Time Series Project

Natalia Pludra, Gasper Pust

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 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).

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

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,n=4), 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

Row	Day	Day.Of.Week	Date	Page.Loads	Unique.Visits	First.Time.Visits	Returning.Visits
4	Wednesday	4	2014-09-17	3.667	2,614	2,327	287

We decided to focus on Daily Page Loads.

Exploratory Data Analysis

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

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

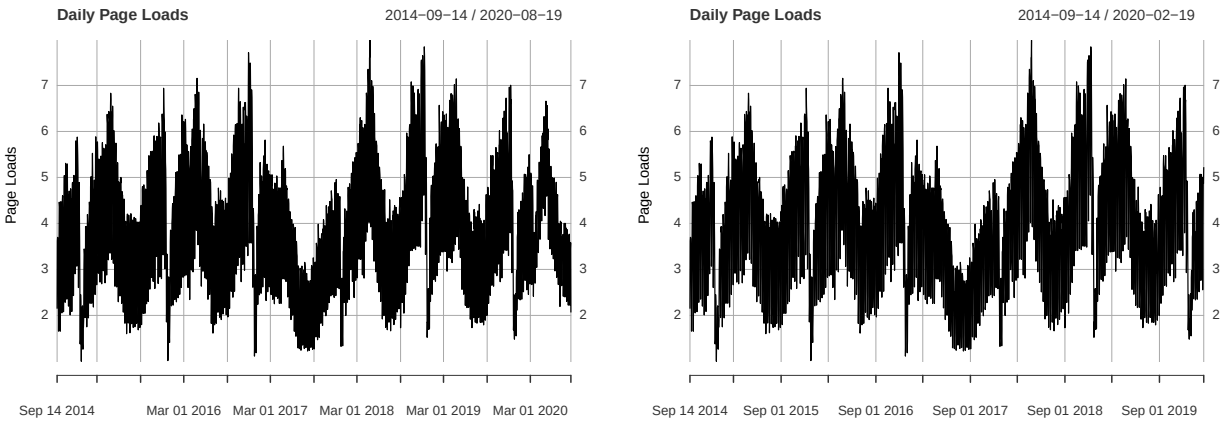
CHANGE THE SIZE OF TRAINING AND TEST SET? 60 DAYS?

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

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

# 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)
```



(a) The entire dataset

(b) Training set

Figure 1: Daily Page Loads

The plot of the training set (Figure 1 (b)) 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 and Figure 2 (a). There is no missing values in our data.

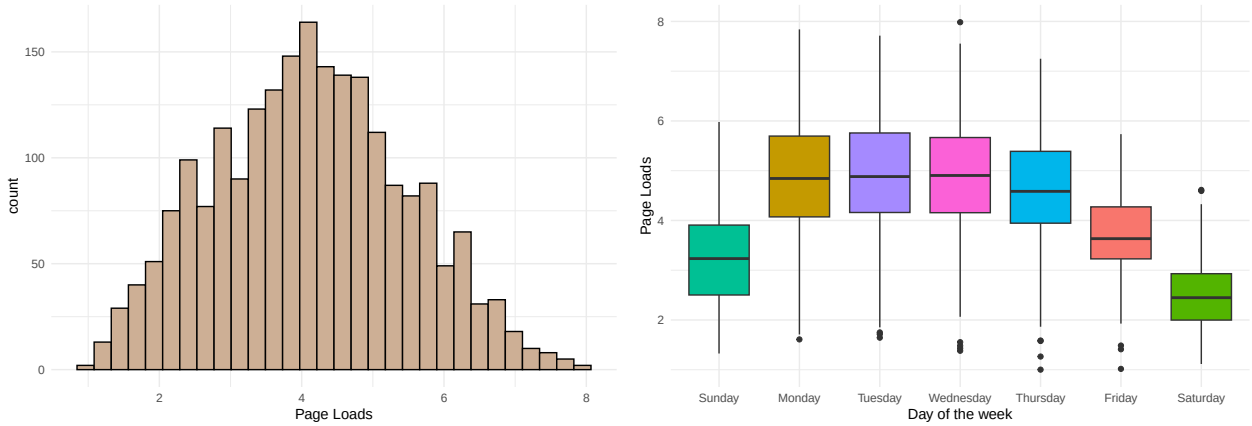
In the Figure 2 (b) we presented boxplots of Page Loads by day of the week.

```
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
```

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

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")
```



(a) Histogram of Daily Page Loads

(b) Boxplot of Page Loads by day of the week

Figure 2: Daily Page Loads Analysis

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 (Figure 3).

```
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.

```
lag1.plot(ts_website,16,cex=0.5)
```

Looking at lag plots in Figure 4, we can see the strongest correlation at lag 1, 7 and 14.

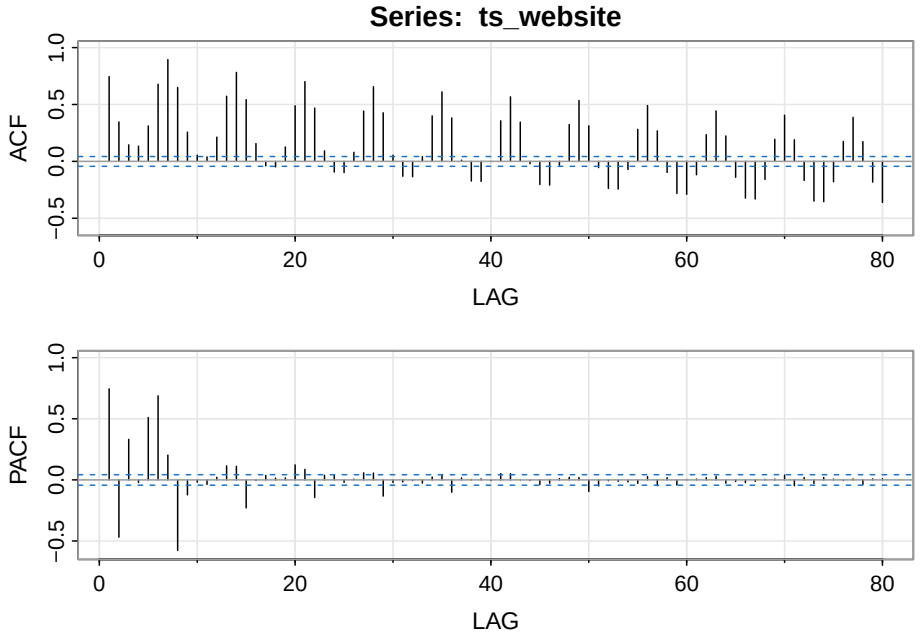


Figure 3: ACF and PACF

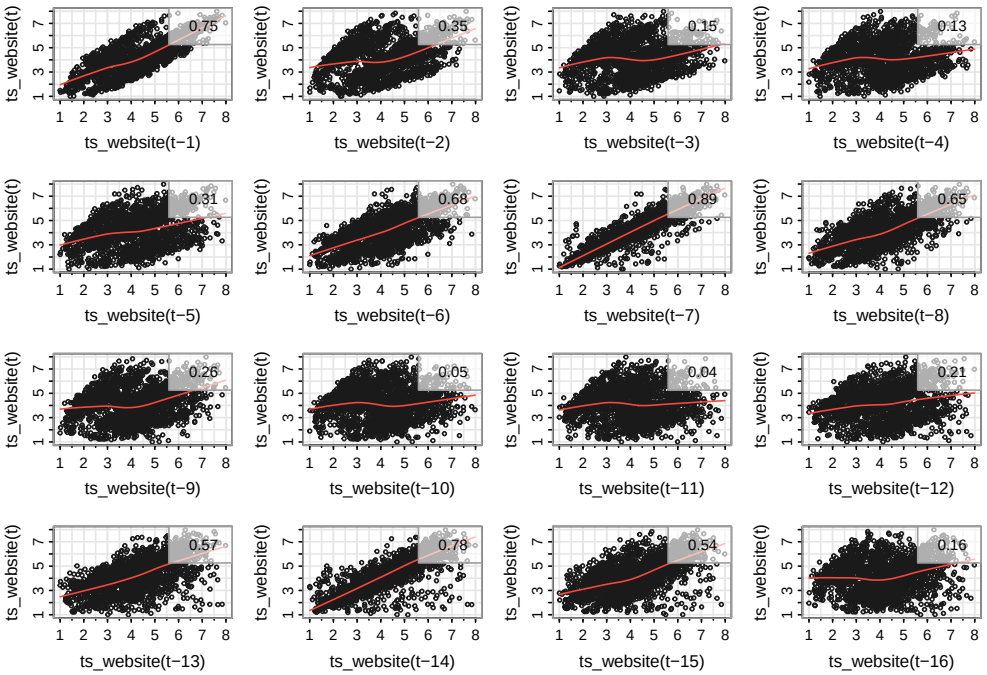
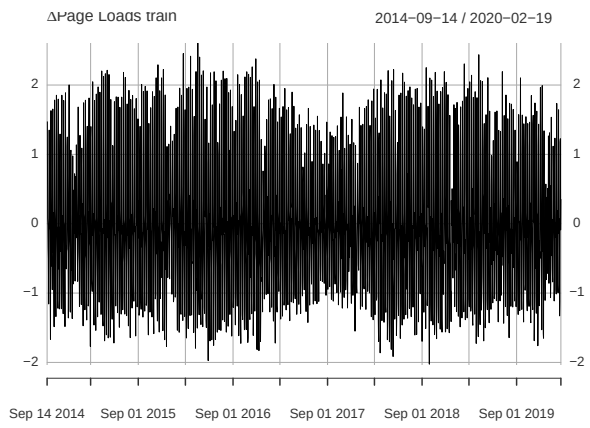


Figure 4: Correlation lag plots

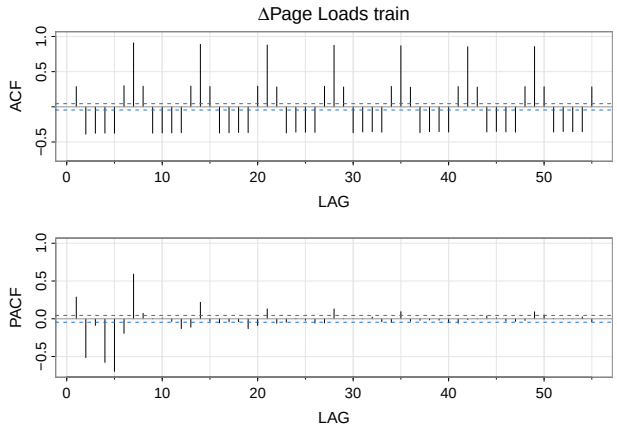
Seasonal Component and Modelling

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"))))
```



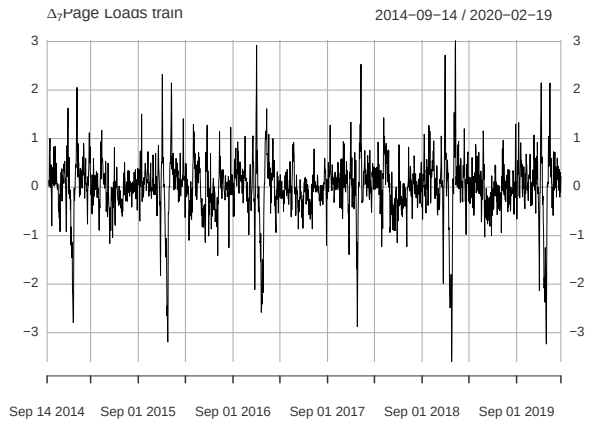
(a) Page Loads over time



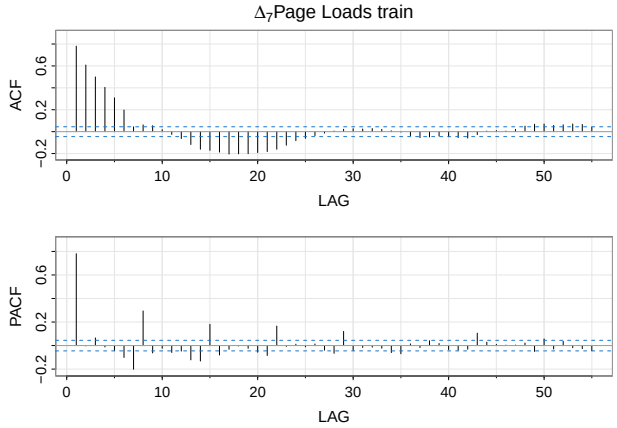
(b) ACF and PACF

Figure 5: Differenced data

```
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"))))
```



(a) Page Loads over time

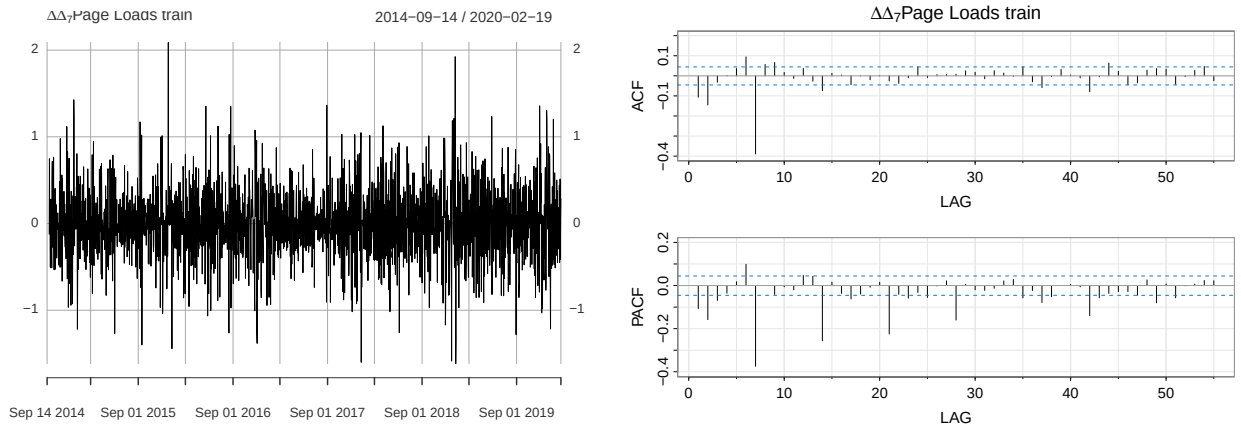


(b) ACF and PACF

Figure 6: Seasonally differenced data (period of 7 days)

```
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 in Figure 7, $\Delta_7\Delta PageLoadstrain$ seems



(a) Page Loads over time

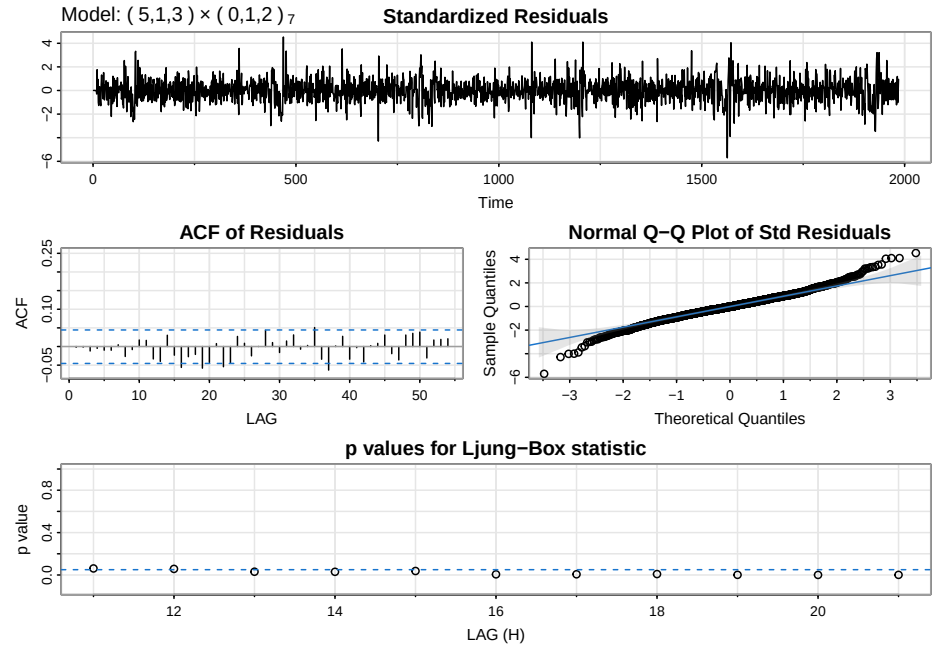
(b) ACF and PACF

Figure 7: Differenced and seasonally differenced (period of 7 days) data

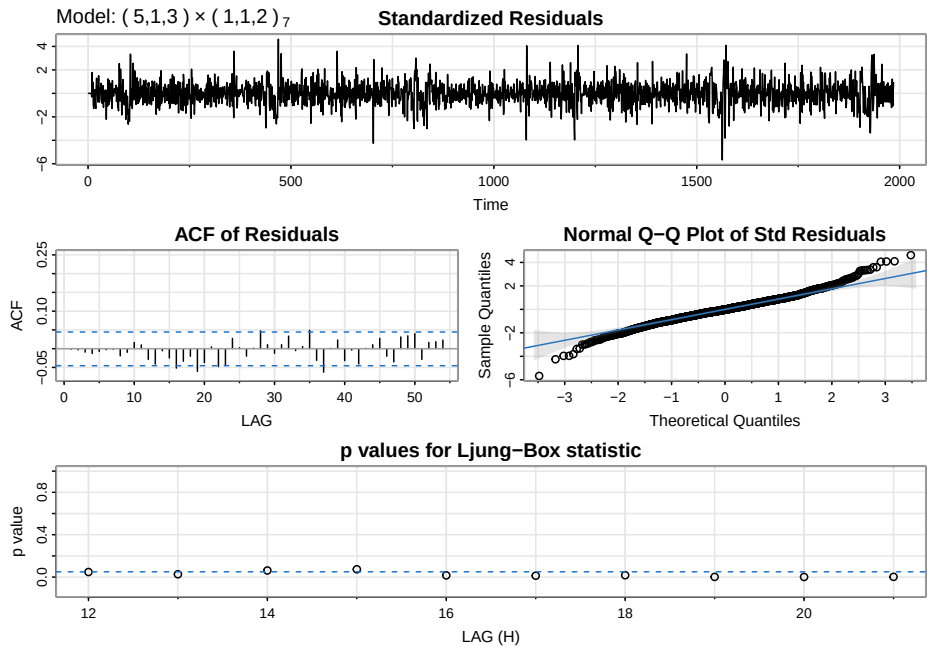
more stationary. This implies a unit root $d = 1$ as well as a seasonal unit root, $D = 1$. We can see that ACF decays to zero quicker than PACF indicating strong MA component of the model. ACF shows significant correlation at lags 7 and 14, which implies $q = 3$ and $Q = 2$. PACF shows significant correlation 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:$SARIMA(5,1,2) \times (0,1,2)_7$; 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. Below are presented the models with best results.

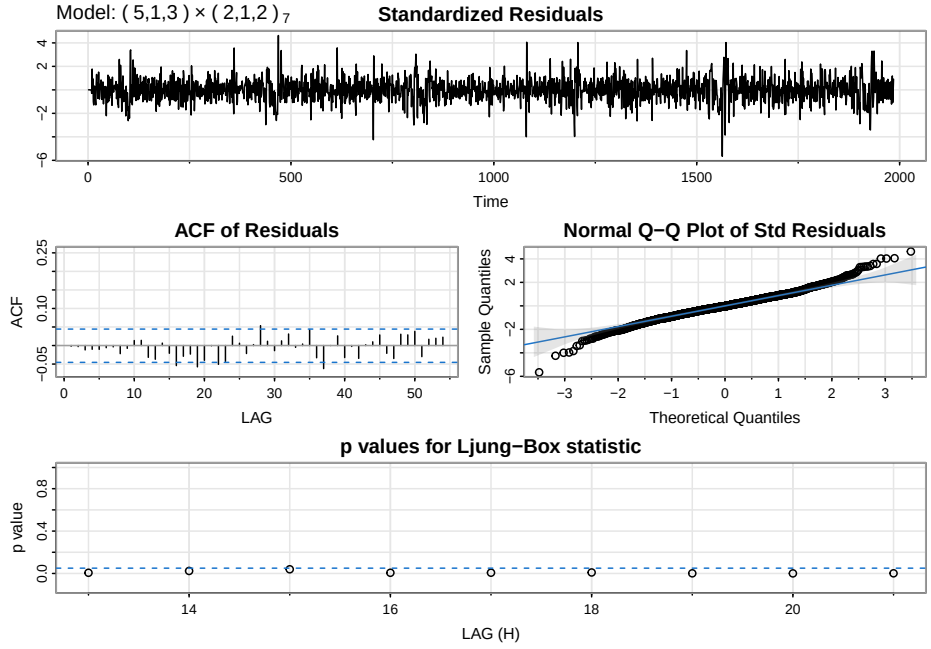
```
visitors.fit3=sarima(train_data,5,1,3,0,1,2,7)
```



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




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



Models 3, 5, and 6 exhibit comparable residual behavior, with no significant patterns in their standardized residuals and minimal autocorrelation in the ACF plots. However, model 3 shows slight deviations from normality in the Q-Q plot and lower p-values in the Ljung-Box test, suggesting it is less effective at capturing serial correlations compared to the others. Models 5 and 6 demonstrate better residual diagnostics, with higher Ljung-Box p-values and closer adherence to normality. Among them, model 5 has slightly lower residual variability and more consistent diagnostic results, making it the most suitable model for forecasting.

```
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.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 confirm that the model that provides the best fit is Model 5 $SARIMA(5,1,3)\times(1,1,2)_7$. In that model, AR3, AR4, AR5, MA3, and seasonal MA1 are significant components and critical for the model's performance. However, AR1, AR2, MA1, MA2, seasonal AR1, and seasonal MA2 are not statistically significant, and therefore contribute less to the model performance.

Forecasting

Forecasting 60-days ahead

This method involves forecasting 60 time steps into the future using a single, fixed model fit. The model is trained on a specified dataset, which remains static throughout the process. By leveraging a unique training dataset, the approach focuses on creating a long-term prediction based on the underlying patterns and dynamics captured during model training. This method is particularly useful for scenarios where computational efficiency is prioritized, as it avoids the need for repeated model fitting.

Forecasting 60 days with 1-step ahead - Interaction

In this approach, it is assumed that new observations become available at each time step during the forecasting process. These new data points are used in conjunction with the already fitted model to generate predictions for the next step. Unlike methods that refit the model with updated data, this strategy relies on the initial model fit to iteratively incorporate new observations into the forecasting pipeline. This approach is ideal for situations where updating the model at each step is unnecessary or computationally expensive, but the presence of real-time data enhances forecast accuracy.

Forecasting 60 days with 1-step ahead - Expanding windows

The expanding window approach gradually increases the size of the training dataset with each forecast step. For each new prediction, one additional time step is appended to the training data, and a new model is fitted to the expanded dataset. This process ensures that the model benefits from the most recent observations while maintaining a cumulative understanding of the past. Although computationally more demanding, this strategy often improves accuracy, especially for time series with evolving patterns or trends.

Forecasting 60 days with 1-step ahead - Recursive windows

The recursive window strategy maintains a fixed-size training dataset, but the set of observations within this window shifts forward by one time step for each forecast. This means that older data points are dropped as new observations are included in the training set. A new model is fitted at each step using the updated dataset. This approach balances the inclusion of recent information with a consistent training window size, making it effective for time series with short-term dependencies or when the focus is on the most recent dynamics in the data.

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))
```

```r
# 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)
}

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))

# 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")
```

```
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,
```

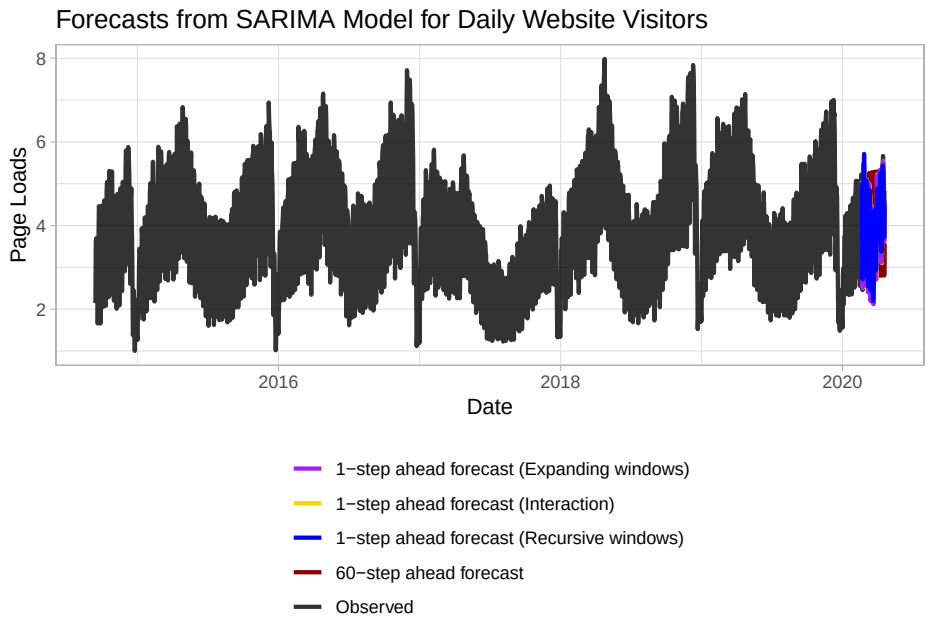


Figure 8: Plot of different forecast methods

```
          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())
```

Choosing the best approach

We checked the forecast errors of each approach used above to decide which technique provides best results for our dataset.

```
errors_1 <- test_df_filtered$Observed - forecast_df_1$Forecast
errors_2 <- test_df_filtered$Observed - forecast_df_2$Forecast
errors_3 <- test_df_filtered$Observed - forecast_df_3$Forecast
errors_4 <- test_df_filtered$Observed - forecast_df_4$Forecast

error_df <- data.frame(
```

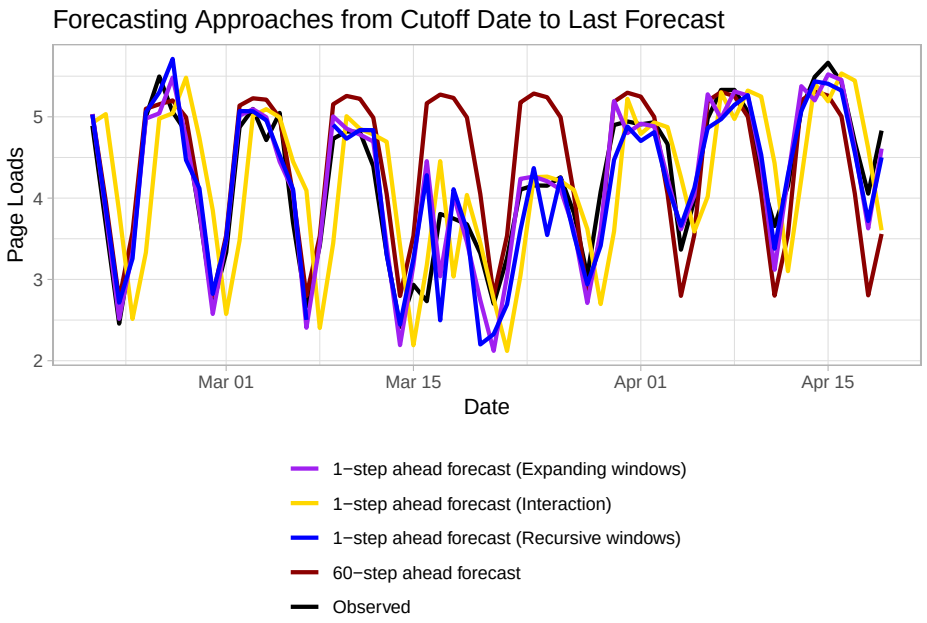


Figure 9: Plot of different forecast methods

```
  Date = test_df_filtered$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)
```

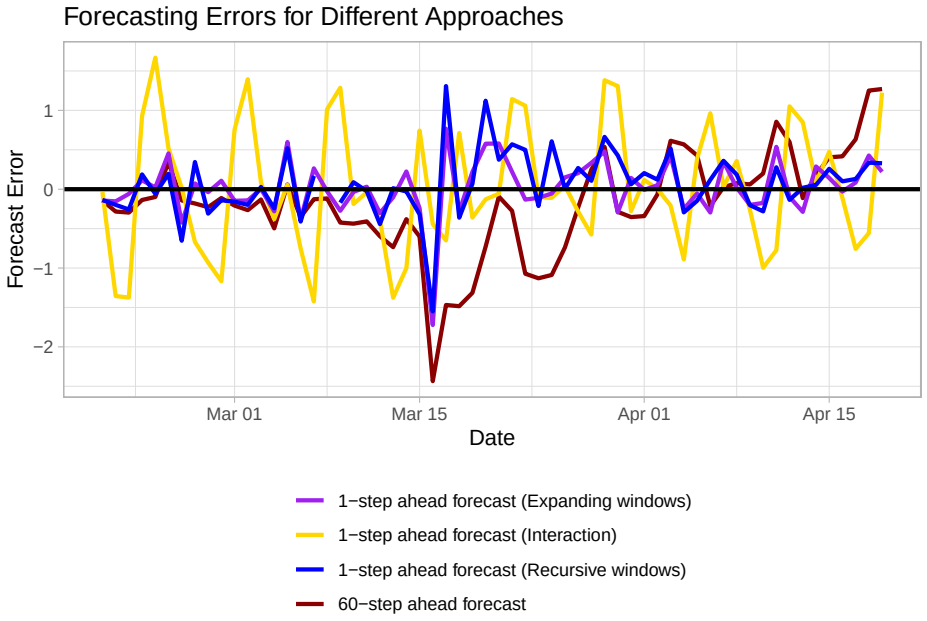


Figure 10: Plot of forecast errors

```
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)')

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

	ME	RMSE	MAE	MPE	MAPE
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 the accuracy measures in the table above, the best approach for forecasting is the expanding windows method. This method achieves the lowest values for root mean squared error (RMSE) and mean absolute error (MAE), which are critical indicators of prediction accuracy. Additionally, it outperforms other approaches in terms of mean absolute percentage error (MAPE), demonstrating its effectiveness in minimizing relative forecast errors. The expanding windows strategy effectively balances the inclusion of cumulative historical data with the integration of recent observations, leading to more reliable and precise forecasts over the 60-day horizon.

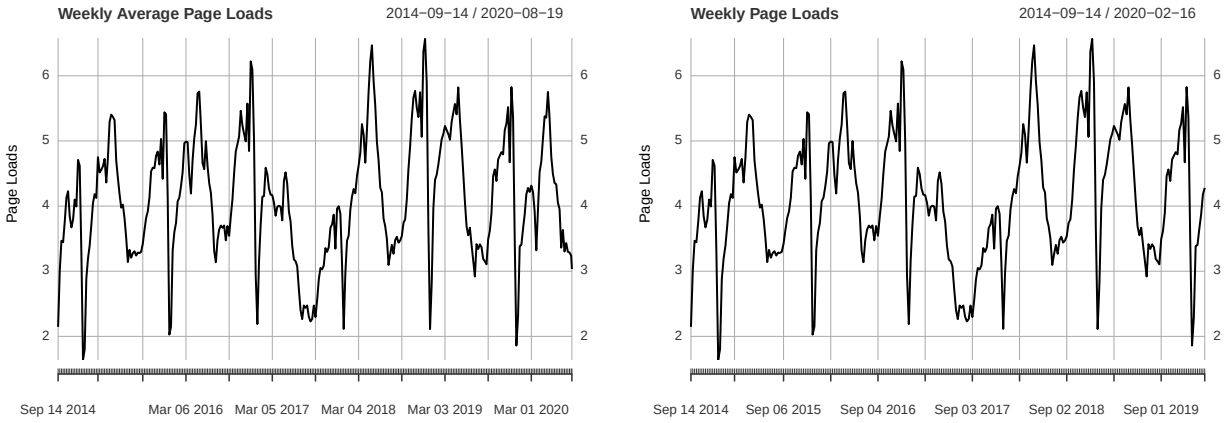
Analysis of Weekly Page Loads

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

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

# Training set: First 4.5 years, Test set: Last 6 months
cutoff_date <- as.Date("2020-02-19")
train_data_weekly <- window(ts_website_weekly, end = cutoff_date)
test_data_weekly <- window(ts_website_weekly, start = cutoff_date + 1)

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



(a) The entire dataset

(b) The training dataset

Figure 11: Weekly Page Loads

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

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

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

```
##
##  Augmented Dickey-Fuller Test
```

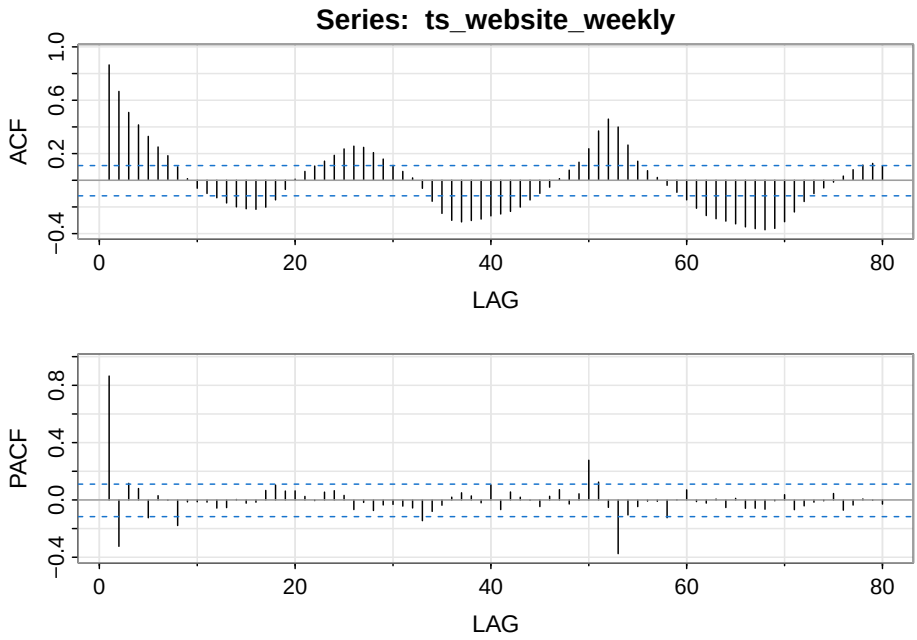
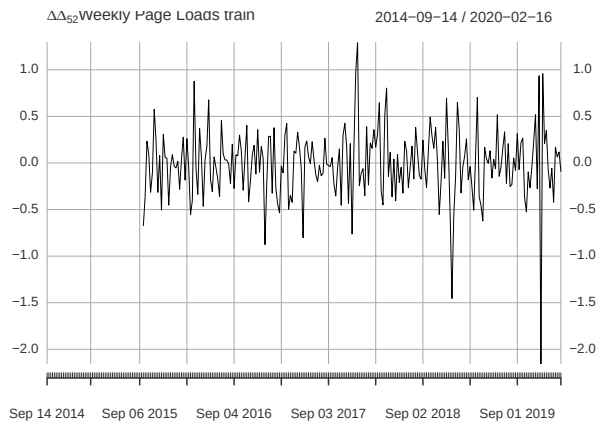


Figure 12: ACF and PACF

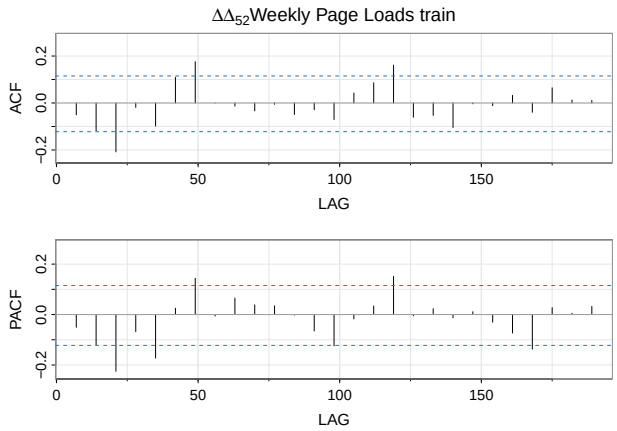
```
##
## data:  ts_website_weekly
## Dickey-Fuller = -4.4322, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

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"))))
```



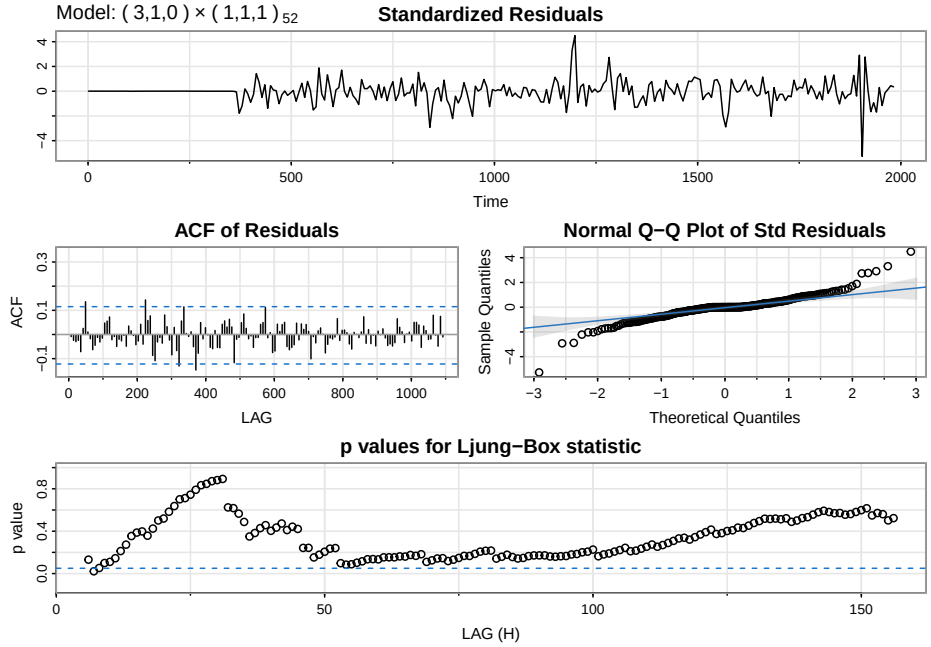
(a) Page Loads over time



(b) ACF and PACF

Figure 13: Differenced data

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




```
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
```

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

# Approach 3: Forecasting 26 weeks with 1-step ahead - Expanding windows
output_3 <- data.frame()
for (i in 0:25) {
  df_select <- ts_df %>% filter(time <= cutoff_date + weeks(i))
  ts_select <- ts(df_select$value, freq = 52, end = c(year(tail(df_select$time, 1)),
    yday(tail(df_select$time, 1))))
  fit <- Arima(ts_select, order = c(3, 1, 0), seasonal = list(order = c(1, 1, 1),
    period = 52))
  forecasted <- forecast(fit, h = 1, level = 95)
  aux <- data.frame(time = cutoff_date + weeks(i + 1), value = as.numeric(forecasted$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))

test_df <- data.frame(Date = index(test_data_weekly), Observed = coredata(test_data_weekly))

last_forecast_date <- max(forecast_df_3$Date)

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

ggplot() +
  geom_line(data = ts_df_filtered, aes(x = time, y = value, color = "Observed"),
            size = 1, alpha = 0.8) +
  geom_line(data = forecast_df_3, aes(x = Date, y = Forecast,
            color = "1-step ahead forecast (Expanding windows)"), size = 1) +
  labs(title = "Forecasts from SARIMA Model for Weekly Website Visitors",
       x = "Date", y = "Page Loads") +
  scale_color_manual(values = c(
    "Observed" = "black",
    "1-step ahead forecast (Expanding windows)" = "green")) +
  theme_light() + theme(legend.title = element_blank(),
    legend.background = element_blank(), legend.position = "bottom",
    legend.direction = "vertical")
```

```r
ggplot() +
  geom_line(data = test_df_filtered, aes(x = Date, y = Observed, color = "Observed"),
            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") +
  labs(title = "Forecasting Approaches from Cutoff Date to Last Forecast",
       x = "Date", y = "Page Loads") +
    scale_color_manual(values = c(
    "Observed" = "black",
    "1-step ahead forecast (Expanding windows)" = "green")) +
  theme_light() + theme(legend.position = "bottom",
    legend.direction = "vertical", legend.title = element_blank())
```

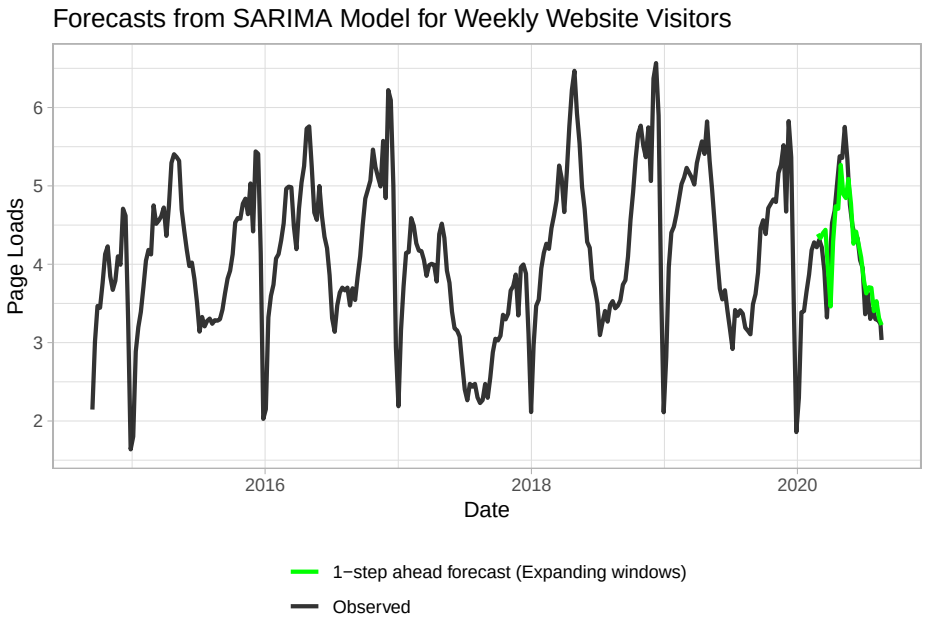


Figure 14: Plot of forecast with expanding windows method

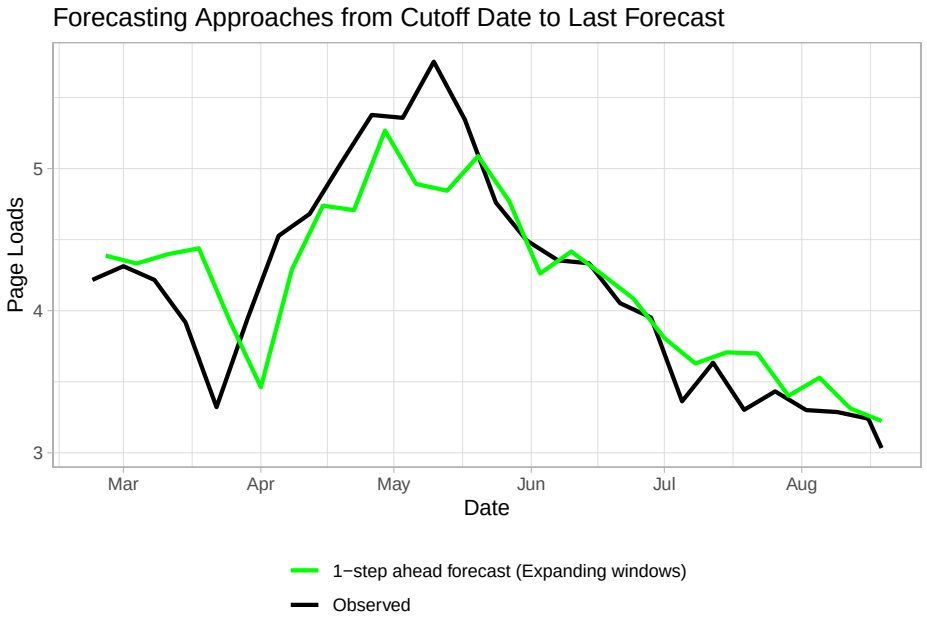


Figure 15: Plot of forecast with expanding windows method

```r
errors_3 <- test_df_filtered$Observed - forecast_df_3$Forecast

error_df <- data.frame(Date = test_df_filtered$Date,
  '1-step ahead forecast (Expanding windows)' = errors_3)

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

colnames(error_df) <- c('Date', '1_step_ahead_forecast_Expanding_windows')

ggplot(error_df, aes(x = Date)) +
  geom_line(aes(y = `1_step_ahead_forecast_Expanding_windows`,
            color = "1-step ahead forecast (Expanding windows)"), size = 1) +
  labs(title = "Forecasting Errors for Exapanding Windows",
       x = "Date", y = "Forecast Error") +
    scale_color_manual(values = c(
    "1-step ahead forecast (Expanding windows)" = "green")) +
  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)
```

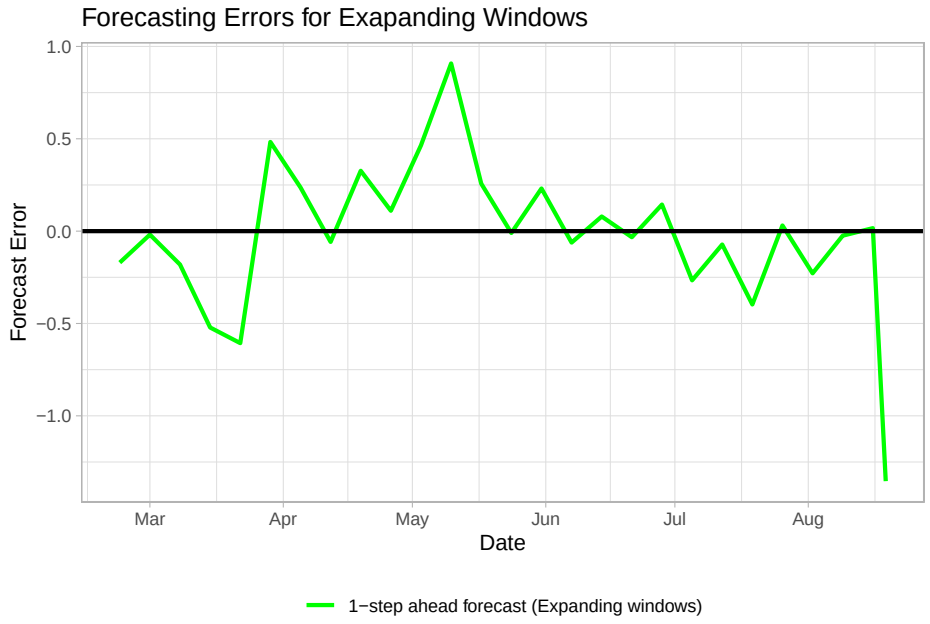


Figure 16: Plot of forecast errors

```r
test_df_filtered <- head(test_df, 26)

accuracy <- accuracy(forecast_df_3$Forecast, test_df_filtered$Observed)

rownames(accuracy) <- c('1-step ahead forecast (Expanding windows)')

knitr::kable(accuracy, digits = 4, align = "ccccccc",
  caption = "Accuracy measures for Exapnding Windows")
```

Table 3: Accuracy measures for Exapnding Windows

	ME	RMSE	MAE	MPE	MAPE
1-step ahead forecast (Expanding windows)	0.0245	0.3158	0.2284	-0.2441	5.4196

Summary

BENEFITS AND LIMITATIONS OF THE MODELS