Time Series Project

Natalia Pludra, Gasper Pust

```r
library(xts)
library(forecast)
library(tseries)
library(astsa)
library(ggplot2)
library(dplyr)
library(knitr)
library(zoo)
library(tsbox)
library(latex2exp)
library(lubridate)
library(scales)
# library(gridExtra)
```

Introduction

Forecasting real-world time series data is a fundamental theme in statistical modeling. This project focuses on analyzing and forecasting website traffic for an academic teaching notes website using robust statistical methods. The objective is to develop accurate models for predicting web traffic, leveraging patterns in the data. The report documents the full analytical process, including data preparation, model building, and validation.

Dataset

This dataset contains five years of daily time series data capturing various traffic metrics for a statistical forecasting teaching notes website (https://regressit.com/statforecasting.com/). The data was collected using StatCounter, a web traffic monitoring tool.

The dataset contains 2 167 rows of data from **September 14, 2014**, to **August 19, 2020** and includes daily counts of:

- **Page Loads:** Total pages accessed on the site.
- **Unique Visitors:** Distinct users visiting the site, identified by IP address.
- **First-Time Visitors:** Users accessing the site for the first time, identified by the absence of prior cookies.
- **Returning Visitors:** Users with prior visits, identified through cookies when accepted.

The data exhibits complex seasonality influenced by both the day of the week and the academic calendar.

The source of the data is Kaggle (https://www.kaggle.com/datasets/bobnau/daily-website-visitors).

```r
df_website <- read.csv("daily-website-visitors.csv")
```

```r
df_website$Page.Loads <- as.numeric(gsub(",", ".", gsub("\\.", "", df_website$Page.Loads)))
df_website$Date <- as.Date(df_website$Date,format = "%m/%d/%Y")

kable(head(df_website), caption="Table1: Sample data")
```

Table 1: Table1: Sample data

Row	Day	Day.Of.Week	Date	Page.Loads	Unique.Visits	First.Time.Visits	Returning.Visits
1	Sunday	1	2014-09-14	2.146	1,582	1,430	152
2	Monday	2	2014-09-15	3.621	2,528	2,297	231
3	Tuesday	3	2014-09-16	3.698	2,630	2,352	278
4	Wednesday	4	2014-09-17	3.667	2,614	2,327	287
5	Thursday	5	2014-09-18	3.316	2,366	2,130	236
6	Friday	6	2014-09-19	2.815	1,863	1,622	241

We decided to focus on Daily Page Loads.

Exploratory Data Analysis

The first step of the project was EDA. Figure 1 shows our time series.

```r
ts_website <- xts(df_website$Page.Loads, df_website$Date)

plot(ts_website, main = "Daily Page Loads", ylab = "Page Loads",lwd=1.2)
```

We divide the data into a training set and a test set. The test set will contain the last 6 months of observations.

```r
# Training set: First 4.5 years, Test set: Last 6 months
cutoff_date <- as.Date("2020-02-19")
train_data <- window(ts_website, end = cutoff_date)
test_data <- window(ts_website, start = cutoff_date + 1)

plot(train_data, main = "Daily Page Loads", ylab = "Page Loads", xlab = "Date",lwd=1.2)
```

The plot of the training set does not indicate the presence of a trend or heteroscedasticity. However, there is evidence of cyclic patterns in the data. We can also notice unusual observations in 2017. The number of page loads was significantly lower than in other years.

Basic statistics and distribution of the Daily Page Loads are presented in Table 2.

```r
summary(df_website$Page.Loads)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.002   3.115   4.106   4.117   5.021   7.984
```

```r
ggplot(df_website, aes(Page.Loads)) +
  geom_histogram(fill="peachpuff3",color="black") +
  labs(x="Page Loads") +
  theme_minimal()
```

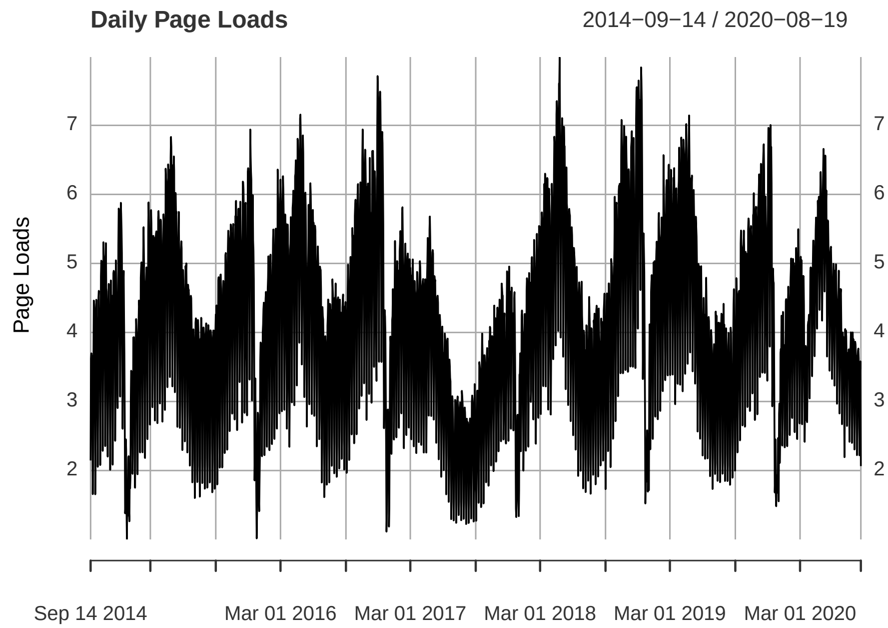


Figure 1: Daily Page Loads

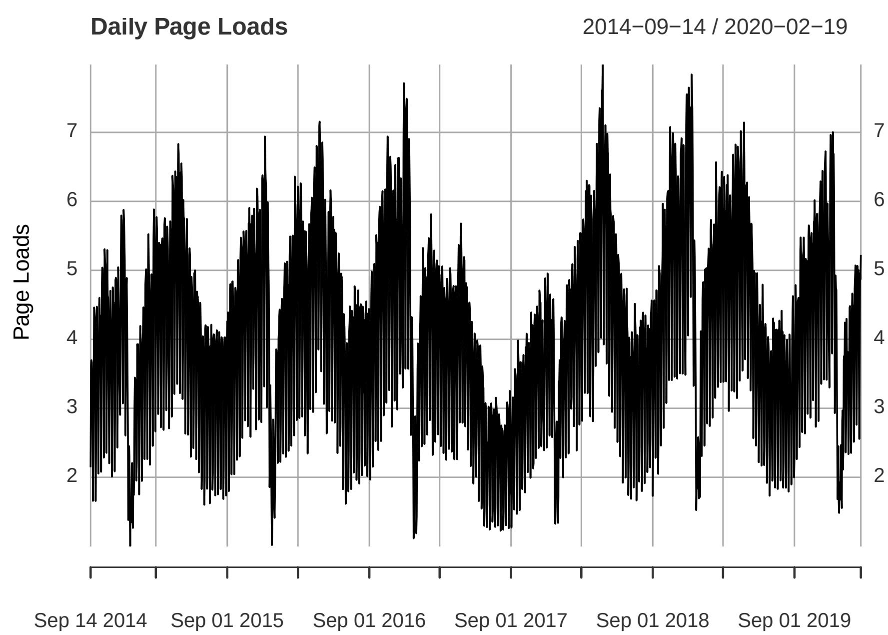


Figure 2: Daily Page Loads - training set

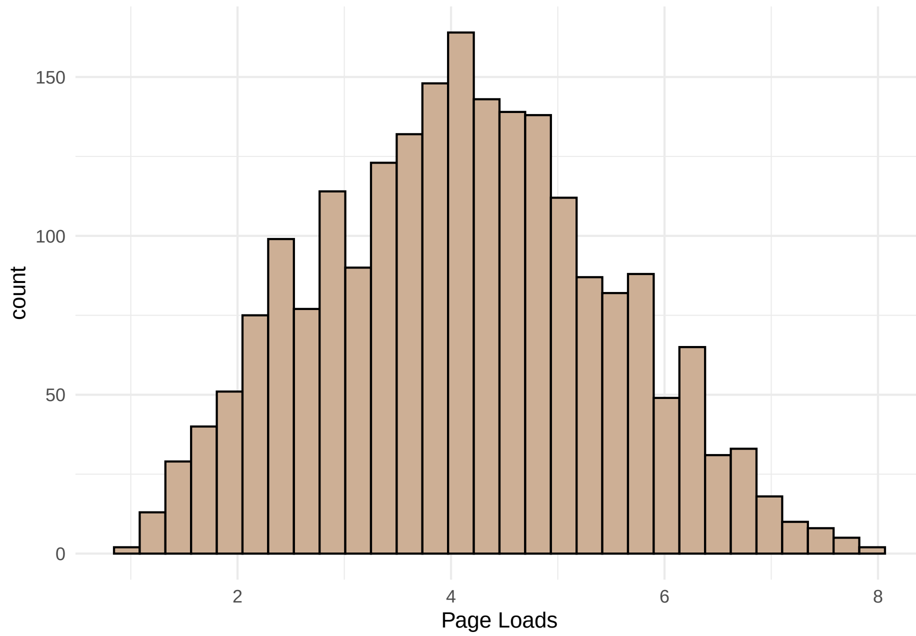


Figure 3: Histogram of Daily Page Loads

There is no missing values in our data.

In the figure below we presented boxplots of Page Loads by day of the week.

```
ggplot(df_website,aes(x=factor(Day,levels=c("Sunday", "Monday", "Tuesday",
      "Wednesday", "Thursday", "Friday", "Saturday")),y=Page.Loads, group=Day)) +
  geom_boxplot(aes(fill=Day)) +
  labs(x="Day of the week", y="Page Loads") +
  theme_minimal() +
  theme(legend.position = "none")
```

We can observe that website traffic is lower during weekends.

We will check if the time series is stationary using Augmented Dickey-Fuller (ADF) test.

```
adf.test(ts_website, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_website
## Dickey-Fuller = -5.4532, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
```

Small p-value indicates that the time series is stationary (assuming significance level of 0.05).

Next, we proceed to compute the sample ACF and PACF for further analysis.

```
invisible(acf2(ts_website,max.lag= 80))
```

The seasonality feature are present in the sample ACF which shows cycles of 7 days - we have weekly seasonality.

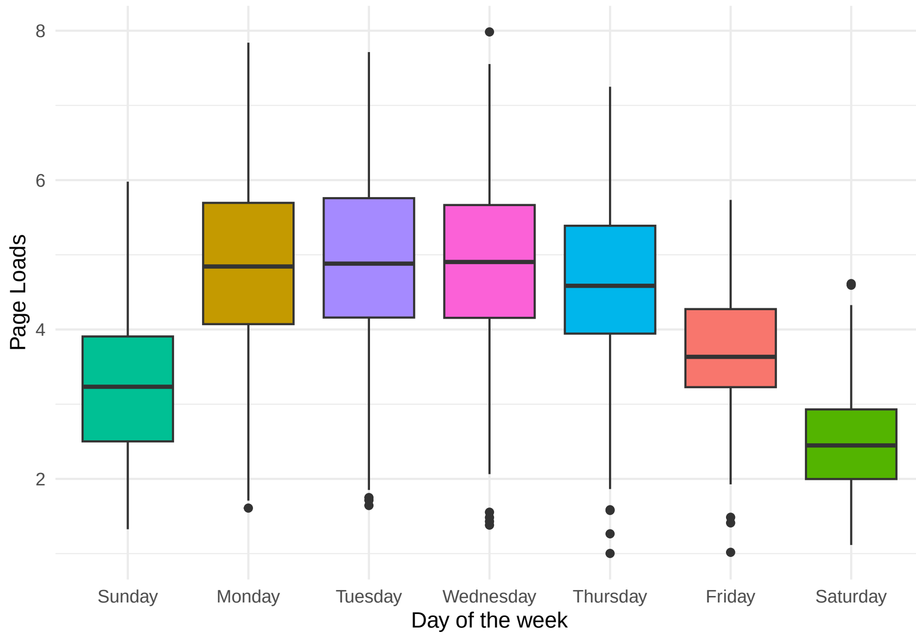


Figure 4: Boxplot of Page Loads by day of the week

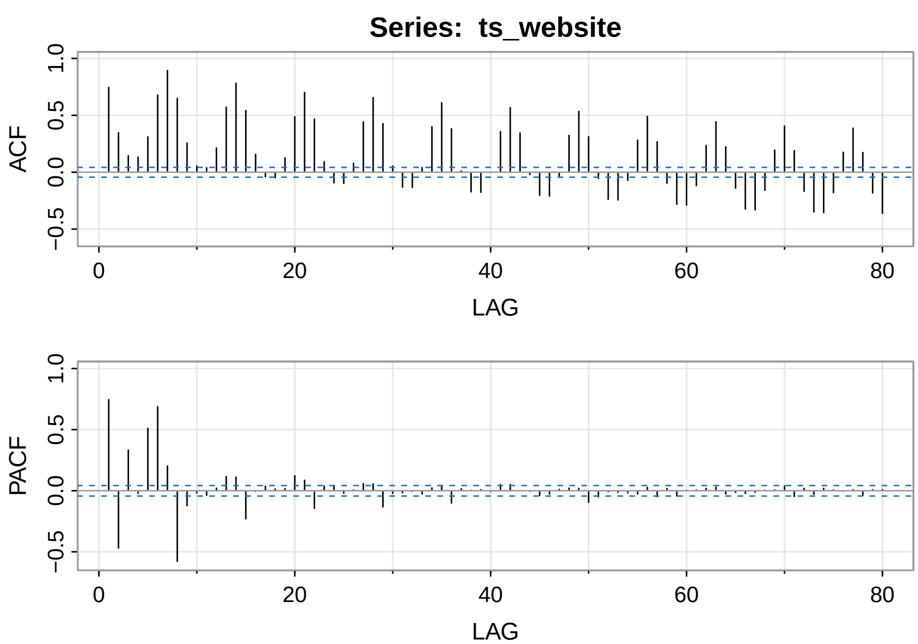


Figure 5: ACF and PACF

```
lag1.plot(ts_website,16)
```

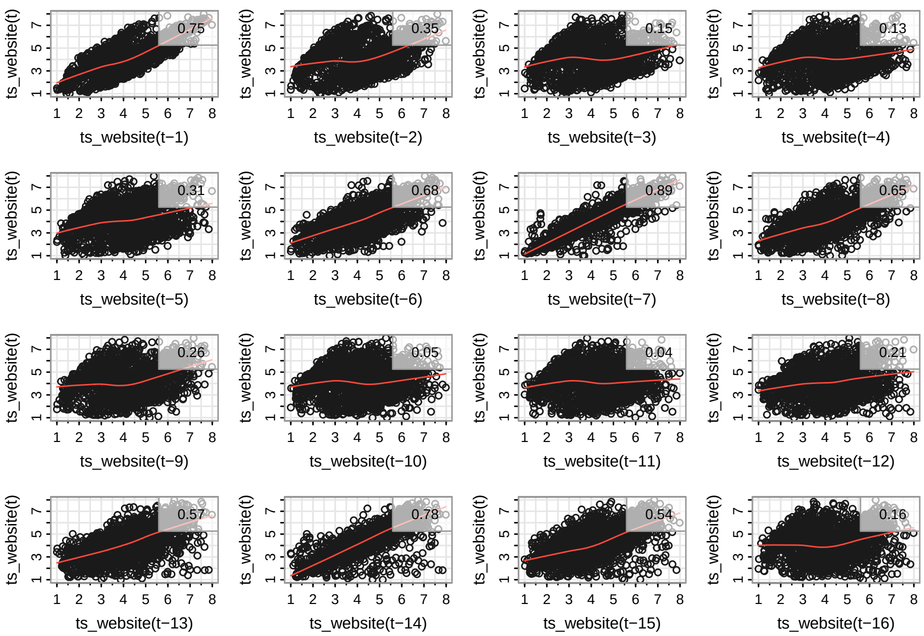


Figure 6: Correlation lag plots

We can see the strongest correlation at lag 1, 7 and 14.

Seasonal Component and Modelling

As we observed in the previous part, there is a weekly seasonal component in our time series. We will try various types of differencing to remove this component.

```
dlm1 <- diff(train_data,1)
plot(dlm1,lwd=1,main=expression(paste(Delta, "Page Loads train")))
```

```
invisible(acf2(dlm1,main=expression(paste(Delta, "Page Loads train"))))
```

```
dlm7 <- diff(train_data,7)
plot(dlm7,lwd=1,,main=expression(paste(Delta[7], "Page Loads train")))
```

```
invisible(acf2(dlm7,main=expression(paste(Delta[7], "Page Loads train"))))
```

```
dlm7.1 <- diff(diff(train_data,7),1)
plot(dlm7.1,lwd=1,main=expression(paste(Delta,Delta[7], "Page Loads train")))
```

```
invisible(acf2(dlm7.1,main=expression(paste(Delta,Delta[7], "Page Loads train"))))
```

Seasonally (with period of 7 days) and regularly differenced data, $\Delta_7 \Delta PageLoadstrain$ seems more stationary. This implies a unit root $d = 1$ as well as a seasonal unit root, $D = 1$. We can see that ACF decays to zero quicker than PACF indicating strong MA component of the model. ACF shows significant correlation at lags 7 and 14, which implies $q = 3$ and $Q = 2$. PACF shows significant correlation at lags 7, 14, 21, 28, 35, 42 and 49, which suggests $p = 7$.

We will try a set of models in order to find the optimal one: M1:$SARIMA(7,1,3) \times (0,1,2)_7$; M2:$SARIMA(6,1,3) \times (0,1,2)_7$; M3:$SARIMA(5,1,3) \times (0,1,2)_7$; M4:$SARIMA(5,1,2) \times (0,1,2)_7$;

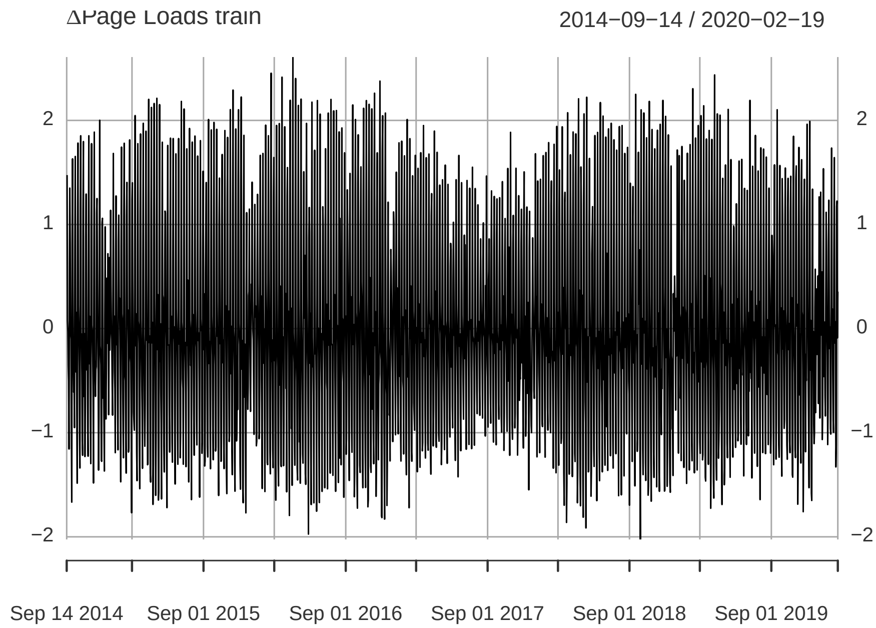


Figure 7: Plots of differenced data

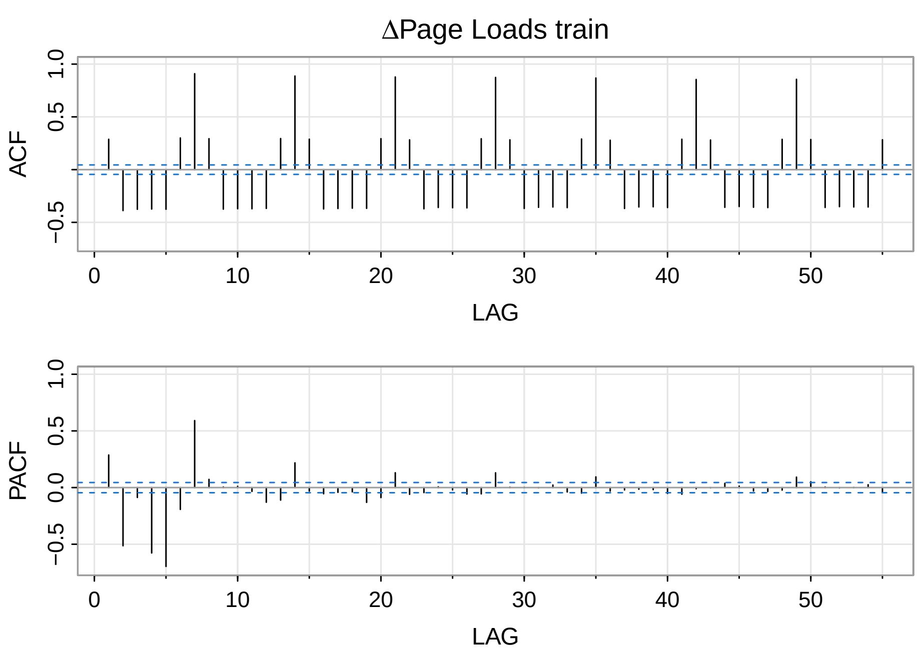


Figure 8: Plots of differenced data

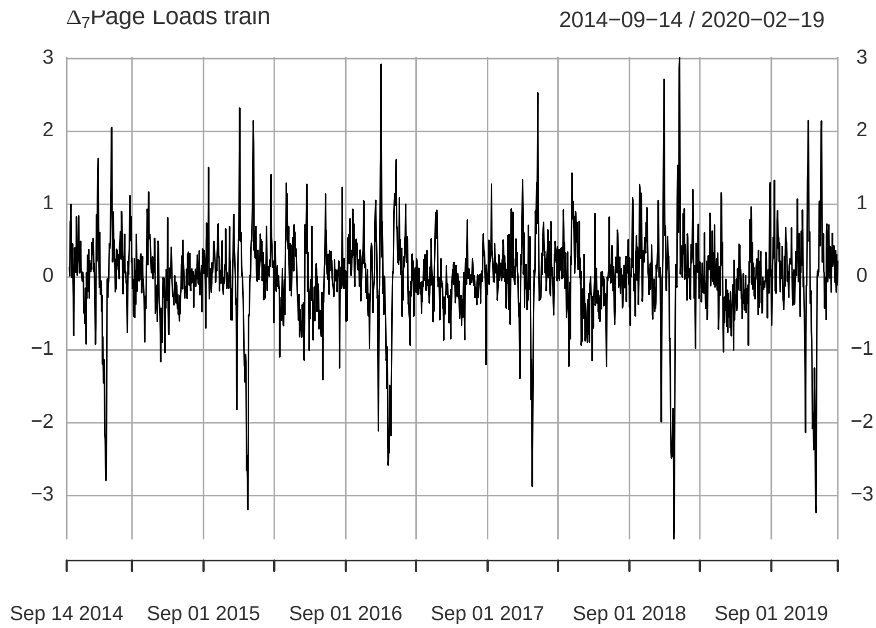


Figure 9: Plots of differenced data

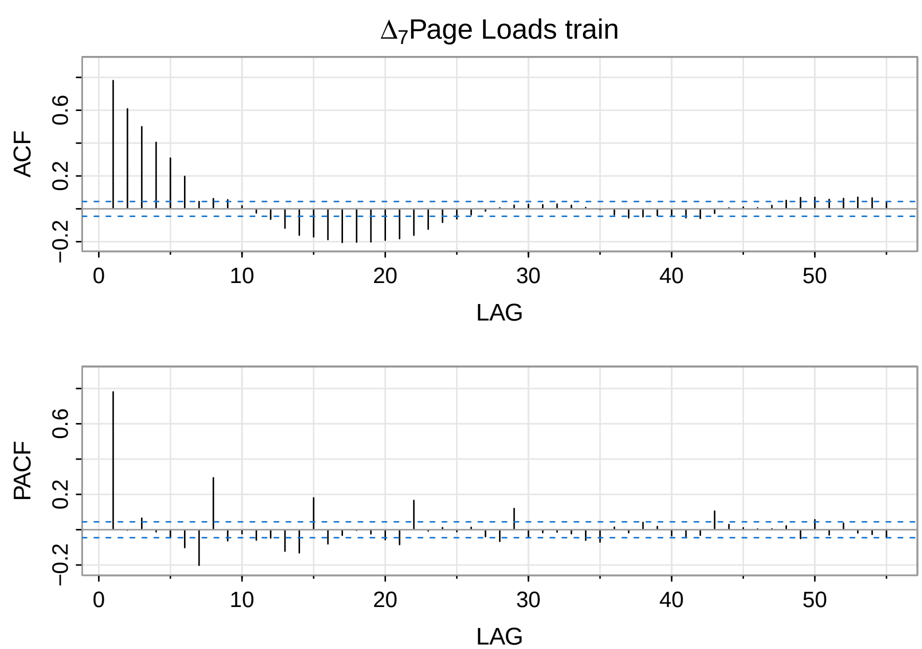


Figure 10: Plots of differenced data

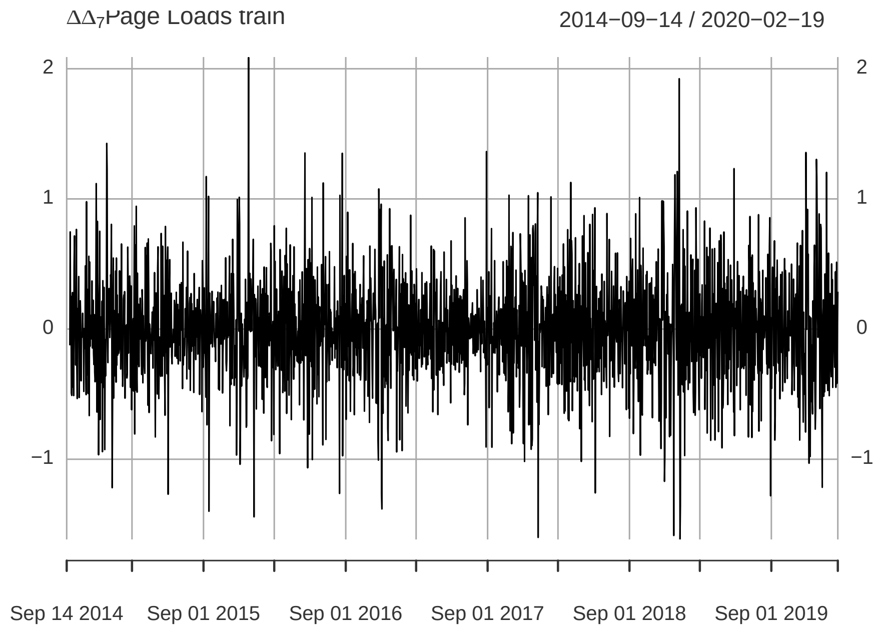


Figure 11: Plots of differenced data

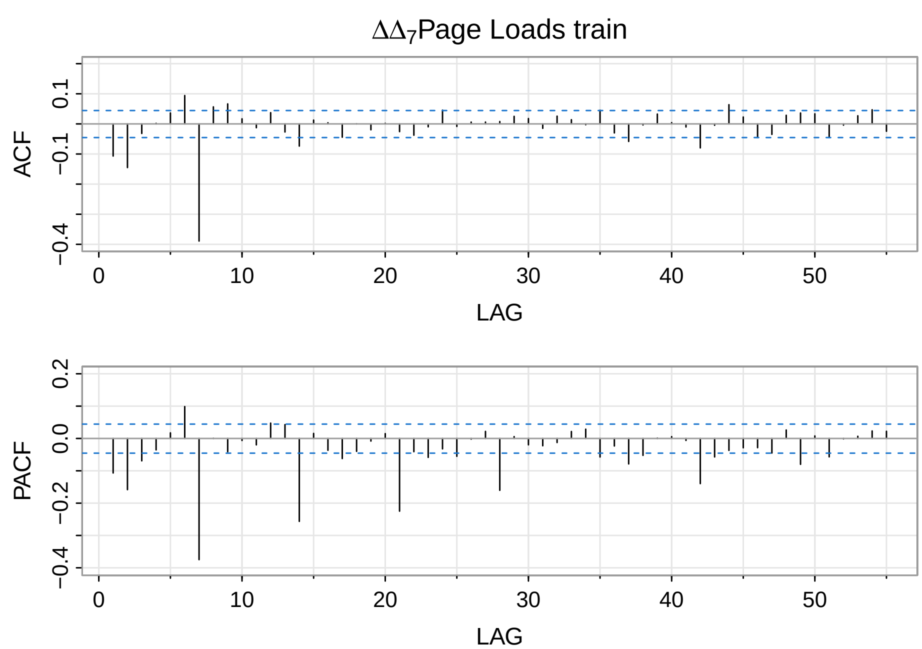
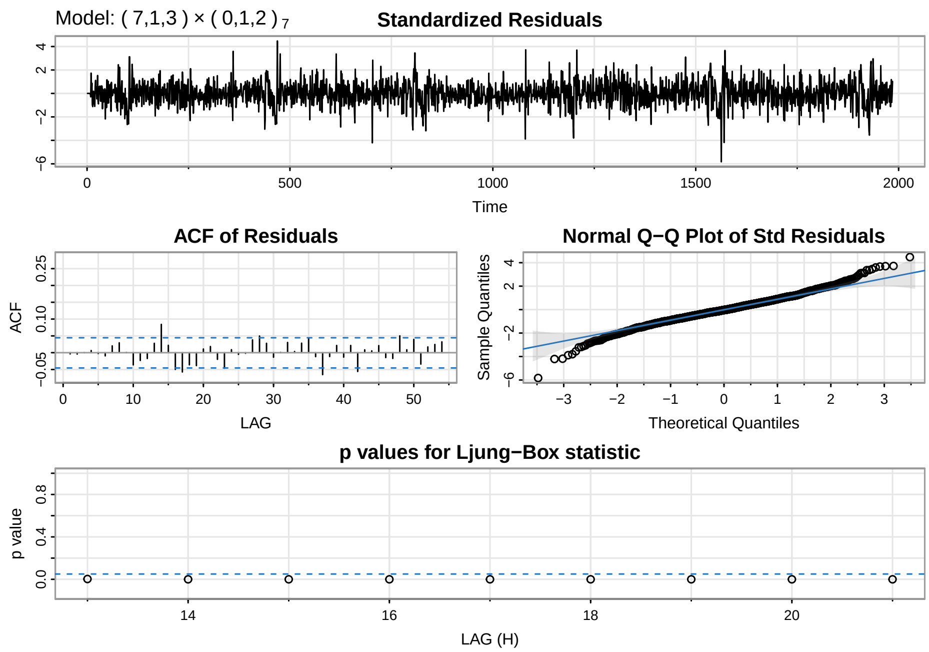


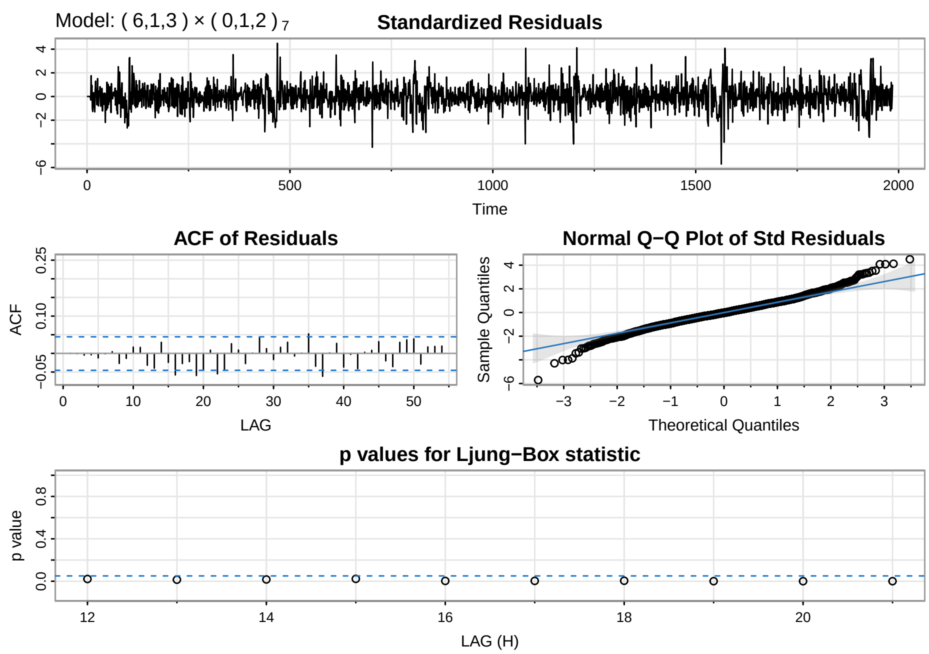
Figure 12: Plots of differenced data

M5:$SARIMA(5,1,3)\times(1,1,2)_7$; M6:$SARIMA(5,1,3)\times(1,1,2)_7$;. For each estimated model check adequacy by analyzing diagnostic on residuals and significance of estimated parameters.

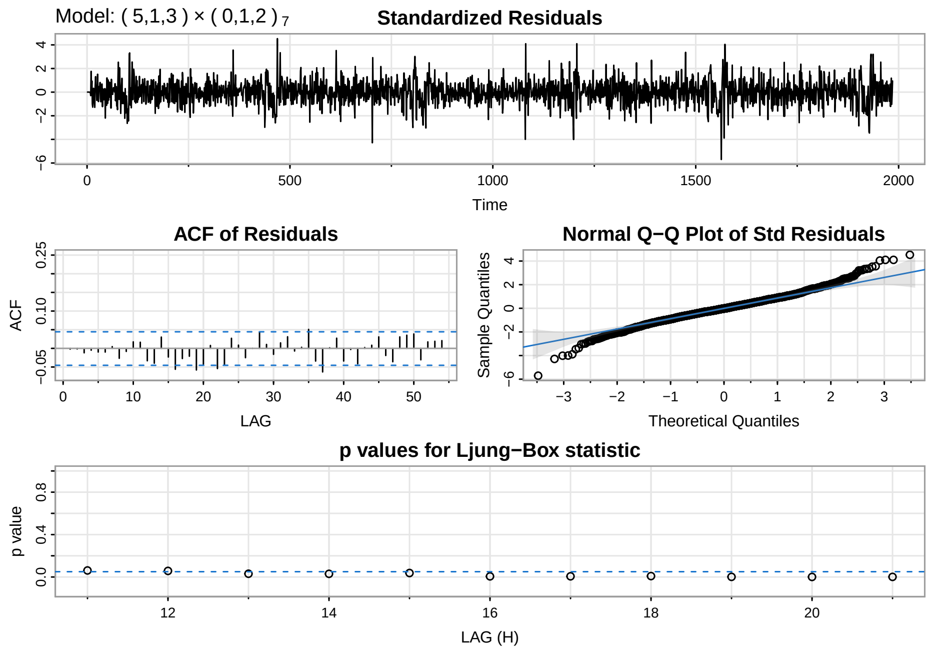
```
visitors.fit1=sarima(train_data,7,1,3,0,1,2,7)
```



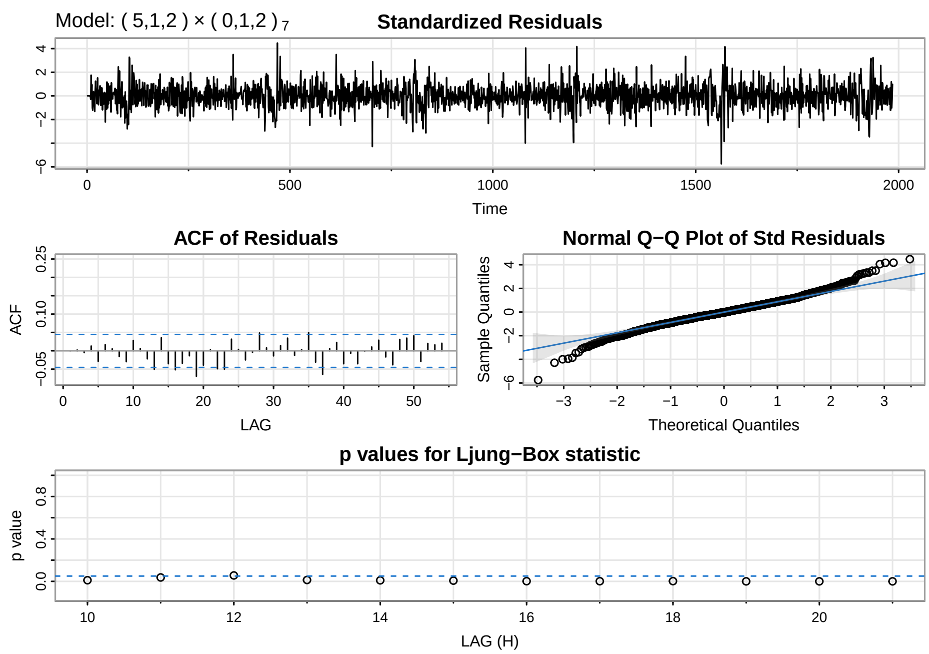

```
visitors.fit2=sarima(train_data,6,1,3,0,1,2,7)
```



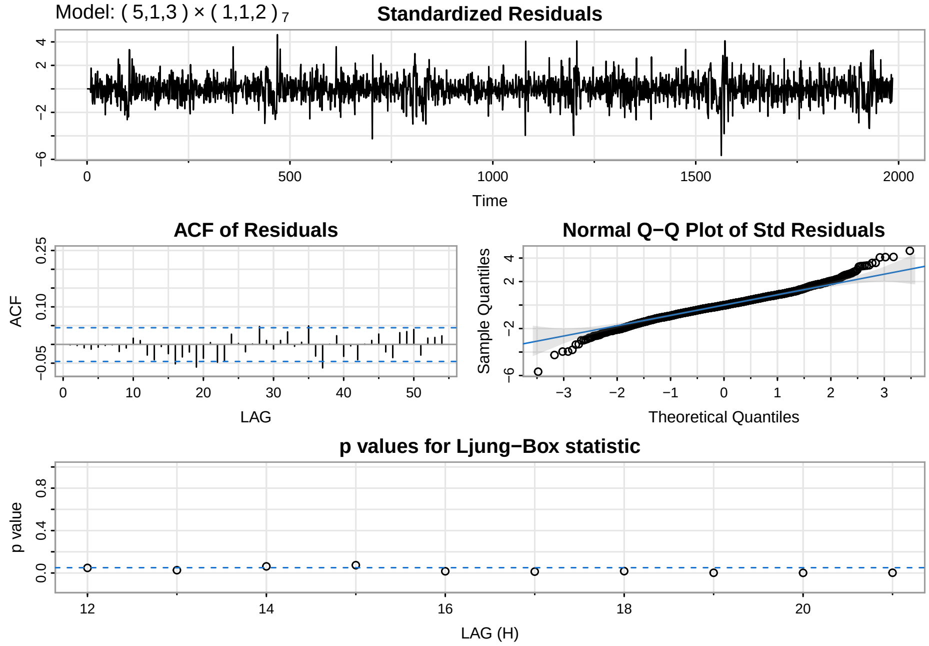

```
visitors.fit3=sarima(train_data,5,1,3,0,1,2,7)
```



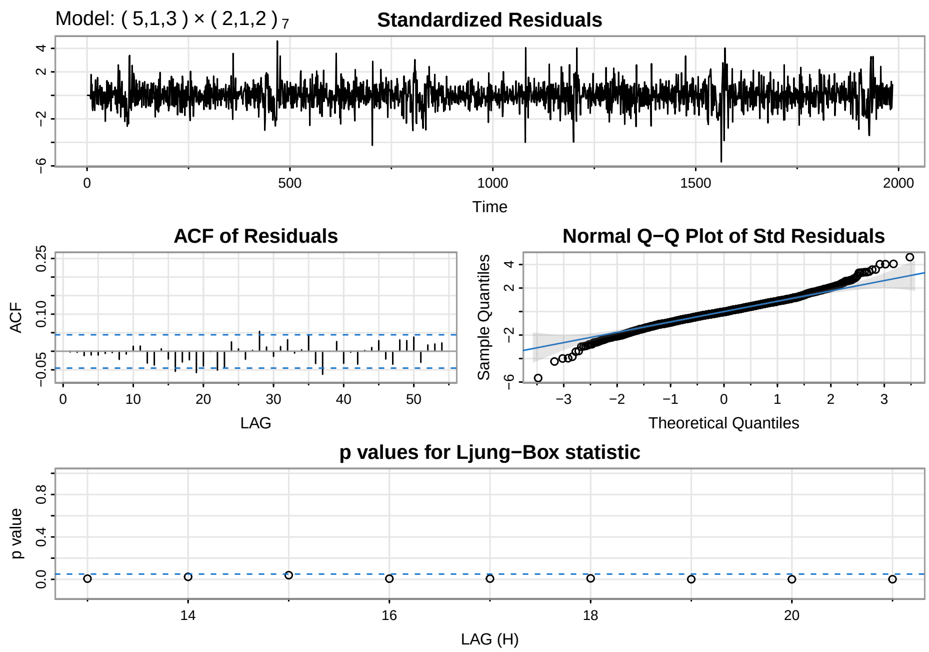

```
visitors.fit4=sarima(train_data,5,1,2,0,1,2,7)
```




```
visitors.fit5=sarima(train_data,5,1,3,1,1,2,7)
```




```
visitors.fit6=sarima(train_data,5,1,3,2,1,2,7)
```




```
visitors.fit1
```

```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1      ar2       ar3      ar4       ar5      ar6       ar7       ma1
```

```
##       0.9267  0.3807  -0.4046  0.0816  -0.0140  0.1366  -0.1354  -1.0275
## s.e.  0.1975  0.2032   0.1273  0.0376   0.0339  0.0336   0.0311   0.1978
##           ma2     ma3     sma1     sma2
##       -0.5149  0.5424  -0.6723  -0.2492
## s.e.   0.2191  0.1445   0.0262   0.0265
##
## sigma^2 estimated as 0.1071:  log likelihood = -606.46,  aic = 1238.92
##
## $degrees_of_freedom
## [1] 1965
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1    0.9267 0.1975   4.6916  0.0000
## ar2    0.3807 0.2032   1.8739  0.0611
## ar3   -0.4046 0.1273  -3.1787  0.0015
## ar4    0.0816 0.0376   2.1719  0.0300
## ar5   -0.0140 0.0339  -0.4125  0.6800
## ar6    0.1366 0.0336   4.0652  0.0000
## ar7   -0.1354 0.0311  -4.3597  0.0000
## ma1   -1.0275 0.1978  -5.1950  0.0000
## ma2   -0.5149 0.2191  -2.3499  0.0189
## ma3    0.5424 0.1445   3.7536  0.0002
## sma1  -0.6723 0.0262 -25.6533  0.0000
## sma2  -0.2492 0.0265  -9.4006  0.0000
##
## $ICs
##       AIC      AICc       BIC
## 0.6266681 0.6267484 0.6634214
```

```
visitors.fit2
```

```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ar2      ar3      ar4      ar5      ar6      ma1      ma2
##       0.0419  -0.0152  -0.8103  -0.1125  -0.2149  -0.0200  -0.1405  -0.1956
## s.e.  0.1533   0.1269   0.1323   0.0304   0.0293   0.0395   0.1502   0.1411
##          ma3     sma1     sma2
##       0.7446  -0.8051  -0.1913
## s.e.  0.1141   0.0272   0.0254
##
## sigma^2 estimated as 0.1055:  log likelihood = -596.56,  aic = 1217.11
##
## $degrees_of_freedom
## [1] 1966
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1    0.0419 0.1533   0.2735  0.7845
```

```
## ar2   -0.0152 0.1269  -0.1201  0.9044
## ar3   -0.8103 0.1323  -6.1239  0.0000
## ar4   -0.1125 0.0304  -3.6959  0.0002
## ar5   -0.2149 0.0293  -7.3417  0.0000
## ar6   -0.0200 0.0395  -0.5069  0.6123
## ma1   -0.1405 0.1502  -0.9354  0.3497
## ma2   -0.1956 0.1411  -1.3861  0.1659
## ma3    0.7446 0.1141   6.5262  0.0000
## sma1  -0.8051 0.0272 -29.6060  0.0000
## sma2  -0.1913 0.0254  -7.5422  0.0000
##
## $ICs
##       AIC      AICc       BIC
## 0.6156360 0.6157039 0.6495621
```

```
visitors.fit3
```

```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ar2      ar3      ar4      ar5      ma1      ma2     ma3
##       0.0780  -0.0251  -0.7731  -0.1076  -0.2085  -0.1743  -0.1812  0.7208
## s.e.  0.0858   0.0913   0.0688   0.0268   0.0262   0.0850   0.0970  0.0718
##          sma1     sma2
##       -0.8043  -0.1920
## s.e.   0.0264   0.0247
##
## sigma^2 estimated as 0.1055:  log likelihood = -596.71,  aic = 1215.43
##
## $degrees_of_freedom
## [1] 1967
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1    0.0780 0.0858   0.9089  0.3635
## ar2   -0.0251 0.0913  -0.2754  0.7830
## ar3   -0.7731 0.0688 -11.2352  0.0000
## ar4   -0.1076 0.0268  -4.0196  0.0001
## ar5   -0.2085 0.0262  -7.9558  0.0000
## ma1   -0.1743 0.0850  -2.0500  0.0405
## ma2   -0.1812 0.0970  -1.8686  0.0618
## ma3    0.7208 0.0718  10.0425  0.0000
## sma1  -0.8043 0.0264 -30.4570  0.0000
## sma2  -0.1920 0.0247  -7.7598  0.0000
##
## $ICs
##       AIC      AICc       BIC
## 0.6147832 0.6148398 0.6458822
```

```
visitors.fit4
```

```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ar2     ar3      ar4      ar5      ma1     ma2     sma1
##       0.9624  -0.9212  0.0419  -0.1717  -0.0439  -1.0637  0.8022  -0.8195
## s.e.  0.0471   0.0412  0.0386   0.0325   0.0286   0.0414  0.0325   0.0257
##          sma2
##       -0.1768
## s.e.   0.0239
##
## sigma^2 estimated as 0.1057:  log likelihood = -598.69,  aic = 1217.38
##
## $degrees_of_freedom
## [1] 1968
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1    0.9624 0.0471  20.4132  0.0000
## ar2   -0.9212 0.0412 -22.3629  0.0000
## ar3    0.0419 0.0386   1.0842  0.2784
## ar4   -0.1717 0.0325  -5.2885  0.0000
## ar5   -0.0439 0.0286  -1.5352  0.1249
## ma1   -1.0637 0.0414 -25.7207  0.0000
## ma2    0.8022 0.0325  24.6565  0.0000
## sma1  -0.8195 0.0257 -31.9097  0.0000
## sma2  -0.1768 0.0239  -7.4098  0.0000
##
## $ICs
##       AIC      AICc       BIC
## 0.6157722 0.6158184 0.6440440
```

```
visitors.fit5
```

```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ar2      ar3      ar4      ar5      ma1      ma2     ma3
##       0.1480  -0.0895  -0.7142  -0.1013  -0.1946  -0.2423  -0.1052  0.6671
## s.e.  0.1441   0.1540   0.1306   0.0273   0.0311   0.1447   0.1667  0.1264
##         sar1     sma1    sma2
##       0.2179  -1.0158  0.0160
## s.e.  0.1390   0.1437  0.1424
##
```

```
## sigma^2 estimated as 0.1051:  log likelihood = -595.37,  aic = 1214.75
##
## $degrees_of_freedom
## [1] 1966
##
## $ttable
##      Estimate     SE t.value p.value
## ar1    0.1480 0.1441  1.0270  0.3046
## ar2   -0.0895 0.1540 -0.5814  0.5610
## ar3   -0.7142 0.1306 -5.4705  0.0000
## ar4   -0.1013 0.0273 -3.7102  0.0002
## ar5   -0.1946 0.0311 -6.2540  0.0000
## ma1   -0.2423 0.1447 -1.6740  0.0943
## ma2   -0.1052 0.1667 -0.6311  0.5281
## ma3    0.6671 0.1264  5.2778  0.0000
## sar1   0.2179 0.1390  1.5679  0.1171
## sma1  -1.0158 0.1437 -7.0695  0.0000
## sma2   0.0160 0.1424  0.1123  0.9106
##
## $ICs
##       AIC      AICc       BIC
## 0.6144408 0.6145088 0.6483670
```

```
visitors.fit6
```

```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ar2      ar3      ar4      ar5      ma1      ma2     ma3
##       0.0876  -0.0235  -0.7700  -0.1002  -0.2017  -0.1816  -0.1768  0.7191
## s.e.  0.1043   0.1113   0.0867   0.0272   0.0267   0.1043   0.1193  0.0887
##          sar1    sar2     sma1     sma2
##       -0.6917  0.1716  -0.1013  -0.8987
## s.e.   0.0882  0.0335   0.0862   0.0857
##
## sigma^2 estimated as 0.105:  log likelihood = -594.81,  aic = 1215.61
##
## $degrees_of_freedom
## [1] 1965
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1    0.0876 0.1043   0.8396  0.4013
## ar2   -0.0235 0.1113  -0.2108  0.8331
## ar3   -0.7700 0.0867  -8.8810  0.0000
## ar4   -0.1002 0.0272  -3.6812  0.0002
## ar5   -0.2017 0.0267  -7.5674  0.0000
## ma1   -0.1816 0.1043  -1.7413  0.0818
## ma2   -0.1768 0.1193  -1.4818  0.1386
## ma3    0.7191 0.0887   8.1056  0.0000
```

```
## sar1  -0.6917 0.0882  -7.8422  0.0000
## sar2   0.1716 0.0335   5.1244  0.0000
## sma1  -0.1013 0.0862  -1.1751  0.2401
## sma2  -0.8987 0.0857 -10.4841  0.0000
##
## $ICs
##       AIC      AICc       BIC
## 0.6148763 0.6149566 0.6516296
```

By comparing the AIC values of the selected models we can see that the model that provides the best fit is Model 5 $SARIMA(5,1,3) \times (1,1,2)_7$. We will do the forecasting with that model.

Forecasting

Forecasting 60-days ahead

To forecast 60 steps ahead, we will use unique training data, which implies using a unique model fit.

Forecasting 60 days with 1-step ahead - Interaction

This forecasting strategy assumes we know the new data at the moment of the forecast, and we will not fit a new model based on new data. The forecast generation uses the fitted model to interact with new data to produce a forecast.

Forecasting 60 days with 1-step ahead - Expanding windows

In the expanding window strategy, the training sample size increases sequentially by a time unit. The forecasts produced are based on multiple fit models.

Forecasting 60 days with 1-step ahead - Recursive windows

In the recursive window strategy, the training sample maintains its size across the forecasting generation. However, with distinct sets of observation. The training sample window slides a time unit to produce a forecast. The forecasts produced are based on multiple fit models.

Plotting of all forecasts

```r
ts_df <- data.frame(time = index(ts_website), value = coredata(ts_website))

# Forecasting 60-days ahead (unique model fit)
M5.fit_plots <- Arima(train_data, order=c(5,1,3), seasonal=list(order=c(1,1,2), period=7))

# Approach 1: Forecasting 60-days ahead
forecast_1 <- forecast(M5.fit_plots, h=60, level=95)
forecast_dates <- seq.Date(from = cutoff_date + days(1), by = "day", length.out = 60)
forecast_df_1 <- data.frame(Date = forecast_dates, Forecast = as.numeric(forecast_1$mean))

# Approach 2: Forecasting 60 days with 1-step ahead - Interaction
ts_df <- data.frame(time = index(ts_website), value = coredata(ts_website))
output_2 <- data.frame()

for (i in 0:59) {
  df_select <- ts_df %>% filter(time <= cutoff_date + days(i))
  ts_select <- ts(df_select$value, freq = 365, end = c(year(tail(df_select$time, 1)),
    yday(tail(df_select$time, 1))))
  fit <- Arima(ts_select, model = M5.fit_plots)
  aux <- bind_cols(time = cutoff_date + days(i + 1), value = tail(fitted(fit), 1))
  output_2 <- bind_rows(output_2, aux)
```

```r
}

forecast_ts_2 <- xts(output_2$value, order.by = output_2$time)
forecast_df_2 <- data.frame(Date = index(forecast_ts_2),
                            Forecast = coredata(forecast_ts_2))

# Approach 3: Forecasting 60 days with 1-step ahead - Expanding windows
output_3 <- data.frame()
for (i in 0:59) {
  df_select <- ts_df %>% filter(time <= cutoff_date + days(i))
  ts_select <- ts(df_select$value, freq = 365, end = c(year(tail(df_select$time, 1)),
    yday(tail(df_select$time, 1))))
  fit <- Arima(ts_select, order = c(5, 1, 3), seasonal = list(order = c(1, 1, 2),
    period = 7))
  forecasted <- forecast(fit, h = 1, level = 95)
  aux <- data.frame(time = cutoff_date + days(i + 1), value = as.numeric(forecasted$mean))
  output_3 <- bind_rows(output_3, aux)
}

forecast_ts_3 <- xts(output_3$value, order.by = output_3$time)
forecast_df_3 <- data.frame(Date = index(forecast_ts_3), Forecast = coredata(forecast_ts_3))

# Approach 4: Forecasting 60 days with 1-step ahead - Recursive windows
output_4 <- data.frame()

for (i in 0:59) {
  start_date <- cutoff_date - months(3) + days(i)
  end_date <- cutoff_date + days(i)
  df_select <- ts_df %>% filter(time >= start_date & time <= end_date)
  ts_select <- ts(df_select$value, frequency = 365, start = c(year(start_date),
    yday(start_date)))
  forecasted <- tryCatch({
    fit <- Arima(ts_select, order = c(5, 1, 3), seasonal = list(order = c(1, 1, 2),
      period = 7))
    forecast(fit, h = 1, level = 95)
  }, error = function(e) {
    NULL # Return NULL if fitting fails
  })
  if (!is.null(forecasted)) {
    aux <- data.frame(time = end_date + days(1), value = as.numeric(forecasted$mean))
    output_4 <- bind_rows(output_4, aux)
  } else {
    aux <- data.frame(time = end_date + days(1), value = NA) # NA for failed forecasts
    output_4 <- bind_rows(output_4, aux)
  }
}

forecast_ts_4 <- xts(output_4$value, order.by = output_4$time)
forecast_df_4 <- data.frame(Date = index(forecast_ts_4),
                            Forecast = coredata(forecast_ts_4))

# Combine observed, forecasts, and test data for plotting
test_df <- data.frame(Date = index(test_data), Observed = coredata(test_data))
```

```r
# Plot for just the test data and forecasts (cutoff_date to last forecast)
last_forecast_date <- max(forecast_df_4$Date)

ts_df_filtered <- ts_df %>% filter(time <= last_forecast_date)
test_df_filtered <- test_df %>% filter(Date <= last_forecast_date)

# Plotting forecasts
ggplot() +
  geom_line(data = ts_df_filtered, aes(x = time, y = value, color = "Observed"),
            size = 1, alpha = 0.8) +
  geom_line(data = forecast_df_1, aes(x = Date, y = Forecast,
            color = "60-step ahead forecast"), size = 1) +
  geom_line(data = forecast_df_2, aes(x = Date, y = Forecast,
            color = "1-step ahead forecast (Interaction)"), size = 1) +
  geom_line(data = forecast_df_3, aes(x = Date, y = Forecast,
            color = "1-step ahead forecast (Expanding windows)"), size = 1) +
  geom_line(data = forecast_df_4, aes(x = Date, y = Forecast,
            color = "1-step ahead forecast (Recursive windows)"), size = 1) +
  labs(title = "Forecasts from SARIMA Model for Daily Website Visitors",
       x = "Date", y = "Page Loads") +
  scale_color_manual(values = c(
    "Observed" = "black",
    "60-step ahead forecast" = "darkred",
    "1-step ahead forecast (Interaction)" = "gold",
    "1-step ahead forecast (Expanding windows)" = "purple",
    "1-step ahead forecast (Recursive windows)" = "blue")) +
  theme_light() + theme(legend.title = element_blank(),
    legend.background = element_blank(), legend.position = "bottom",
    legend.direction = "vertical")
```

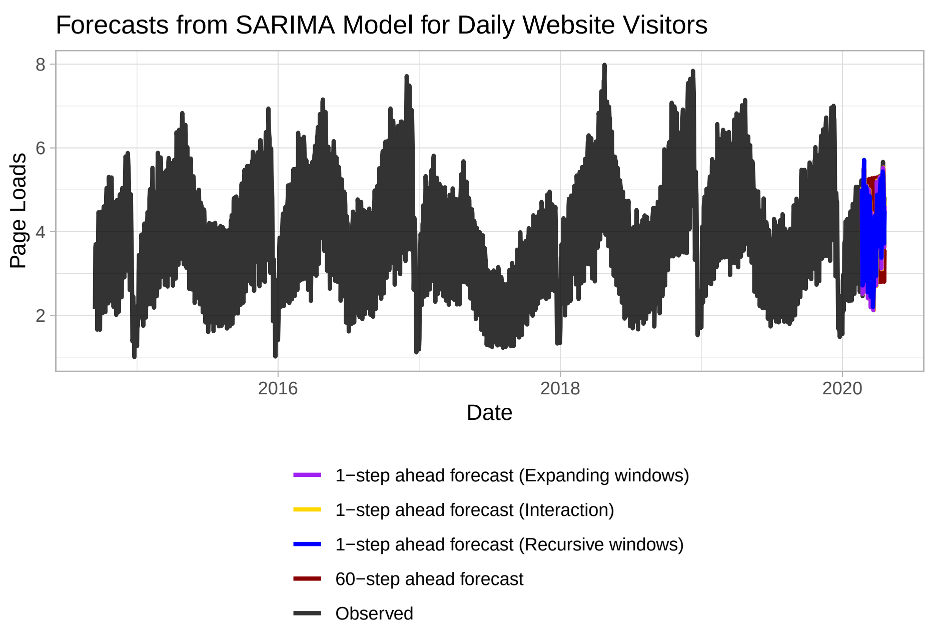


Figure 13: Plot of different forecast methods

```r
ggplot() +
  geom_line(data = test_df_filtered, aes(x = Date, y = Observed, color = "Observed"),
            size = 1, linetype = "solid") +
  geom_line(data = forecast_df_1, aes(x = Date, y = Forecast,
            color = "60-step ahead forecast"),
            size = 1, linetype = "solid") +
  geom_line(data = forecast_df_2, aes(x = Date, y = Forecast,
            color = "1-step ahead forecast (Interaction)"),
            size = 1, linetype = "solid") +
  geom_line(data = forecast_df_3, aes(x = Date, y = Forecast,
            color = "1-step ahead forecast (Expanding windows)"),
            size = 1, linetype = "solid") +
  geom_line(data = forecast_df_4, aes(x = Date, y = Forecast,
            color = "1-step ahead forecast (Recursive windows)"),
            size = 1, linetype = "solid") +
  labs(title = "Forecasting Approaches from Cutoff Date to Last Forecast",
       x = "Date", y = "Page Loads") +
    scale_color_manual(values = c(
    "Observed" = "black",
    "60-step ahead forecast" = "darkred",
    "1-step ahead forecast (Interaction)" = "gold",
    "1-step ahead forecast (Expanding windows)" = "purple",
    "1-step ahead forecast (Recursive windows)" = "blue")) +
  theme_light() + theme(legend.position = "bottom",
    legend.direction = "vertical", legend.title = element_blank())
```

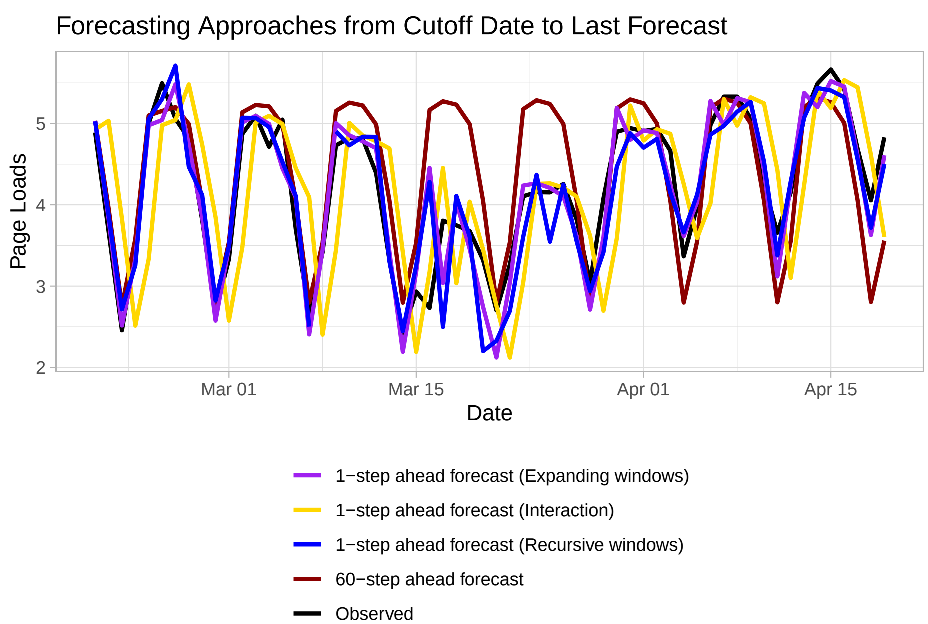


Figure 14: Plot of different forecast methods

Choosing the best approach

```r
errors_1 <- test_df$Observed - forecast_df_1$Forecast
errors_2 <- test_df$Observed - forecast_df_2$Forecast
```

```r
errors_3 <- test_df$Observed - forecast_df_3$Forecast
errors_4 <- test_df$Observed - forecast_df_4$Forecast

error_df <- data.frame(
  Date = test_df$Date,
  '60-step ahead forecast' = errors_1,
  '1-step ahead forecast (Interaction)' = errors_2,
  '1-step ahead forecast (Expanding windows)' = errors_3,
  '1-step ahead forecast (Recursive windows)' = errors_4)

error_df$Date <- as.Date(error_df$Date)

colnames(error_df) <- c(
  'Date',
  '60_step_ahead_forecast',
  '1_step_ahead_forecast_Interaction',
  '1_step_ahead_forecast_Expanding_windows',
  '1_step_ahead_forecast_Recursive_windows')

ggplot(error_df, aes(x = Date)) +
  geom_line(aes(y = `60_step_ahead_forecast`,
           color = "60-step ahead forecast"), size = 1) +
  geom_line(aes(y = `1_step_ahead_forecast_Interaction`,
           color = "1-step ahead forecast (Interaction)"), size = 1) +
  geom_line(aes(y = `1_step_ahead_forecast_Expanding_windows`,
           color = "1-step ahead forecast (Expanding windows)"), size = 1) +
  geom_line(aes(y = `1_step_ahead_forecast_Recursive_windows`,
           color = "1-step ahead forecast (Recursive windows)"), size = 1) +
  labs(title = "Forecasting Errors for Different Approaches",
       x = "Date", y = "Forecast Error") +
    scale_color_manual(values = c(
    "60-step ahead forecast" = "darkred",
    "1-step ahead forecast (Interaction)" = "gold",
    "1-step ahead forecast (Expanding windows)" = "purple",
    "1-step ahead forecast (Recursive windows)" = "blue")) +
  theme_light() + theme(legend.title = element_blank(),
      legend.background = element_blank(), legend.position = "bottom",
      legend.direction = "vertical") +
  geom_hline(yintercept = 0, color = "black", size = 1)
```

```r
test_df_filtered <- head(test_df, 60)

accuracy <- rbind(
  accuracy(forecast_df_1$Forecast, test_df_filtered$Observed),
  accuracy(forecast_df_2$Forecast, test_df_filtered$Observed),
  accuracy(forecast_df_3$Forecast, test_df_filtered$Observed),
  accuracy(forecast_df_4$Forecast, test_df_filtered$Observed))

rownames(accuracy) <- c(
  '60-step ahead forecast',
  '1-step ahead forecast (Interaction)',
  '1-step ahead forecast (Expanding windows)',
  '1-step ahead forecast (Recursive windows)')
```

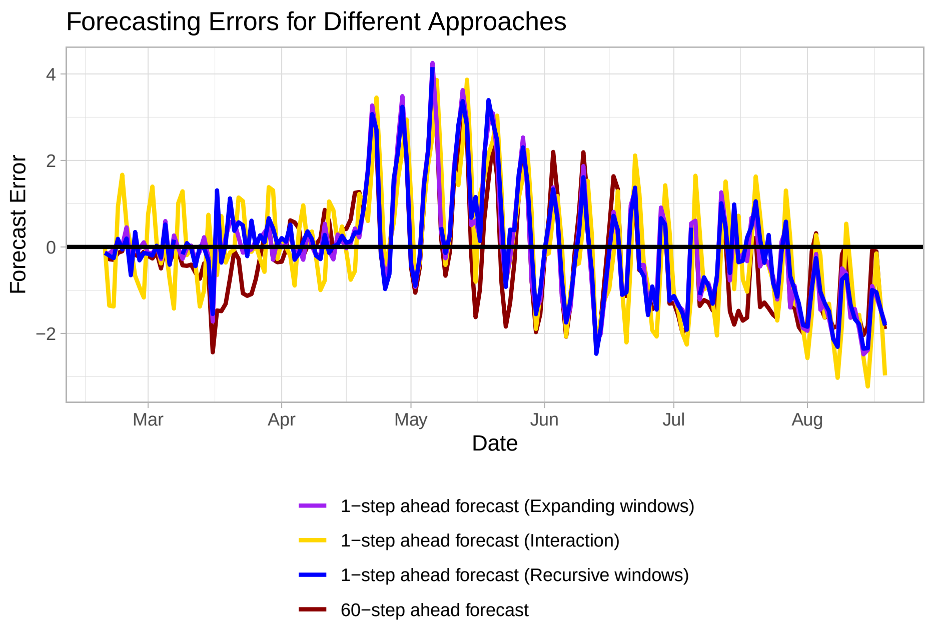


Figure 15: Plot of forecast errors

```r
knitr::kable(accuracy, digits = 4, align = "ccccccc",
  caption = "Accuracy measures for different forecasting approaches")
```

Table 2: Accuracy measures for different forecasting approaches

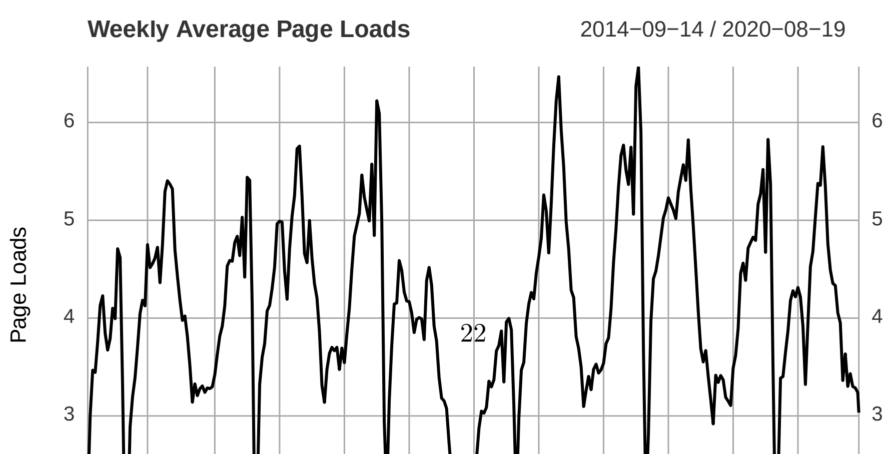
	ME	RMSE	MAE	MPE	MAPE
60-step ahead forecast	-0.1928	0.6643	0.4851	-5.8354	12.6326
1-step ahead forecast (Interaction)	0.0172	0.7950	0.6380	-1.4756	16.9379
1-step ahead forecast (Expanding windows)	0.0224	0.3572	0.2477	0.4678	6.5906
1-step ahead forecast (Recursive windows)	0.0628	0.4174	0.2978	1.1666	7.7851

Based on accuracy measures in the table above, the best approach for forecasting is expanding windows.

Analysis of Weekly Page Loads

Now, we are aggregating our data to weekly time series.

```r
ts_website_weekly <- apply.weekly(ts_website, mean)
plot(ts_website_weekly, main = "Weekly Average Page Loads",
     ylab = "Page Loads", xlab = "Date")
```



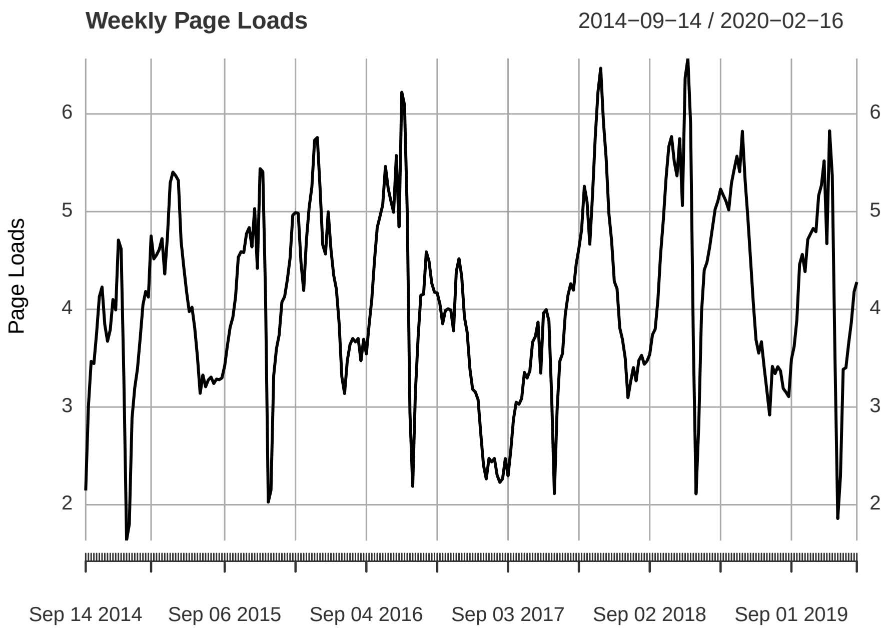


Figure 17: Weekly Page Loads - training set

```
invisible(acf2(ts_website_weekly,max.lag= 80))
```

```
adf.test(ts_website_weekly, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_website_weekly
## Dickey-Fuller = -4.4322, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

Similar to the process before we try to find the best model for the weekly data. Here the seasonality seems yearly (period of 52 weeks). We try with the model $SARIMA(3,1,0) \times (1,1,1)_{52}$.

```
dlm52.1 <- diff(diff(train_data_weekly,52),1)
plot(dlm52.1,lwd=1,main=expression(paste(Delta,Delta[52], "Weekly Page Loads train")))
```

```
invisible(acf2(dlm52.1,main=expression(paste(Delta,Delta[52],
                                             "Weekly Page Loads train"))))
```

```
weekly_visitors.fit1=sarima(train_data_weekly,3,1,0,1,1,1,52)
```

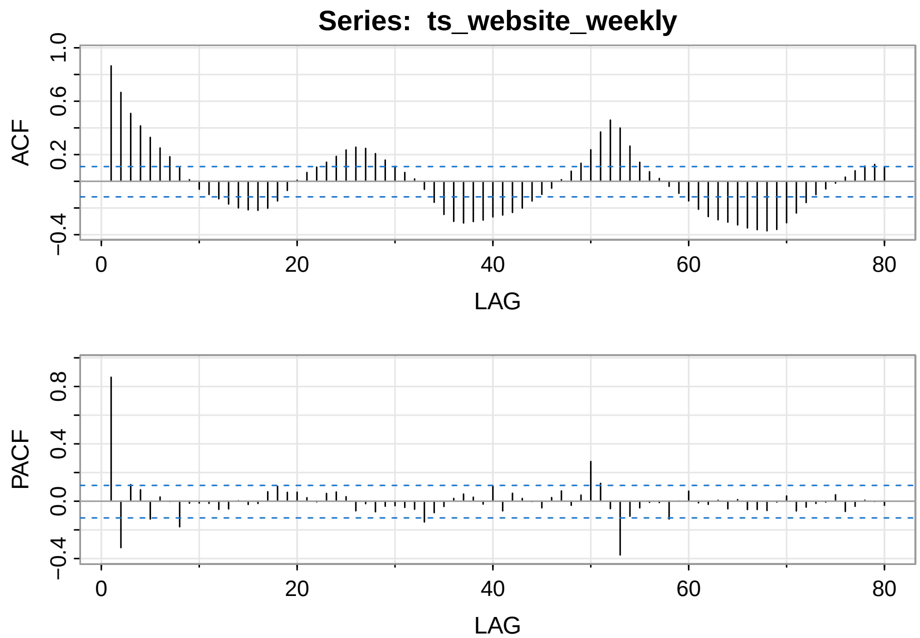


Figure 18: ACF and PACF

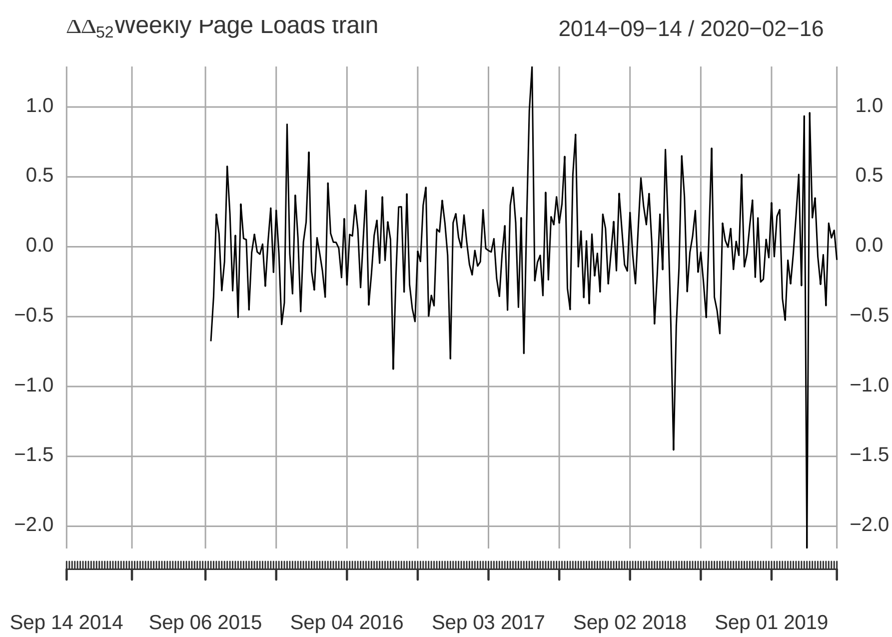


Figure 19: Plots of differenced data

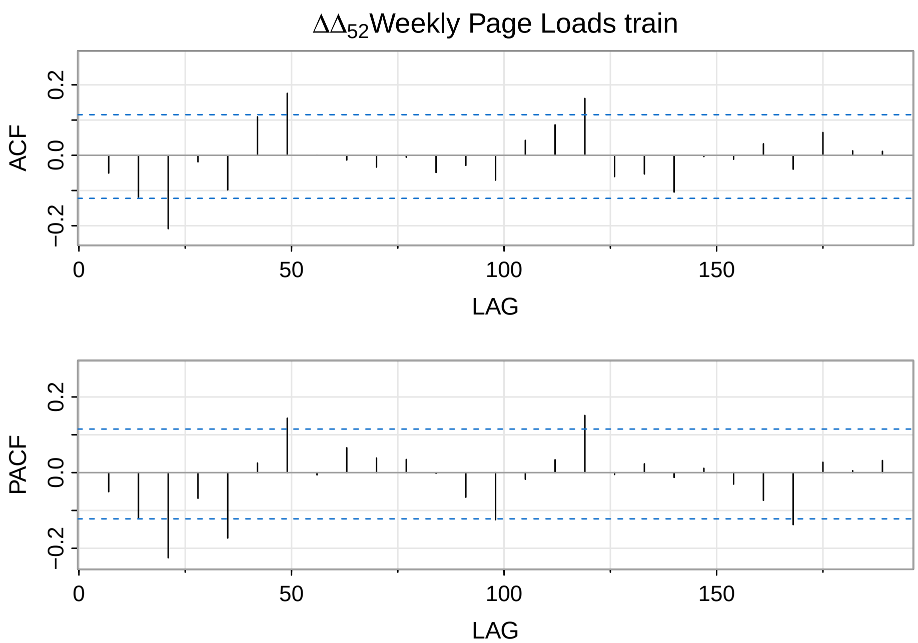
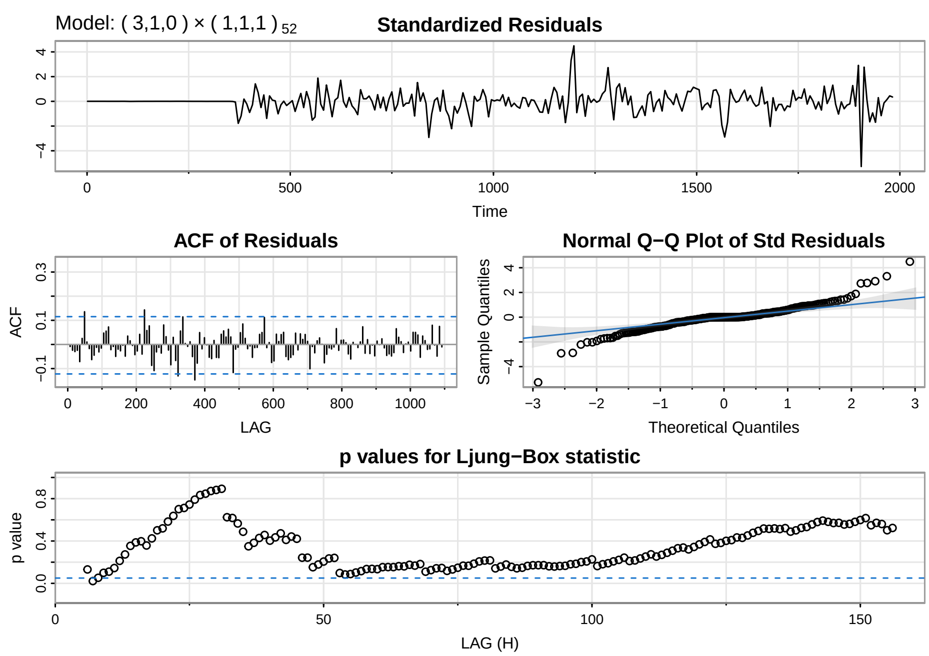


Figure 20: Plots of differenced data



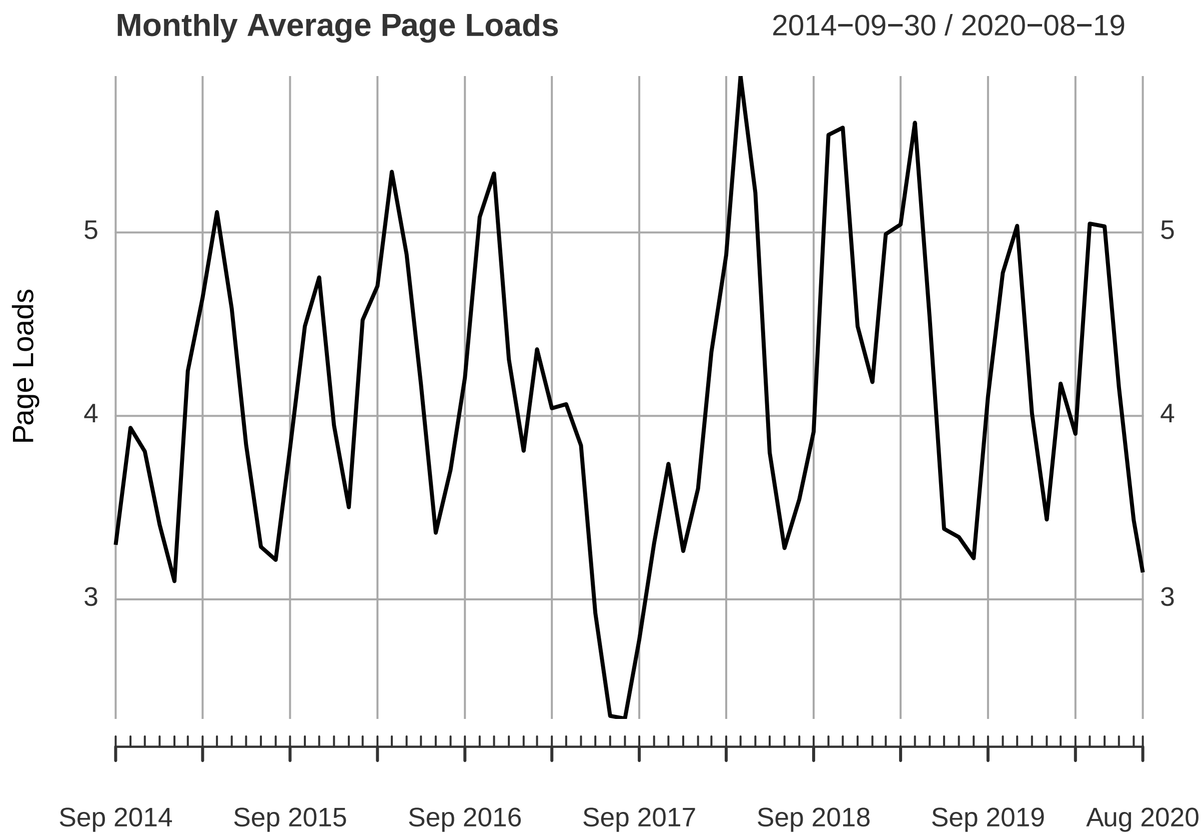

```
weekly_visitors.fit1
```

```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
```

```
## Coefficients:
##           ar1      ar2      ar3     sar1     sma1
##       -0.1867  -0.1935  -0.2096  -0.3665  -0.2132
## s.e.   0.0655   0.0653   0.0657   0.1333   0.1512
##
## sigma^2 estimated as 0.09495:  log likelihood = -64.81,  aic = 141.63
##
## $degrees_of_freedom
## [1] 226
##
## $ttable
##      Estimate     SE t.value p.value
## ar1   -0.1867 0.0655 -2.8528  0.0047
## ar2   -0.1935 0.0653 -2.9612  0.0034
## ar3   -0.2096 0.0657 -3.1890  0.0016
## sar1  -0.3665 0.1333 -2.7493  0.0065
## sma1  -0.2132 0.1512 -1.4098  0.1600
##
## $ICs
##       AIC      AICc       BIC
## 0.6131008 0.6142553 0.7025143
```

Analysis of Monthly Page Loads

```r
# Monthly Aggregation
ts_website_monthly <- apply.monthly(ts_website, mean)
plot(ts_website_monthly, main = "Monthly Average Page Loads",
     ylab = "Page Loads", xlab = "Date")
```



We divide the data into a training set and a test set. The test set will contain the last 6 months of observations.

```
# Training set: First 4.5 years, Test set: Last 6 months
cutoff_date <- as.Date("2020-02-29")
train_data_monthly <- window(ts_website_monthly, end = cutoff_date)
test_data_monthly <- window(ts_website_monthly, start = cutoff_date + 1)

plot(train_data_monthly, main = "Weekly Page Loads",
     ylab = "Page Loads", xlab = "Date")
```

```
invisible(acf2(ts_website_monthly))
```

```
adf.test(ts_website_monthly, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_website_monthly
## Dickey-Fuller = -2.5767, Lag order = 4, p-value = 0.3405
## alternative hypothesis: stationary
```

Similar to the process before we try to find the best model for the weekly data. Here the seasonality seems yearly (period of 12 months). We try with the model $SARIMA(0,1,0) \times (0,1,1)_{12}$.

```
dlm12.1 <- diff(diff(train_data_weekly,12),1)
plot(dlm12.1,lwd=1,main=expression(paste(Delta,Delta[12], "Monthly Page Loads train")))
```

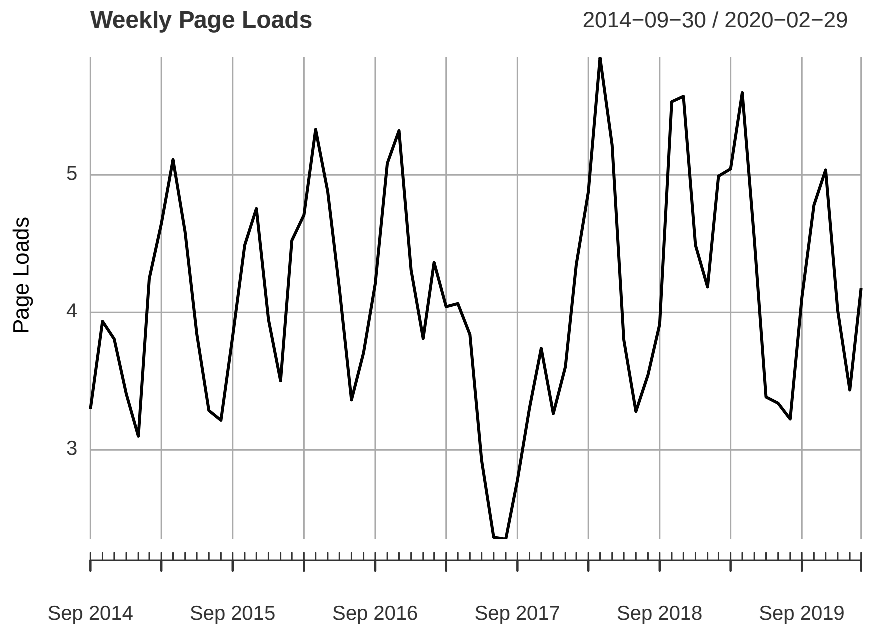


Figure 21: Weekly Page Loads - training set

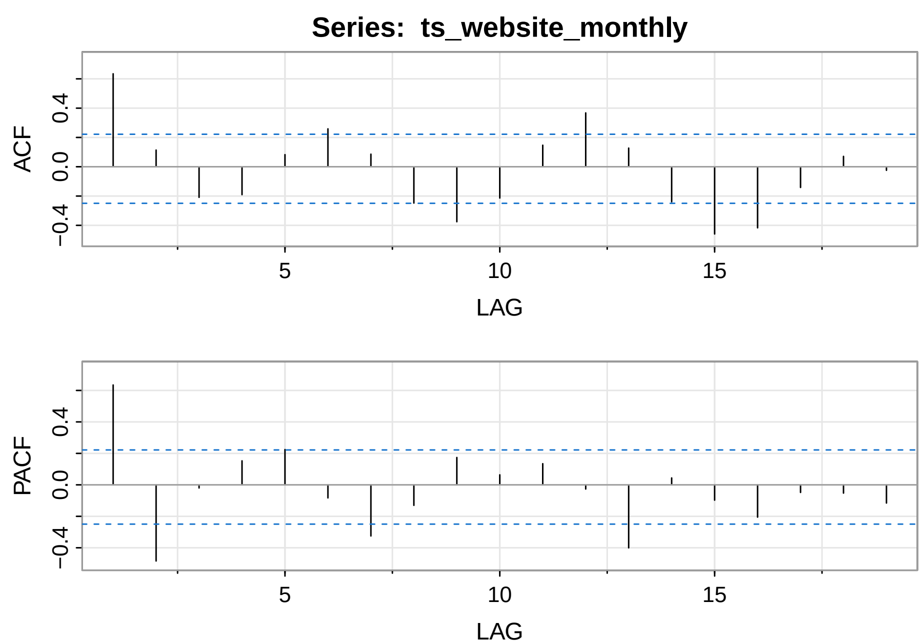


Figure 22: ACF and PACF

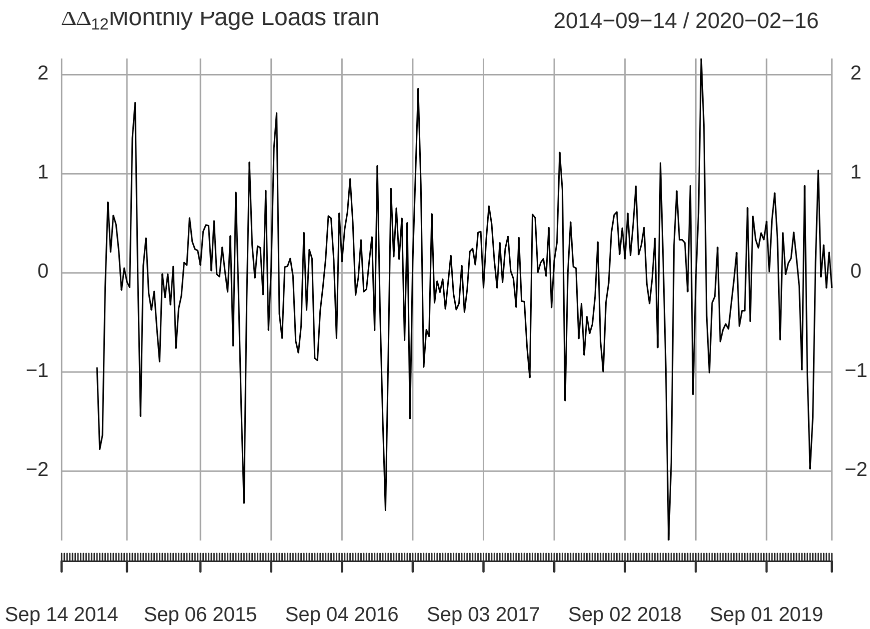
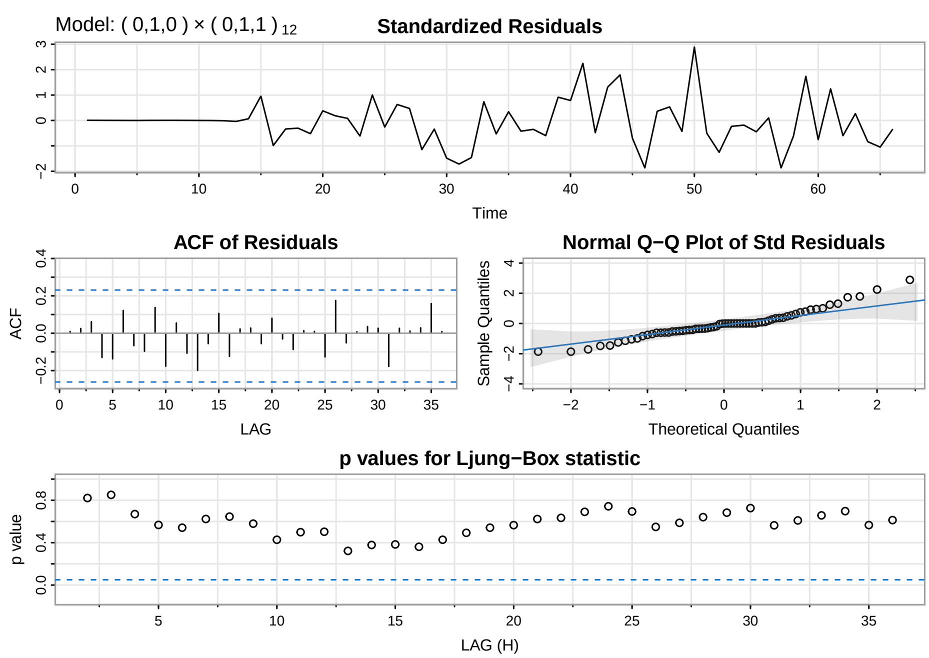


Figure 23: Plots of differenced data

```
invisible(acf2(dlm12.1,main=expression(paste(Delta,Delta[12], "Monthly Page Loads train"))))
```

```
monthly_visitors.fit1=sarima(train_data_monthly,0,1,0,0,1,1,12)
```




```
monthly_visitors.fit1
```

```
## $fit
##
## Call:
```

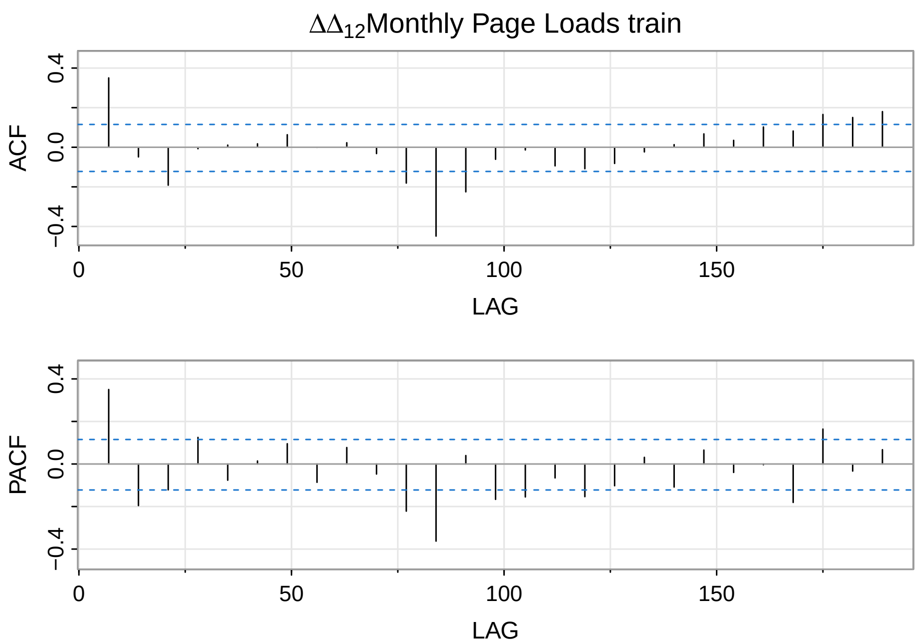


Figure 24: Plots of differenced data

```
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          sma1
##       -0.9999
## s.e.   0.6719
##
## sigma^2 estimated as 0.08519:  log likelihood = -20.05,  aic = 44.1
##
## $degrees_of_freedom
## [1] 52
##
## $ttable
##      Estimate     SE t.value p.value
## sma1  -0.9999 0.6719 -1.4882  0.1427
##
## $ICs
##       AIC      AICc       BIC
## 0.8319862 0.8334660 0.9063368
```