time series project

"Alessandro Scarpato (5206880), Matteo Barbieri (5208190), Lorenzo Bonatti (5208784)"

2024-02-19

```
library(forecast)
```

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

```
library(tseries)
library(knitr)
library(kableExtra)
```

London bike sharing dataset

Dataset description

```
head(full_dataset)
```

```
##             timestamp cnt  t1  t2   hum wind_speed weather_code is_holiday
## 1 2015-01-04 00:00:00 182 3.0 2.0  93.0        6.0            3          0
## 2 2015-01-04 01:00:00 138 3.0 2.5  93.0        5.0            1          0
## 3 2015-01-04 02:00:00 134 2.5 2.5  96.5        0.0            1          0
## 4 2015-01-04 03:00:00  72 2.0 2.0 100.0        0.0            1          0
## 5 2015-01-04 04:00:00  47 2.0 0.0  93.0        6.5            1          0
## 6 2015-01-04 05:00:00  46 2.0 2.0  93.0        4.0            1          0
##   is_weekend season
## 1          1      3
## 2          1      3
## 3          1      3
## 4          1      3
## 5          1      3
## 6          1      3
```

To perform the analysis we are using a dataset regarding *London bike sharing*, which takes into consideration the number of bike shares in London every hour of the day starting from January 4th, 2015, to January 3rd, 2017.
To explain the number of bike shares several covariates of different types are taken into consideration:

- **"cnt"** - the count of a new bike shares

- **“t1”** - real temperature in C
- **“t2”** - temperature in C “feels like” “hum” - humidity in percentage
- **“wind_speed”** - wind speed in km/h
- **“weather_code”** - category of the weather **
- **“is_holiday”** - boolean field - 1 holiday / 0 non holiday
- **“is_weekend”** - boolean field - 1 if the day is weekend
- **“season”** - category field meteorological seasons: 0-spring ; 1-summer; 2-fall; 3-winter.

** **“weather_code”** category description:

1 = Clear ; mostly clear but have some values with haze/fog/patches of fog/ fog in vicinity

2 = scattered clouds / few clouds

3 = Broken clouds

4 = Cloudy

7 = Rain/ light Rain shower/ Light rain

10 = rain with thunderstorm

26 = snowfall

94 = Freezing Fog

Data sources

The data is acquired from 3 sources:

- https://cycling.data.tfl.gov.uk/ ‘Contains OS data © Crown copyright and database rights 2016’ and Geomni UK Map data © and database rights [2019] ‘Powered by TfL Open Data’
- freemeteo.com - weather data
- https://www.gov.uk/bank-holidays From 1/1/2015 to 31/12/2016

Project goal

The goal of our project is to perform a multi-step strategy in order to build an appropriate and efficient time series analysis.

Preliminary analysis

In preliminary analysis for time series analysis (TSA), we begin by visually examining the data to identify any prominent patterns or trends. This often involves plotting the time series data and its associated autocorrelation and partial autocorrelation functions to understand the autocorrelation structure. Descriptive statistics are then computed to summarize key characteristics such as the mean, variance, and standard deviation, providing insights into the central tendency and variability of the data. These initial steps serve as the foundation for selecting suitable modeling approaches and refining model specifications in TSA.

Check for NA

Firstly, it is imperative to inspect for any anomalous values, such as missing data (e.g., NAs), which may affect our analytical process. Fortunately, upon examination, our dataset reveals the absence of such values. Consequently, we can proceed confidently with the time series analysis (TSA)

```
na_observations <- is.na(full_dataset)
na_rows <- which(na_observations)
na_rows
```

```
## integer(0)
```

Reduced dataset

Given that the full dataset comprises over 17,000 observations, we have opted to cut off our analysis by focusing on a subset of 705 observations corresponding to the month of June (From June 4th, 2016, to July 5th, 2016) . This approach allows for a more focused representation of hourly bike rentals without the burden of managing the extensive volume of data present in the original dataset.

```
bike <- full_dataset[12340:13044,]
y = bike$cnt
```

```
head(bike)
```

```
##                 timestamp cnt t1 t2 hum wind_speed weather_code is_holiday
## 12340 2016-06-04 00:00:00 451 12 12  94        7.5            3          0
## 12341 2016-06-04 01:00:00 344 12 12  94        7.5            3          0
## 12342 2016-06-04 02:00:00 265 12 12  94        8.0            3          0
## 12343 2016-06-04 03:00:00 193 12 12  94        6.5            3          0
## 12344 2016-06-04 04:00:00 110 12 12  94        6.5            4          0
## 12345 2016-06-04 05:00:00 105 12 12  94        6.5            3          0
##       is_weekend season
## 12340          1      1
## 12341          1      1
## 12342          1      1
## 12343          1      1
## 12344          1      1
## 12345          1      1
```

To perform our time series analysis, we decide to use as response variable *cnt* (the number of bikes rented for each hour), in order to predict the evolution of the number of bikes rented in London during the different hours of day in the summer season.

Statistical analysis

We also perfomed some exploratory analysis on the variable of interest *cnt*, calculating the mean, the variance, the standard deviation, the range of values and the days when it occured the minimum and the maximum number of bike renting per hour.

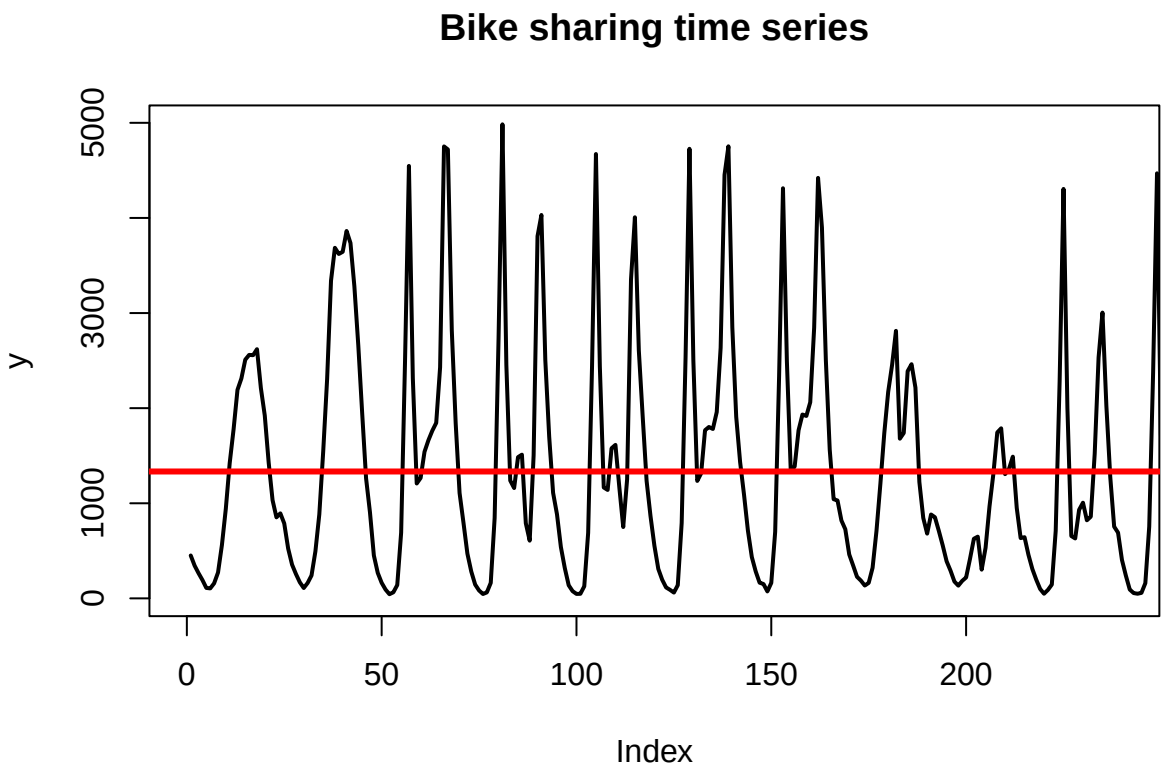
```r
# Find the index of the maximum bike shares
max_bike_shares_index <- which.max(y)
max_date <- bike$timestamp[max_bike_shares_index]

# Find the index of the minimum bike shares
min_bike_shares_index <- which.min(y)
min_date <- bike$timestamp[min_bike_shares_index]

# Calculate other statistics
mean_value <- mean(y)
variance_value <- var(y)
sd_value <- sd(y)
min_value <- min(y)
max_value <- max(y)

# Create a data frame with the statistics
statistics_df <- data.frame(
  Statistic = c("Mean", "Variance", "Standard Deviation", "Minimum", "Maximum", "Max Date", "Min Date"),
  Value = c(mean_value, variance_value, sd_value, min_value, max_value, max_date, min_date)
)

# Print the table using kable
kable(statistics_df, caption = "Summary Statistics of Bike Shares Data")
```

Table 1: Summary Statistics of Bike Shares Data

Statistic	Value
Mean	1334.14468085106
Variance	1318086.94778772
Standard Deviation	1148.07967832713
Minimum	12
Maximum	4984
Max Date	2016-06-07 08:00:00
Min Date	2016-06-23 03:00:00

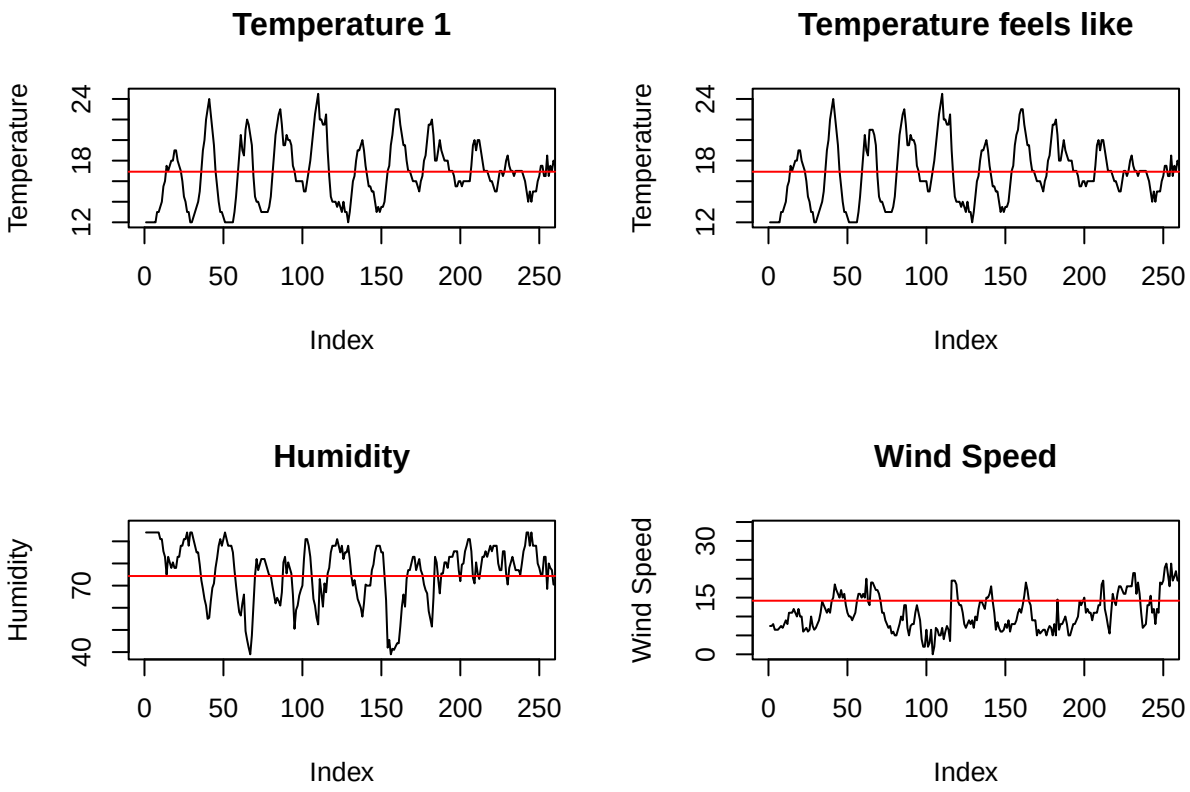
Plots

```r
plot(y,type = "l",xlim = c(0,240), lwd = 2,  main = "Bike sharing time series")
abline(h = mean(bike$cnt), col = "red", lwd = 3)
```



In this plot, we can observe the rental behavior throughout each hour of the day. A clear seasonal pattern emerges, repeating every 24 hours. However, this seasonal pattern could pose several challenges during analysis, potentially leading to non-stationarity attributable to seasonality. Furthermore, examining the red line, we notice that the mean value of rentals appears to remain constant over time.

```
par(mfrow = c(2, 2))

plot(bike$t1, type = "l", main = "Temperature 1", ylab = "Temperature",xlim = c(0,250))
abline(h = mean(bike$t1), col = "red")

plot(bike$t2, type = "l", main = "Temperature feels like", ylab = "Temperature",xlim = c(0,250))
abline(h = mean(bike$t2), col = "red")

plot(bike$hum, type = "l", main = "Humidity", ylab = "Humidity",xlim = c(0,250))
abline(h = mean(bike$hum), col = "red")

plot(bike$wind_speed, type = "l", main = "Wind Speed", ylab = "Wind Speed",xlim = c(0,250))
abline(h = mean(bike$wind_speed), col = "red")
```



In this above plot, we can observe four exogenous factors (in this case: actual temperature, perceived temperature, humidity, and wind speed). Exogenous variables in time series refer to additional variables that may influence the main time series but are not directly caused by it. By observing the temperature graphs in particular, we can notice some sort of coincidence with the bike rentals graph, which could imply some form of dependency between the two variables.

Check for stationarity assuption

With a statistical test we check the assumption of stationarity of our time series.
Null-hypothesis H0: non-stationarity Alternative hypothesis H1: stationarity

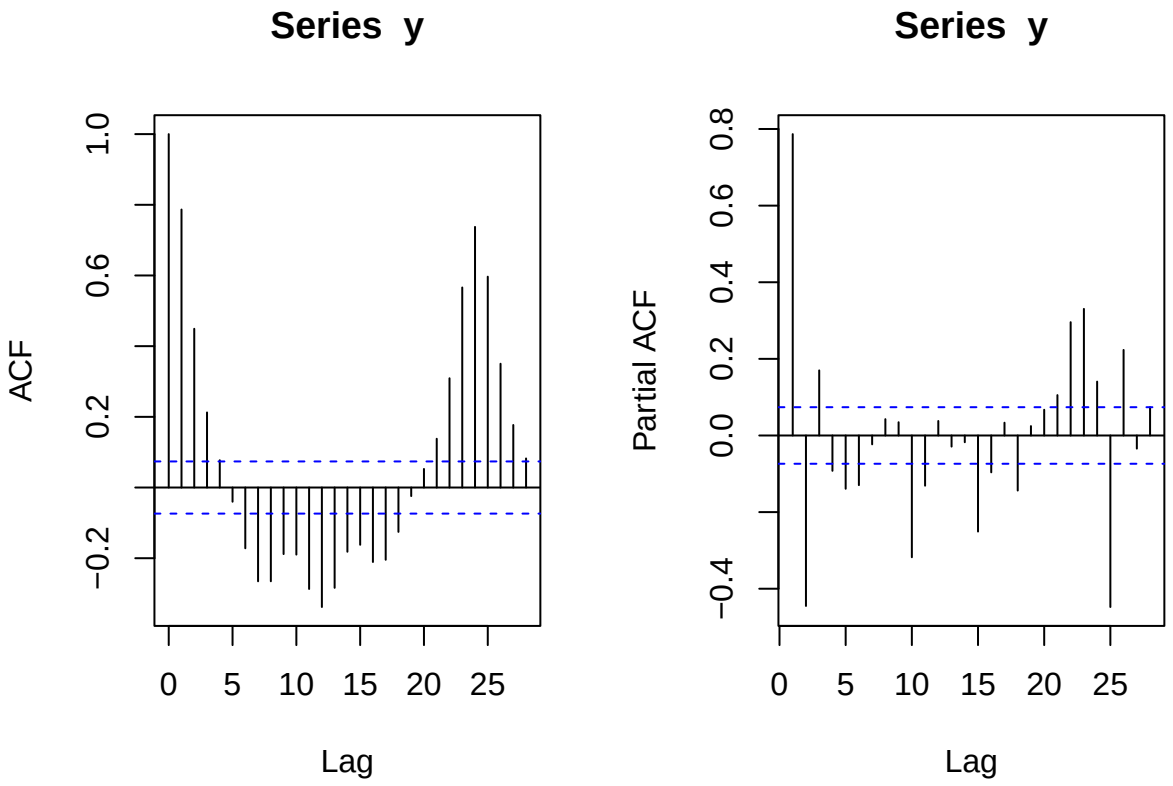
```
adf.test(y)
```

```
## Warning in adf.test(y): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  y
## Dickey-Fuller = -8.9491, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

This discrepancy implies that the p-value is significant at a lower level than reported. In summary, the test results indicate that the time series data (represented by 'y') is stationary, as the calculated p-value is significant at the chosen significance level.

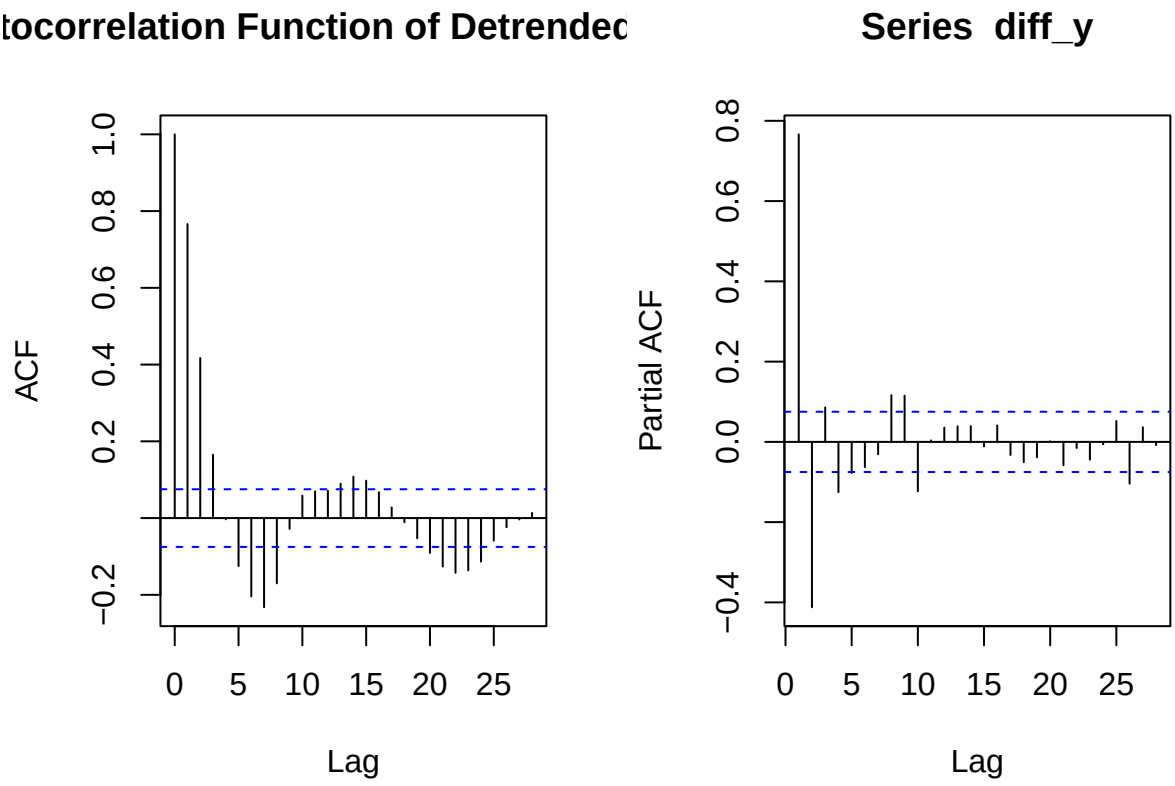
Model specification

```
par(mfrow = c(1, 2))
acf(y)
pacf(y)
```



Based on the previous autocorrelation plot, a strong correlation between observations is evident every 24 observations (equivalent to 24 hours), indicating a clear seasonal daily trend. To address this, we performed differentiation on the series at intervals of 24 observations and proceeded to plot the autocorrelation function. The objective was to determine whether this differentiation successfully mitigated the peaks observed at every 24-hour interval.

```
diff_y <- diff(y, lag = 24)  # Differencing with a lag of 24 (corresponding to one day)

par(mfrow = c(1,2))

acf(diff_y, main = "Autocorrelation Function of Detrended Series")
pacf(diff_y)
```



From the plot, we can observe that the peaks every 24 observations have practically disappeared, the autocorrelation function seems much more smooth. This suggests that there is indeed a daily seasonal trend that can be removed by differencing the series. This means that when fitting the model later on, we will use the SARIMA function with a first-order seasonal differenciation.

The seasonal differencing aims to remove the seasonal component of the time series, while ordinary differencing (or first-order differencing) aims to make the time series stationary by removing the trend component, in this case we didn't applied the first order differentiation since from the test result the series was stationary and there was non trend component to be removed.

By observing the autocorrelation function, we were able to diagnose that the stochastic process generating our series is autoregressive rather than moving average. This is because we observe a slow decay in autocorrelation as the lags increase. If, for example, we were facing a moving average process of order 2, we would have observed high correlation at lags 1 and 2, and suddenly we would have observed a huge drop in autocorrelation at lag 3, but this is not the case.

Now, instead, we need to diagnose what type of autoregressive process we are dealing with. To do this, we should look not at the autocorrelation function but at the partial autocorrelation function. From the plot of the PACF, we observe that the last nonzero partial autocorrelation is at lag 4, although it is slightly above the threshold. In any case, we can compare some of the best models using AIC and BIC information criteria. The models we will compare are: SARMA(2,1), SARMA(3,1), SARMA(4,1).

By disposing the plot of the partial autocorrelation function, we can observe how the spikes that we could observe in the original graph are no longer visible, that means that a seasonal differentiation is a god way to solve our seasonality problem.

Trying different models

After observing the autocorrelation and partial autocorrelation plots, we can infer that our model could be a SARIMA model where the regular autoregressive component might be of order 4, and in the seasonal component, we would differentiate only once to remove the daily seasonality of our hourly data, as suggested by the ACF plot on the differenced series shown earlier. To be sure about this reasoning, we decide to fit several plausible models with different orders and compare them according to some different metrics criteria.

The SARMA (Seasonal Autoregressive Moving Average) model is a time series model that combines autoregressive (AR) and moving average (MA) components with a seasonal pattern. The model is represented by the equation:

$$X_t = \psi_1 X_{t-1} + \psi_2 X_{t-2} + \ldots + \psi_p X_{t-p} + A_t + \theta_1 Z_{t-1} + \theta_2 A_{t-2} + \ldots + \theta_q A_{t-q}$$

Where:

- X_t - represents the observed value of the time series at time t.
- $\psi_1, \psi_2, \ldots, \psi_p$ - are the autoregressive coefficients, representing the effect of past values of the time series (X) on the current value (X_t). For example, ψ_1 represents the effect of the value at time $t-1$ on X_t, ψ_2 represents the effect of the value at time $t-2$ on X_t, and so on up to p time steps back.
- A_t - represents the white noise at time t, which is a random term contributing to the model's error.
- $\theta_1, \theta_2, \ldots, \theta_q$ - are the moving average coefficients, representing the effect of past values of the white noise (A) on the current value of the time series (X_t). For example, θ_1 represents the effect of the white noise at time $t-1$ on X_t, θ_2 represents the effect of the white noise at time $t-2$ on X_t, and so on up to q time steps back.

In summary, the SARMA model explains how past values of the time series and white noise influence the current value of the series, taking into account both autoregressive and moving average components. The presence of terms such as $X_{t-1}, X_{t-2}, \ldots$ indicates the dependence of the time series on its past values, while terms such as $Z_{t-1}, A_{t-2}, \ldots$ indicate the effect of white noise.

```
SARMA2 = arima(y,order = c(2,0,1), seasonal = list(order = c(0,1,1),period = 24))
SARMA3 = arima(y,order = c(3,0,1), seasonal = list(order = c(0,1,1),period = 24))
SARMA4 = arima(y,order = c(4,0,1), seasonal = list(order = c(0,1,1),period = 24))
```

These codes aim to estimate three seasonal ARIMA (SARIMA) models with different autoregressive (AR) orders, but with the same seasonal specification. The goal is to evaluate which model fits the data best by comparing their performances.

To do so, three SARIMA models with different AR orders (2, 3, and 4) are tested while keeping the seasonal specification constant. The seasonal specification indicates that there is a seasonal differencing of order 1 (D = 1) and a seasonal moving average term of order 1 (Q = 1), with a seasonal period of 24 hours, suggesting that the data exhibit daily seasonality.

After estimating the three models, their performances can be compared using evaluation criteria such as the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC). The model with the lowest AIC or BIC value is typically considered the best-fitting model as it seeks to balance the model's goodness of fit with its complexity.

```
coef(SARMA2)
```

```
##        ar1        ar2        ma1       sma1
##  0.8294193 -0.2268228  0.3137785 -0.1780428
```

$$X_t = 0.8294X_{t-1} - 0.2268X_{t-2} + A_t + 0.3138A_{t-1} - 0.1780A_{t-2}$$

```
coef(SARMA3)
```

```
##         ar1         ar2         ar3         ma1        sma1
##  0.69010974 -0.06520502 -0.07437455  0.45074435 -0.17640756
```

$$X_t = 0.6901X_{t-1} - 0.0652X_{t-2} - 0.0744X_{t-3} + A_t + 0.4507A_{t-1} - 0.1764A_{t-2}$$

```
coef(SARMA4)
```

```
##        ar1        ar2        ar3        ar4        ma1       sma1
##  1.5980694 -1.1203788  0.4995448 -0.1864500 -0.4694984 -0.1907888
```

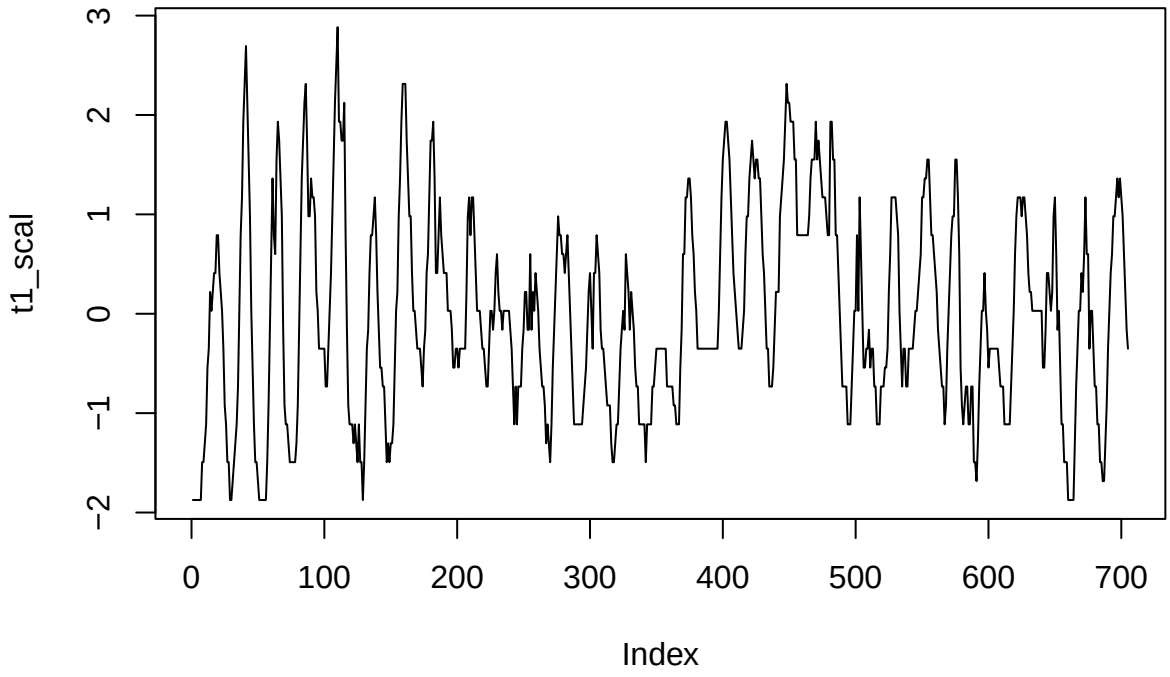
$$X_t = 1.5981X_{t-1} - 1.1204X_{t-2} + 0.4995X_{t-3} - 0.1865X_{t-4} + A_t - 0.4695A_{t-1} - 0.1908A_{t-2}$$

Try to introduce exogenous variables

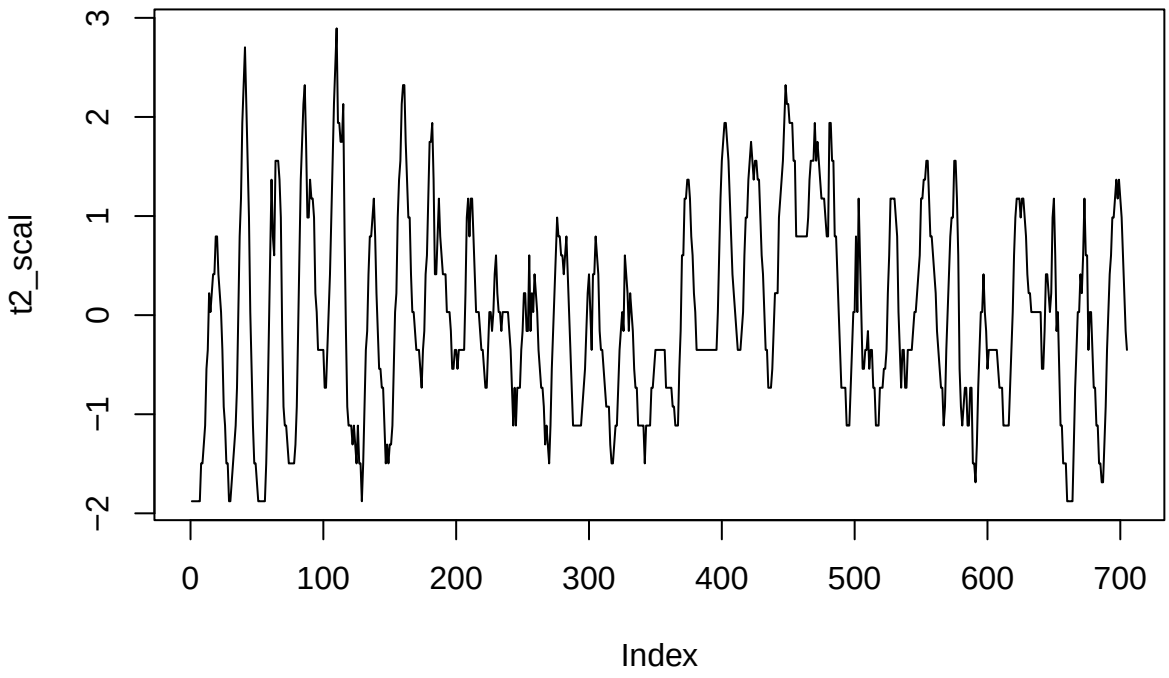
Taking into account the exogenous variables presented earlier is important when constructing the ARIMA model, as these variables can impact the model's goodness of fit and the accuracy of future forecasting. In this case, we are working with two specific variables: the exact temperature and the perceived temperature:

```
t1_scal <- scale(bike$t1)
t2_scal <- scale(bike$t2)
```

```r
plot(t1_scal, type = "l")
```




```r
plot(t2_scal, type = "l")
```




```
SARMAX4_with_t <- arima(y, order = c(4, 0, 1), seasonal = list(order = c(0, 1, 1), period = 24), xreg =
```

```
coef(SARMAX4_with_t)
```

```
##                       ar1                       ar2                       ar3
##                 1.5214737                -1.0357778                 0.4539862
##                       ar4                       ma1                      sma1
##                -0.1686904                -0.4145802                -0.1631596
## cbind(t1_scal, t2_scal)1 cbind(t1_scal, t2_scal)2
##             -756.2023551               948.8704368
```

$$X_t = 1.521X_{t-1} - 1.0358_{t-2} + 0.454X_{t-3} - 0.169X_{t-4} - 0.415A_{t-1} - 0.1631596A_{t-2} - 756.202Z_{t_1} + 948.870Z_{t_2}$$

Model comparison

Model comparison is a crucial aspect of time series analysis, aimed at selecting the most appropriate model for forecasting or understanding the underlying dynamics of the data. Time series data often exhibit complex patterns and structures, making it essential to evaluate different modeling approaches to capture these characteristics effectively. In our project, we will utilize a range of metrics for model comparison in time series analysis. Specifically, we will employ the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) to assess the goodness-of-fit and complexity of candidate models. Additionally, we will use the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to evaluate the accuracy of

model forecasts. These metrics will help us identify the most suitable model for our data, considering both goodness-of-fit and predictive performance.

```
library(knitr)

corrected_AIC <- function(AIC_value, k, n) {
  corr_AIC <- AIC_value + (2 * k * (k + 1)) / (n - k - 1)
  return(corr_AIC)
}

AIC2 <- AIC(SARMA2)
AIC3 <- AIC(SARMA3)
AIC4 <- AIC(SARMA4)
AICarx <- AIC(SARMAX4_with_t)

AICC2 <- corrected_AIC(10299.67, 4, 705)
AICC3 <- corrected_AIC(10301.03, 5, 705)
AICC4 <- corrected_AIC(10289.99, 6, 705)
AICCarx <- corrected_AIC(AICarx, 8, 705)


BIC2 <- BIC(SARMA2)
BIC3 <- BIC(SARMA3)
BIC4 <- BIC(SARMA4)
BICarx <- BIC(SARMAX4_with_t)

res2 <- SARMA2$residuals
res3 <- SARMA3$residuals
res4 <- SARMA4$residuals
resarx <- SARMAX4_with_t$residuals

RMSE2 <- sqrt(mean(res2^2))
MAE2 <- mean(abs(res2))
RMSE3 <- sqrt(mean(res3^2))
MAE3 <- mean(abs(res3))
RMSE4 <- sqrt(mean(res4^2))
MAE4 <- mean(abs(res4))
RMSEarx <- sqrt(mean(resarx^2))
MAEarx <- mean(abs(resarx))

model_comp = data.frame(model = c("SARMA2","SARMA3","SARMA4", "SARMAX"),
AIC = c(AIC2, AIC3, AIC4, AICarx),
AICC = c(AICC2, AICC3, AICC4, AICCarx),
BIC = c(BIC2, BIC3, BIC4, BICarx),
MAE = c(MAE2, MAE3, MAE4, MAEarx),
RMSE = c(RMSE2, RMSE3, RMSE4, RMSEarx))

kable(model_comp)
```

model	AIC	AICC	BIC	MAE	RMSE
SARMA2	10299.67	10299.73	10322.29	267.5563	453.5185
SARMA3	10301.03	10301.12	10328.17	267.4579	453.3085

model	AIC	AICC	BIC	MAE	RMSE
SARMA4	10289.99	10290.11	10321.66	267.6150	448.9159
SARMAX	10270.01	10270.22	10310.73	269.1212	441.1915

Note that the corrected AIC is commonly employed in situations with small sample sizes. In our scenario, as our sample size is sufficiently large, we anticipate similar values compared to the standard AIC. Examining the provided indices, we observe that the SARMAX model demonstrates the lowest AIC and BIC values, indicating superior model fit and selection. However, it's important to note that incorporating two exogenous variables in our SARMAX model results in a marginal decrease of 0.19%. This decrease, though small, prompts us to consider the balance between model complexity and predictive performance. Consequently, we have opted to conduct our predictions using the simpler SARIMA(4,0,1)(0,1,1)[24] model, excluding the exogenous variables.

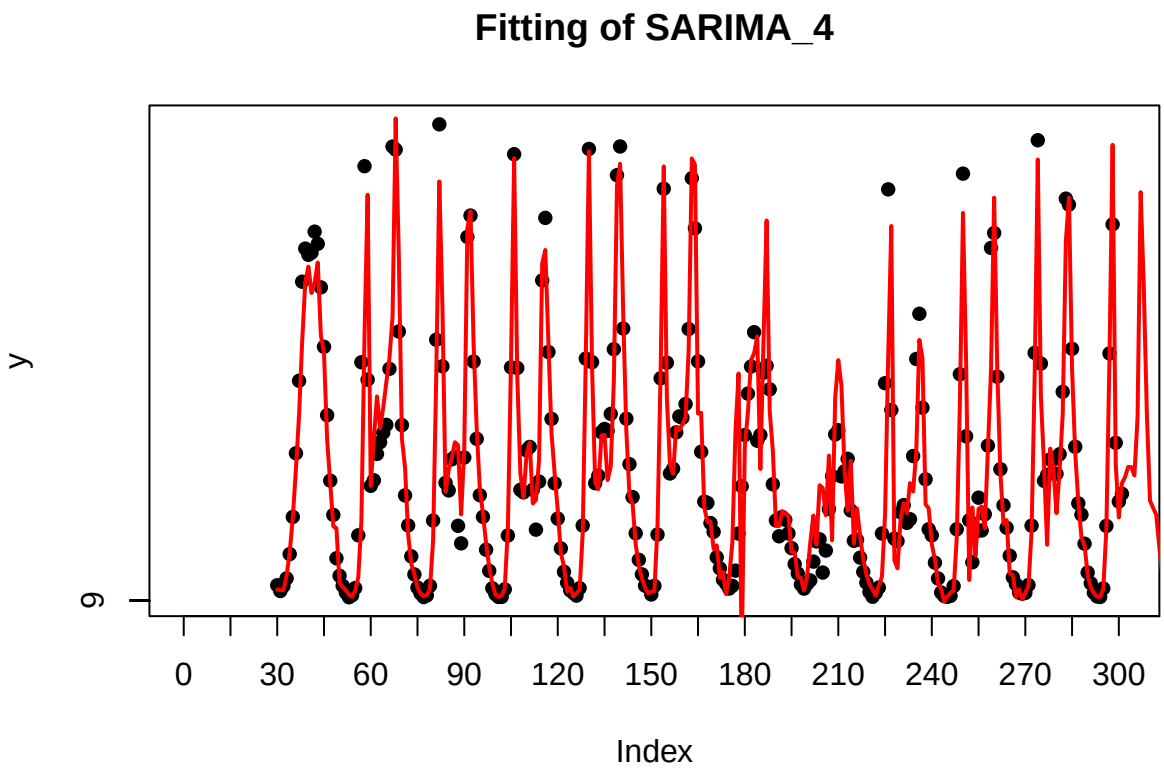
Model fitting

Our model is a SARMA(4,0,1) with a seasonal order of (0,1,1) and a seasonal period of 24. The number of observations we would need to consider before starting to plot the fitted values and residuals would be 29.

This is because the order of the autoregressive component is 4 (p=4), which means the model uses past values up to 4 time periods ago to predict the current value. Additionally, there are no moving average components (q=0) for the non-seasonal part of the model. As for seasonality, the model has a seasonal autoregressive component of order 0 (P=0) and a seasonal moving average component of order 1 (Q=1).

So, to ensure the model has enough data to make accurate predictions, we would need to consider the maximum of the seasonal period (24) and the order of the autoregressive component (4) plus 1 (to include the current forecasting period). Therefore, we would need 29 observations before starting to plot the fitted values and residuals. This ensures the model has enough data to accurately calculate both autoregressive and seasonal components.

```
n = 705
y.fitted = c(rep(NA,29),fitted(SARMA4)[29:n])
y.residual = c(rep(NA,29),residuals(SARMA4)[29:n])
y.plot = c(rep(NA,29),y[29:300])
x_val=c(0,(1:24)*15)
y_val=(seq(9,11,.25))
plot(y.plot,main="Fitting of SARIMA_4",xlab="Index",
ylab="y",
xaxt="n",yaxt="n",type="p",lwd=2,pch=16)
axis(1,x_val,x_val)
axis(2,y_val,y_val)
lines(y.fitted,col="red",lwd=2)
```



This plot illustrates the goodness of the SARMA(4,1) model in comparison to the actual values, where the red line represents the fitted values and the black dots represent the actual values. As observed, our model appears to be reasonably accurate in capturing our data, as evidenced by its ability to reflect the daily trend present in our dataset.

```r
library(knitr)

true_values <- c()
fitted_values <- c()

specific_dates <- c("2016-06-07 12:00:00", "2016-06-07 13:00:00","2016-06-07 14:00:00", "2016-06-07 15:

for (date in specific_dates) {
  true_value <- bike[bike$timestamp == date, "cnt"]
  fitted_value <- fitted(SARMA4)[bike$timestamp == date]
  true_values <- c(true_values, true_value)
  fitted_values <- c(fitted_values, fitted_value)
}

results <- data.frame(
  "Date" = specific_dates,
  "True Value" = true_values,
  "Fitted Value" = fitted_values
)
```

```
kable(results)
```

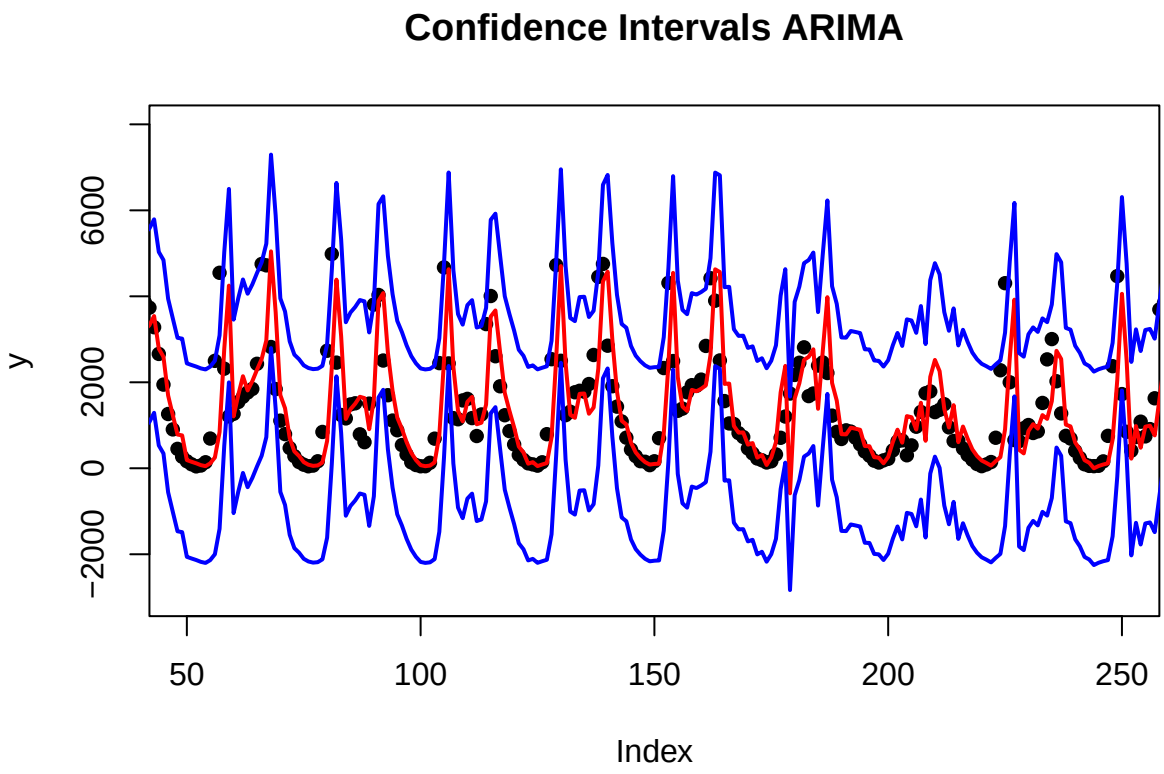
Date	True.Value	Fitted.Value
2016-06-07 12:00:00	1485	1500.8330
2016-06-07 13:00:00	1511	1663.2829
2016-06-07 14:00:00	790	1632.2554
2016-06-07 15:00:00	607	909.1486
2016-06-07 16:00:00	1504	1599.2708
2016-06-07 17:00:00	3808	3897.3475
2016-06-07 18:00:00	4031	4075.1342
2016-06-07 19:00:00	2506	2689.8627
2016-06-07 20:00:00	1699	1800.5734
2016-06-07 21:00:00	1110	1178.4763
2016-06-07 22:00:00	884	917.8185
2016-06-07 23:00:00	540	608.7670
2016-06-08 00:00:00	320	359.2365

In general, upon reviewing the table above, it appears that the predicted values align quite well with the true values in some instances, but there are also significant discrepancies in others. For example, for certain observations such as those at 1:00 PM and 5:00 PM, the predicted values are considerably higher than the true values, while for others like those at 3:00 PM and 11:00 PM, the predicted values are notably lower.

This suggests that the model employed may struggle to accurately capture the data variation under certain circumstances. Maybe further scrutiny of the model, conducting residual analysis, or considering the use of more sophisticated or adaptive models may be necessary to enhance even better predictions.

Confidence interval

```
left_bound=rep(NA,times=n)
for (t in 29:n) {
   left_bound[t]=y.fitted[t]-qnorm(.975)*sd(y, na.rm = TRUE)
}

right_bound=rep(NA,times=n)
for (t in 29:n) {
   right_bound[t]=y.fitted[t]+qnorm(.975)*sd(y, na.rm = TRUE)
}


plot(y,main="Confidence Intervals ARIMA",xlab="Index",ylab="y",xlim=c(50,250), ylim = c(-3000,8000),
type="p",lwd=2, pch=16)
lines(y.fitted,col="red",lwd=2)
lines(left_bound,col="blue",lwd=2)
lines(right_bound,col="blue",lwd=2)
```



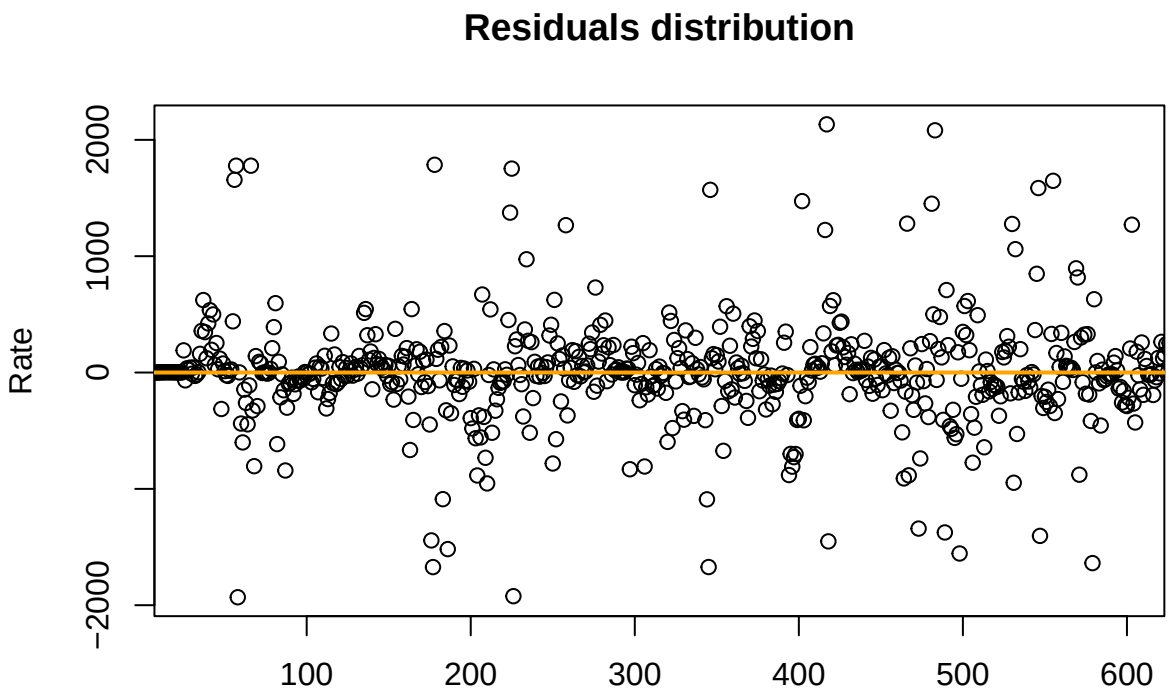
This code creates a plot that displays the observed values of the time series y along with the fitted values from the ARIMA model, as well as the 95% confidence intervals for the fitted values. This way, the plot provides a visualization of the observed values, fitted values, and associated confidence intervals, aiding in assessing the goodness of fit of the ARIMA model relative to the observed data. We see that, since the fitting seems to be good, a 95% confidence interval contains all the true values for our series.

Model diagnostic

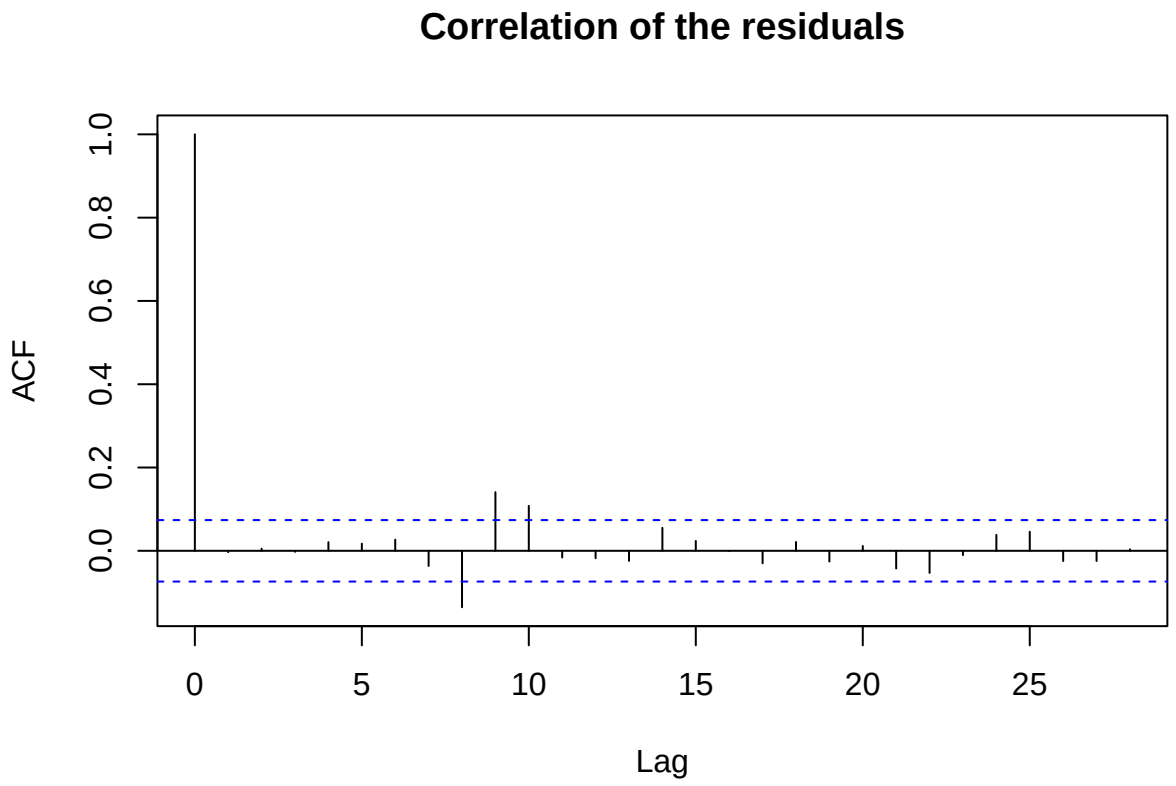
Independence and correlation of the residuals

After fitting the model, it is crucial to conduct diagnostic to ensure the validity of the model. Model diagnostics, in fact, is concerned with assessing the quality of the model that we have specified and estimated. To do so we start with the study of the residuals. Residual analysis is a common diagnostic tool, involving the examination of model residuals for patterns or systematic errors.

```
res=SARMA4$residuals
n = 705
# Create residuals plot
plot(res, main = "Residuals distribution", xlab = "", ylab = "Rate", xlim = c(30, 600),type = "p")

lines(rep(0, times = n), col = "orange", lwd = 2)
```



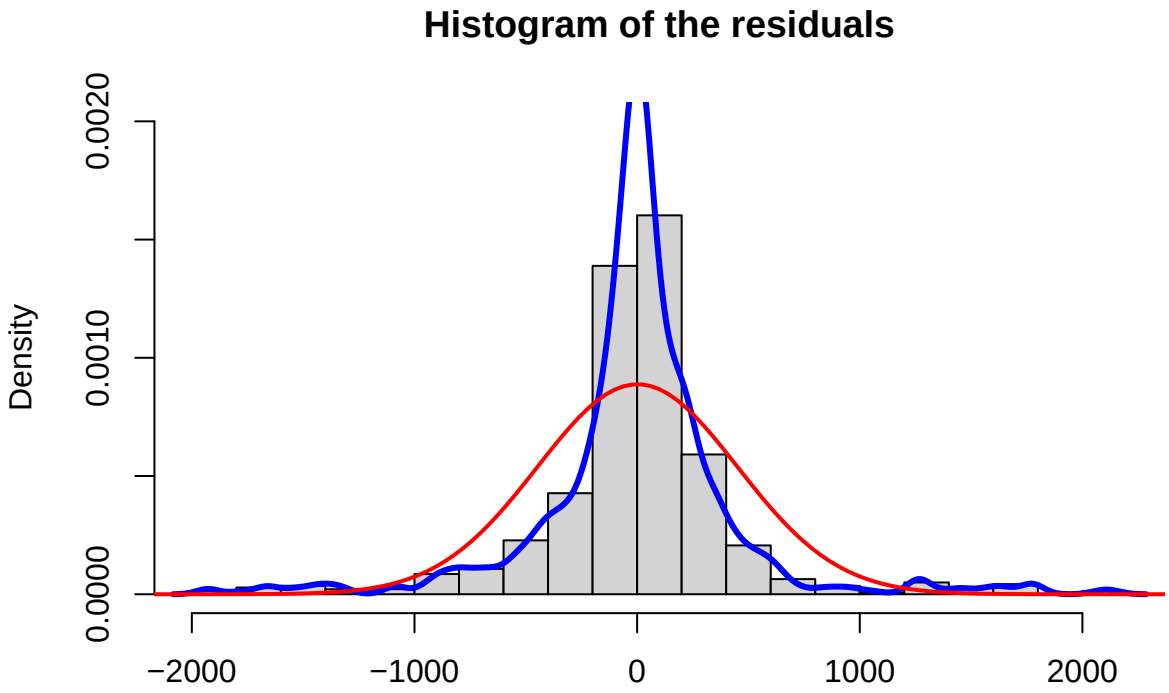

```
acf(res, main = "Correlation of the residuals")
```



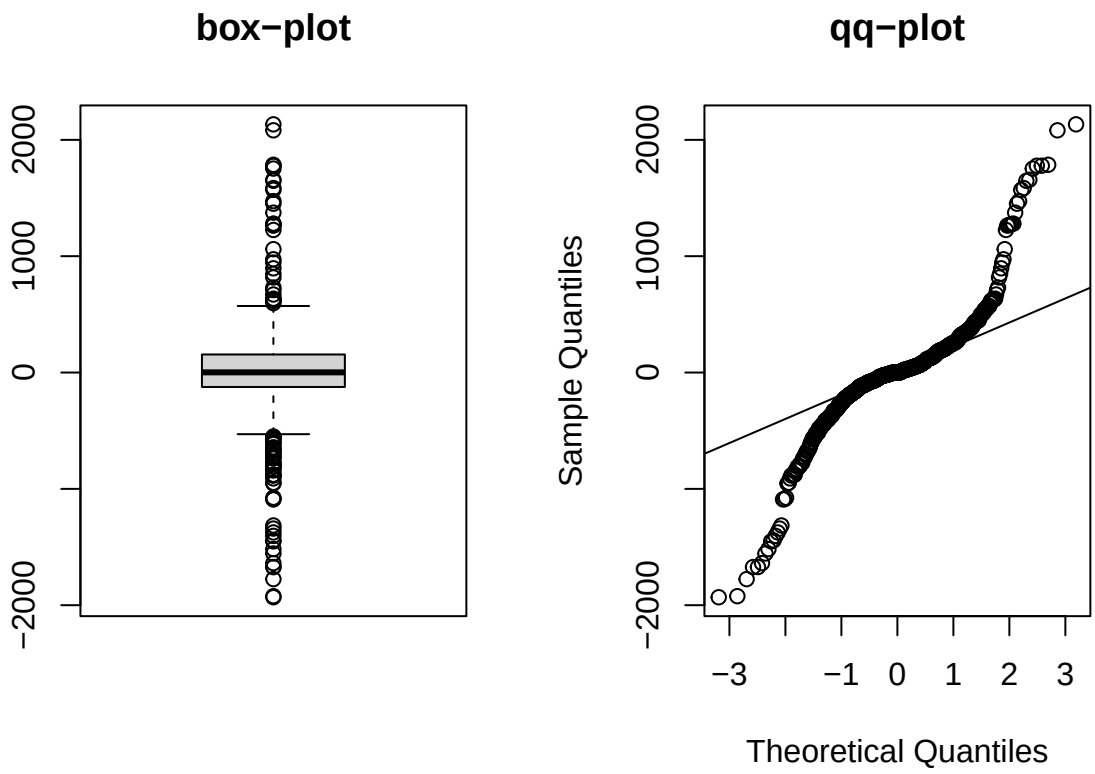
These plots are useful for examining the behavior of the residuals over time. In the first plot, the orange line at y = 0 serves as a reference for identifying patterns or trends in the residuals. Patterns in the residuals could indicate that the model is not capturing all the information in the data, suggesting areas for improvement. As we can observe, the residuals are evenly distributed around the y = 0 axis and the values seems to be independent. Therefore, no transformation is necessary on them. In the second plot, it is visible from the ACF that the residuals are mostly uncorrelated.

Normality assumption

```
hist(res[4:n],main="Histogram of the residuals",xlab="",freq=F,breaks=20, ylim = c(0, 0.0020))
lines(density(res[4:n]),col="blue",lwd=3)
zz=seq(-4000, 4000,length=200)
f.zz=dnorm(zz,mean(res, na.rm = TRUE),sd(res, na.rm = TRUE))
lines(zz,f.zz,col="red",lwd=2)
```




```
par(mfrow=c(1,2))
boxplot(res,main="box-plot")
qqnorm(res,main="qq-plot")
qqline(res)
```



From the boxplot we can state that the distribution of the residuals is symmetric but not normal, it is much more concetrated in the middle with respect to the normal distribution.

```
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.85364, p-value < 2.2e-16
```

The Shapiro-Wilk normality test is another test commonly used to assess the normality of residuals in a statistical model. In this case, it's applied to the residuals (res) of the SARMA(4) model. The extremely low p-value (much smaller than the commonly used significance level of 0.05) indicates strong evidence against the null hypothesis of normality. Therefore, we reject the null hypothesis and conclude that the residuals are not normally distributed according to the Shapiro-Wilk test.

```
jarque.bera.test(res)
```

```
##
##  Jarque Bera Test
##
## data:  res
## X-squared = 1020.7, df = 2, p-value < 2.2e-16
```

The Jarque Bera Test is a test for normality of residuals in a statistical model. In this case, it's applied to the residuals (res) of the SARMA(4) model. The extremely low p-value (much smaller than the commonly used significance level of 0.05) indicates strong evidence against the null hypothesis of normality. Therefore, we reject the null hypothesis and conclude that the residuals are not normally distributed. This suggests that the SARMA(4) model may not fully capture the underlying distribution of the data, and there may be room for improvement in the modeling approach or the inclusion of additional variables. Further diagnostics and model refinement may be necessary to address this issue.

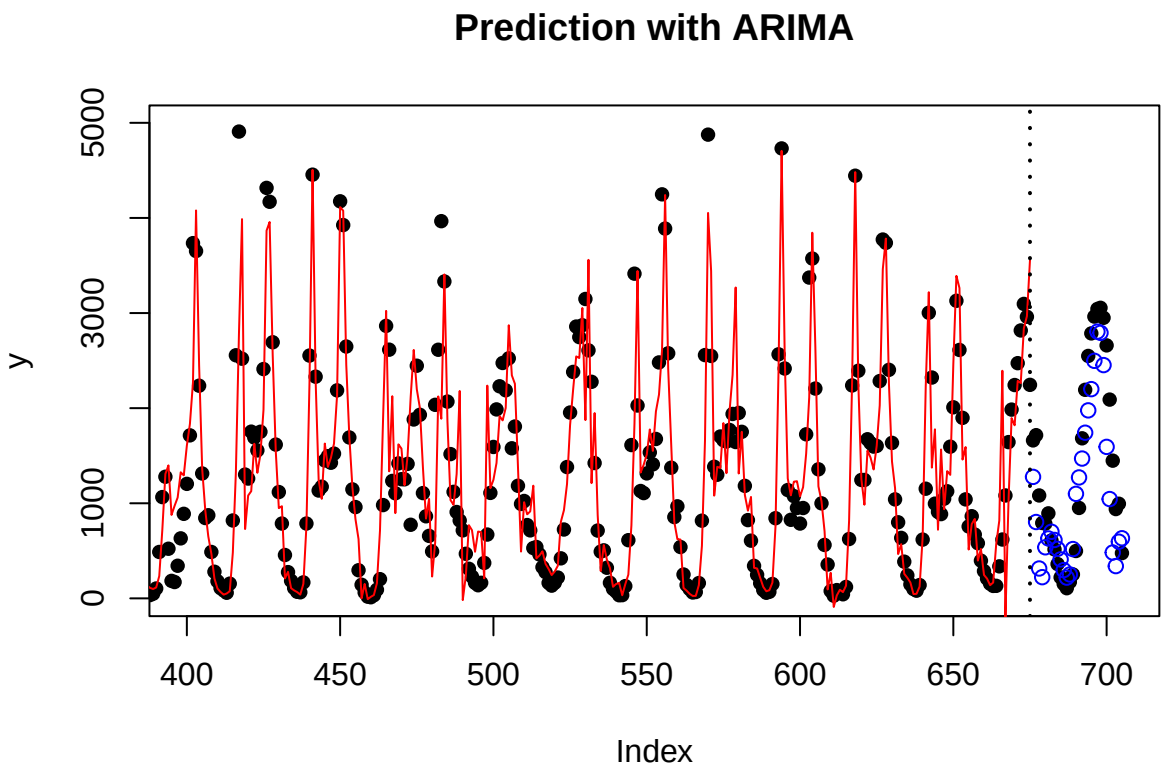
Forecasting

Forecasting plays a pivotal role in Time Series Analysis, serving as a fundamental technique for predicting future values based on historical data patterns. Time series data represents observations collected sequentially over time, where the data points are often dependent on previous observations. This temporal dependency necessitates specialized methods for modeling and predicting future values, in our case, a SARMA model.

Prediction using the SARMA model

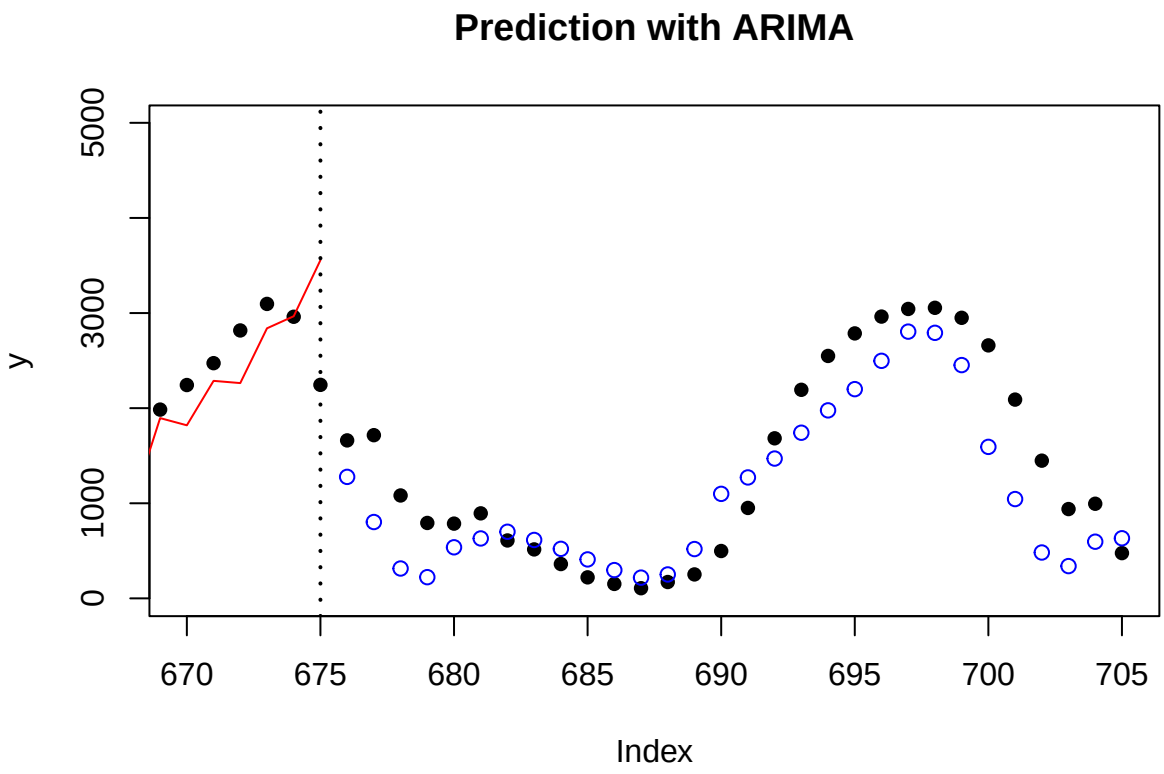
```
# Original number of observations
n <- 705
nn <- n
# Number of observations used for training
pr <- 30
n <- nn - pr

# Training the SARIMA model
SARMA_forecast <- arima(y[1:n], order = c(4, 0, 1),
                        seasonal = list(order = c(0, 1, 1), period = 24), method = "ML")

# Calculating the fitted values
y.fitted <- c(rep(NA, 28), fitted(SARMA_forecast)[29:n])

# Forecasting future values
y.for <- c(rep(NA, n), forecast(SARMA_forecast, h = pr)$mean)
```

```
plot(y, pch = 16, main = 'Prediction with ARIMA', xlim= c(400,705) )
lines(y.fitted, type = 'l', col = 'red')
lines(y.for, type = 'p',col = 'blue')
abline(v = n, col = 'black', lty = 3, lwd = 2)
```



The resulting plot shows the original time series (y), the fitted values (predictions on the training data) in red, and the future forecasts in blue. Additionally, a black vertical line is drawn to separate the training data from the forecasted data. The original time series exhibits some seasonality and short-term variations. The fitted values closely follow the original time series, indicating a good fit to the training data. The future forecasts extend beyond the training data, following the general pattern of the time series. However, as the forecasts move away from the training data, the uncertainty in the predictions increases. Overall, the plot suggests that the SARIMA model captures the main characteristics of the time series. However, it's important to carefully assess the validity of the future forecasts, particularly as they extend beyond the training data. Further evaluation of the model's accuracy and the uncertainty of future forecasts would be beneficial for a better understanding of the model's performance.

```
plot(y, xlim = c(670,705),pch = 16, main = 'Prediction with ARIMA' )
lines(y.fitted, type = 'l', col = 'red')
lines(y.for, type = 'p',col = 'blue')
abline(v = n, col = 'black', lty = 3, lwd = 2)
```



This plot provides a focused view of the prediction within the specified range of the time series.

Prediction confidence interval

To compute the confidence intervals, we first need to calculate the Psi vector, which is used to determine the variance of the forecast errors. The Psi vector, initialized with a length equal to the number of forecast periods (pr), starts with the first element set to 1. Subsequent elements of the Psi vector are computed using the autoregressive (AR) and moving average (MA) coefficients obtained from the SARIMA model.

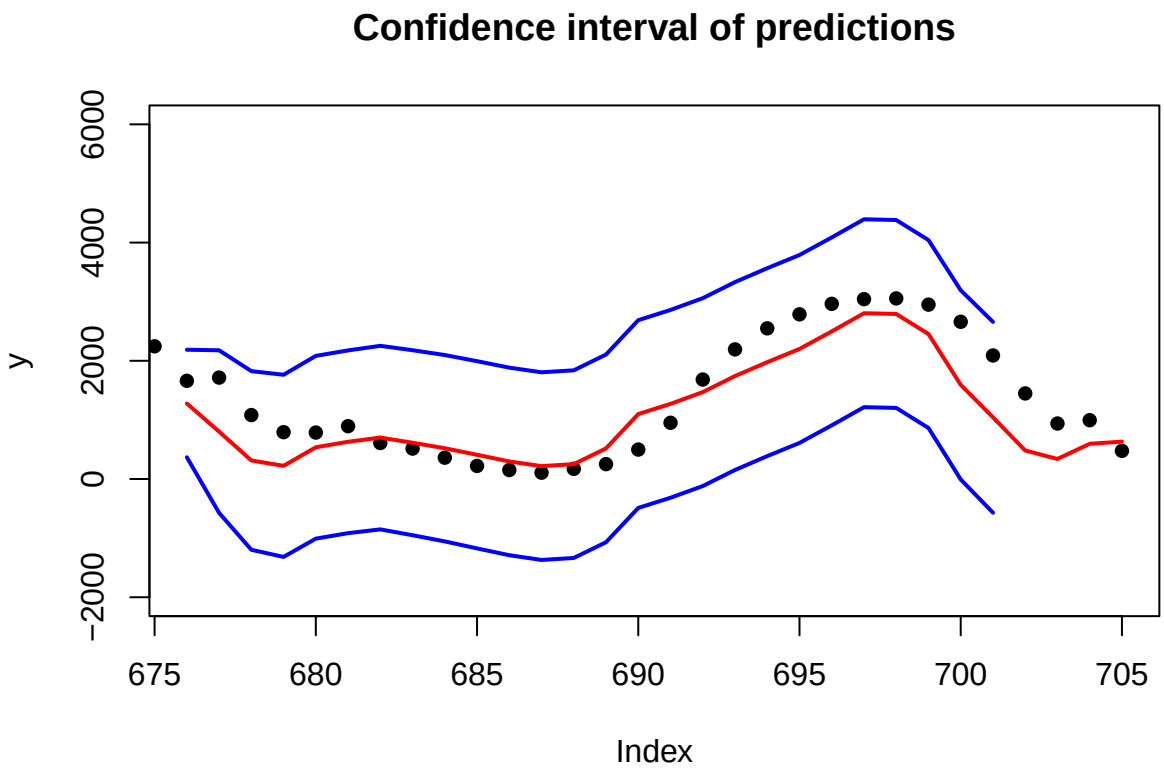
Next, we calculate the variance of the errors at each forecasting step. This is achieved by utilizing the Psi coefficients along with the standard deviation of errors derived from the trained SARIMA model. The variance of errors is computed for each forecasting step and stored in the var.er vector.

```
phi = SARMA_forecast$model$phi ; theta = SARMA_forecast$model$theta

psi = rep(NA,pr)
psi[1] = 1

for(j in 2:pr){
       candidate = 0
       for(p in 1:length(phi)){
               if(j > p){
                  candidate = candidate + psi[j-p] * phi[p]
   }
}
       candidate=candidate+theta[j-1]
```

```
       psi[j]=candidate
}
var.er=rep(NA,times=pr)
sigma.square = summary(SARMA_forecast)$sigma
var.er=rep(0,times=pr)
var.er[1]=sigma.square

for (j in 2:pr) {
 var.er[j]=sigma.square*(psi[1]+sum(psi[2:(j)]^2))
}
```

Plot the results

```
left_ci.er=rep(NA,times=nn)

for (t in (n+1):nn) {
        left_ci.er[t]=y.for[t]-qnorm(.975)*sqrt(var.er[t-n])
}

right_ci.er=rep(NA,times=nn)

for (t in (n+1):nn) {
       right_ci.er[t]=y.for[t]+qnorm(.975)*sqrt(var.er[t-n])
}

plot(y,xlim=c(n+1,nn),type="p",pch=16,ylim=c(-2000, 6000),
main='Confidence interval of predictions',ylab='y')
lines(y.for,type="l",col="red",lwd=2)
lines(left_ci.er,type="l",col="blue",lwd=2)
lines(right_ci.er,type="l",col="blue",lwd=2)
```



The ability to construct and interpret confidence intervals is fundamental in time series analysis. Confidence intervals provide insights into the accuracy of model predictions and the expected variability in future data. In the context of time series forecasting with models like SARIMA, confidence intervals allow us to assess how reliable the predictions are and how they may vary over time. In this specific case, we observe that our 95% confidence interval consistently contains all actual values of the time series. This suggests that the SARIMA model accurately captures the data variability and provides reliable estimates.

In summary, monitoring confidence intervals not only provides a measure of model prediction accuracy but also indicates the expected variability in future data, thereby assisting analysts in making informed decisions based on forecasts.