IE 360 HW 4-5 - Developing alternative strategies for forecasting how many products will Trendyol sell each day

Yunus Emre Erdogan - Ugur Uzunoglu - IE360 - Spring 2021

Introduction

The aim of this study is to develop and assess different methods for predicting Trendyol's product sales amount of nine different products.

Data Reading & Preprocessing

Importing necessary libraries

```
library(jsonlite)
library(httr)
library(ellipsis)
library(stats)
library(zoo)
library(plotly)
library(ggplot2)
library(readxl)
library(dplyr)
library(tidyr)
library(hrbrthemes)
library(lubridate)
library(data.table)
library(forecast)
library(fpp)
library(xts)
library(tidyverse)
library(caret)
library(leaps)
library(gridExtra)
library(GGally)
library(urca)
library(plyr)
library(gridExtra)
library(forecast)
library(ggcorrplot)
```

Target variable and training variables is gathered as an csv file from the data Trendyol provided:

```
data_path='ProjectRawData.csv'
trendyol=fread(data_path)
trendyol$event_date=as.Date(trendyol$event_date)
```

```
str(trendyol)
```

```
## Classes 'data.table' and 'data.frame':   4331 obs. of  13 variables:
##  $ event_date         : Date, format: "2021-05-31" "2021-05-31" ...
##  $ product_content_id : int  85004 4066298 6676673 7061886 31515569 32737302 32939029 48740784 733185
##  $ price              : num  87.4 65.8 120.7 294.4 60.3 ...
##  $ sold_count         : int  80 1398 345 24 454 44 125 2 175 85 ...
##  $ visit_count        : int  6002 19102 19578 1485 20114 5017 5301 221 20256 5265 ...
##  $ basket_count       : int  744 5096 1261 81 2871 316 608 15 965 633 ...
##  $ favored_count      : int  1642 1703 1510 122 2102 607 698 22 1960 1104 ...
##  $ category_sold      : int  5048 6547 4944 951 8155 5198 930 1684 5198 4117 ...
##  $ category_visits    : int  236197 108811 306462 95645 637143 1010634 41973 329087 1010634 215260 .
##  $ category_basket    : int  33681 28558 23418 4704 49389 33728 3911 12614 33728 25181 ...
##  $ category_favored   : int  40472 11913 26597 8886 62460 96699 5791 24534 96699 36225 ...
##  $ category_brand_sold: int  743 4286 786 179 1759 3665 875 12 3665 430 ...
##  $ ty_visits          : int  125439876 125439876 125439876 125439876 125439876 125439876 125439876 1
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
trendyol = trendyol[!is.na(trendyol$event_date)]
trendyol = trendyol[!(ty_visits == 1)]

trendyol$price[trendyol$price == -1 ] <- 0
trendyol$price[is.na(trendyol$price)] <- 0


trendyol[,wday:=weekdays(trendyol$event_date)]
trendyol[,mon:=months(trendyol$event_date)]
```

Missing values containing no date information and are disregarded. Also, data points in 'ty_visits' column with '-1' value does not match with the majority of the data; therefore, they are eliminated too. ## 2. Ploting sales information of all products

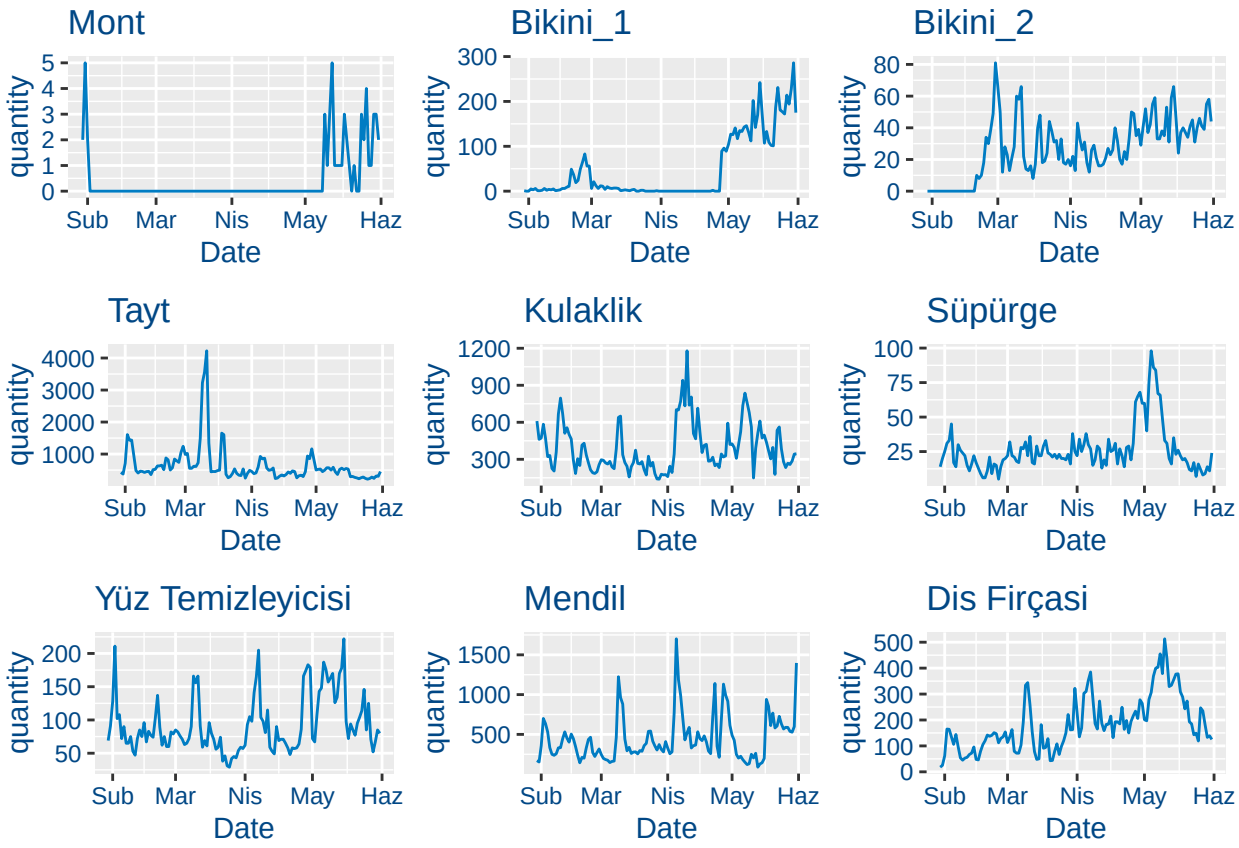
In this homework, we are dealing with 9 different products. Assigning different dataframes for all nine products:

```
mont = trendyol[product_content_id=="48740784"]
bikini_1 = trendyol[product_content_id=="73318567"]
bikini_2 = trendyol[product_content_id=="32737302"]
tayt = trendyol[product_content_id=="31515569"]
kulaklık = trendyol[product_content_id=="6676673"]
supurge = trendyol[product_content_id=="7061886"]
yuz_temizleyicisi = trendyol[product_content_id=="85004"]
mendil = trendyol[product_content_id=="4066298"]
dis_fircasi = trendyol[product_content_id=="32939029"]
```

Time series visualization of 9 different products can be seen below:

```
par(mfrow=c(3,3))
a<-ggplot(data = mont, aes(x = event_date, y = sold_count ,group=1))+
  geom_line(color = '#007cc3') + labs(title = 'Mont', x= "Date", y = "quantity") + theme(text=element_t

b<-ggplot(data = bikini_1, aes(x = event_date, y = sold_count ,group=1))+
```

```
  geom_line(color = '#007cc3') + labs(title = 'Bikini_1', x= "Date", y = "quantity") + theme(text=elemer

c<-ggplot(data = bikini_2, aes(x = event_date, y = sold_count ,group=1))+
  geom_line(color = '#007cc3') + labs(title = 'Bikini_2', x= "Date", y = "quantity") + theme(text=elemer

d<-ggplot(data = tayt, aes(x = event_date, y = sold_count ,group=1))+
  geom_line(color = '#007cc3') + labs(title = 'Tayt', x= "Date", y = "quantity") + theme(text=element_te

e<-ggplot(data = kulaklık, aes(x = event_date, y = sold_count ,group=1))+
  geom_line(color = '#007cc3') + labs(title = 'Kulaklık', x= "Date", y = "quantity") + theme(text=elemer

f<-ggplot(data = supurge, aes(x = event_date, y = sold_count ,group=1))+
  geom_line(color = '#007cc3') + labs(title = 'Süpürge', x= "Date", y = "quantity") + theme(text=elemen

g<-ggplot(data = yuz_temizleyicisi, aes(x = event_date, y = sold_count ,group=1))+
  geom_line(color = '#007cc3') + labs(title = 'Yüz Temizleyicisi', x= "Date", y = "quantity") + theme(te

h<-ggplot(data = mendil, aes(x = event_date, y = sold_count ,group=1))+
  geom_line(color = '#007cc3') + labs(title = 'Mendil', x= "Date", y = "quantity") + theme(text=element_

j<-ggplot(data = dis_fircasi, aes(x = event_date, y = sold_count ,group=1))+
  geom_line(color = '#007cc3') + labs(title = 'Diş Fırçası', x= "Date", y = "quantity") + theme(text=ele

grid.arrange(a,b,c,d,e,f,g,h,j ,
             ncol = 3, nrow = 3)
```



The initial visual inspection is that some of the products such as Bikini_2 has monthly seasonality. Furthermore, some products are not sold not at all in certain periods, such as 'mont' product is not sold at all between March and May, indicating no stock is available in Trendyol's website. To interpret the data more carefully, seasonality analysis for different time periods should be done for each product separately.

Analyzing the seasonality of the sales for each 9 product

Weekly decomposition will be done for products, since yearly and hourly decomposition could not be performed due to granularity of the data. Furthermore, due to selecting data for only 5 months, monthly seasonality couldn't be done too.

ACF will give information about the 'q' parameter of the ARIMA model, i.e the moving average part. Furthermore, PACF will give information about the 'p' parameter, i.e the autoregression part. The most probable 'p' and 'q' parameter will be selected by ACF and PACF analysis and 5 ARIMA models will be tried to find best ARIMA model for each product.

For all products, KPSS test, ACF, PACF and seasonality analysis will be done with the function below:

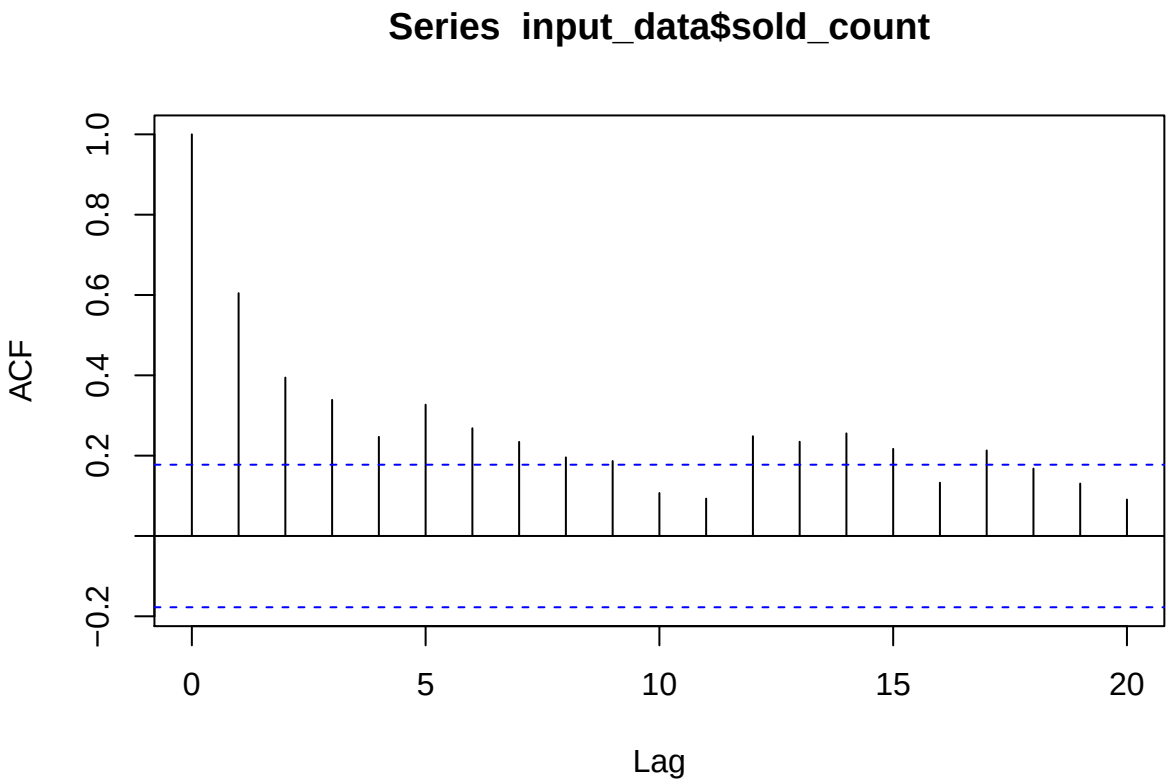
```
Seasonality_Analysis <- function(X){
  input_data = X
  len = length(input_data$price)
  KPSS_test= ur.kpss(input_data$sold_count)
  print(summary(KPSS_test))
  acf(input_data$sold_count)
  pacf(input_data$sold_count)
  #weekly decomposition
  df_new <- ts(input_data$sold_count,frequency=7)
  df_additive <- decompose(df_new,type="additive")
  plot(df_additive)

}
```

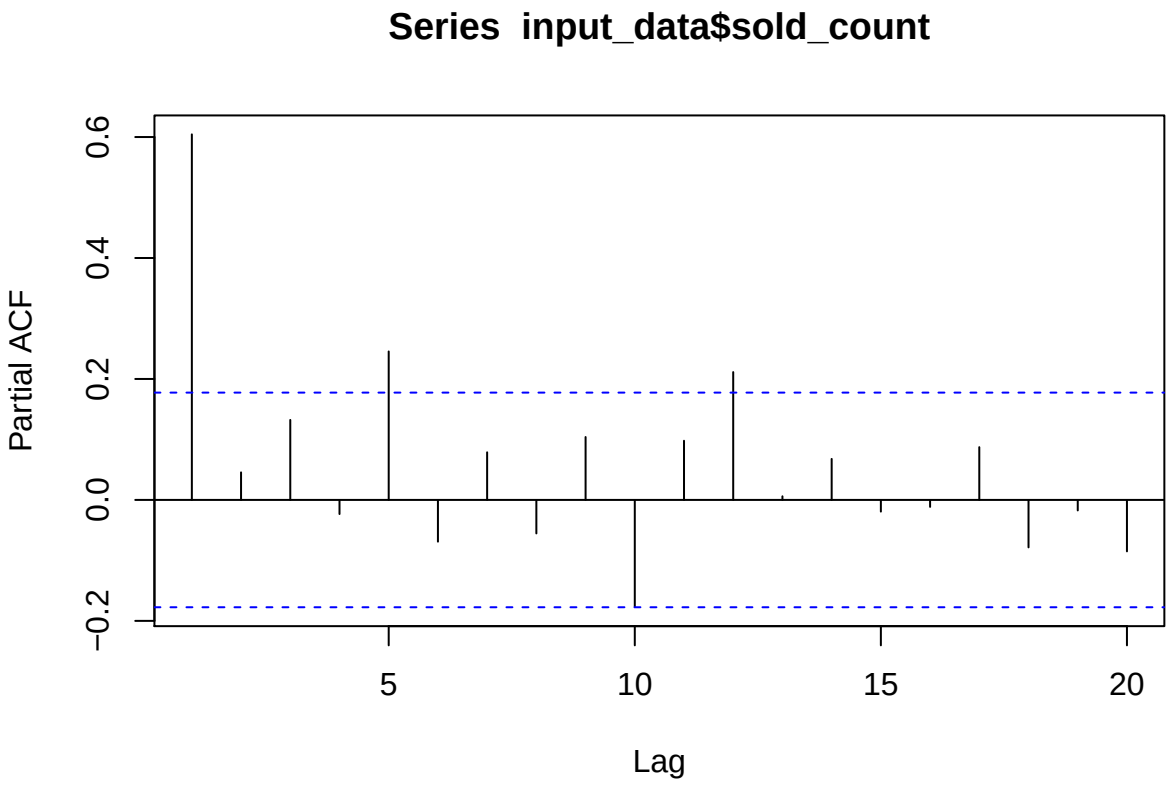
Seasonality analysis for all nine Products

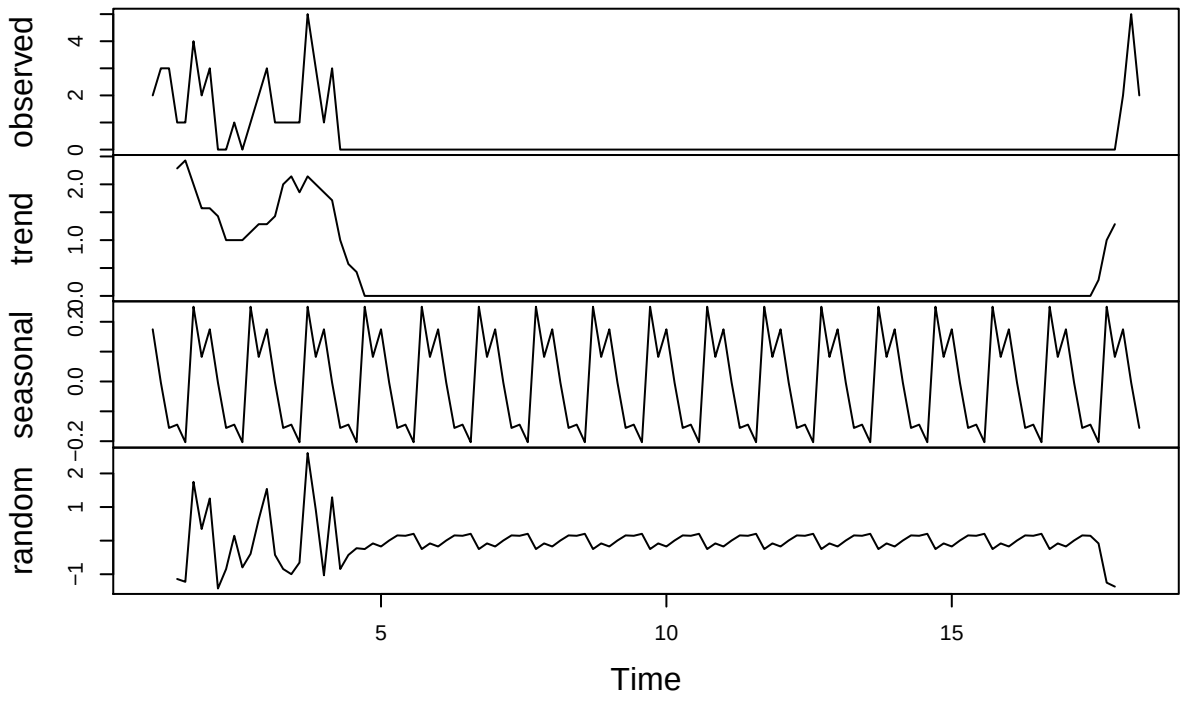
Seasonality analyis for 'Mont Product':

```
Seasonality_Analysis(mont)
```

```
##
## ######################
## # KPSS Unit Root Test #
## ######################
##
## Test is of type: mu with 4 lags.
##
## Value of test-statistic is: 0.7966
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```


Series input_data$sold_count
ACF
1.0
0.8
0.6
0.4
0.2
−0.2
0
5
10
15
20
Lag


Series input_data$sold_count
Partial ACF
0.6
0.4
0.2
0.0
−0.2
5
10
15
20
Lag

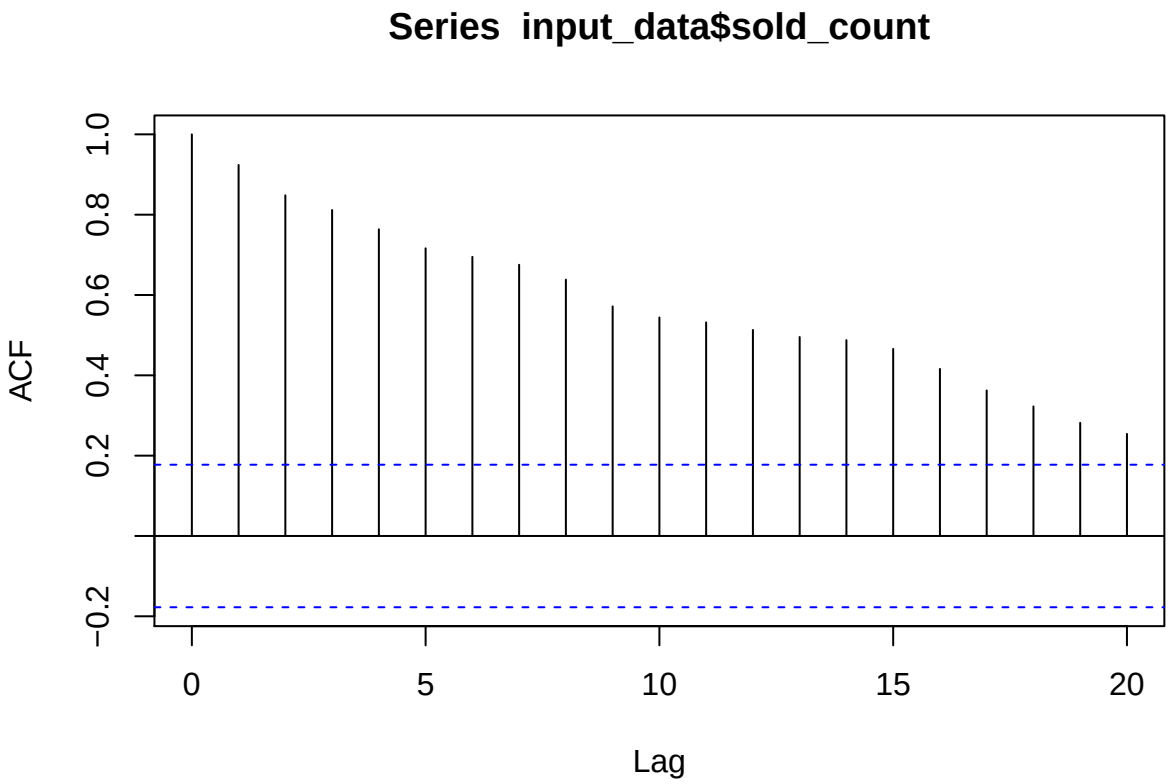


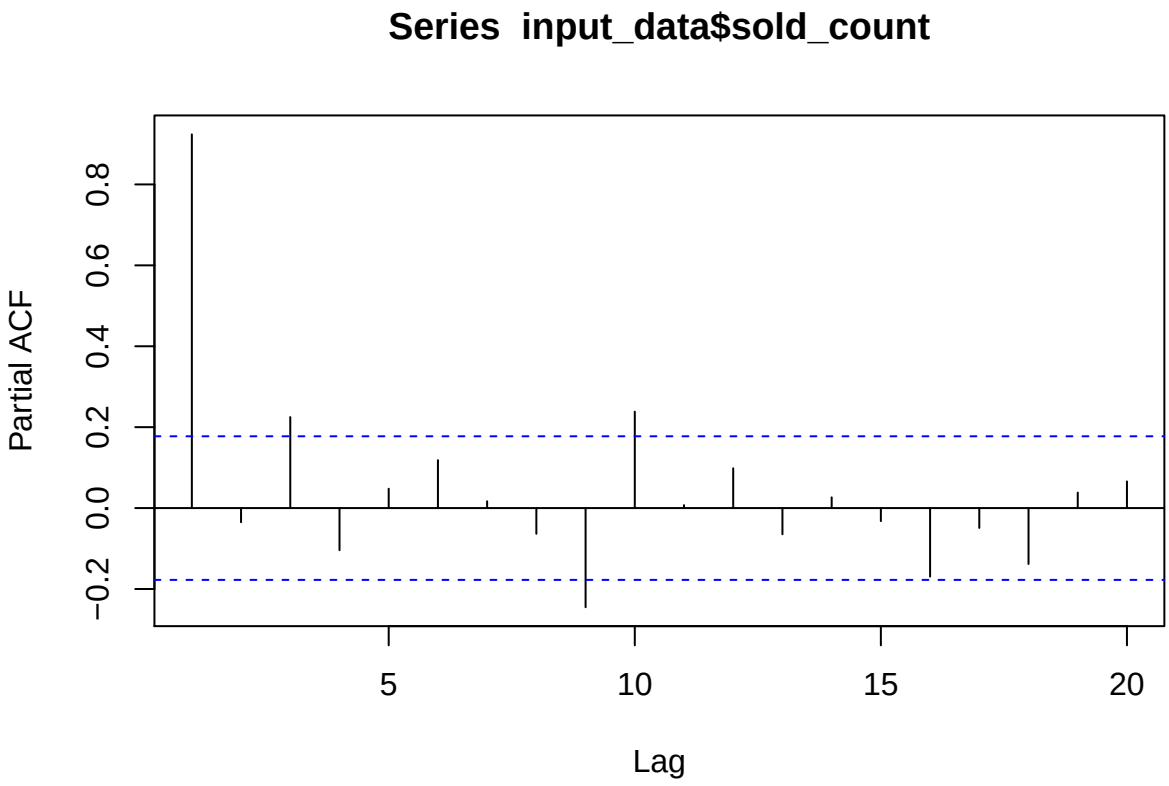
For the 'Mont' product, the null hypothesis of the KPSS test is that the data is stationary, and the value of test-statistic exceeds critical values. Therefore, the null hypothesis is rejected and can be concluded that the data is non-stationary. The same conclusion logic will be applied for the rest of the products. For the seasonality, the weekly decomposition shows that the data has weekly seasonal component clearly. Random, trend and the observed data points could not give precise information, since the 'sold_count' is zero for at least 3 months, presumably due to 'out of stock' status of the product. Best 'p' and 'q' parameters are 1 and 1-2-3, respectively.

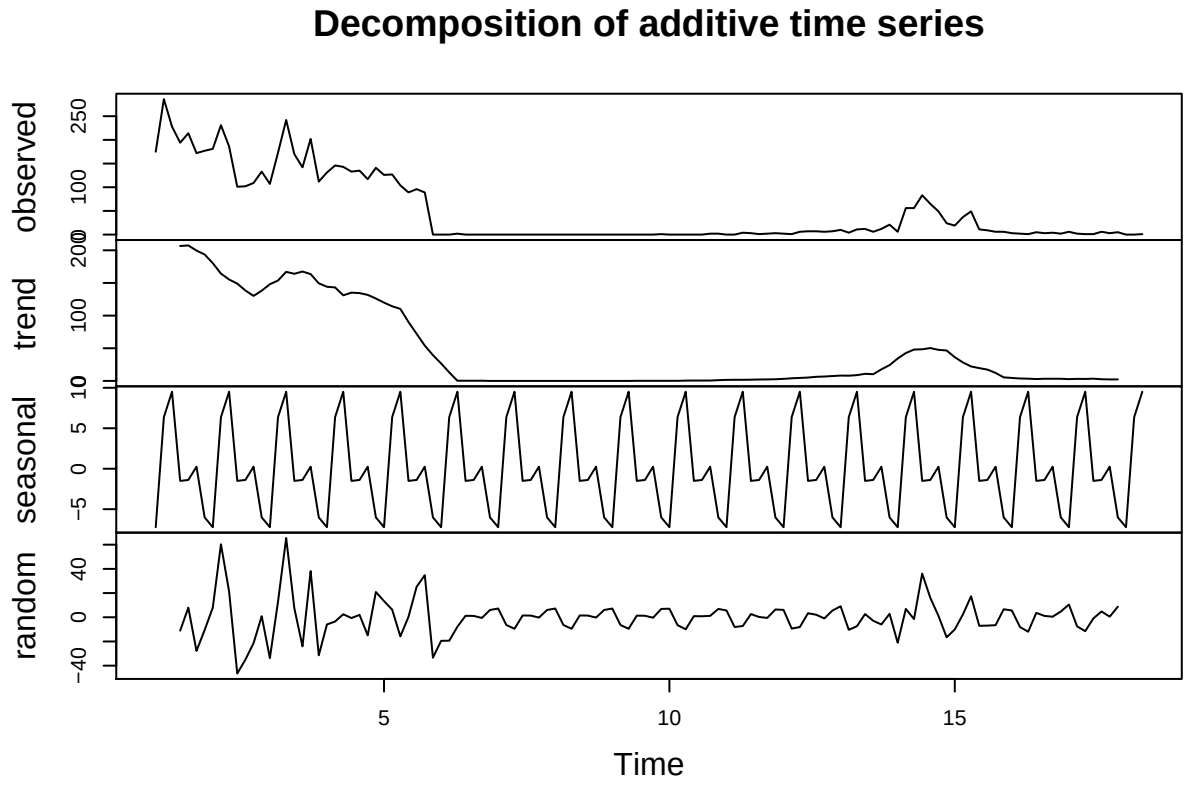
Seasonality analyis for 'Bikini_1 Product':

```
Seasonality_Analysis(bikini_1)
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 4 lags.
##
## Value of test-statistic is: 1.5333
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```


Series input_data$sold_count
ACF
1.0
0.8
0.6
0.4
0.2
−0.2
0
5
10
15
20
Lag

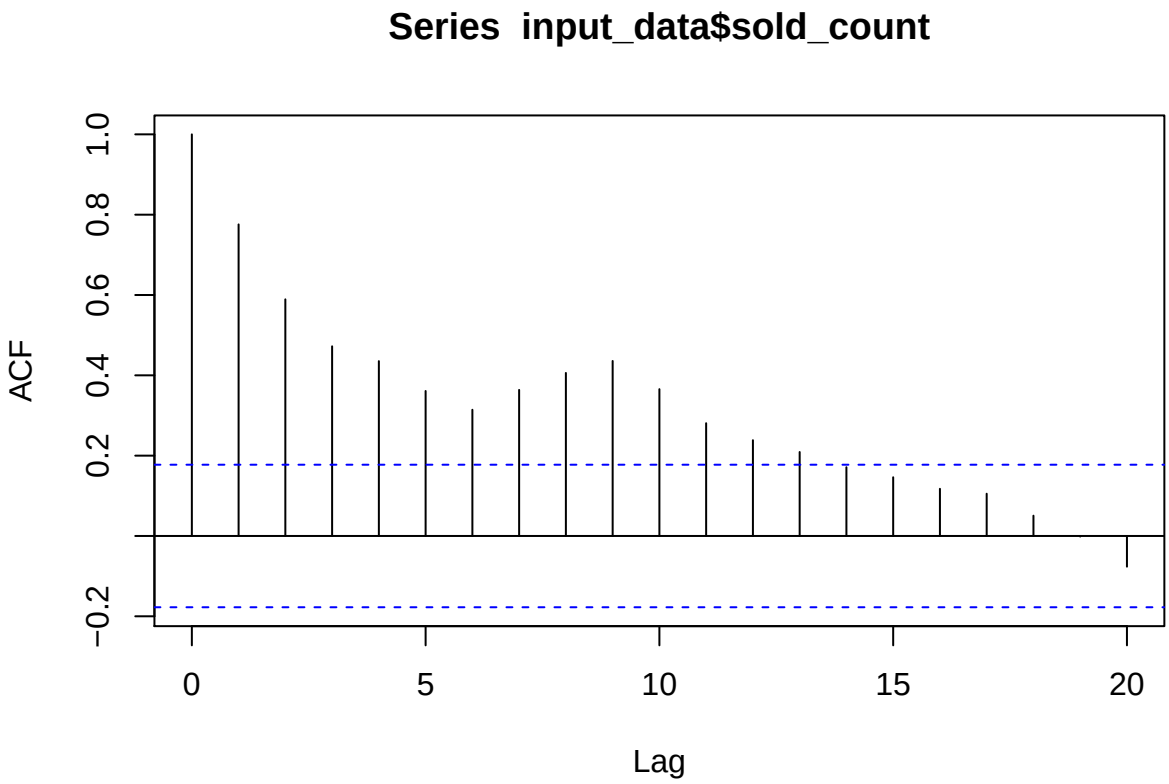

Series input_data$sold_count
Partial ACF
0.8
0.6
0.4
0.2
0.0
−0.2
5
10
15
20
Lag

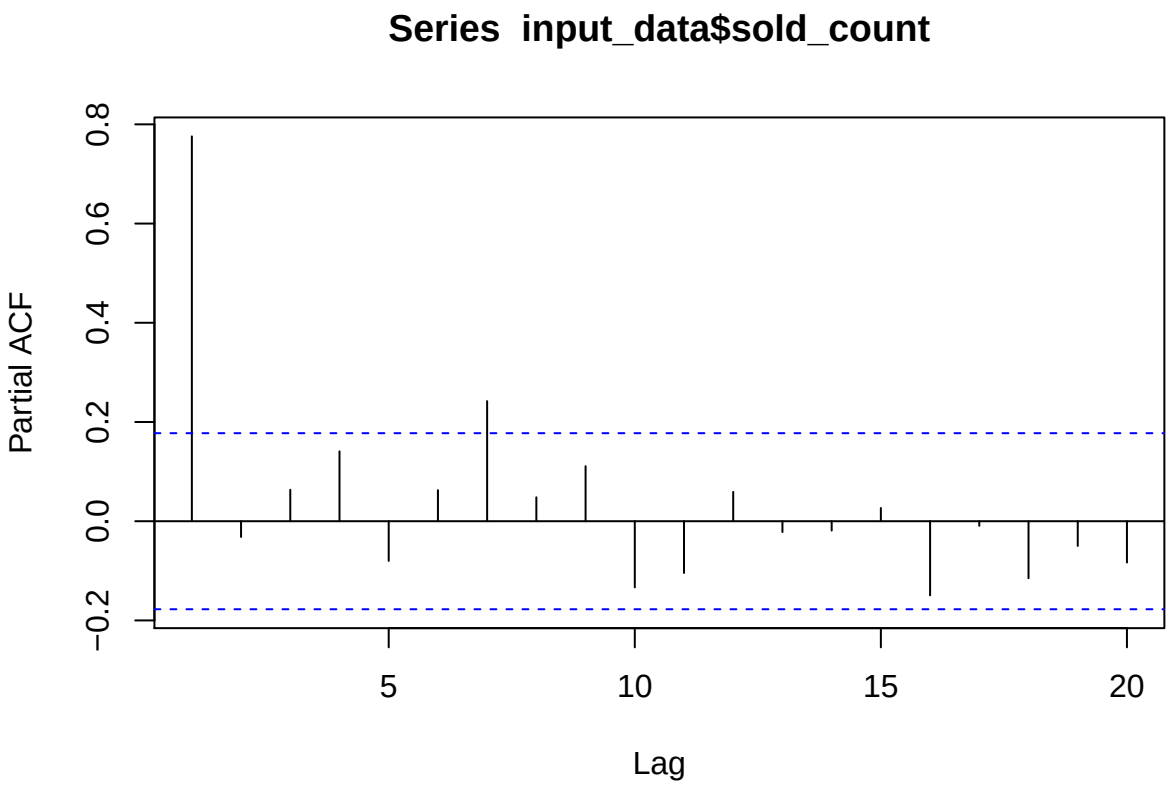


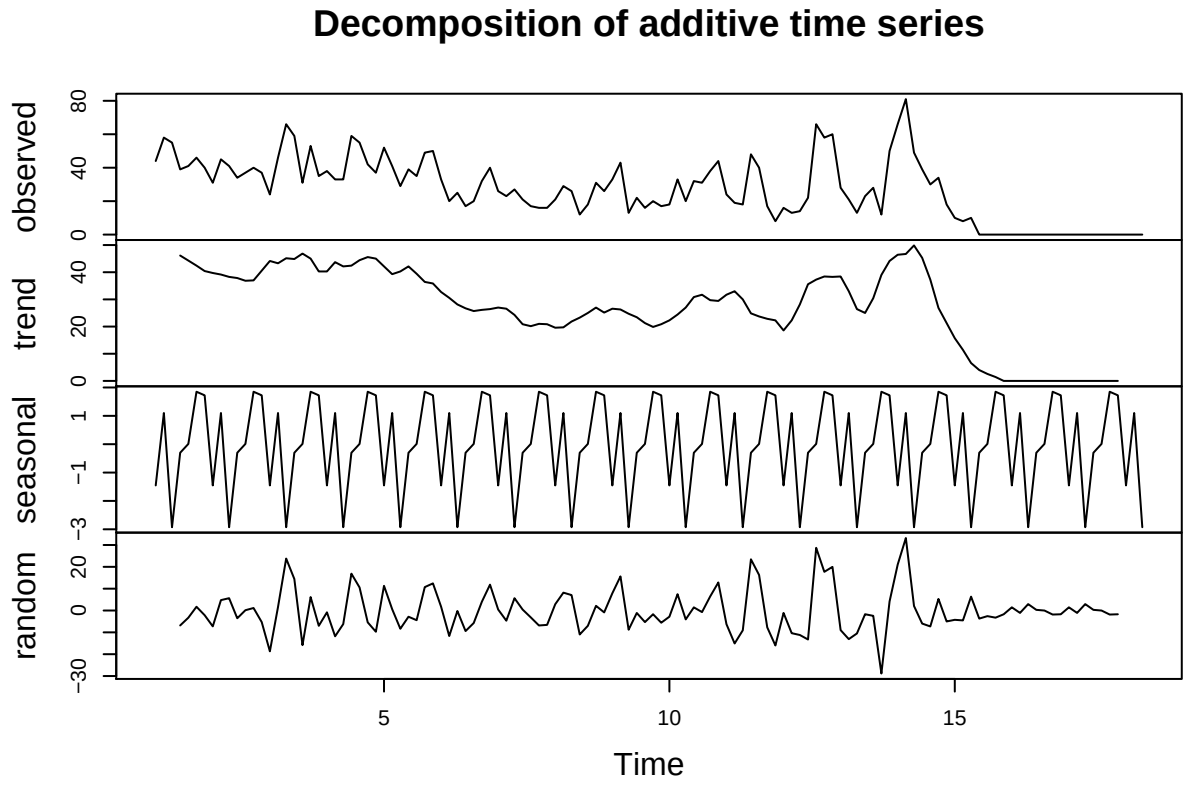
The data is non-stationary. Seasonality is clearly observed; however the rest can not be observed due to most probably 'out of stock'. Best 'p' and 'q' parameters are 1 and 1-2-3, respectively. ACF also shows that significantly lags are observed even in 20. day, i.e the data has the trend component.

```
Seasonality_Analysis(bikini_2)
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 4 lags.
##
## Value of test-statistic is: 1.243
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```


Series input_data$sold_count
ACF
1.0
0.8
0.6
0.4
0.2
−0.2
0
5
10
15
20
Lag

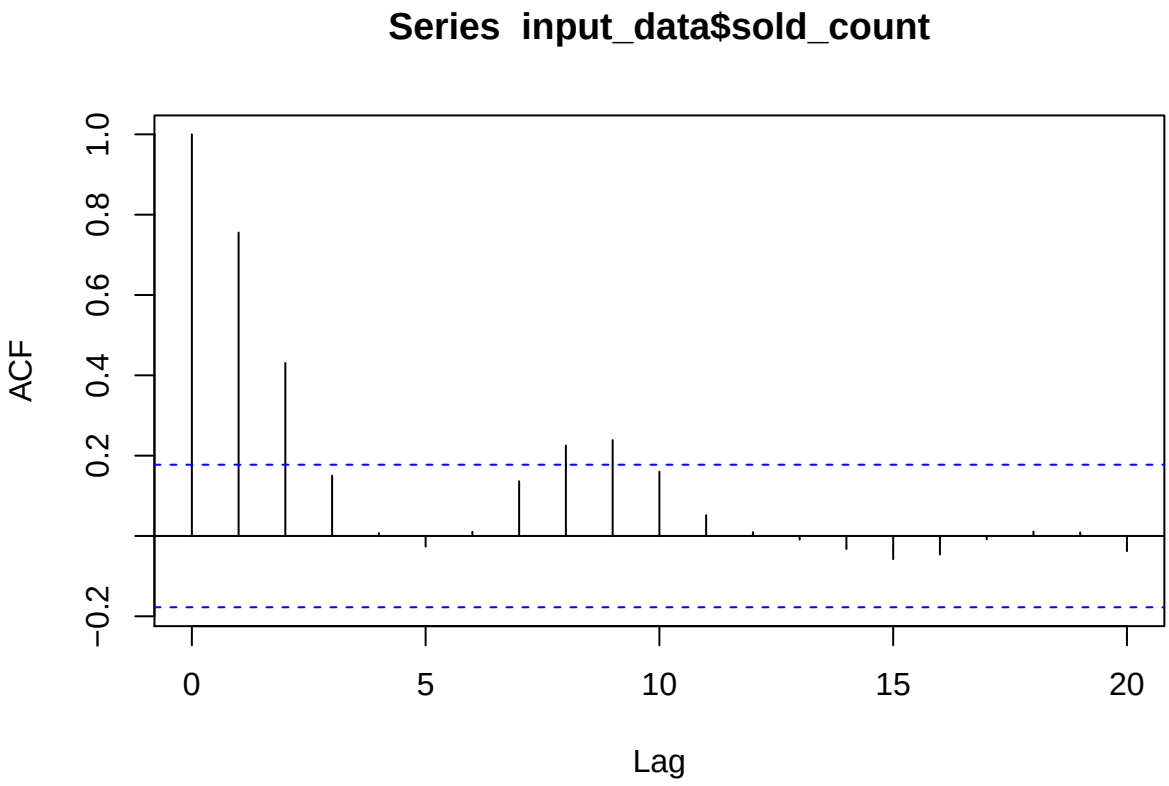

Series input_data$sold_count
Partial ACF
0.8
0.6
0.4
0.2
0.0
−0.2
5
10
15
20
Lag

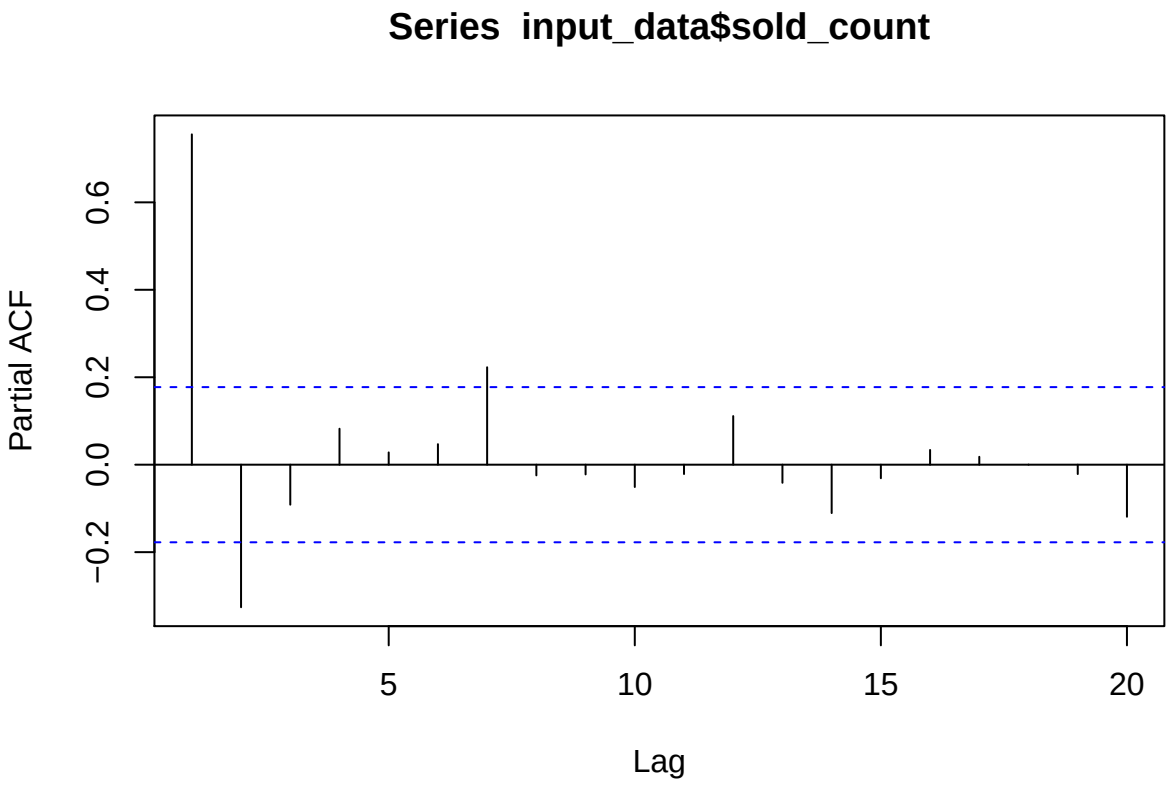


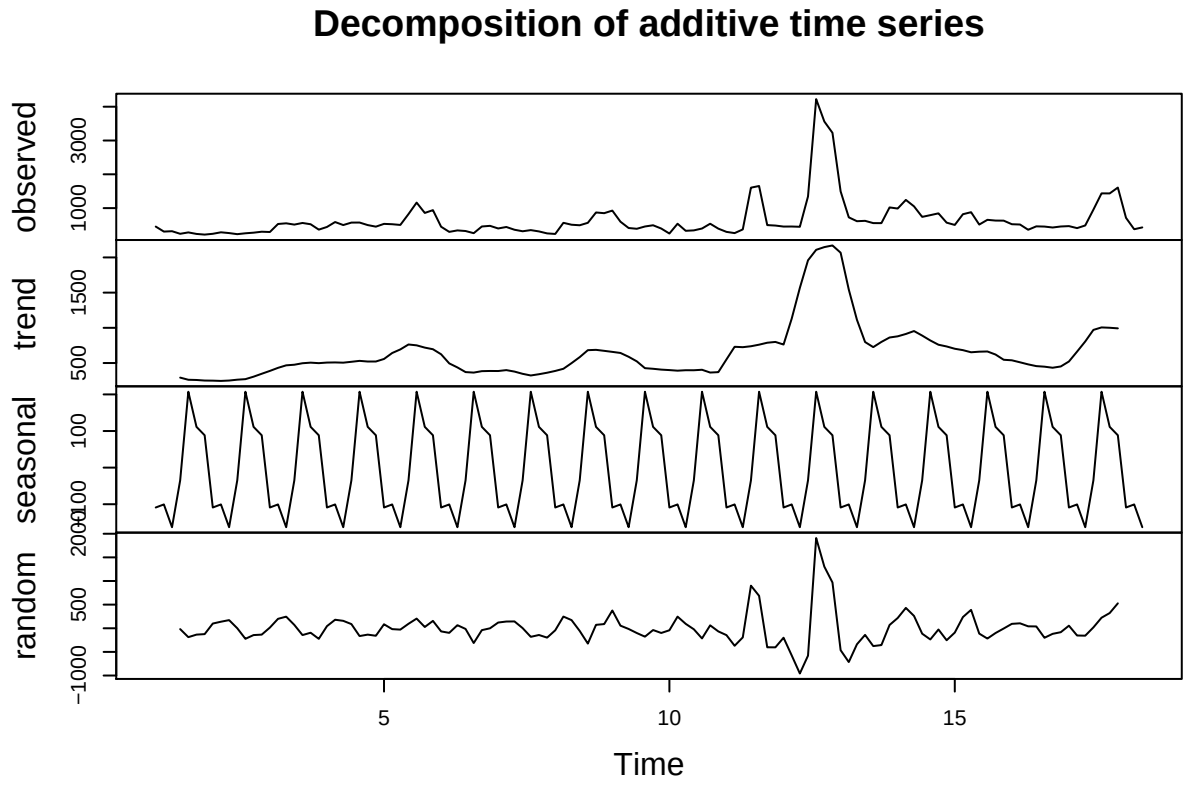
The data is non-stationary. Ignoring the 'out of stock' period at the end, seasonality and trend variation is clearly observed. Best 'p' and 'q' parameters are 1 and 1-2-3, respectively.

```
Seasonality_Analysis(tayt)
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 4 lags.
##
## Value of test-statistic is: 0.4962
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```


Series input_data$sold_count
ACF
1.0
0.8
0.6
0.4
0.2
−0.2
0
5
10
15
20
Lag

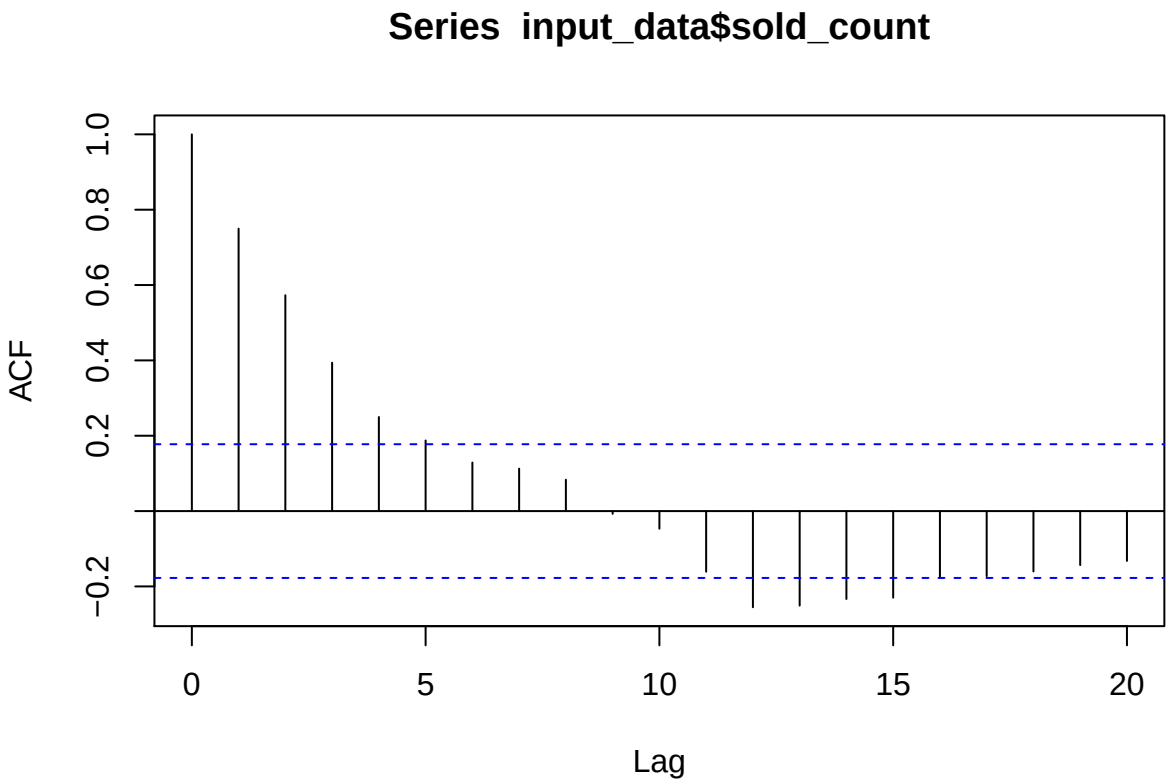

Series input_data$sold_count
Partial ACF
0.6
0.4
0.2
0.0
−0.2
5
10
15
20
Lag

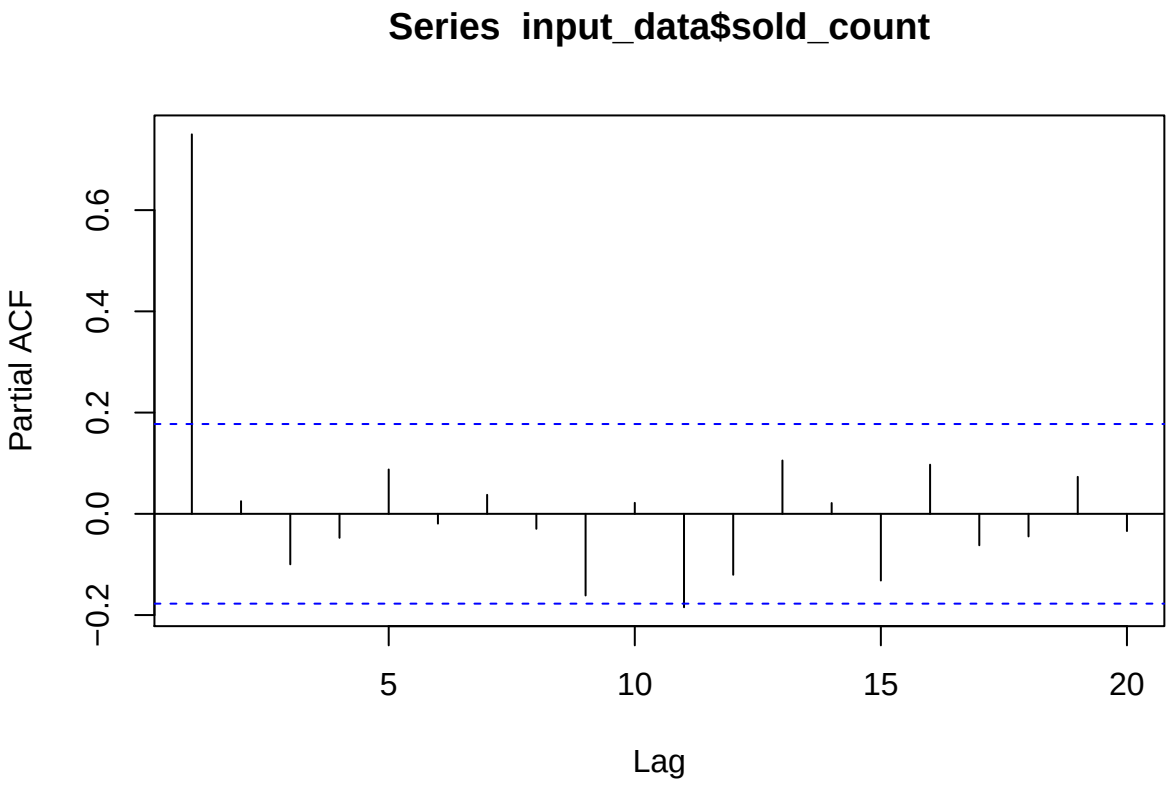


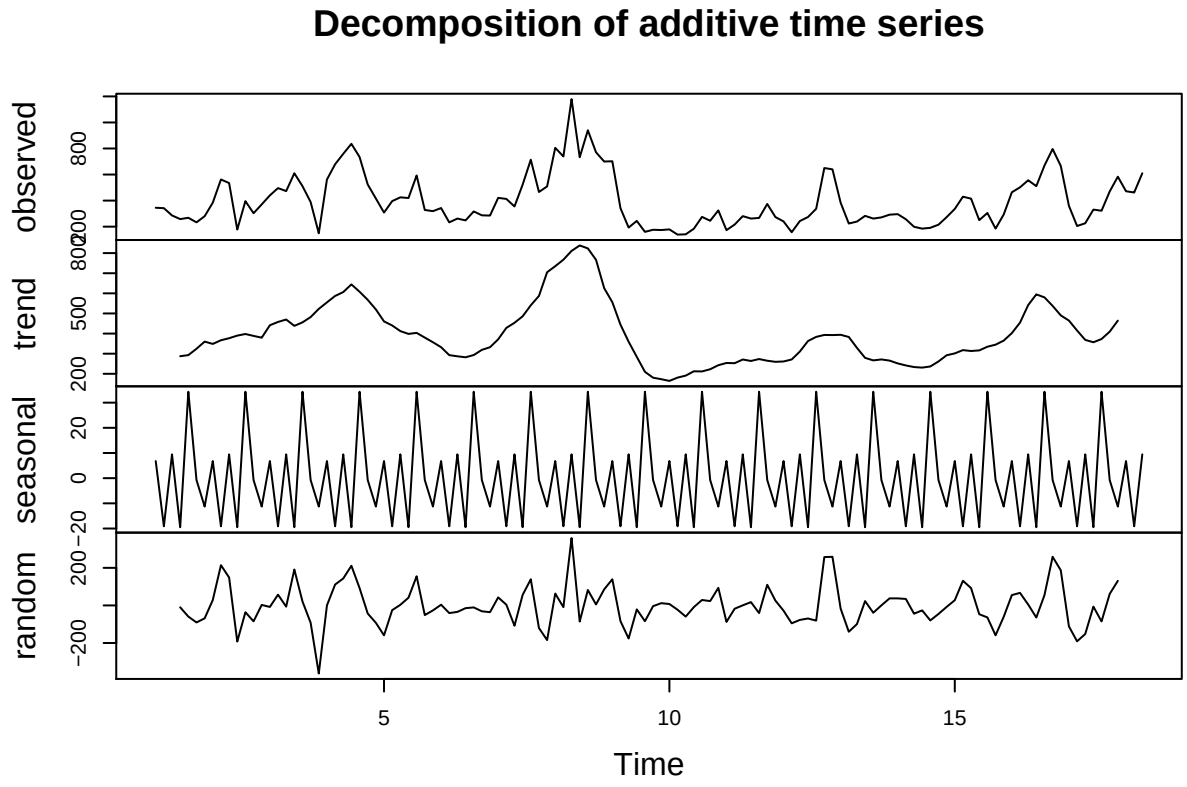
The data is more stationary comparing to the products above (stationary wrt. 2.5 and 1 pct confidence interval). Seasonality is clearly observed. There is a significant peak in the data, can occur from surge in the discount(resulting from special discount days of Trendyol such as 'Muhteşem Kasım'. Best 'p' and 'q' parameters are 1 and 1-2-3, respectively. Comparing the random and the actual observation, the data resembles to white noise, meaning that it can't be predicted well by the statistical models.

```
Seasonality_Analysis(kulaklık)
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 4 lags.
##
## Value of test-statistic is: 0.1654
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```


Series input_data$sold_count
ACF
1.0
0.8
0.6
0.4
0.2
−0.2
0
5
10
15
20
Lag

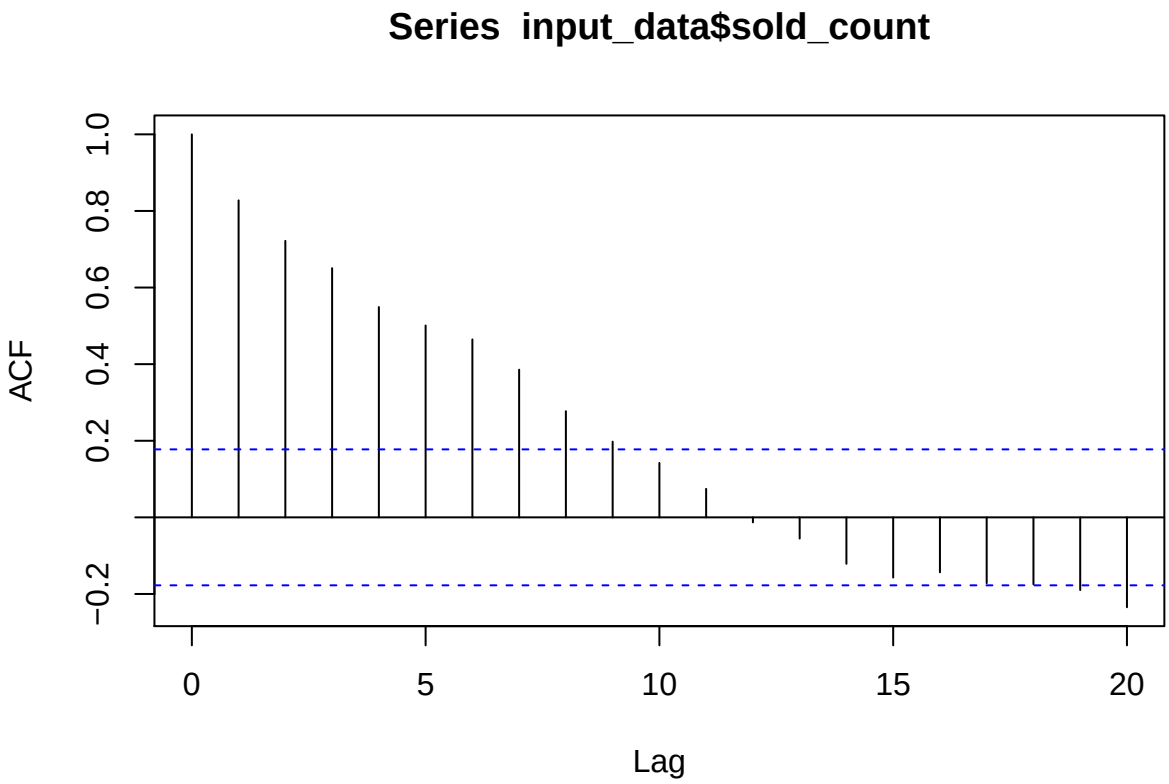

Series input_data$sold_count
Partial ACF
0.6
0.4
0.2
0.0
−0.2
5
10
15
20
Lag

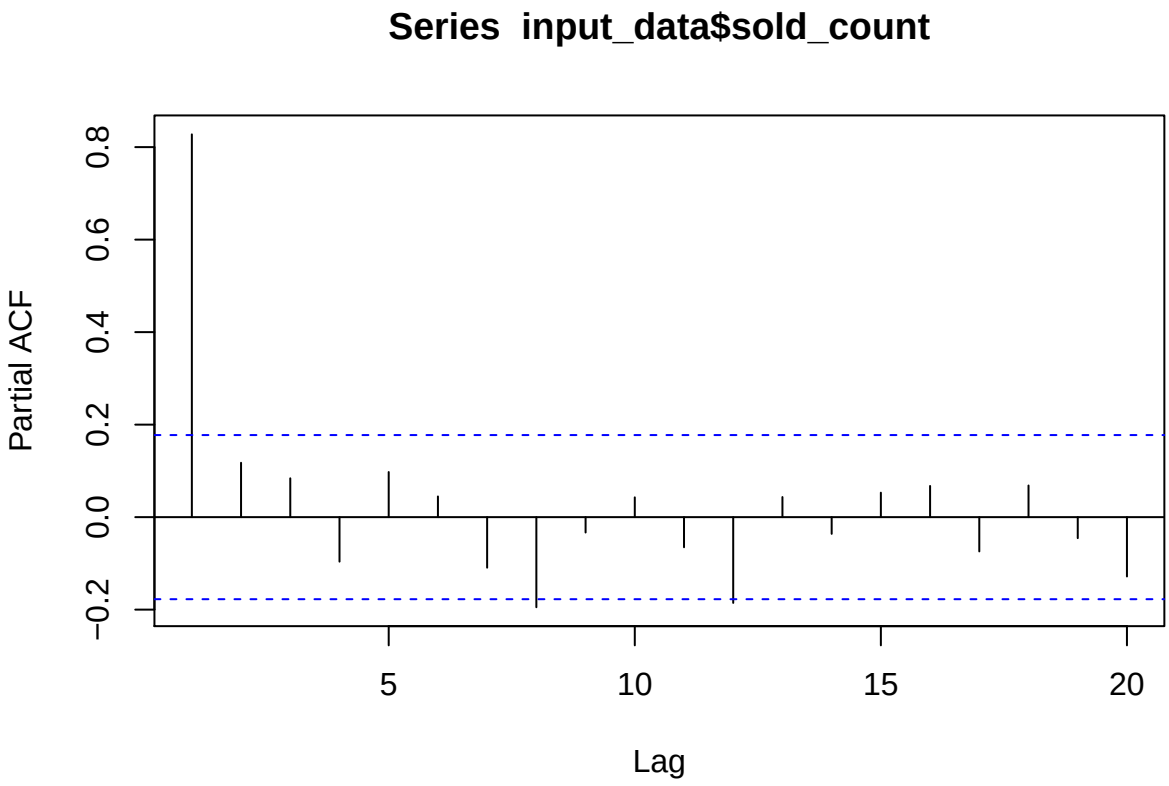


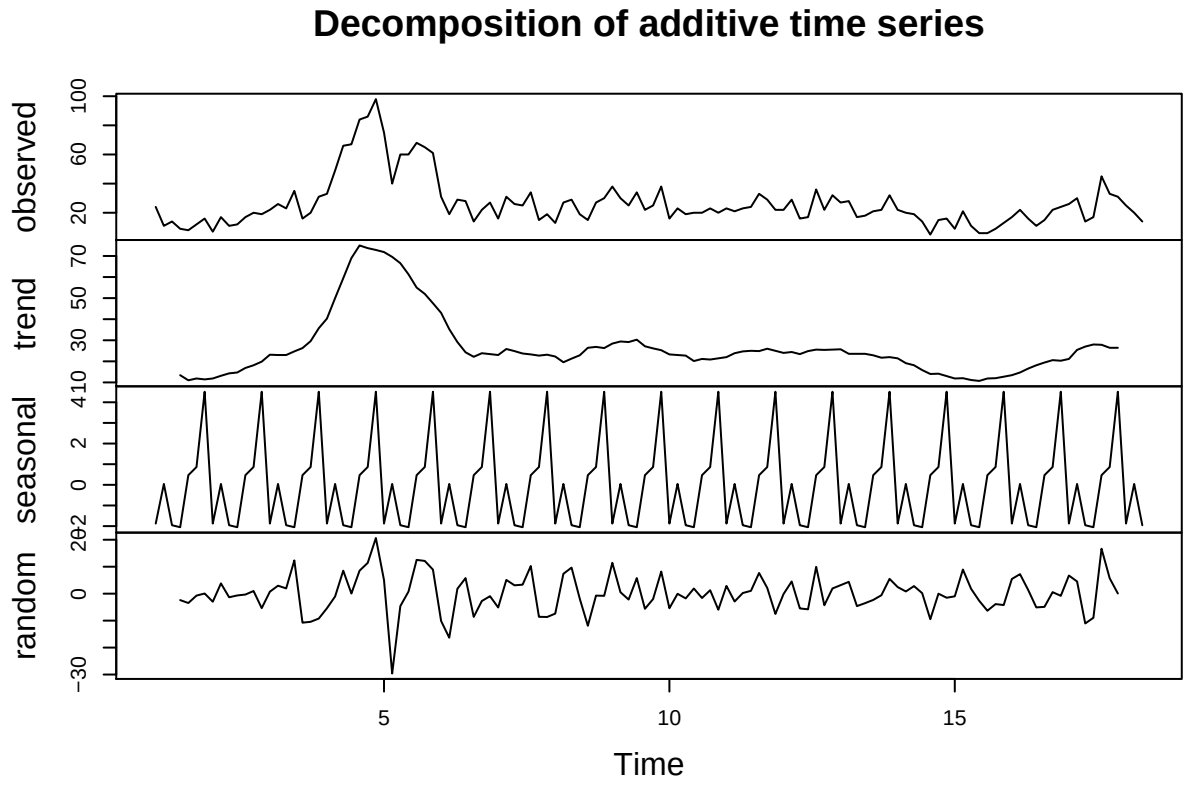
The data is stationary. Seasonality is clearly observed. There is a significant peak in the data, can occur from surge in the discount(resulting from special discount days of Trendyol such as 'Muhteşem Kasım'. Best 'p' and 'q' parameters are 1 and 1-2-3, respectively.

```
Seasonality_Analysis(supurge)
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 4 lags.
##
## Value of test-statistic is: 0.4004
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```


Series input_data$sold_count
ACF
1.0
0.8
0.6
0.4
0.2
−0.2
0
5
10
15
20
Lag

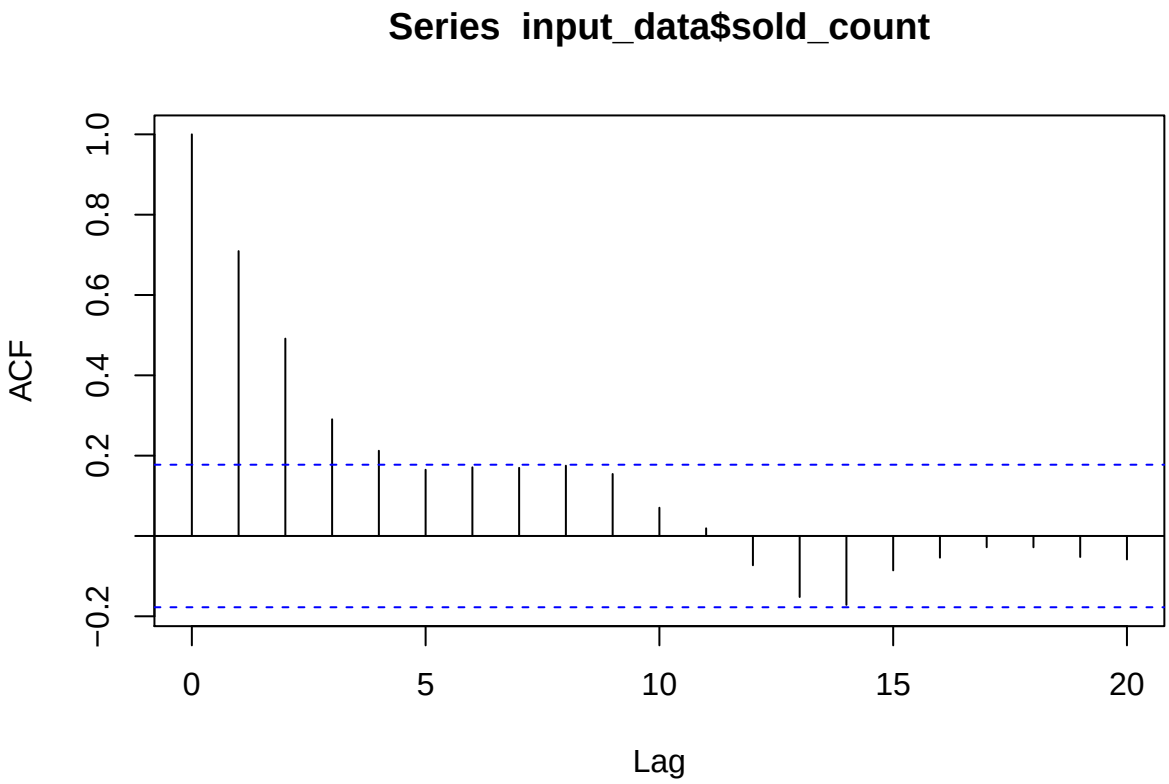

Series input_data$sold_count
Partial ACF
0.8
0.6
0.4
0.2
0.0
−0.2
5
10
15
20
Lag

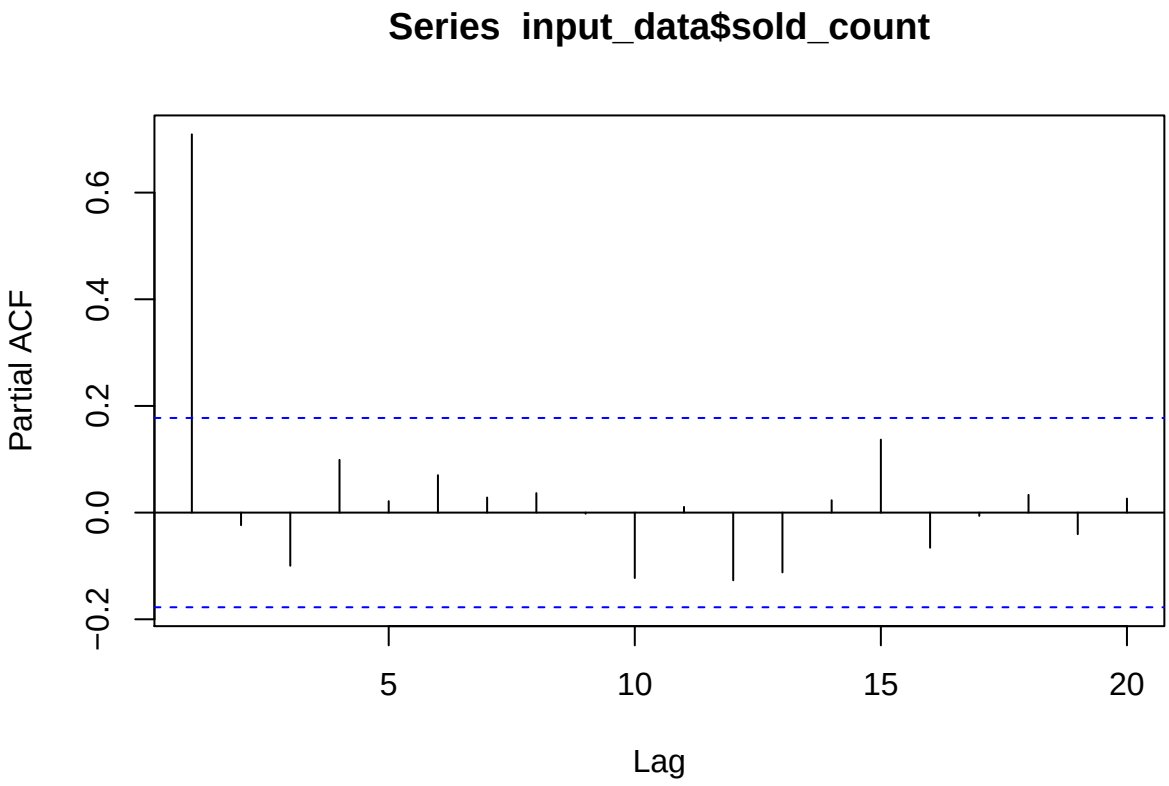


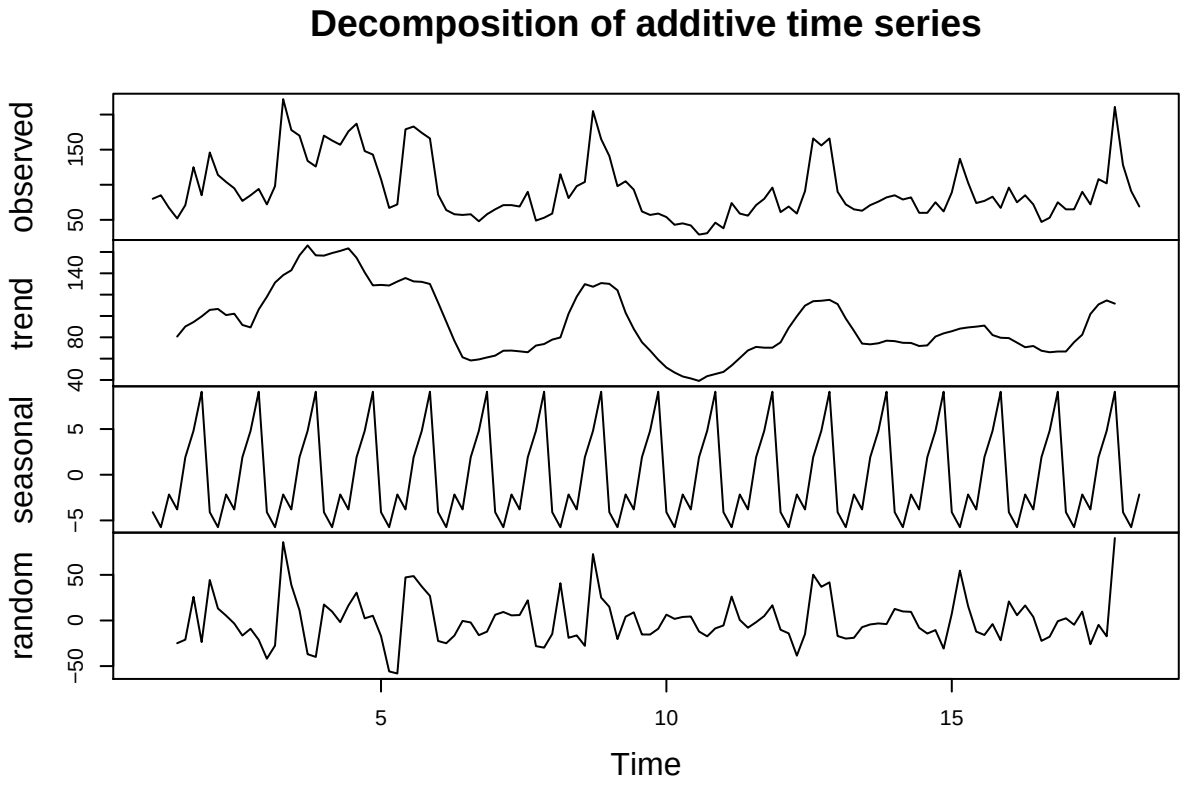
The data is more stationary comparing to the products above (stationary wrt. 5, 2.5 and 1 pct confidence interval). Seasonality is clearly observed. There is a significant peak in the data, can occur from surge in the discount(resulting from special discount days of Trendyol such as 'Muhteşem Kasım'. Best 'p' and 'q' parameters are 1 and 1-2-3, respectively.

```
Seasonality_Analysis(yuz_temizleyicisi)
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 4 lags.
##
## Value of test-statistic is: 0.4218
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```


Series input_data$sold_count
ACF
1.0
0.8
0.6
0.4
0.2
−0.2
0
5
10
15
20
Lag

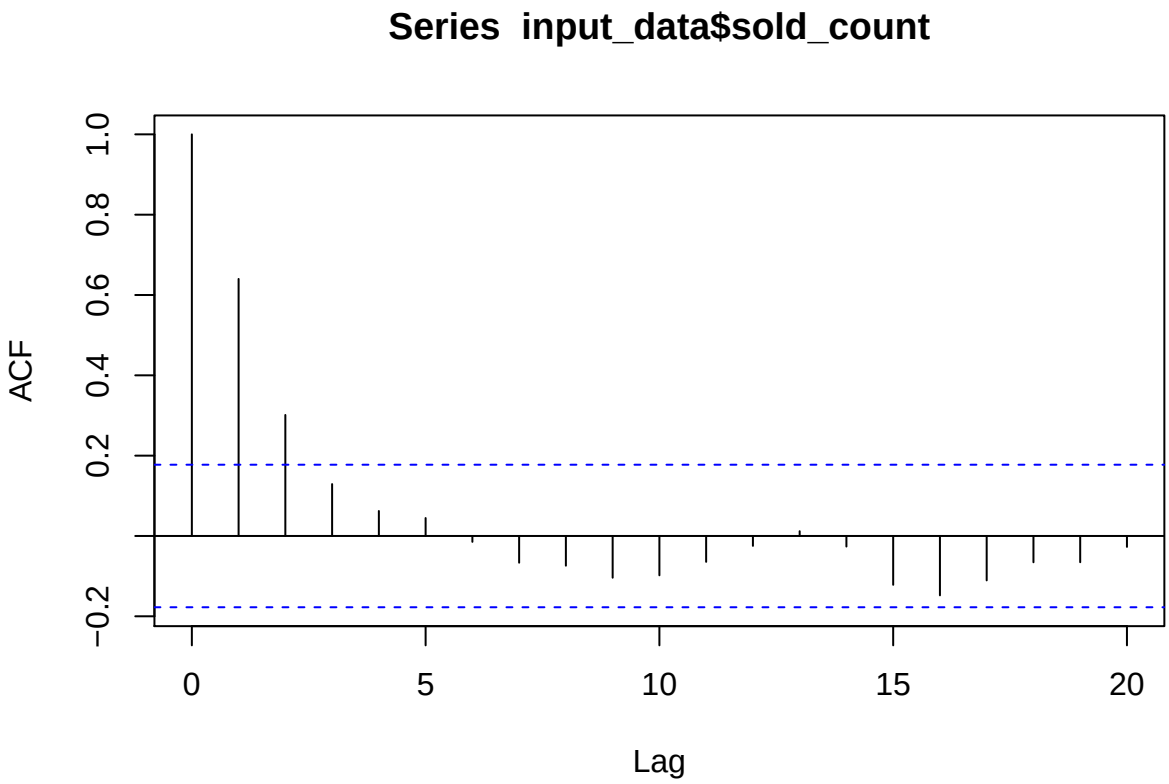

Series input_data$sold_count
Partial ACF
0.6
0.4
0.2
0.0
−0.2
5
10
15
20
Lag

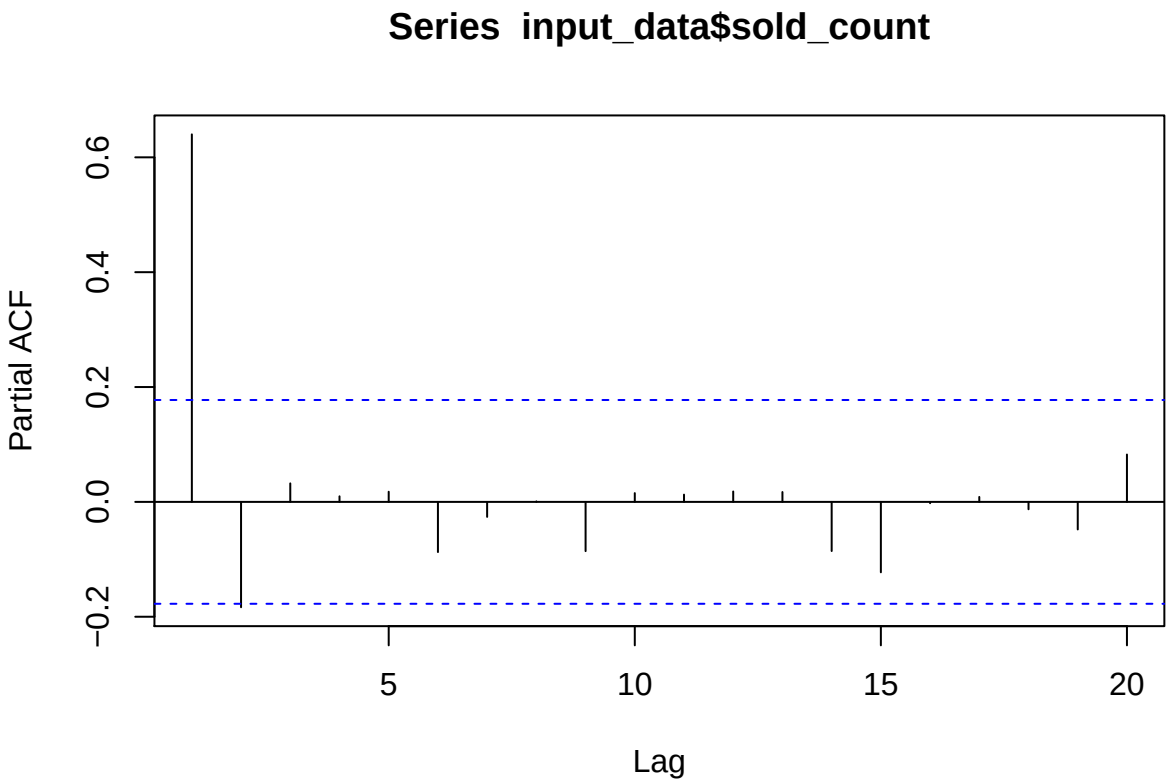


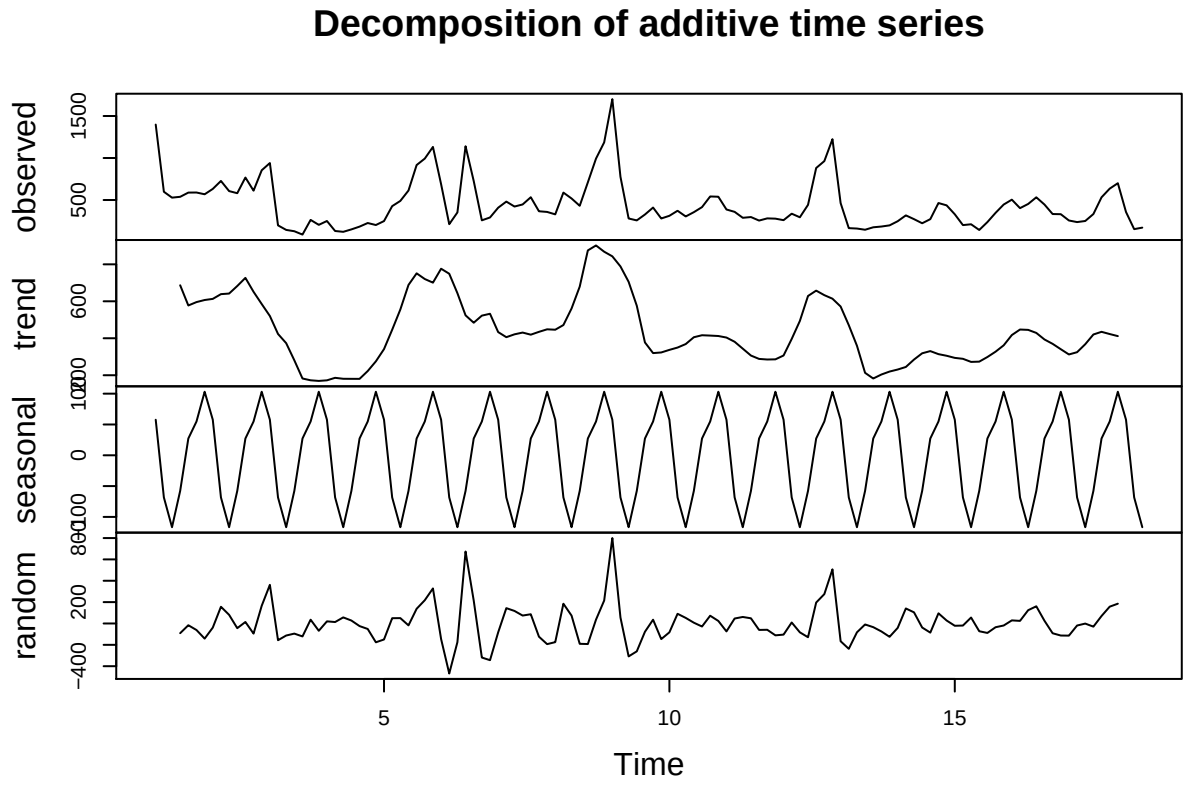
The data is more stationary comparing to the products above (stationary wrt. 5, 2.5 and 1 pct confidence interval). Seasonality is clearly observed. Best 'p' and 'q' parameters are 1 and 1-2-3, respectively.

```
Seasonality_Analysis(mendil)
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 4 lags.
##
## Value of test-statistic is: 0.323
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```


Series input_data$sold_count
ACF
1.0
0.8
0.6
0.4
0.2
−0.2
0
5
10
15
20
Lag

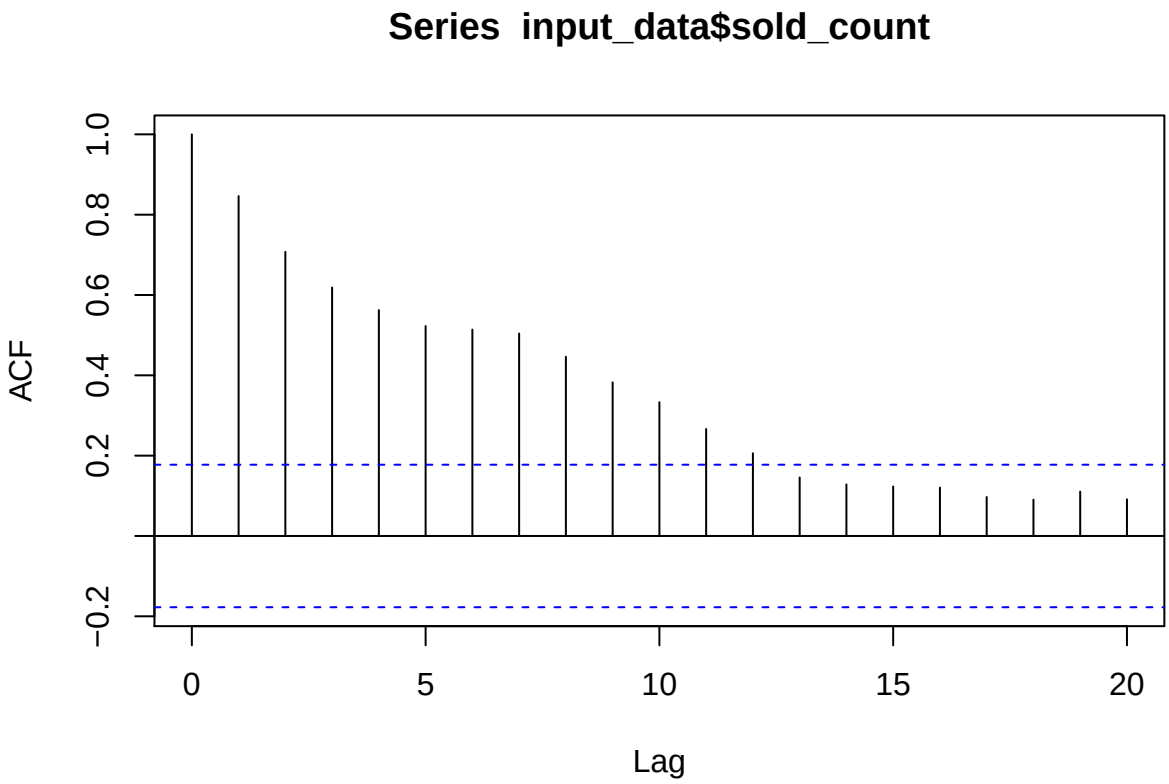

Series input_data$sold_count
Partial ACF
0.6
0.4
0.2
0.0
−0.2
5
10
15
20
Lag

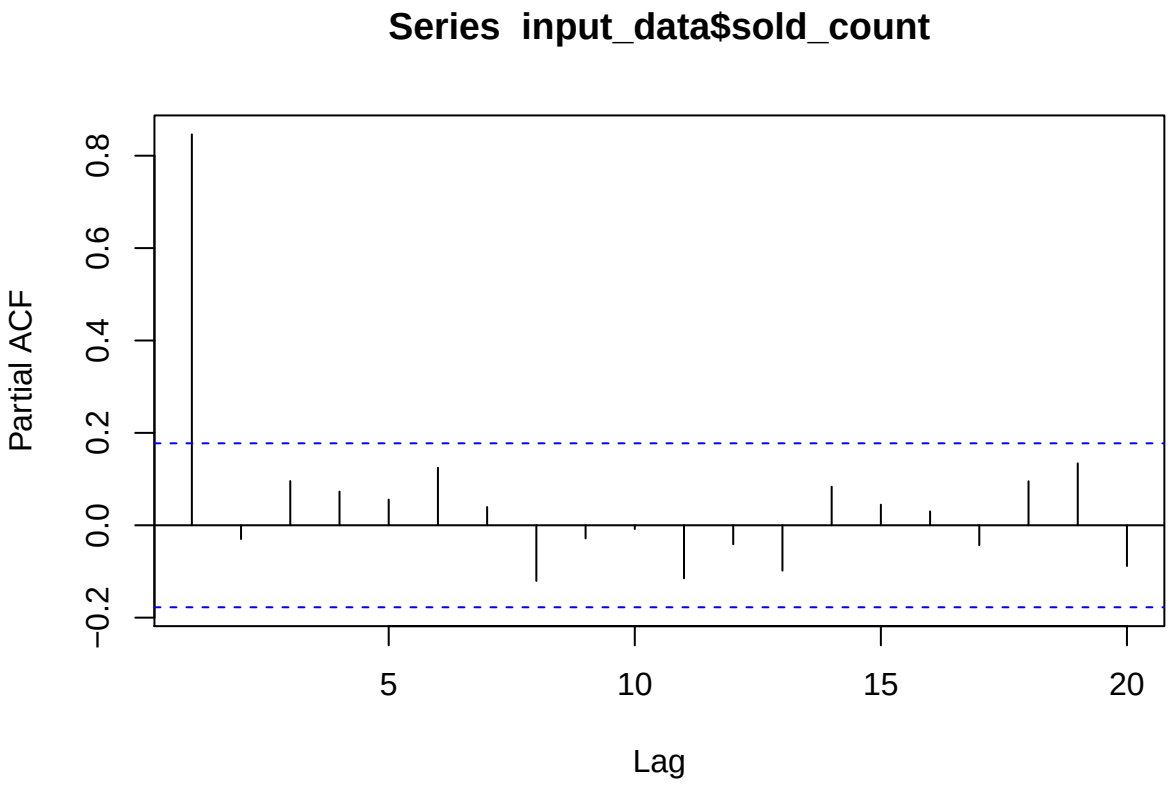


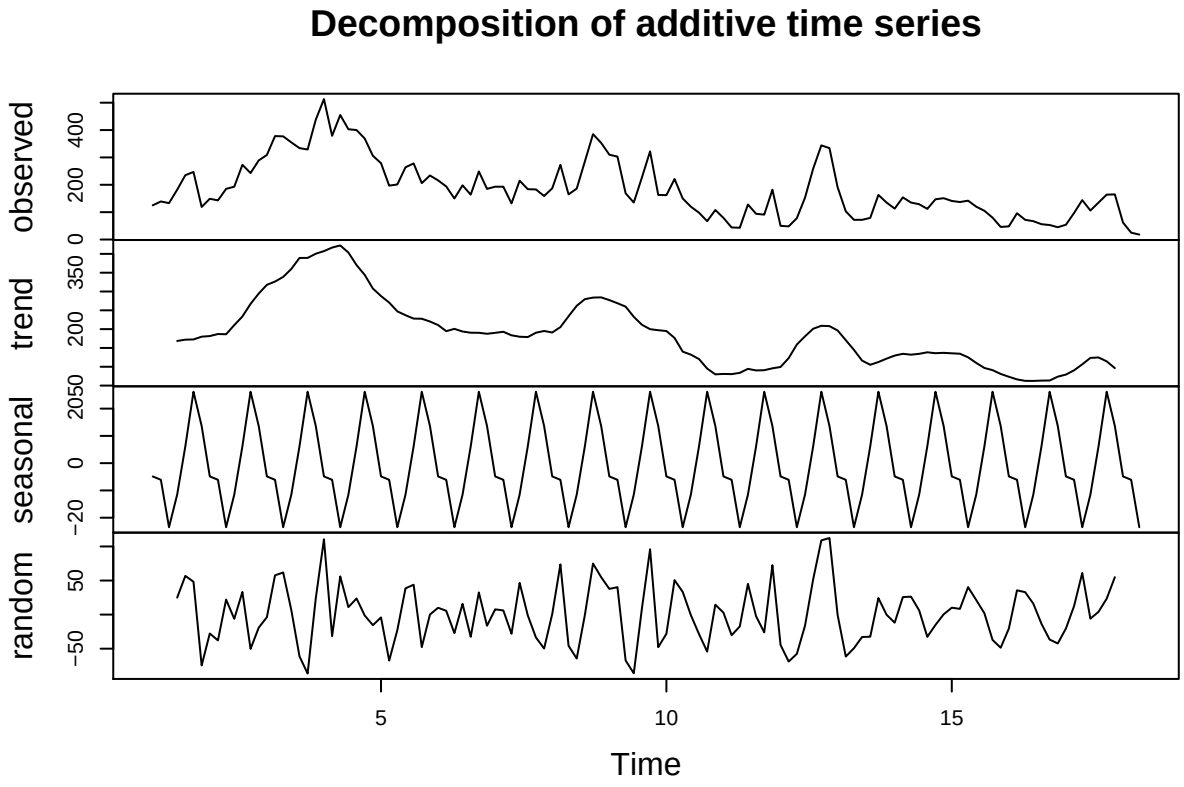
The data is stationary. Seasonality is clearly observed. Best 'p' and 'q' parameters are 1 and 1-2-3, respectively.

```
Seasonality_Analysis(dis_fircasi)
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 4 lags.
##
## Value of test-statistic is: 1.296
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```


Series input_data$sold_count
ACF
1.0
0.8
0.6
0.4
0.2
−0.2
0
5
10
15
20
Lag


Series input_data$sold_count
Partial ACF
0.8
0.6
0.4
0.2
0.0
−0.2
5
10
15
20
Lag



The data is non-stationary. Unlike to other 8 products, it has a decreasing trend over the months. Seasonality is clearly observed. Best 'p' and 'q' parameters are 1 and 1-2-3, respectively.

Proposing ARIMA models

We proposed 'p' and 'q' parameters as 1 and 1-2-3, respectively for all products, since we observed similar ACF and PACF characteristics. Also, differencing of 1 will be tried for each model. The function for decomposing data and creating ARIMA models can be seen below:

```
ARIMA_proposing <- function(X){
  df_new <- ts(X$sold_count,frequency=7)
  df_additive <- decompose(df_new,type="additive")
  df_deseasoned_detrended <- df_new-df_additive$seasonal-df_additive$trend

  model <- arima(df_deseasoned_detrended, order=c(1,0,1))
  print(model)
  AIC(model)

  model <- arima(df_deseasoned_detrended, order=c(1,1,1))
  print(model)
  AIC(model)

  model <- arima(df_deseasoned_detrended, order=c(1,0,2))
  print(model)
  AIC(model)
```

```
  model <- arima(df_deseasoned_detrended, order=c(1,1,2))
  print(model)
  AIC(model)

  model <- arima(df_deseasoned_detrended, order=c(1,0,3))
  print(model)
  AIC(model)

  model <- arima(df_deseasoned_detrended, order=c(1,1,3))
  print(model)
  AIC(model)
}
```

Best ARIMA model for 'Mont Product':

```
ARIMA_proposing(mont)
```

```
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 1))
##
## Coefficients:
##          ar1     ma1  intercept
##       -0.7975  0.9918    -0.0311
## s.e.   0.0832  0.0734     0.0524
##
## sigma^2 estimated as 0.2593:  log likelihood = -87.23,  aic = 182.47
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 1))
##
## Coefficients:
##          ar1      ma1
##       0.0691  -1.0000
## s.e.  0.0978   0.0253
##
## sigma^2 estimated as 0.2879:  log likelihood = -93.89,  aic = 193.79
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 2))
##
## Coefficients:
##          ar1      ma1      ma2  intercept
##       0.0796  -0.3615  -0.6385    -0.0040
## s.e.  0.2732   0.2484   0.2477     0.0023
##
## sigma^2 estimated as 0.2073:  log likelihood = -75.46,  aic = 160.92
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 2))
##
## Coefficients:
##          ar1      ma1      ma2
```

```
##       -0.7946  -0.0095  -0.9905
## s.e.   0.0836   0.0700   0.0699
##
## sigma^2 estimated as 0.2617:  log likelihood = -89.26,  aic = 186.52
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 3))
##
## Coefficients:

## Warning in sqrt(diag(x$var.coef)): NaNs üretimi

##           ar1      ma1      ma2      ma3  intercept
##       -0.1192  -0.1756  -0.7402  -0.0841    -0.0039
## s.e.      NaN      NaN      NaN      NaN     0.0021
##
## sigma^2 estimated as 0.2072:  log likelihood = -75.5,  aic = 163
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 3))
##
## Coefficients:
##          ar1      ma1      ma2     ma3
##       0.1293  -1.3912  -0.2140  0.6072
## s.e.  0.2560   0.2385   0.4621  0.2332
##
## sigma^2 estimated as 0.2117:  log likelihood = -80.37,  aic = 170.74

## [1] 170.741
```

Best AIC is observed in the ARIMA model (1, 0, 2)

Best ARIMA model for 'Bikini_1 Product':

```
ARIMA_proposing(bikini_1)
```

```
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 1))
##
## Coefficients:
##           ar1     ma1  intercept
##       -0.2688  0.4722    -0.2146
## s.e.   0.2159  0.1858     1.6364
##
## sigma^2 estimated as 231.1:  log likelihood = -480.32,  aic = 968.64
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 1))
##
## Coefficients:
##          ar1      ma1
##       0.1315  -1.0000
```

```
## s.e.  0.0930   0.0222
##
## sigma^2 estimated as 239.9:  log likelihood = -480.54,  aic = 967.08
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 2))
##
## Coefficients:
##          ar1      ma1      ma2  intercept
##       0.4177  -0.5629  -0.4371    -0.1258
## s.e.  0.1073   0.0953   0.0928     0.0867
##
## sigma^2 estimated as 173.4:  log likelihood = -465.47,  aic = 940.93
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 2))
##
## Coefficients:
##           ar1      ma1      ma2
##       -0.2571  -0.5329  -0.4671
## s.e.   0.2160   0.1867   0.1857
##
## sigma^2 estimated as 233.2:  log likelihood = -478.91,  aic = 965.81
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 3))
##
## Coefficients:
##          ar1      ma1      ma2      ma3  intercept
##       0.0292  -0.1386  -0.5376  -0.3237    -0.1239
## s.e.  0.3075   0.2951   0.1359   0.1843     0.0786
##
## sigma^2 estimated as 168.8:  log likelihood = -464.05,  aic = 940.09
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 3))
##
## Coefficients:
##          ar1      ma1     ma2     ma3
##       0.4534  -1.5682  0.1417  0.4295
## s.e.  0.1073   0.1094  0.1876  0.0956
##
## sigma^2 estimated as 177.4:  log likelihood = -466.55,  aic = 943.1

## [1] 943.1041
```

Best AIC is observed in the ARIMA model (1, 0, 3)

Best ARIMA model for 'Bikini_2 Product':

```
ARIMA_proposing(bikini_2)
```

```
##
## Call:
```

```
## arima(x = df_deseasoned_detrended, order = c(1, 0, 1))
##
## Coefficients:
##           ar1     ma1  intercept
##       -0.0778  0.3876    -0.1427
## s.e.   0.1795  0.1502     1.1073
##
## sigma^2 estimated as 86.11:  log likelihood = -423.08,  aic = 854.16
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 1))
##
## Coefficients:
##          ar1      ma1
##       0.2319  -1.0000
## s.e.  0.0913   0.0218
##
## sigma^2 estimated as 90.65:  log likelihood = -424.48,  aic = 854.95
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 2))
##
## Coefficients:
##          ar1      ma1      ma2  intercept
##       0.4065  -0.5247  -0.4753    -0.0052
## s.e.  0.1029   0.0828   0.0804     0.0525
##
## sigma^2 estimated as 62.57:  log likelihood = -406.37,  aic = 822.75
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 2))
##
## Coefficients:
##           ar1      ma1      ma2
##       -0.0645  -0.6175  -0.3825
## s.e.   0.1790   0.1507   0.1494
##
## sigma^2 estimated as 86.87:  log likelihood = -422.05,  aic = 852.1
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 3))
##
## Coefficients:
##          ar1     ma1      ma2      ma3  intercept
##       -0.116  0.1390  -0.4389  -0.7001    -0.0036
## s.e.   0.143  0.1099   0.0617   0.1017     0.0496
##
## sigma^2 estimated as 52.73:  log likelihood = -396.88,  aic = 805.77
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 3))
##
## Coefficients:
##           ar1     ma1      ma2      ma3
```

```
##       -0.5668  0.0559  -0.5910  -0.4649
## s.e.   0.1556  0.1328   0.1253   0.1273
##
## sigma^2 estimated as 84.1:  log likelihood = -420.13,  aic = 850.26

## [1] 850.2578
```

Best AIC is observed in the ARIMA model (1, 0, 3)

Best ARIMA model for 'Tayt Product':

```
ARIMA_proposing(tayt)
```

```
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 1))
##
## Coefficients:
##          ar1     ma1  intercept
##       0.1879  0.3977     6.3537
## s.e.  0.1331  0.1077    46.7351
##
## sigma^2 estimated as 86264:  log likelihood = -823.95,  aic = 1655.91
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 1))
##
## Coefficients:
##           ar1     ma1
##       -0.2815  0.3621
## s.e.   0.3937  0.3739
##
## sigma^2 estimated as 130270:  log likelihood = -840.38,  aic = 1686.76
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 2))
##
## Coefficients:
##           ar1     ma1     ma2  intercept
##       -0.2355  0.9154  0.4737     6.7127
## s.e.   0.1707  0.1390  0.1125    50.7810
##
## sigma^2 estimated as 80798:  log likelihood = -820.3,  aic = 1650.59

## Warning in log(s2): NaNs üretimi

## Warning in log(s2): NaNs üretimi

## Warning in log(s2): NaNs üretimi

##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 2))
```

```
##
## Coefficients:
##          ar1      ma1      ma2
##       0.2024  -0.6069  -0.3931
## s.e.  0.1336   0.1099   0.1079
##
## sigma^2 estimated as 87017:  log likelihood = -819.18,  aic = 1646.36
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 3))
##
## Coefficients:
##          ar1      ma1      ma2      ma3  intercept
##       0.3125  -0.0505  -0.3673  -0.5821     0.4462
## s.e.  0.1139   0.0874   0.0883   0.0964     2.3406
##
## sigma^2 estimated as 57481:  log likelihood = -802.31,  aic = 1616.62
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 3))
##
## Coefficients:
##           ar1      ma1      ma2      ma3
##       -0.2234  -0.0886  -0.4347  -0.4767
## s.e.   0.1706   0.1402   0.1131   0.1131
##
## sigma^2 estimated as 81496:  log likelihood = -815.44,  aic = 1640.89

## [1] 1640.887
```

Best AIC is observed in the ARIMA model (1, 0, 3)

Best ARIMA model for 'Kulaklık Product':

```
ARIMA_proposing(kulaklık)
```

```
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 1))
##
## Coefficients:
##          ar1     ma1  intercept
##       0.0186  0.2827    -0.0662
## s.e.  0.2043  0.1825    12.2493
##
## sigma^2 estimated as 10229:  log likelihood = -700.16,  aic = 1408.31
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 1))
##
## Coefficients:
##          ar1      ma1
##       0.2668  -1.0000
## s.e.  0.0909   0.0218
```

```
##
## sigma^2 estimated as 10522:  log likelihood = -697.81,  aic = 1401.61
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 2))
##
## Coefficients:
##          ar1      ma1      ma2  intercept
##       0.5128  -0.5763  -0.4236    -0.0162
## s.e.  0.0950   0.0849   0.0825     0.6866
##
## sigma^2 estimated as 7888:  log likelihood = -686.8,  aic = 1383.6
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 2))
##
## Coefficients:
##          ar1      ma1      ma2
##       0.0354  -0.7252  -0.2747
## s.e.  0.2030   0.1822   0.1810
##
## sigma^2 estimated as 10318:  log likelihood = -696.72,  aic = 1401.45
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 3))
##
## Coefficients:
##          ar1      ma1      ma2      ma3  intercept
##       0.2527  -0.2700  -0.3010  -0.4289     0.0246
## s.e.  0.1398   0.1177   0.1104   0.1322     0.6552
##
## sigma^2 estimated as 7246:  log likelihood = -682.02,  aic = 1376.04
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 3))
##
## Coefficients:
##           ar1     ma1      ma2      ma3
##       -0.8867  0.2250  -0.8818  -0.3432
## s.e.   0.2377  0.2368   0.1999   0.0901
##
## sigma^2 estimated as 10142:  log likelihood = -695.75,  aic = 1401.49

## [1] 1401.495
```

Best AIC is observed in the ARIMA model (1, 0, 3)

Best ARIMA model for 'Süpürge Product':

```
ARIMA_proposing(supurge)
```

```
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 1))
```

```
##
## Coefficients:
##           ar1     ma1  intercept
##       -0.2871  0.5393     0.0212
## s.e.   0.2274  0.1934     0.7399
##
## sigma^2 estimated as 44.5:  log likelihood = -384.78,  aic = 777.55
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 1))
##
## Coefficients:
##          ar1      ma1
##       0.1738  -1.0000
## s.e.  0.0921   0.0221
##
## sigma^2 estimated as 46.75:  log likelihood = -386.45,  aic = 778.91
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 2))
##
## Coefficients:
##          ar1      ma1      ma2  intercept
##       0.4453  -0.5214  -0.4786     0.0132
## s.e.  0.1043   0.0954   0.0931     0.0412
##
## sigma^2 estimated as 33.81:  log likelihood = -370.64,  aic = 751.29
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 2))
##
## Coefficients:
##           ar1      ma1      ma2
##       -0.2729  -0.4675  -0.5325
## s.e.   0.2269   0.1937   0.1928
##
## sigma^2 estimated as 44.89:  log likelihood = -384.15,  aic = 776.31
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 3))
##
## Coefficients:
##          ar1      ma1      ma2      ma3  intercept
##       0.2384  -0.2930  -0.4965  -0.2105     0.0126
## s.e.  0.2124   0.2079   0.1063   0.1505     0.0389
##
## sigma^2 estimated as 33.26:  log likelihood = -369.76,  aic = 751.53
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 3))
##
## Coefficients:
##          ar1      ma1     ma2     ma3
##       0.4783  -1.5286  0.0607  0.4699
```

```
## s.e.  0.1049   0.1037  0.1867  0.0951
##
## sigma^2 estimated as 34.55:  log likelihood = -372.63,  aic = 755.25

## [1] 755.2503
```

Best AIC is observed in the ARIMA model (1, 0, 2)

Best ARIMA model for 'Yüz Temizleyicisi Product':

```
ARIMA_proposing(yuz_temizleyicisi)
```

```
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 1))
##
## Coefficients:
##          ar1     ma1  intercept
##       0.0149  0.2607     0.4339
## s.e.  0.2080  0.1847     2.9539
##
## sigma^2 estimated as 619.9:  log likelihood = -537.55,  aic = 1083.1
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 1))
##
## Coefficients:
##          ar1      ma1
##       0.2430  -1.0000
## s.e.  0.0969   0.0227
##
## sigma^2 estimated as 635.3:  log likelihood = -536.42,  aic = 1078.84
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 2))
##
## Coefficients:
##          ar1     ma1      ma2  intercept
##       0.4495  -0.586  -0.4140     0.0164
## s.e.  0.1037   0.091   0.0887     0.1478
##
## sigma^2 estimated as 466:  log likelihood = -522.79,  aic = 1055.57
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 2))
##
## Coefficients:
##          ar1      ma1      ma2
##       0.0322  -0.7468  -0.2532
## s.e.  0.2064   0.1840   0.1827
##
## sigma^2 estimated as 625.3:  log likelihood = -535.54,  aic = 1079.08
##
## Call:
```

```
## arima(x = df_deseasoned_detrended, order = c(1, 0, 3))
##
## Coefficients:
##          ar1      ma1      ma2      ma3  intercept
##       0.0770  -0.1224  -0.3801  -0.4975     0.0080
## s.e.  0.1922   0.1741   0.1084   0.1116     0.1407
##
## sigma^2 estimated as 415.8:  log likelihood = -516.36,  aic = 1044.72
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 3))
##
## Coefficients:
##           ar1     ma1      ma2      ma3
##       -0.6627  0.0450  -0.6302  -0.4148
## s.e.   0.2160  0.2068   0.1654   0.1072
##
## sigma^2 estimated as 600.4:  log likelihood = -533.18,  aic = 1076.36

## [1] 1076.363
```

Best AIC is observed in the ARIMA model (1, 0, 3)

Best ARIMA model for 'Mendil Product':

```
ARIMA_proposing(mendil)
```

```
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 1))
##
## Coefficients:
##           ar1     ma1  intercept
##       -0.0684  0.5721     0.9858
## s.e.   0.1426  0.1049    20.9777
##
## sigma^2 estimated as 23696:  log likelihood = -748.99,  aic = 1505.99
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 1))
##
## Coefficients:
##          ar1      ma1
##       0.3094  -1.0000
## s.e.  0.0898   0.0224
##
## sigma^2 estimated as 27372:  log likelihood = -752.73,  aic = 1511.46
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 2))
##
## Coefficients:
##          ar1      ma1      ma2  intercept
##       0.4376  -0.3687  -0.6313    -0.0757
```

```
## s.e.  0.0953   0.0696   0.0675     1.0184
##
## sigma^2 estimated as 17445:  log likelihood = -733,  aic = 1476.01
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 2))
##
## Coefficients:
##           ar1      ma1      ma2
##       -0.0567  -0.4313  -0.5687
## s.e.   0.1432   0.1074   0.1058
##
## sigma^2 estimated as 23902:  log likelihood = -745.03,  aic = 1498.05
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 3))
##
## Coefficients:
##          ar1      ma1      ma2      ma3  intercept
##       0.1625  -0.0173  -0.6108  -0.3720    -0.0882
## s.e.  0.1536   0.1338   0.0877   0.1082     0.9599
##
## sigma^2 estimated as 16300:  log likelihood = -729.17,  aic = 1470.33
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 3))
##
## Coefficients:
##           ar1     ma1      ma2      ma3
##       -0.8711  0.4486  -0.9192  -0.5293
## s.e.   0.1931  0.1941   0.1046   0.1129
##
## sigma^2 estimated as 23695:  log likelihood = -744.5,  aic = 1499

## [1] 1499.002
```

Best AIC is observed in the ARIMA model (1, 0, 3)

Best ARIMA model for 'Diş Fırçası Product':

```
ARIMA_proposing(dis_fircasi)
```

```
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 1))
##
## Coefficients:
##           ar1     ma1  intercept
##       -0.1234  0.4415     1.2884
## s.e.   0.1957  0.1660     4.8151
##
## sigma^2 estimated as 1639:  log likelihood = -593.96,  aic = 1195.91
##
## Call:
```

```
## arima(x = df_deseasoned_detrended, order = c(1, 1, 1))
##
## Coefficients:
##          ar1      ma1
##       0.2390  -1.0000
## s.e.  0.0917   0.0221
##
## sigma^2 estimated as 1729:  log likelihood = -593.98,  aic = 1193.97
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 2))
##
## Coefficients:
##          ar1      ma1      ma2  intercept
##       0.4590  -0.5111  -0.4889     0.1899
## s.e.  0.1006   0.0861   0.0838     0.2580
##
## sigma^2 estimated as 1239:  log likelihood = -579.51,  aic = 1169.02

## Warning in log(s2): NaNs üretimi

##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 2))
##
## Coefficients:
##           ar1      ma1      ma2
##       -0.1092  -0.5648  -0.4352
## s.e.   0.1954   0.1667   0.1655
##
## sigma^2 estimated as 1653:  log likelihood = -591.46,  aic = 1190.92
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 0, 3))
##
## Coefficients:
##          ar1      ma1      ma2      ma3  intercept
##       0.0602  -0.0526  -0.4849  -0.4625     0.1626
## s.e.  0.1793   0.1553   0.0827   0.1349     0.2342
##
## sigma^2 estimated as 1159:  log likelihood = -575.8,  aic = 1163.61
##
## Call:
## arima(x = df_deseasoned_detrended, order = c(1, 1, 3))
##
## Coefficients:

## Warning in sqrt(diag(x$var.coef)): NaNs üretimi

##           ar1      ma1      ma2      ma3
##       -0.3053  -0.3435  -0.5499  -0.1066
## s.e.      NaN      NaN      NaN      NaN
##
## sigma^2 estimated as 1659:  log likelihood = -591.62,  aic = 1193.25
```

```
## [1] 1193.248
```

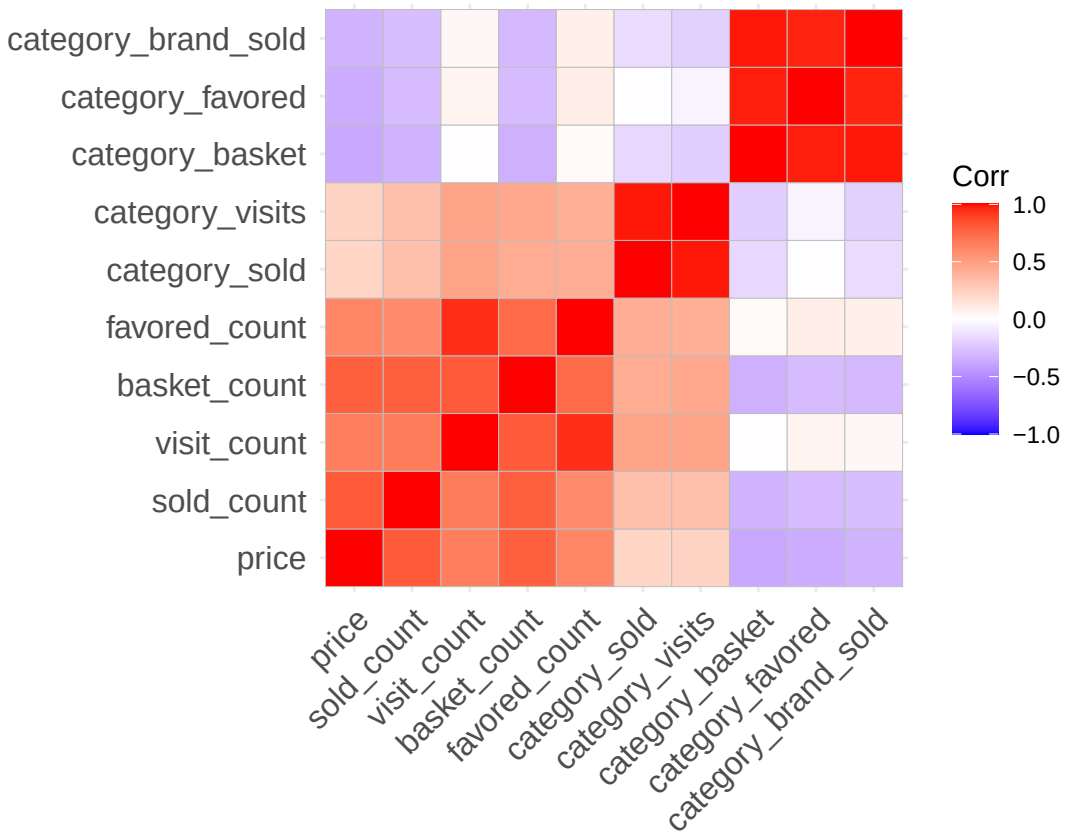
Best AIC is observed in the ARIMA model (1, 0, 3)

The ARIMA models with (1,0,3) parameters are the best performers in the most of the Trendyol products in terms of AIC.

Possible Regressors for Improving the Model

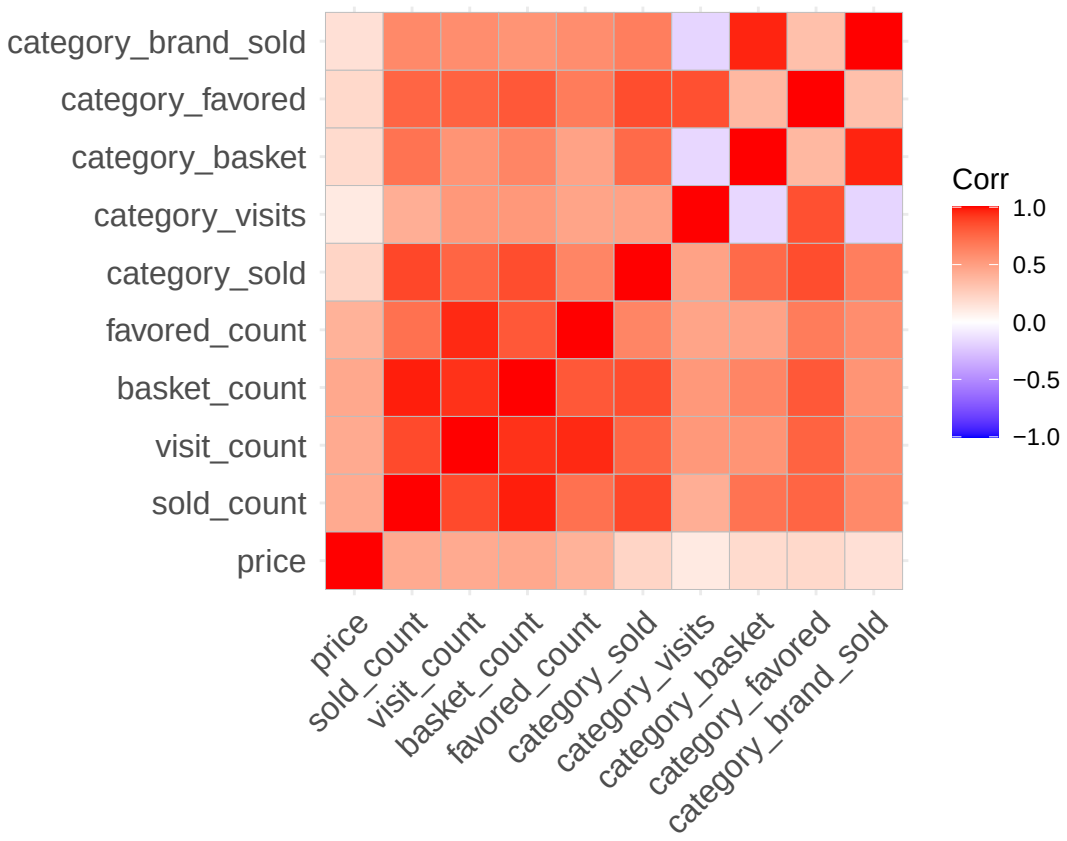
For deciding the possible regressors for each product, correlation matrix will be built and most positive and negative correlated variables will be selceted for the ARIMA models with external regressors.

```
ggcorrplot(cor(mont[,3:12]) )
```



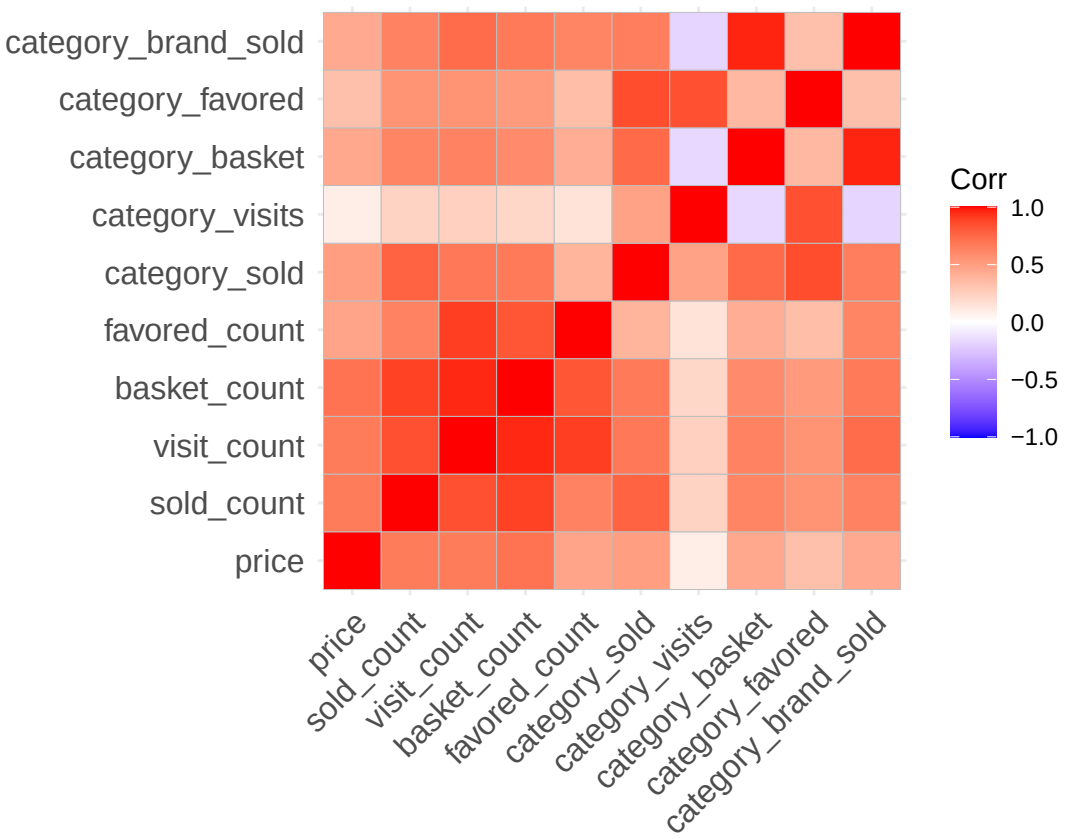
For 'Mont product', 'price' and 'basket_count' variables are the most correlated variables for the sold_count.

```
ggcorrplot(cor(bikini_1[,3:12]) )
```



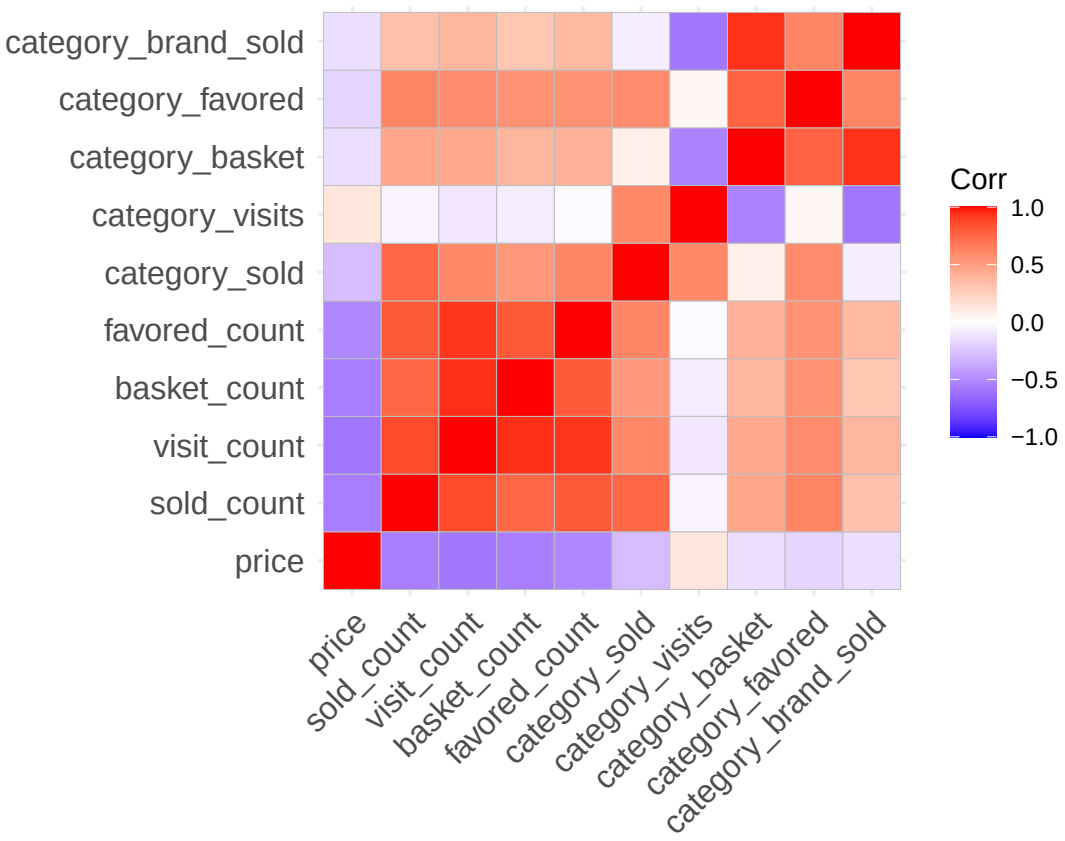
For 'Bikini_1 product', 'basket_count' and 'category_sold' variables are the most correlated variables for the sold_count.

```
ggcorrplot(cor(bikini_2[,3:12]) )
```



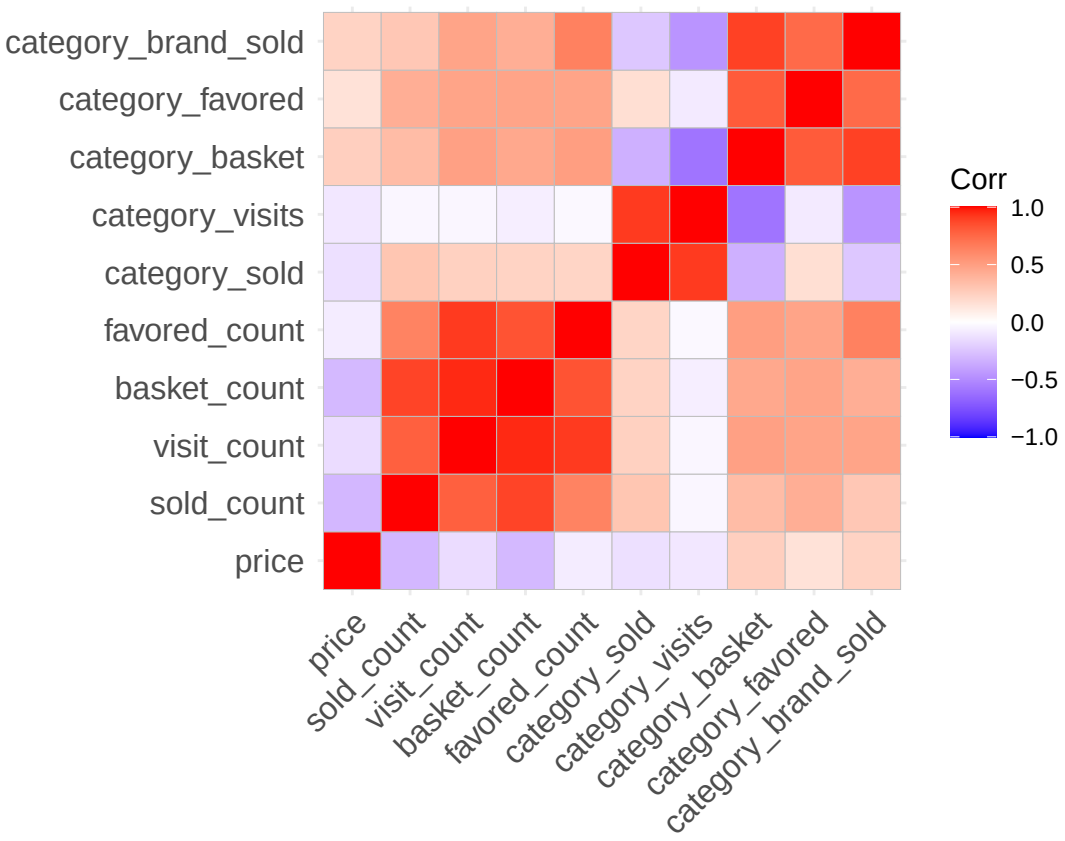
For 'Bikini_2 product', 'visit_count' and 'basket_count' variables are the most correlated variables for the sold_count.

```
ggcorrplot(cor(tayt[,3:12]) )
```



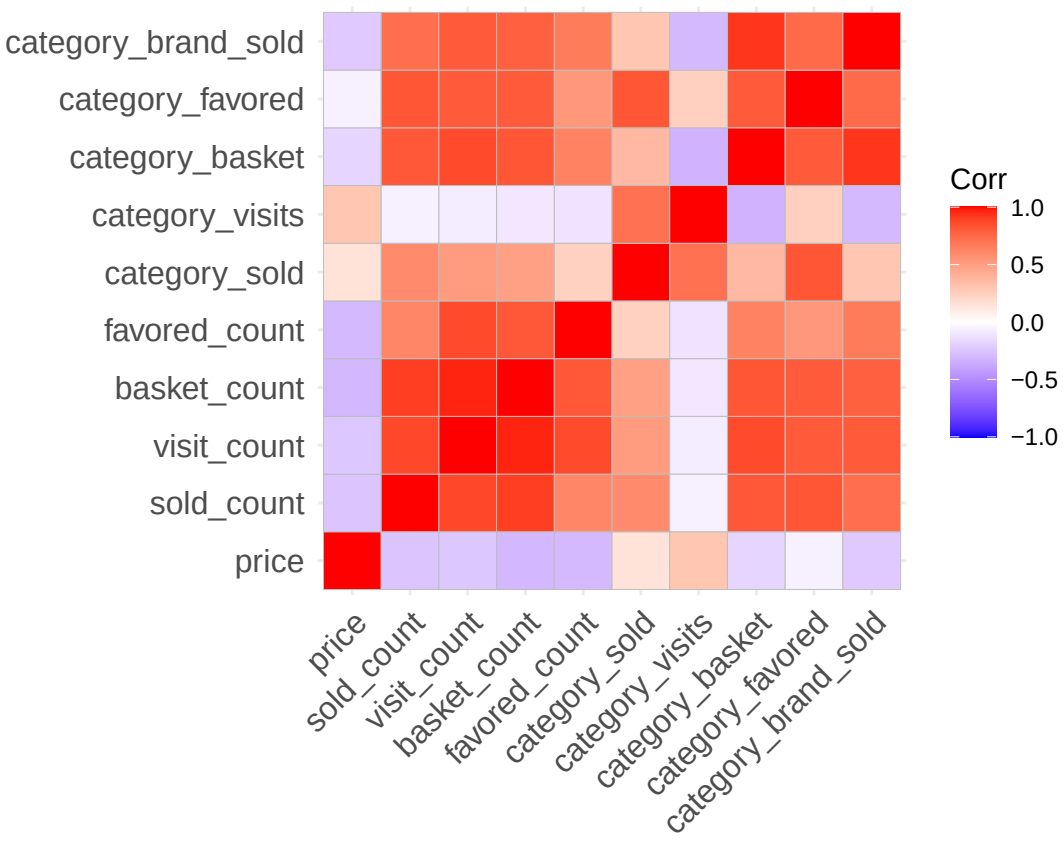
For 'Tayt product', 'price' and 'favored_count' variables are the most correlated variables for the sold_count.

```
ggcorrplot(cor(kulaklık[,3:12]) )
```



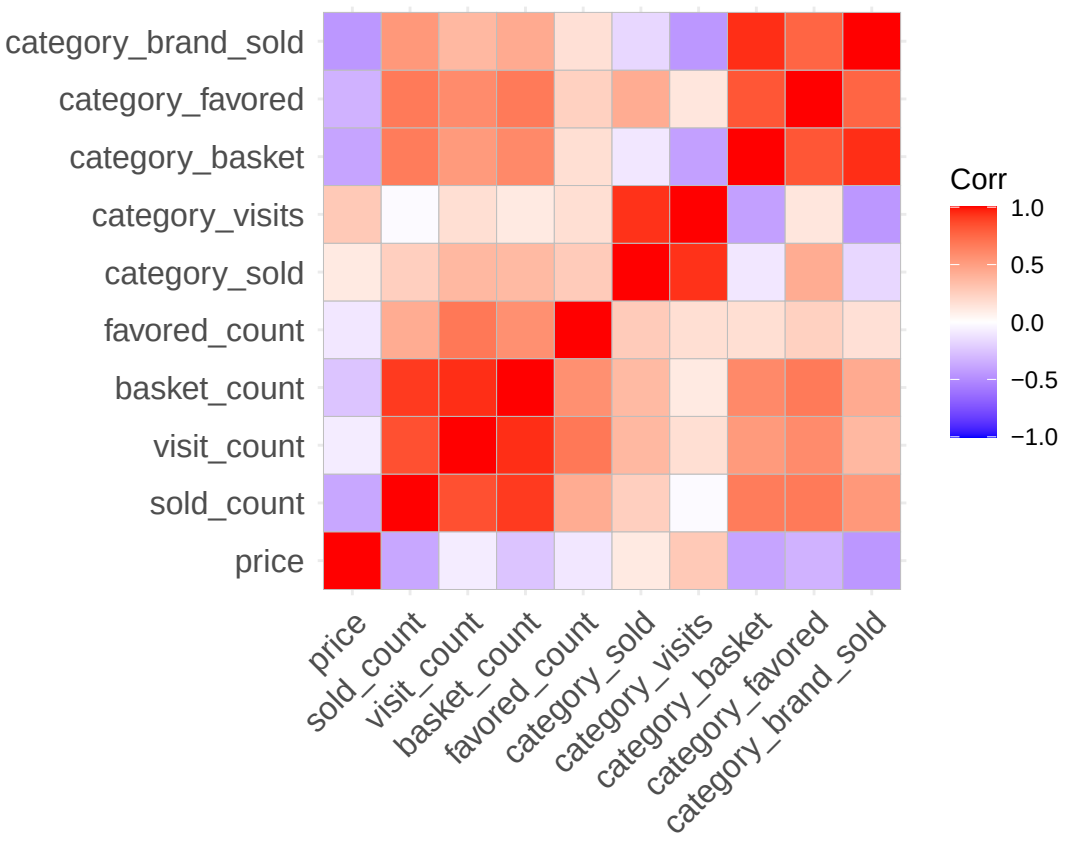
For 'Kulaklık product', 'price' and 'basket_count' variables are the most correlated variables for the sold_count.

```
ggcorrplot(cor(supurge[,3:12]) )
```



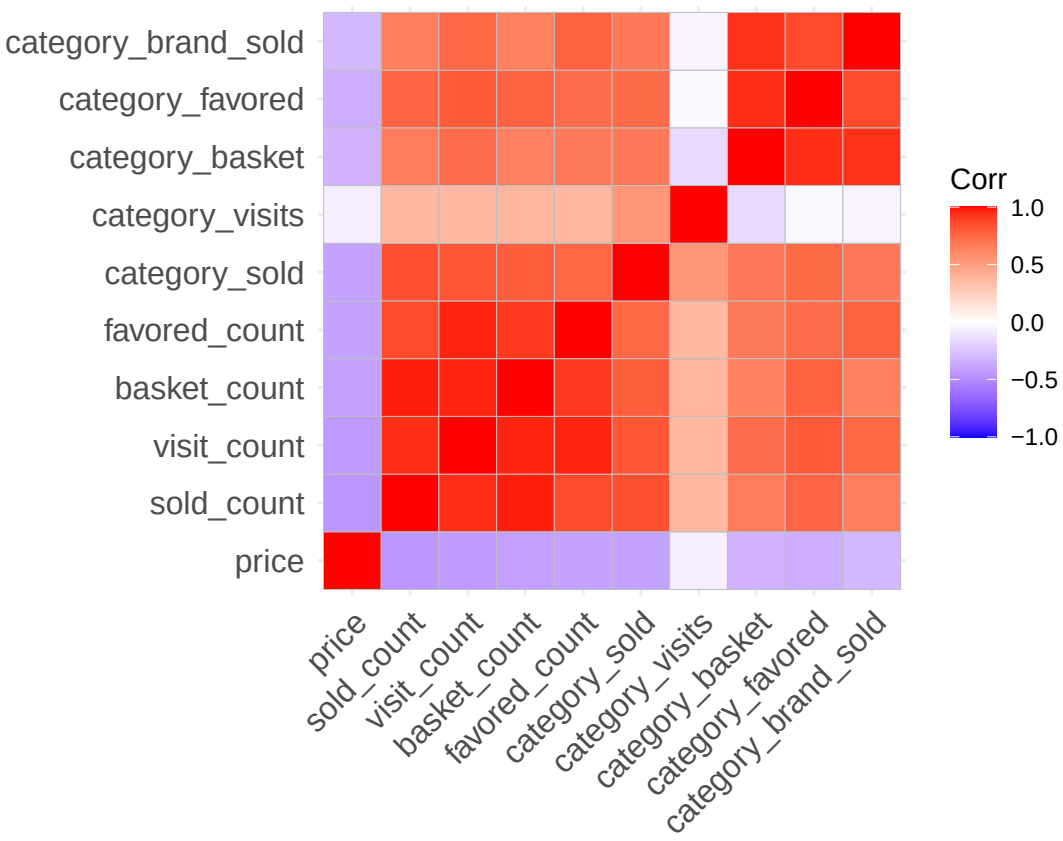
For 'Süpürge product', 'price' and 'visit_count' variables are the most correlated variables for the sold_count.

```
ggcorrplot(cor(yuz_temizleyicisi[,3:12]) )
```



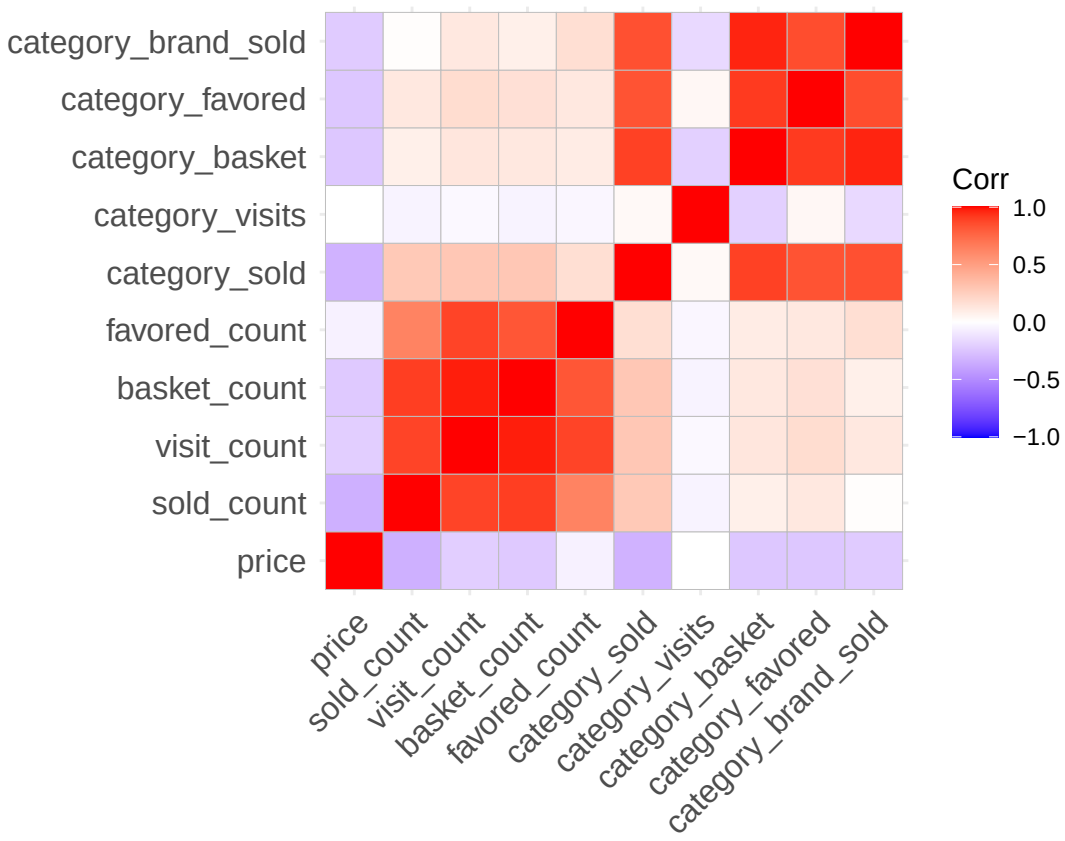
For 'Yüz Temizleyicisi product', 'price' and 'basket_count' variables are the most correlated variables for the sold_count.

```
ggcorrplot(cor(mendil[,3:12]) )
```



For 'Mendil product', 'price' and 'visit_count' variables are the most correlated variables for the sold_count.

```
ggcorrplot(cor(dis_fircasi[,3:12]) )
```



For 'Diş Fırçası product', 'price' and 'visit_count' variables are the most correlated variables for the sold_count.

To conclude this section, the price is negatively correlated for almost al products, i.e the price increase results in less products sales. Furthermore, basket and visit_count are positive correlated with total unit sold of the products.

Using the potential regressors in the selected ARIMA models

In this section, the best ARIMA model in terms of AIC will be compared with ARIMAX models. ARIMAX models will be built with most 2 positively or negatively correlated external regressors for each product. Functin for comparing the two models is below:

```
ARIMAX_function <- function(X, regressor_1,regressor_2,p,d,q){
  X <- X %>% map_df(rev)
  df_train <- X[!(X$event_date >= "2021-05-25"),]
  df_test <- X[!(X$event_date < "2021-05-25"),]
  df_new <- ts(df_train$sold_count,frequency=7)
  df_additive <- decompose(df_new,type="additive")
  df_deseasoned_detrended <- df_new-df_additive$seasonal-df_additive$trend
  df_deseasoned_detrended[1:3] <- mean(df_deseasoned_detrended[4:6])
  df_deseasoned_detrended[113:115] <- mean(df_deseasoned_detrended[110:112])
  forecasted_seasonal = (df_additive$seasonal)[108:114]
  forecasted_trend = rep(mean(tail(df_additive$trend[!is.na(df_additive$trend)],2)),7)

  pf_reg1<-forecast(regressor_1,h=7)$mean[1]
```

```
  pf_reg2<-forecast(regressor_2,h=7)$mean[1]
  newxreg <- as.matrix(cbind(pf_reg1, pf_reg2))
  matrix = cbind(regressor_1[1:115],regressor_2[1:115])

  old_model=arima(df_deseasoned_detrended,order=c(p,d,q))
  model_forecast_old <- predict(old_model, n.ahead = 7)$pred
  #print(model_forecast_old)
  print(AIC(old_model))

  fitted_arimax=auto.arima(df_deseasoned_detrended,xreg=matrix)
  model_forecast_new <- forecast(fitted_arimax,xreg=newxreg)$mean[1]
  model_forecast_new<- rep(model_forecast_new,7)
  #print(model_forecast_new)
  print(AIC(fitted_arimax))



  new_restored <- forecasted_seasonal+forecasted_trend+model_forecast_new
  old_restored <- forecasted_seasonal+forecasted_trend+model_forecast_old



  error_new=X$sold_count[116:122]-new_restored
  error_old=X$sold_count[116:122]-old_restored

  mean=mean(X$sold_count)
  MAD_2=sum(abs(error_old))/length(X$sold_count)
  WMAPE_old=MAD_2/mean
  print(WMAPE_old)

  mean=mean(X$sold_count)
  MAD_1=sum(abs(error_new))/length(X$sold_count)
  WMAPE_new=MAD_1/mean
  print(WMAPE_new)
}
```

For the 'Mont Product':

```
ARIMAX_function(mont,mont$price,mont$basket_count,1,0,2)
```

```
## [1] 149.6967

## Warning in forecast.forecast_ARIMA(fitted_arimax, xreg = newxreg): xreg contains
## different column names from the xreg used in training. Please check that the
## regressors are in the same order.

## [1] 143.2324
## [1] 0.1846941
## [1] 0.2353858
```

Here we see that arima model has 149 AIC and arimax model has 143 AIC. New model is slightly better. In terms of prediction , the new model suffers much more than old model. we see that old model's wmape

is around 18 pct and new model's wmape is around 23 pct. This may be occured from pointforecasting the regressor values since it accumulates the error.

For the 'Bikini_1 Product':

```
ARIMAX_function(bikini_1,bikini_1$basket_count,bikini_1$category_sold,1,0,3)
```

```
## [1] 951.8343
```

```
## Warning in forecast.forecast_ARIMA(fitted_arimax, xreg = newxreg): xreg contains
## different column names from the xreg used in training. Please check that the
## regressors are in the same order.
```

```
## [1] 944.04
## [1] 0.08170744
## [1] 0.09390802
```

Here we see that arima model has 951 AIC and arimax model has 944 AIC. New model is slightly better. In terms of prediction , the new model suffers much more than old model. we see that old model's wmape is around 8 pct and new model's wmape is around 9 pct. This may be occured from pointforecasting the regressor values since it accumulates the error same as in previous one.

For the 'Bikini_2 Product':

```
ARIMAX_function(bikini_2,bikini_2$visit_count,bikini_2$basket_count,1,0,3)
```

```
## [1] 797.075
```

```
## Warning in forecast.forecast_ARIMA(fitted_arimax, xreg = newxreg): xreg contains
## different column names from the xreg used in training. Please check that the
## regressors are in the same order.
```

```
## [1] 845.5657
## [1] 0.01756959
## [1] 0.01893382
```

Here we see that arima model has 797 AIC and arimax model has 845 AIC. New model is slightly worse. In terms of prediction , the new model suffers more than old model. we see that old model's wmape is around 1.7 pct and new model's wmape is around 1.8 pct. This may be occured from pointforecasting the regressor values since it accumulates the error.

For the 'Tayt Product':

```
ARIMAX_function(tayt,tayt$price,tayt$favored_count,1,0,3)
```

```
## [1] 1613.902
```

```
## Warning in forecast.forecast_ARIMA(fitted_arimax, xreg = newxreg): xreg contains
## different column names from the xreg used in training. Please check that the
## regressors are in the same order.
```

```
## [1] 1639.689
## [1] 0.01965264
## [1] 0.01470704
```

Here we see that arima model has 1613 AIC and arimax model has 1639 AIC. New model is slightly worse. In terms of prediction , the new model predicts much better than old model. we see that old model's wmape is around 1.9 pct and new model's wmape is around 1.4 pct.

For the 'Kulaklık Product':

```
ARIMAX_function(kulaklık,kulaklık$price,kulaklık$basket_count,1,0,3)
```

```
## [1] 1370.454

## Warning in forecast.forecast_ARIMA(fitted_arimax, xreg = newxreg): xreg contains
## different column names from the xreg used in training. Please check that the
## regressors are in the same order.

## [1] 1390.838
## [1] 0.02519358
## [1] 0.01110617
```

Here we see that arima model has 1370 AIC and arimax model has 1390 AIC. New model is slightly worse. In terms of prediction , the new model predicts much better than old model. we see that old model's wmape is around 2.5 pct and new model's wmape is around 1.1 pct.

For the 'Süpürge Product':

```
ARIMAX_function(supurge,supurge$price,supurge$visit_count,1,0,2)
```

```
## [1] 753.9106

## Warning in forecast.forecast_ARIMA(fitted_arimax, xreg = newxreg): xreg contains
## different column names from the xreg used in training. Please check that the
## regressors are in the same order.

## [1] 769.2982
## [1] 0.01188456
## [1] 0.01182937
```

Here we see that arima model has 753 AIC and arimax model has 769 AIC. New model is slightly worse. In terms of prediction , the new model predicts slightly better than old model. we see that old model's wmape is around 1.1 pct and new model's wmape is around 1.1 pct.

For the 'Yüz Temizleyicisi Product':

```
ARIMAX_function(yuz_temizleyicisi,yuz_temizleyicisi$price,yuz_temizleyicisi$basket_count,1,0,3)
```

```
## [1] 1038.381

## Warning in forecast.forecast_ARIMA(fitted_arimax, xreg = newxreg): xreg contains
## different column names from the xreg used in training. Please check that the
## regressors are in the same order.
```

```
## [1] 1066.679
## [1] 0.01478238
## [1] 0.01235416
```

Here we see that arima model has 1038 AIC and arimax model has 1066 AIC. New model is slightly worse. In terms of prediction , the new model predicts much better than old model. we see that old model's wmape is around 1.4 pct and new model's wmape is around 1.2 pct.

For the 'Mendil Product':

```
ARIMAX_function(mendil,mendil$price,mendil$visit_count,1,0,3)
```

```
## [1] 1464.796

## Warning in forecast.forecast_ARIMA(fitted_arimax, xreg = newxreg): xreg contains
## different column names from the xreg used in training. Please check that the
## regressors are in the same order.

## [1] 1479.522
## [1] 0.03463817
## [1] 0.03178024
```

Here we see that arima model has 1464 AIC and arimax model has 1479 AIC. New model is slightly worse. In terms of prediction , the new model predicts much better than old model. we see that old model's wmape is around 3.4 pct and new model's wmape is around 3.1 pct.

For the 'Diş Fırçası Product':

```
ARIMAX_function(dis_fircasi,dis_fircasi$price,dis_fircasi$visit_count,1,1,2)
```

```
## [1] 1172.028

## Warning in forecast.forecast_ARIMA(fitted_arimax, xreg = newxreg): xreg contains
## different column names from the xreg used in training. Please check that the
## regressors are in the same order.

## [1] 1175.958
## [1] 0.01759382
## [1] 0.0157276
```

Here we see that arima model has 1172 AIC and arimax model has 1175 AIC. New model is slightly worse. In terms of prediction , the new model predicts better than old model. we see that old model's wmape is around 1.7 pct and new model's wmape is around 1.5 pct.

6. Conclusion

In conclusion, we see that when the model is improved then we see that the prediction gets worse but when the model gets worse the prediction gives us better results. This may be occured from not selecting the correct regressors or accumulated error in the point forecasting.Overall, we investigated the sales quantity data from Trendyol and applied some statistical analysis and furthermore we predicted last 7 days' sales quantity and compared them on the test data.ARIMAX is better than ARIMA in general in terms of performance, but in our case , this was not entirely true. One more reason could be having some data points which equals to zero sales quantity, this may gave us some non-ideal results.All in all, other than mentioned reasons we can say that arimax performed better.