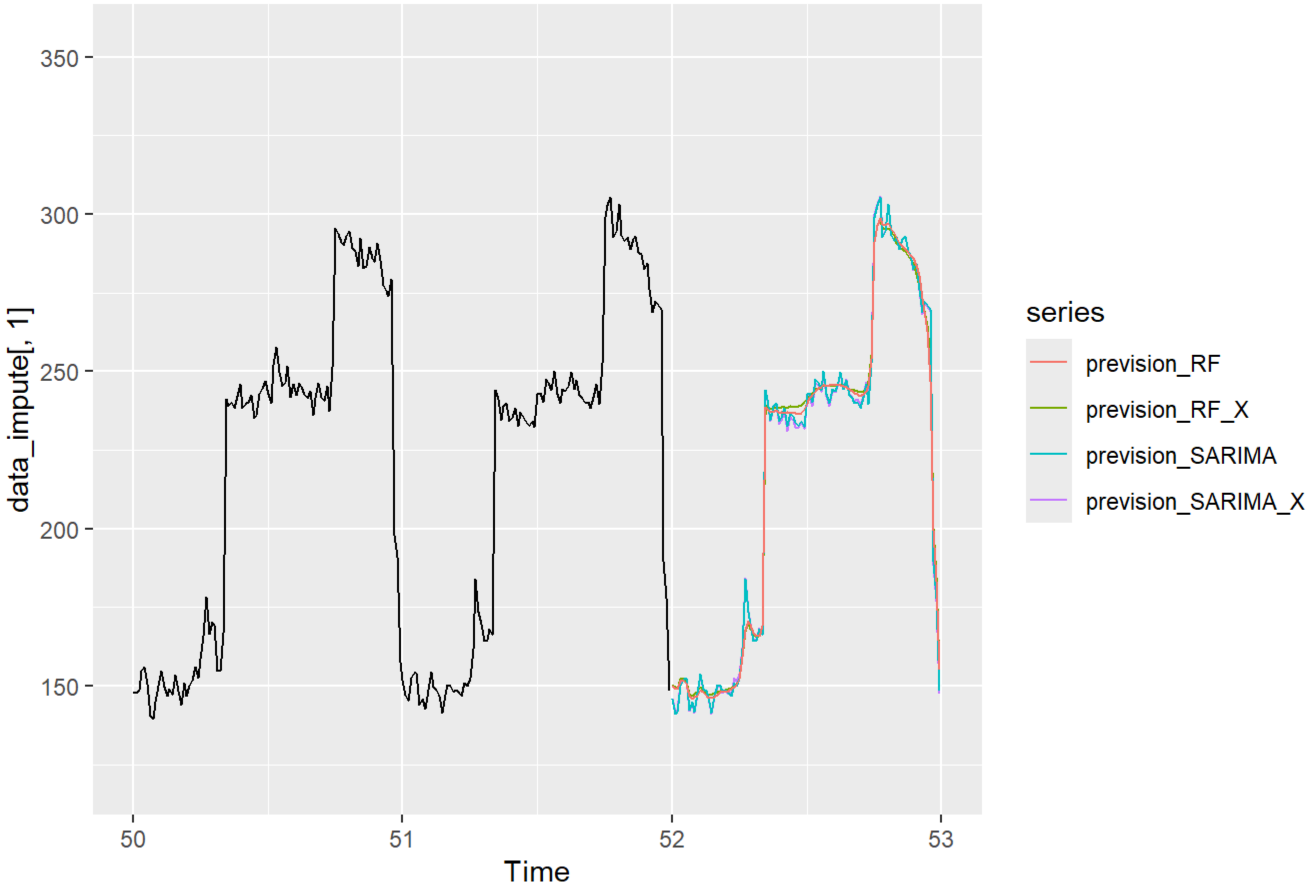
```
saveRDS(fit, file = "Final_model_with_covariate_RF_daily.rds")
```

```
# forecast 96 next values
horizon = 96
freq = 96
newdata_ML = c(as.vector(window(data_impute, start = c(51,1), end = c(51,96))))
xreg = as.vector(data_forecast[,2])

prevision_RF_X = ts(
  forecast_ML_X(readRDS("Final_model_with_covariate_RF_daily.rds"),
              newdata = matrix(newdata_ML,1),
              horizon,
              xreg),
  start = c(52,1),
  frequency = freq)
```

Plots

```
# Plots
autoplot(data_impute[,1]) +
  autolayer(prevision_SARIMA_X) +
  autolayer(prevision_RF_X) +
  autolayer(prevision_SARIMA) +
  autolayer(prevision_RF) +
  xlim(c(50,53))
```

```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```



Save forecast results

```
df_results$Forecast_covariate_SARIMA = prevision_SARIMA_X
# df_results$Forecast_covariate_RF = prevision_RF_X
```

Export forecast results

```
write_xlsx(df_results, "SamdGuizani.xlsx", col_names = FALSE)
```