Group 3 Project_ Rossmann Sales

Emma

3/1/2021

Prepare the Data

- Format the data

Data Exploration

- Annual Sales
- Monthly Aggregated Sales
- Aggregate Days of Week Sales

Time Series Analysis

- Correlation Heatmap
- Spectrum Analysis
- Decompositon of the Data
- Stationary

Models

- Differencing and EACF
- Auto ARIMA
- Comparison between Auto Arima and Bestfit_arima6
- Moving Average Smoothing
- Holt-Winter Model
- TBATS
- Fourier Term

```
library(fpp)
```

```
## Loading required package: forecast
```

```
## Warning: package 'forecast' was built under R version 3.6.2
```

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

```
## Loading required package: fma
```

```
## Loading required package: expsmooth
```

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

```
## Warning: package 'lmtest' was built under R version 3.6.2
```

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

```
## Warning: package 'zoo' was built under R version 3.6.2
```

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

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

```
## Loading required package: tseries
```

```
## Warning: package 'tseries' was built under R version 3.6.2
```

```
library(anytime)
```

```
## Warning: package 'anytime' was built under R version 3.6.2
```

```
library(lubridate)# for year function
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tseries)
library(fpp)
library(ggplot2)
library(scales) # for pretty breaks plot
library(TSA)
```

```
## Warning: package 'TSA' was built under R version 3.6.2
```

```
## Registered S3 methods overwritten by 'TSA':
##   method       from
##   fitted.Arima forecast
##   plot.Arima   forecast
```

```
##
## Attaching package: 'TSA'
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

```
## The following object is masked from 'package:utils':
##
##     tar
```

```
source('/Users/emmali/Library/Mobile Documents/com~apple~CloudDocs/Time Series/TSA_1.2/TSA/R/eacf.R')
setwd('/Users/emmali/Library/Mobile Documents/com~apple~CloudDocs/Time Series/project/rossmann-store-sales/')
train <- read.csv('train.csv')
test <- read.csv('test.csv')
store <- read.csv('store.csv')
head(train)
```

```
##   Store DayOfWeek       Date Sales Customers Open Promo StateHoliday
## 1     1         5 2015-07-31  5263       555    1     1            0
## 2     2         5 2015-07-31  6064       625    1     1            0
## 3     3         5 2015-07-31  8314       821    1     1            0
## 4     4         5 2015-07-31 13995      1498    1     1            0
## 5     5         5 2015-07-31  4822       559    1     1            0
## 6     6         5 2015-07-31  5651       589    1     1            0
##   SchoolHoliday
## 1             1
## 2             1
## 3             1
## 4             1
## 5             1
## 6             1
```

```
#check na values
sum(is.na(train$sales)) # Returns 0 na values so good to go.
```

```
## [1] 0
```

Prepare the Data

Summary:

we have 1115 stores, 4 store type(a,b,c,d) and 3 Assortment # Store 3

```
summary(store)
```

```
##      Store        StoreType Assortment CompetitionDistance
##  Min.   :   1.0   a:602     a:593      Min.   :   20.0
##  1st Qu.: 279.5   b: 17     b:  9      1st Qu.:  717.5
##  Median : 558.0   c:148     c:513      Median : 2325.0
##  Mean   : 558.0   d:348                Mean   : 5404.9
##  3rd Qu.: 836.5                        3rd Qu.: 6882.5
##  Max.   :1115.0                        Max.   :75860.0
##                                        NA's   :3
##  CompetitionOpenSinceMonth CompetitionOpenSinceYear     Promo2
##  Min.   : 1.000            Min.   :1900             Min.   :0.0000
##  1st Qu.: 4.000            1st Qu.:2006             1st Qu.:0.0000
##  Median : 8.000            Median :2010             Median :1.0000
##  Mean   : 7.225            Mean   :2009             Mean   :0.5121
##  3rd Qu.:10.000            3rd Qu.:2013             3rd Qu.:1.0000
##  Max.   :12.000            Max.   :2015             Max.   :1.0000
##  NA's   :354               NA's   :354
##  Promo2SinceWeek Promo2SinceYear          PromoInterval
##  Min.   : 1.0    Min.   :2009                    :544
##  1st Qu.:13.0    1st Qu.:2011    Feb,May,Aug,Nov :130
##  Median :22.0    Median :2012    Jan,Apr,Jul,Oct :335
##  Mean   :23.6    Mean   :2012    Mar,Jun,Sept,Dec:106
##  3rd Qu.:37.0    3rd Qu.:2013
##  Max.   :50.0    Max.   :2015
##  NA's   :544     NA's   :544
```

```
#store[which(store$StoreType =='a'),]
#train[which(train$Open ==0),]
#train[which(train$SchoolHoliday==1),]
#train[which(train$Sales ==0),]
#summary(train)
#sales_total<- aggregate(Sales~Date,  train, FUN = mean) # we could use mean or sum #
####
# check data point for each store

train %>% count(Store,sort = TRUE)
```

```
## # A tibble: 1,115 x 2
##    Store     n
##    <int> <int>
##  1     1   942
##  2     2   942
##  3     3   942
##  4     4   942
##  5     5   942
##  6     6   942
##  7     7   942
##  8     8   942
##  9     9   942
## 10    10   942
## # … with 1,105 more rows
```

```
#unique(train[which(train$Open==0),]$DayOfWeek)
#unique(train[which(train$DayOfWeek==7),]$Sales)
#temp = train %>% filter(Store==8) %>% arrange(Date)
#unique(temp[which(temp$DayOfWeek==7),]$Sales)

#unique(train[which(train$StateHoliday==1),]$Sales)
#unique(train[which(train$SchoolHoliday==1),]$Sales)
```

Choose store 8 as sample

possible store chosen criteria: no close during Holiday

```
#temp = train %>% filter(Store==85) %>% arrange(Date)
#temp[which(temp$DayOfWeek==7),]$Sales
##unique(temp$Open)
#temp_ts<-ts(temp, f = 7)
#temp_ts
#autoplot(temp_ts[,4])
#tsdisplay(temp_ts[,4])
```

Format the data

```
data <- read.csv('train.csv')
store_8<-data %>% filter(Store==8) %>% arrange(Date)

# Convert Date column to type "Date", get the first day of our data
store_8$Date = as.Date(store_8$Date, format = "%Y-%m-%d")

# Convert "03-01" to day of the year
dayOfYear = as.numeric(format(store_8[1,3], "%j"))

#keep the date information for future use
store_8$Date = anydate(store_8$Date)
store_8$year = year(store_8$Date)
store_8$month = month(store_8$Date)
store_8$week <- week(store_8$Date)
store_8$day = day(store_8$Date)

# double check 365 dates each year without missing
store_8 %>% count(year,sort = TRUE)
```

```
## # A tibble: 3 x 2
##    year     n
##   <dbl> <int>
## 1  2013   365
## 2  2014   365
## 3  2015   212
```

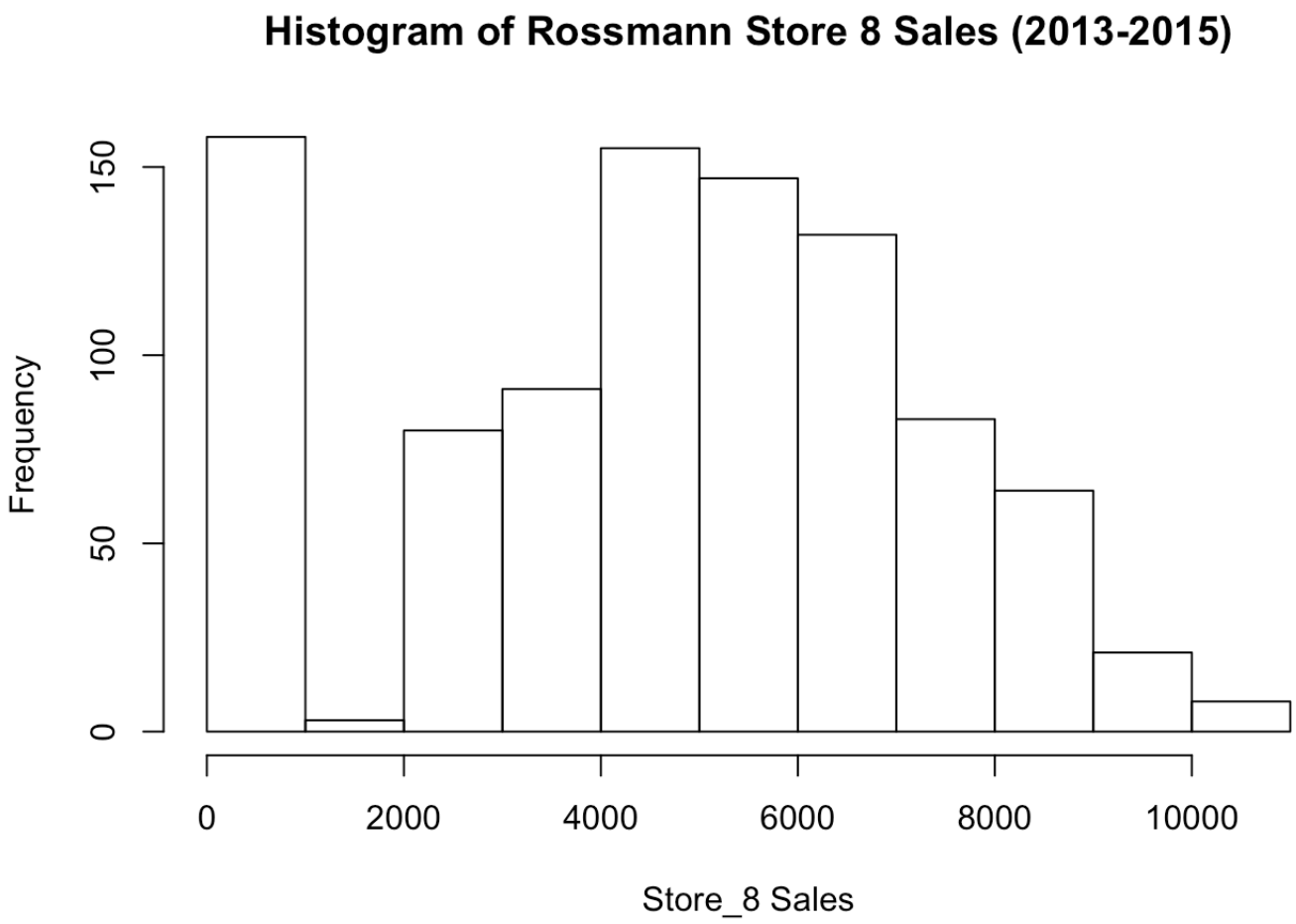
```
#take a look at the data: no missing data
summary(store_8)
```

```
##      Store     DayOfWeek          Date                Sales
##  Min.   :8   Min.   :1.000   Min.   :2013-01-01   Min.   :    0
##  1st Qu.:8   1st Qu.:2.000   1st Qu.:2013-08-24   1st Qu.: 2917
##  Median :8   Median :4.000   Median :2014-04-16   Median : 4919
##  Mean   :8   Mean   :3.998   Mean   :2014-04-16   Mean   : 4610
##  3rd Qu.:8   3rd Qu.:6.000   3rd Qu.:2014-12-07   3rd Qu.: 6558
##  Max.   :8   Max.   :7.000   Max.   :2015-07-31   Max.   :10971
##    Customers           Open            Promo        StateHoliday
##  Min.   :   0.0   Min.   :0.0000   Min.   :0.0000   0:918
##  1st Qu.: 374.2   1st Qu.:1.0000   1st Qu.:0.0000   a: 14
##  Median : 630.5   Median :1.0000   Median :0.0000   b:  6
##  Mean   : 547.8   Mean   :0.8323   Mean   :0.3822   c:  4
##  3rd Qu.: 759.8   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1133.0   Max.   :1.0000   Max.   :1.0000
##  SchoolHoliday         year          month             week
##  Min.   :0.0000   Min.   :2013   Min.   : 1.000   Min.   : 1.00
##  1st Qu.:0.0000   1st Qu.:2013   1st Qu.: 3.000   1st Qu.:12.00
##  Median :0.0000   Median :2014   Median : 6.000   Median :23.00
##  Mean   :0.1688   Mean   :2014   Mean   : 5.962   Mean   :24.11
##  3rd Qu.:0.0000   3rd Qu.:2014   3rd Qu.: 9.000   3rd Qu.:36.00
##  Max.   :1.0000   Max.   :2015   Max.   :12.000   Max.   :53.00
##       day
##  Min.   : 1.00
##  1st Qu.: 8.00
##  Median :16.00
##  Mean   :15.71
##  3rd Qu.:23.00
##  Max.   :31.00
```

```
#drop store number
store_8 <- subset(store_8, select=-1)
# adjust the col sequence
store_8 <- subset(store_8,select= c(2:ncol(store_8),1))
```

Data Exploration

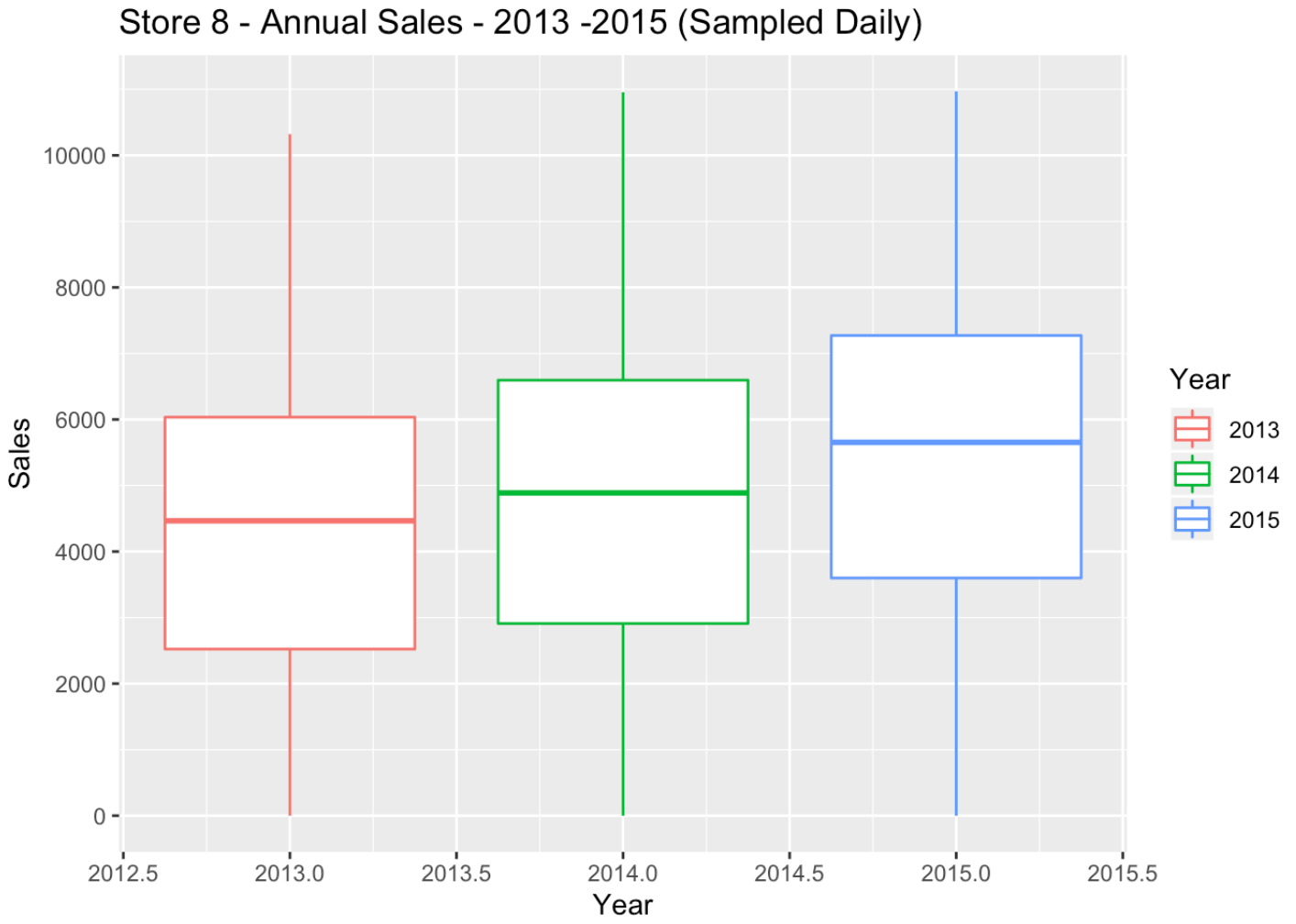
```
hist(store_8$Sales, xlab="Store_8 Sales", main = 'Histogram of Rossmann Store 8 Sales
(2013-2015)')
```



From the histogram we could see there is large portion of the 0 value which means it is the store closed due to some reasons.

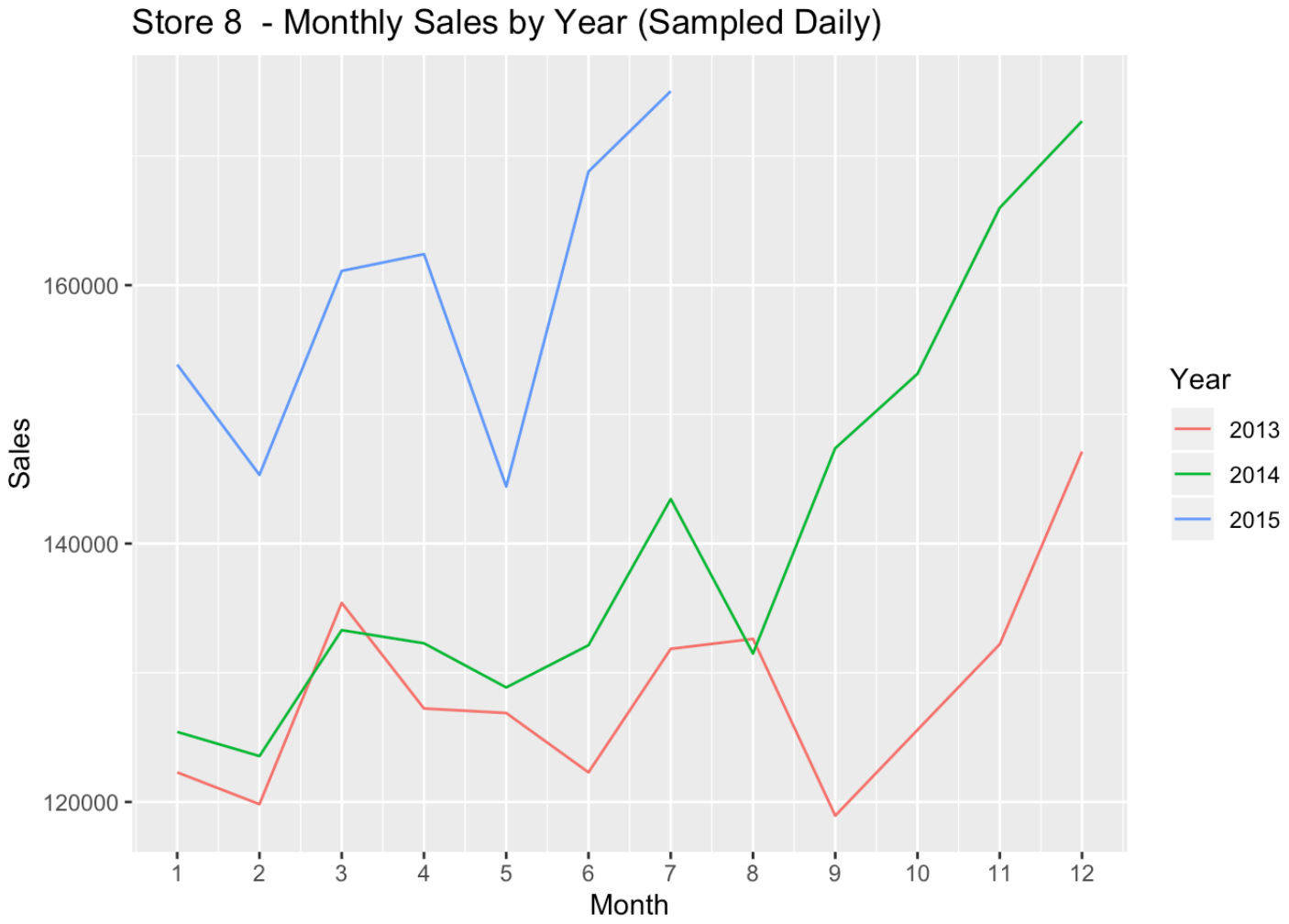
Annual Sales

```
title = 'Store 8 - Annual Sales - 2013 -2015 (Sampled Daily)'
ym_box = ggplot(store_8, aes(x = year, y = Sales, color = factor(sort(year)))) + geom
_boxplot()
print(ym_box + ggtitle(title) + labs(y = 'Sales', x = 'Year', color ='Year') + scale_
x_continuous(breaks = pretty_breaks())+ scale_y_continuous(breaks = pretty_breaks() )
)
```



Monthly Aggregate Sales

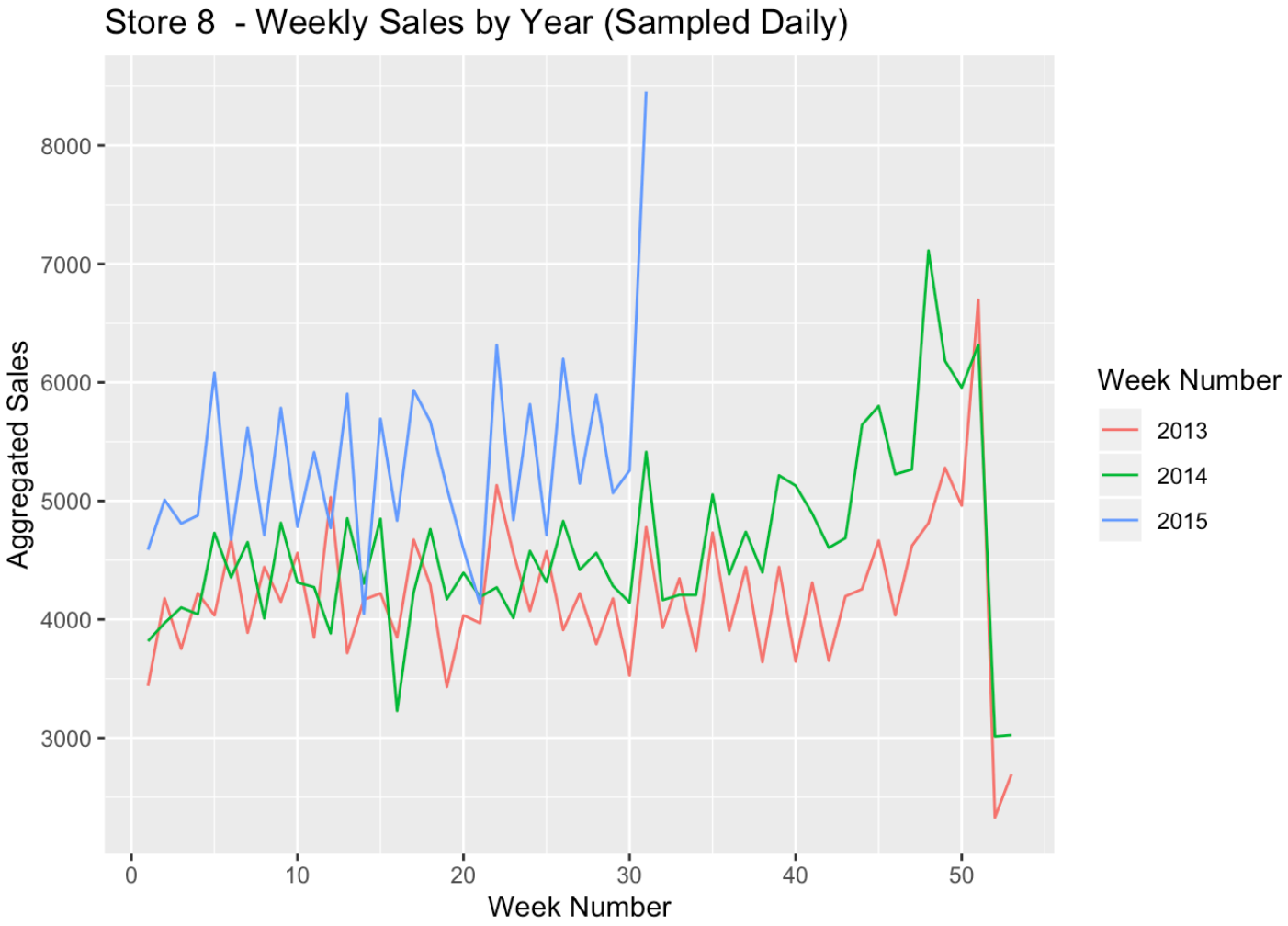
```
title = 'Store 8  - Monthly Sales by Year (Sampled Daily)'
ym = subset(store_8, select =c('year','month','Sales'))
ym = aggregate(Sales~year+month, ym, FUN = sum)
ym_line = ggplot(ym, aes(x = month, y =Sales, color = as.character(year))) + geom_lin
e()
print(ym_line + ggtitle(title) + labs(y = 'Sales', x = 'Month', color='Year') + scale
_x_continuous(breaks = seq(1, 12, by = 1)))
```



From monthly plot, there is a trend from 2013-2015. and it looks End of Year, the sales will go up and considering the flu season, it makes sense. in the future works, if we could have more data, we could including this monthly seasonality.

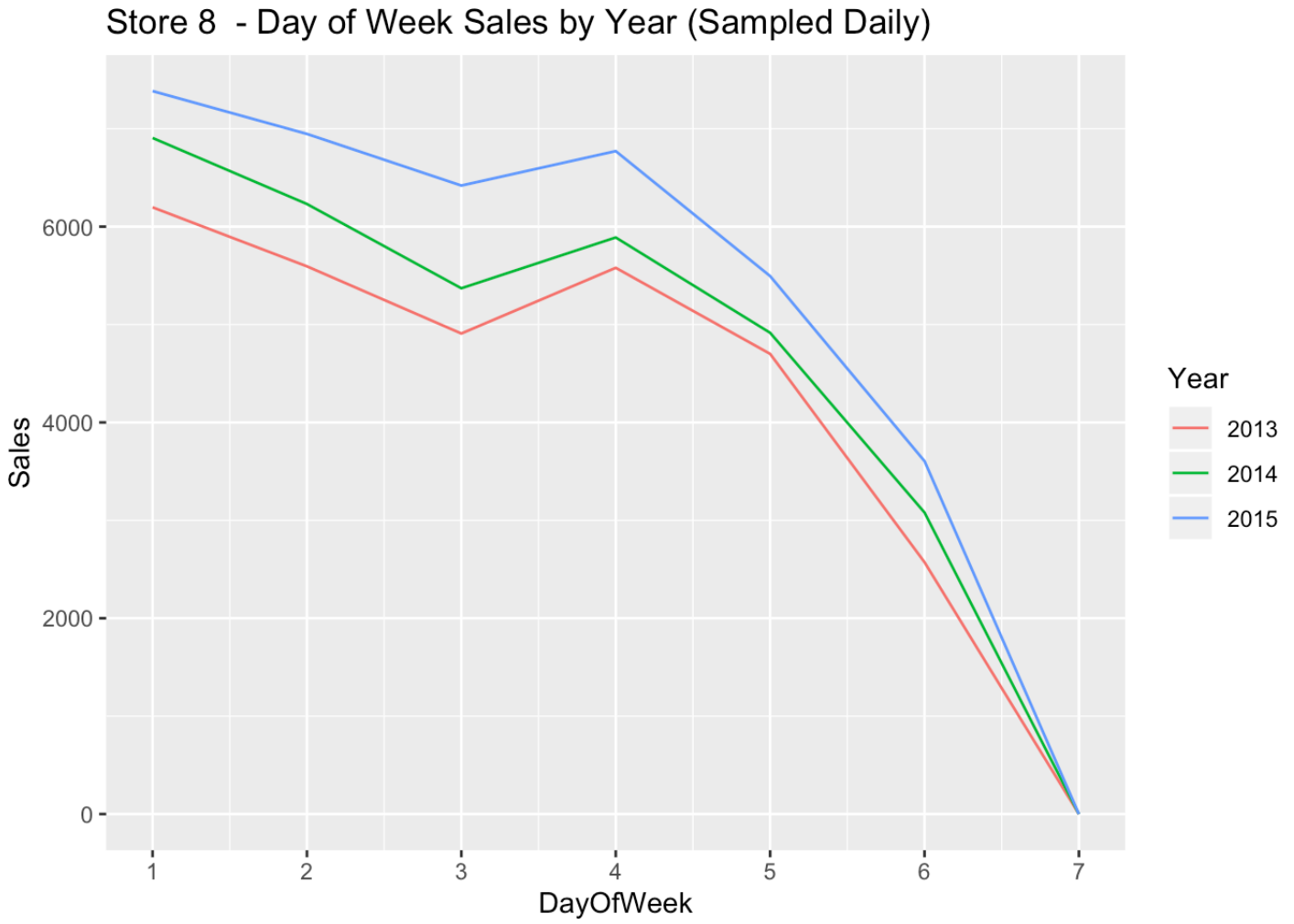
Weekly Aggreageted Sales

```
title = 'Store 8  - Weekly Sales by Year (Sampled Daily)'
ym = subset(store_8, select = c('year',  'week', 'Sales'))
ym = aggregate(Sales ~ year +  week, ym, FUN = mean)
ym_line= ggplot(ym, aes(x = week, y = Sales, color = as.character(year))) + geom_line
()
print(ym_line + ggtitle(title)+labs(y = 'Aggregated Sales', x = 'Week Number', color
='Week Number') +scale_x_continuous(breaks  = pretty_breaks())+ scale_y_continuous(br
eaks = pretty_breaks()))
```



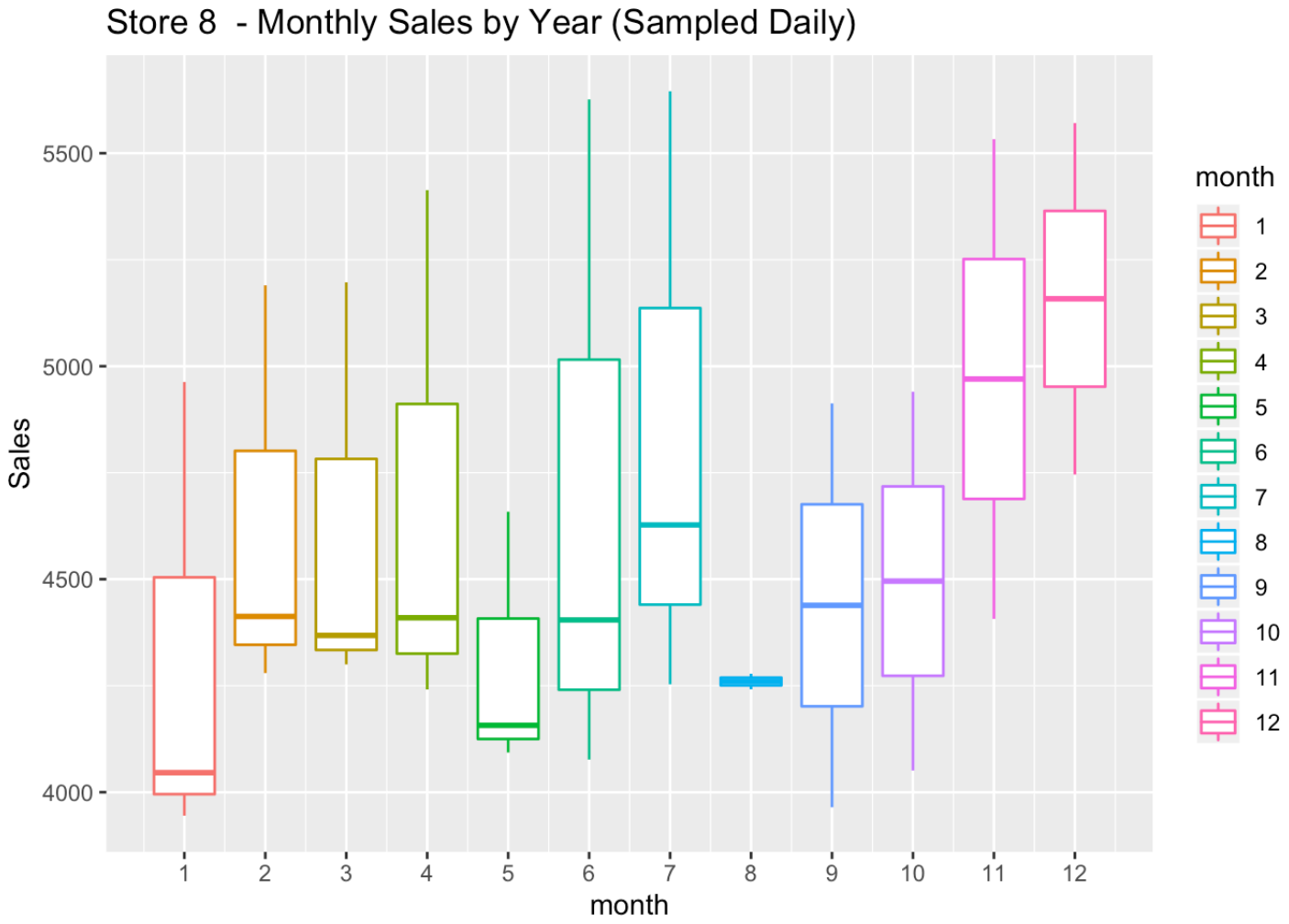
Aggregate Days of Week Sales

```
title = 'Store 8  - Day of Week Sales by Year (Sampled Daily)'
ym = subset(store_8, select =c('week','year','DayOfWeek','Sales'))
ym = aggregate(Sales~year+DayOfWeek, ym, FUN = mean)
ym_line = ggplot(ym, aes(x = DayOfWeek, y =Sales, color = as.character(year))) + geom
_line()
print(ym_line + ggtitle(title) + labs(y = 'Sales', x = 'DayOfWeek', color='Year') + s
cale_x_continuous(breaks = seq(1, 7, by = 1)))
```



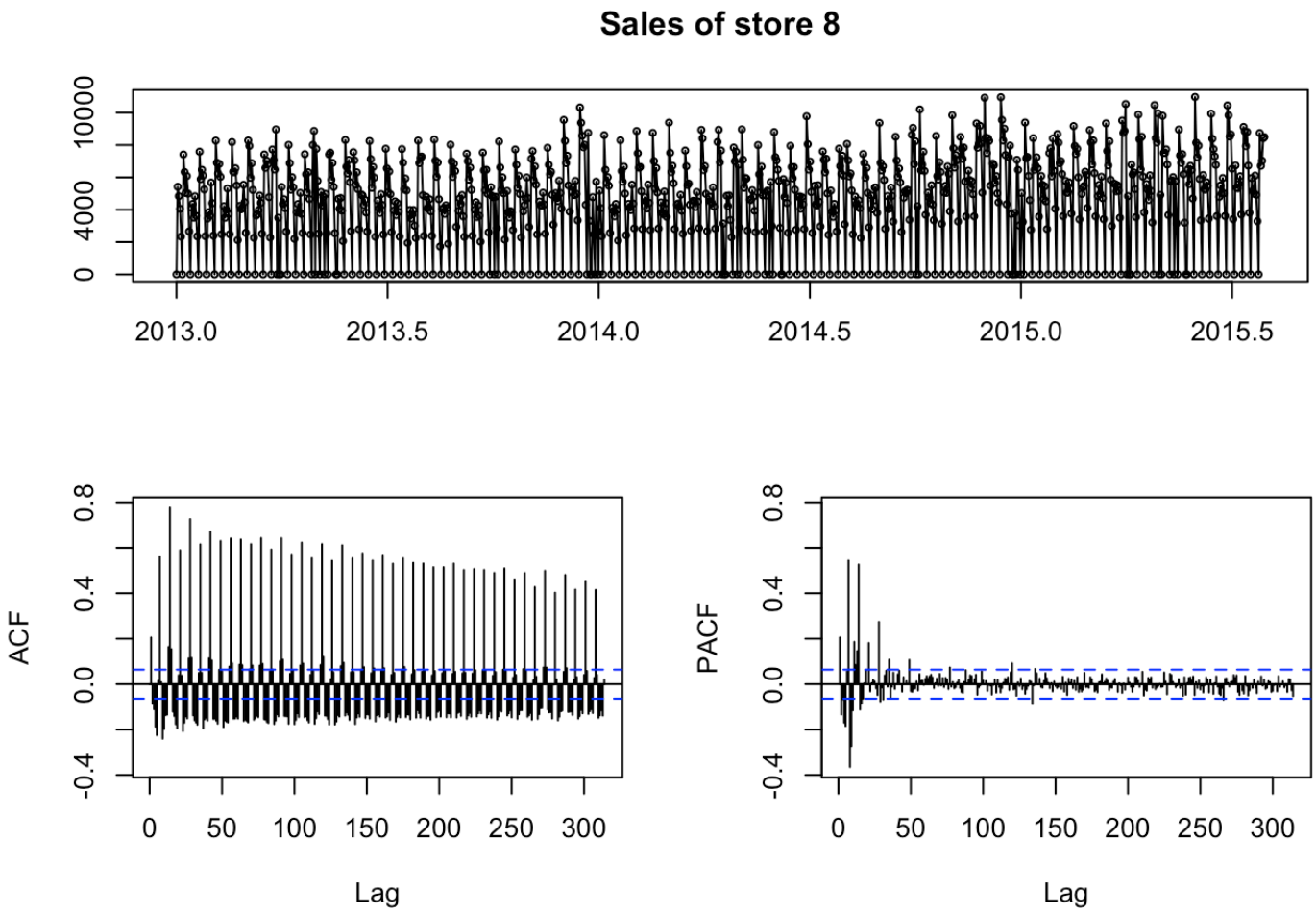
Day of week is a important factor box plot for variation

```
title = 'Store 8  - Monthly Sales by Year (Sampled Daily)'
ym = subset(store_8, select =c('year','month','Sales'))
ym = aggregate(Sales~year+month, ym, FUN = mean)
ym_box = ggplot(ym, aes(x = month, y = Sales, color = factor(sort(month)))) + geom_bo
xplot()
print(ym_box + ggtitle(title) + labs(y = 'Sales', x = 'month', color ='month') + scal
e_x_continuous(breaks = seq(1, 12, by = 1)))
```



Time Series Analysis

```
#convert to time series data
store_8_ts <- ts(store_8, start = c(2013, dayOfYear),frequency = 365.25)
tsdisplay(store_8_ts[,2], main = 'Sales of store 8')
```

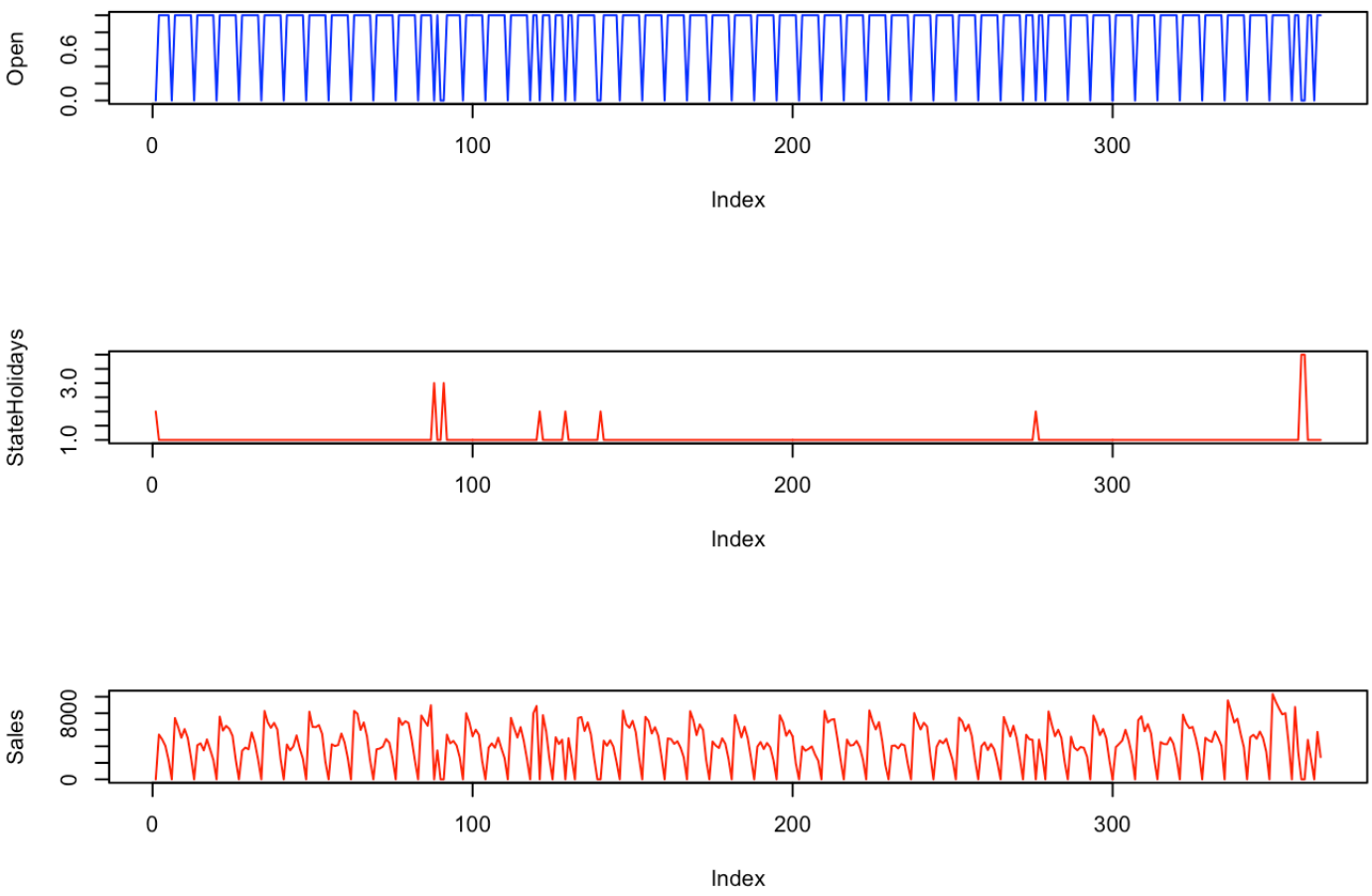



```
#split train test data
train <- window(store_8_ts, start = c(2013, 1), end = c(2015, 23))
test <-window(store_8_ts, start = c(2015, 24))
```

Sales and Opens

Store closed will lead the sales to 0

```
par(mfrow = c(3,1))
plot(train[1:365,4], type = 'l', col = 'blue', ylab = 'Open' )
plot(train[1:365,6],type = 'l', col = 'red',ylab='StateHolidays')
plot(train[1:365, 2], type = 'l', col = 'red',ylab='Sales')
```



on stateholiday, the store have 0 sales.

Correlation Heatmap

```
cormat <- round(cor(train[,c(2:7,12)]),2)
head(cormat)
```

```
##               Sales Customers  Open Promo StateHoliday SchoolHoliday
## Sales          1.00      0.97  0.77  0.70        -0.25          0.13
## Customers      0.97      1.00  0.84  0.59        -0.27          0.13
## Open           0.77      0.84  1.00  0.31        -0.32          0.10
## Promo          0.70      0.59  0.31  1.00        -0.05          0.08
## StateHoliday  -0.25     -0.27 -0.32 -0.05         1.00          0.22
## SchoolHoliday  0.13      0.13  0.10  0.08         0.22          1.00
##               DayOfWeek
## Sales             -0.72
## Customers         -0.73
## Open              -0.55
## Promo             -0.39
## StateHoliday      -0.04
## SchoolHoliday     -0.23
```

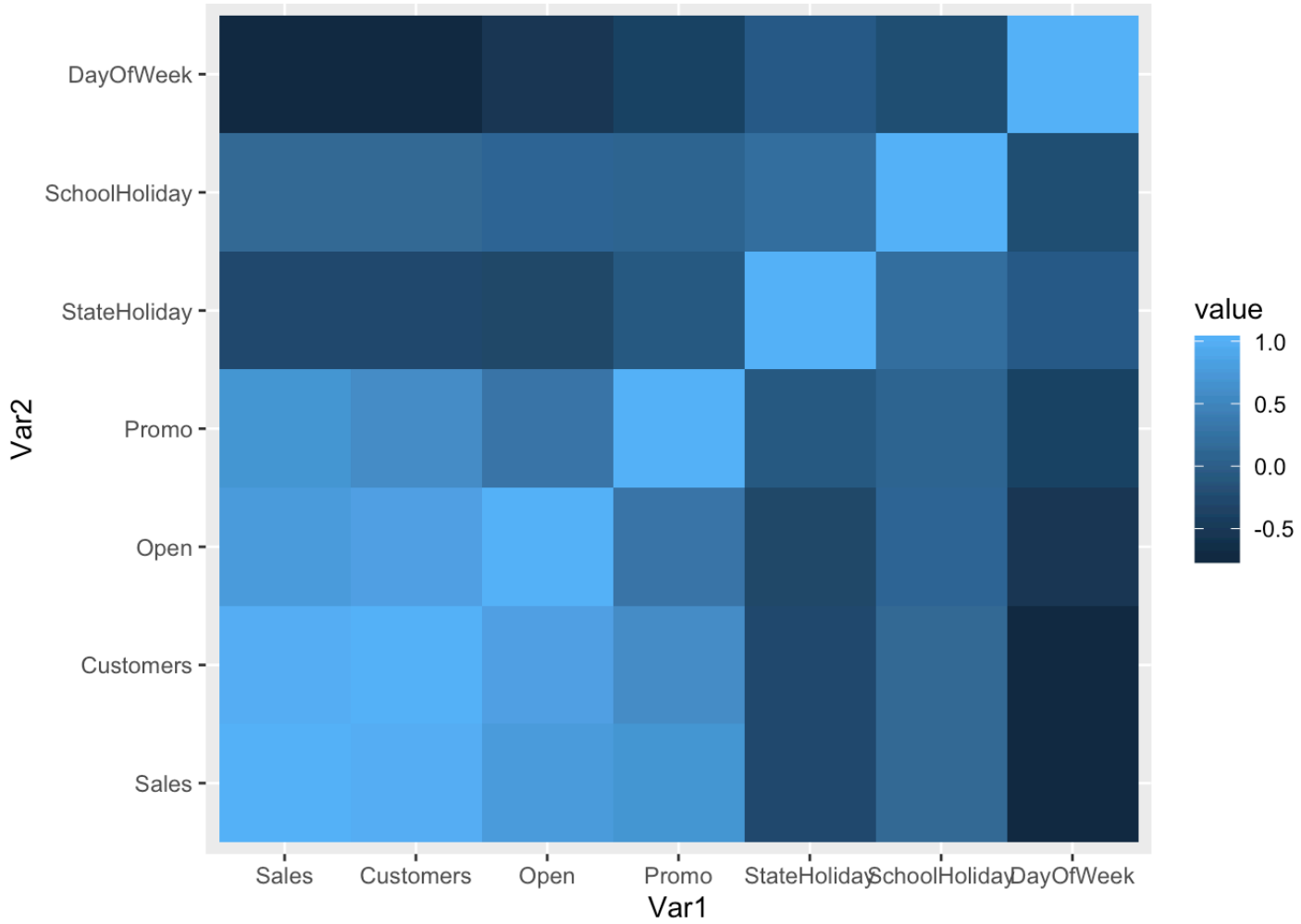
```
cormat
```

```
##               Sales Customers  Open Promo StateHoliday SchoolHoliday
## Sales          1.00      0.97  0.77  0.70        -0.25          0.13
## Customers      0.97      1.00  0.84  0.59        -0.27          0.13
## Open           0.77      0.84  1.00  0.31        -0.32          0.10
## Promo          0.70      0.59  0.31  1.00        -0.05          0.08
## StateHoliday  -0.25     -0.27 -0.32 -0.05         1.00          0.22
## SchoolHoliday  0.13      0.13  0.10  0.08         0.22          1.00
## DayOfWeek     -0.72     -0.73 -0.55 -0.39        -0.04         -0.23
##               DayOfWeek
## Sales             -0.72
## Customers         -0.73
## Open              -0.55
## Promo             -0.39
## StateHoliday      -0.04
## SchoolHoliday     -0.23
## DayOfWeek          1.00
```

```
library(reshape2)
melted_cormat <- melt(cormat)
head(melted_cormat)
```

```
##            Var1  Var2 value
## 1         Sales Sales  1.00
## 2     Customers Sales  0.97
## 3          Open Sales  0.77
## 4         Promo Sales  0.70
## 5  StateHoliday Sales -0.25
## 6 SchoolHoliday Sales  0.13
```

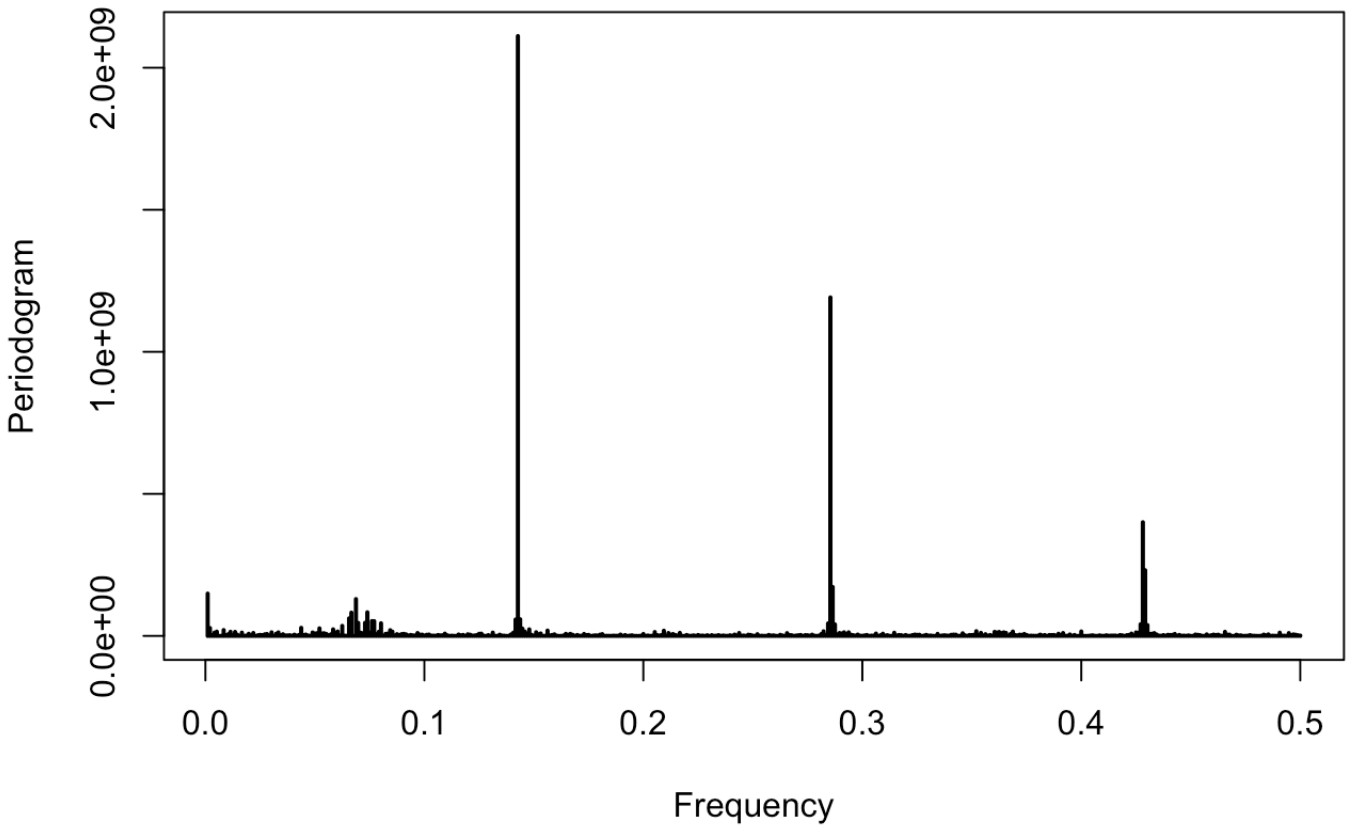
```
ggplot(data = melted_cormat , aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()
```



Spectrum Analysis

Identify the Seasonality

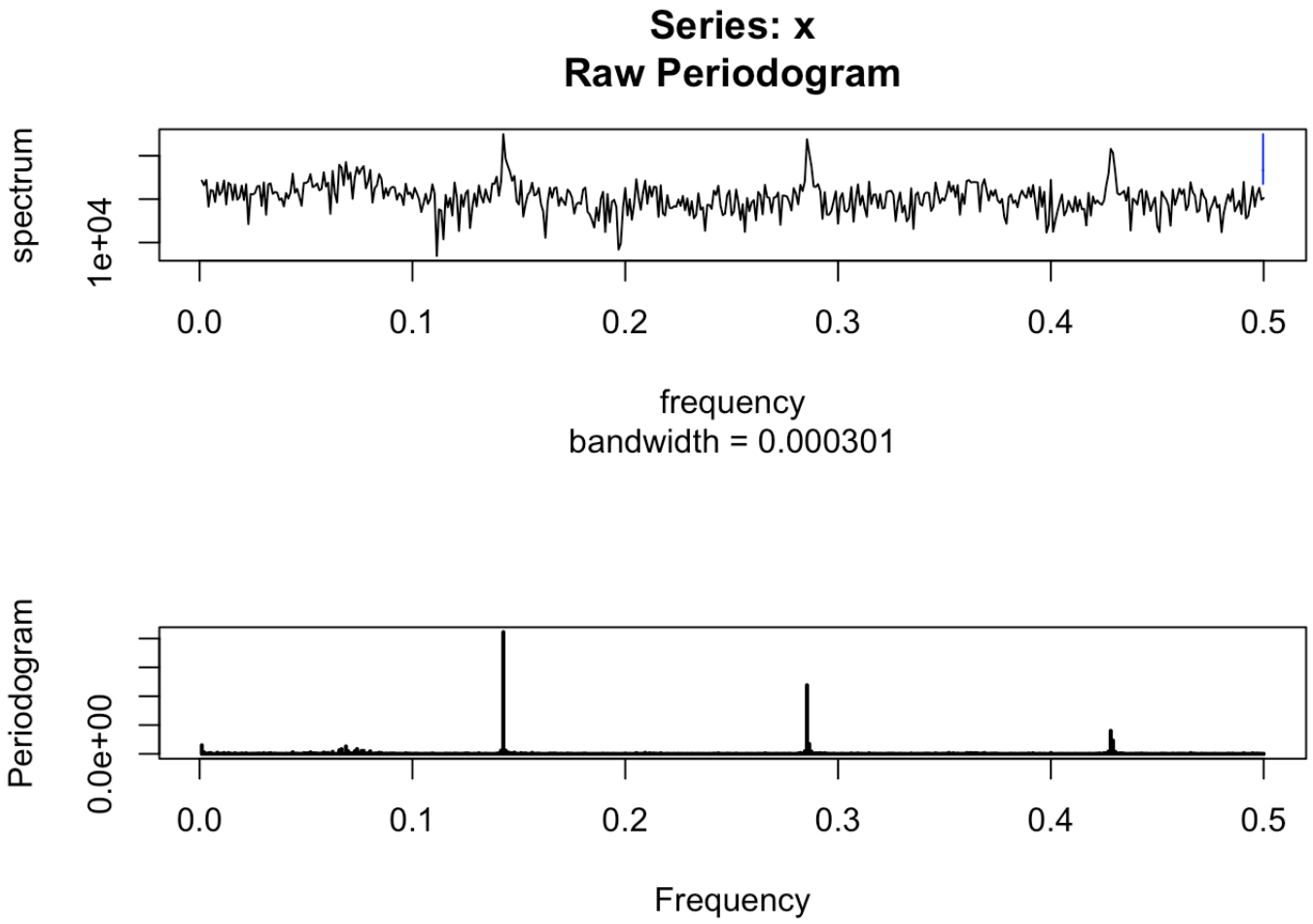
```
periodogram(store_8[,2])
temp <-periodogram(store_8[,2])
```




```
max_freq <- temp$freq[which.max(temp$spec)]
seasonality = 1/max_freq
seasonality
```

```
## [1] 7.007299
```

```
par(mfrow=c(2,1))
spectrum(store_8[,2])
periodogram(store_8[,2])
```



from above, it compose three fundamental frequency. 0.15 contribute more which is 7. so we will use 7 for our Arima model.

Decompositon of the Data

HW couldn't handle long seasonality such as 365. so let's try to use 7 as it frequency

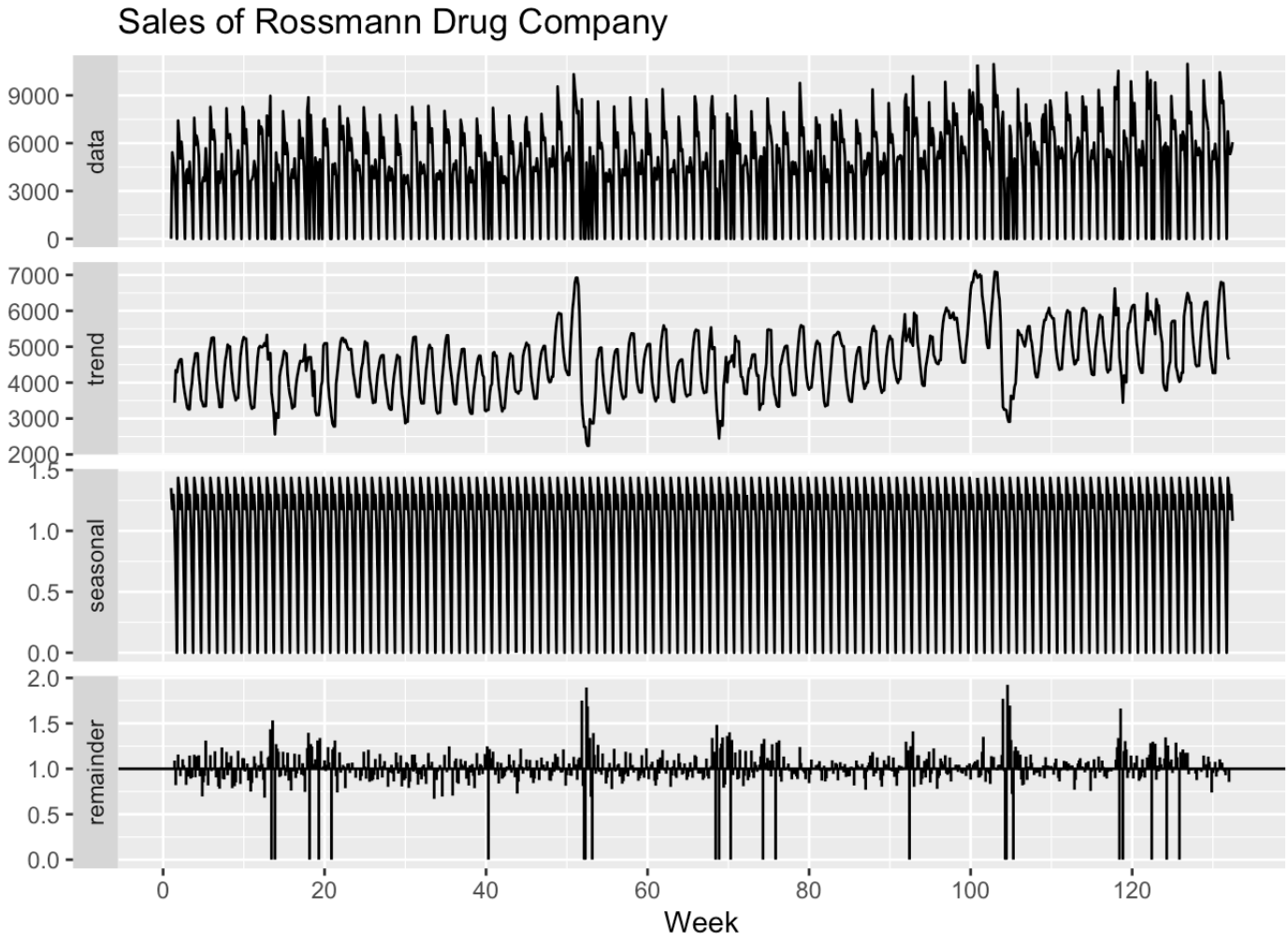
```
store_8_tsw<-ts(store_8, f=7)
tail(store_8_tsw,n=21)
```

```
## Time Series:
## Start = c(132, 5)
## End = c(135, 4)
## Frequency = 7
##           Date Sales Customers Open Promo StateHoliday SchoolHoliday year
## 132.5714 16627  3707       425    1     0            1             0 2015
## 132.7143 16628     0         0    0     0            1             0 2015
## 132.8571 16629  9117       935    1     1            1             0 2015
```

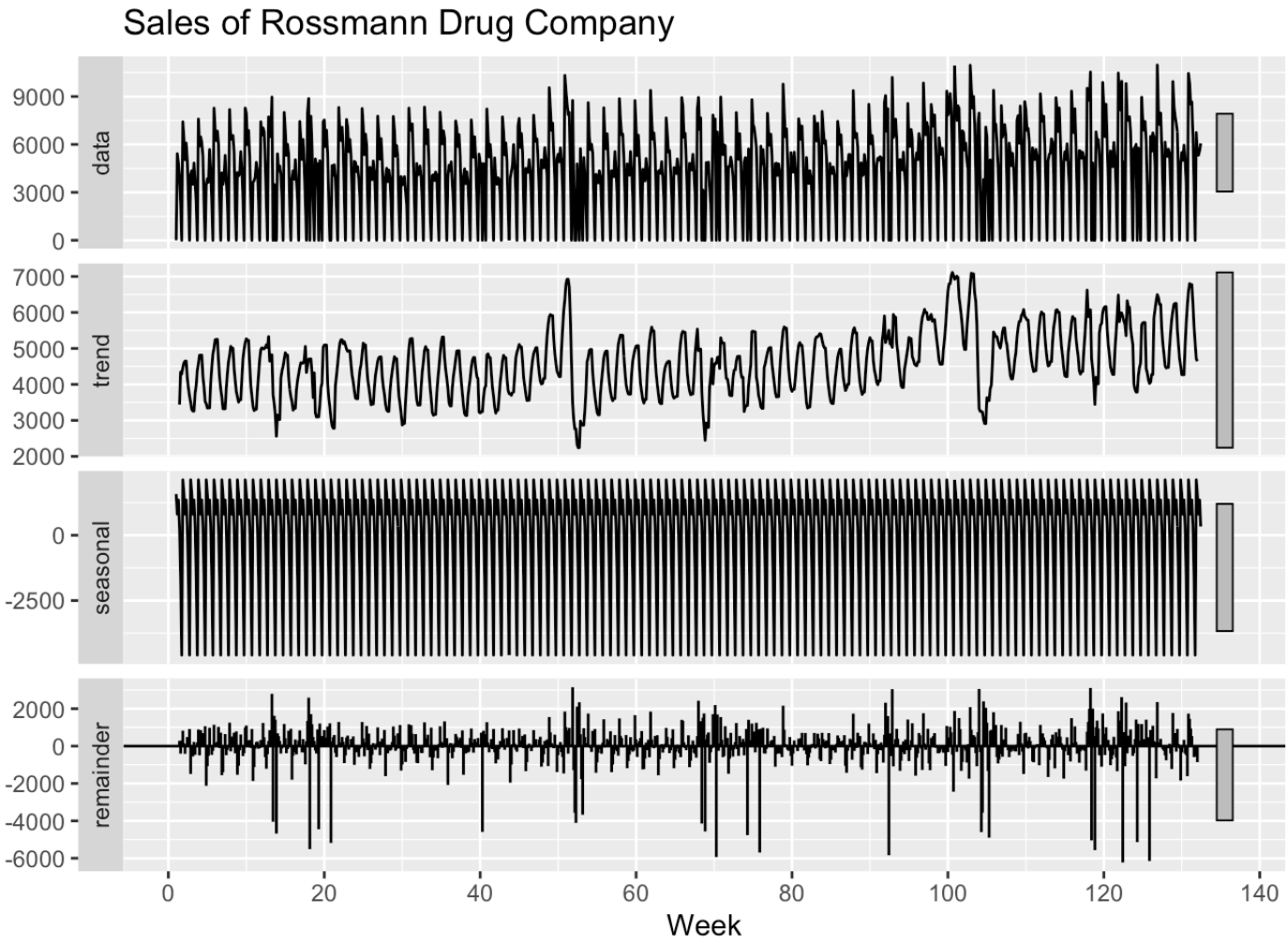
```
## 133.0000 16630  8845       840    1     1              1              0 2015
## 133.1429 16631  7932       802    1     1              1              0 2015
## 133.2857 16632  8808       988    1     1              1              0 2015
## 133.4286 16633  6728       684    1     1              1              0 2015
## 133.5714 16634  3813       442    1     0              1              0 2015
## 133.7143 16635     0         0    0     0              1              0 2015
## 133.8571 16636  5956       733    1     0              1              1 2015
## 134.0000 16637  5168       673    1     0              1              1 2015
## 134.1429 16638  4996       626    1     0              1              1 2015
## 134.2857 16639  6109       783    1     0              1              1 2015
## 134.4286 16640  4917       558    1     0              1              1 2015
## 134.5714 16641  3288       398    1     0              1              0 2015
## 134.7143 16642     0         0    0     0              1              0 2015
## 134.8571 16643  8739       841    1     1              1              1 2015
## 135.0000 16644  6717       695    1     1              1              1 2015
## 135.1429 16645  7029       698    1     1              1              1 2015
## 135.2857 16646  8420       882    1     1              1              1 2015
## 135.4286 16647  8492       833    1     1              1              1 2015
##          month week day DayOfWeek
## 132.5714     7   28  11         6
## 132.7143     7   28  12         7
## 132.8571     7   28  13         1
## 133.0000     7   28  14         2
## 133.1429     7   28  15         3
## 133.2857     7   29  16         4
## 133.4286     7   29  17         5
## 133.5714     7   29  18         6
## 133.7143     7   29  19         7
## 133.8571     7   29  20         1
## 134.0000     7   29  21         2
## 134.1429     7   29  22         3
## 134.2857     7   30  23         4
## 134.4286     7   30  24         5
## 134.5714     7   30  25         6
## 134.7143     7   30  26         7
## 134.8571     7   30  27         1
## 135.0000     7   30  28         2
## 135.1429     7   30  29         3
## 135.2857     7   31  30         4
## 135.4286     7   31  31         5
```

```
train_w<-window(store_8_tsw, start = c(1,1), end = c(132,4))
test_w<-window(store_8_tsw, start = c(132,5) )# test 3 weeks data
```

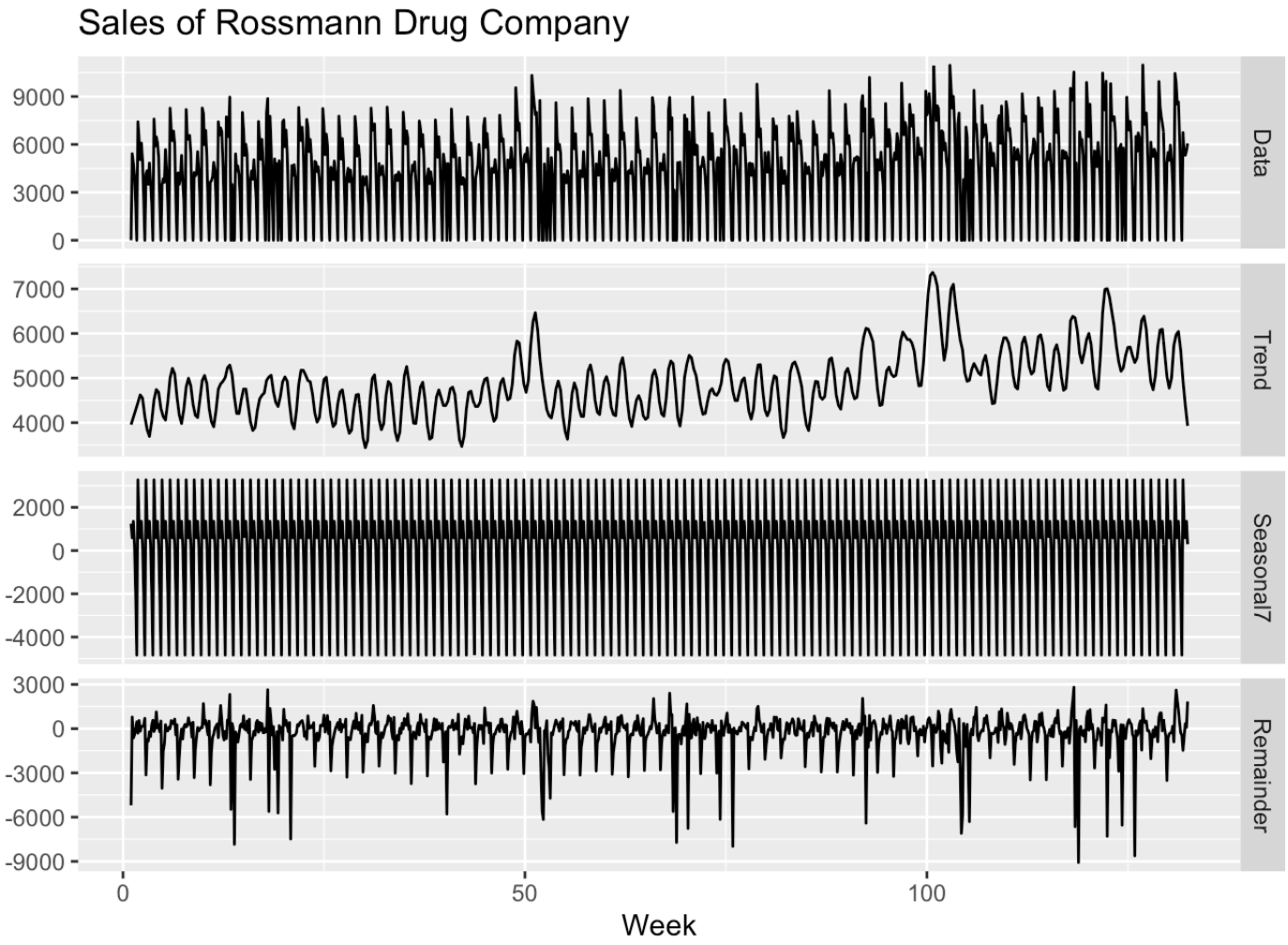
```
train_w[,2] %>% decompose(type= 'mult') %>% autoplot() + xlab("Week") + ggtitle("Sale
s of Rossmann Drug Company")
```




```
train_w[,2] %>% decompose(type= 'add') %>% autoplot() + xlab("Week") + ggtitle("Sales
of Rossmann Drug Company")
```



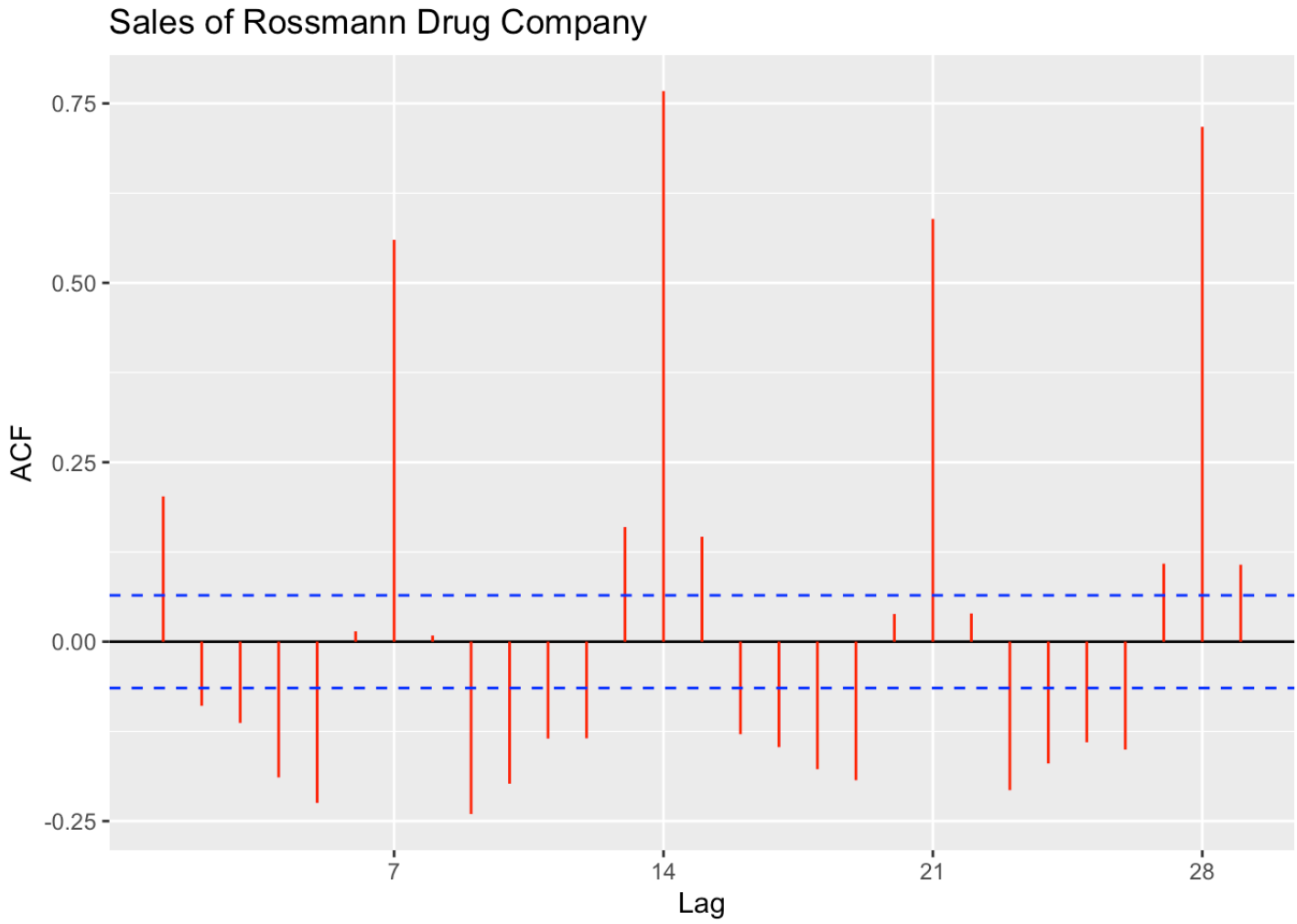

```
train_w[,2] %>% mstl(t.window=13, s.window="periodic", robust=TRUE)  %>% autoplot() +
xlab("Week") + ggtitle("Sales of Rossmann Drug Company")
```



Stationary

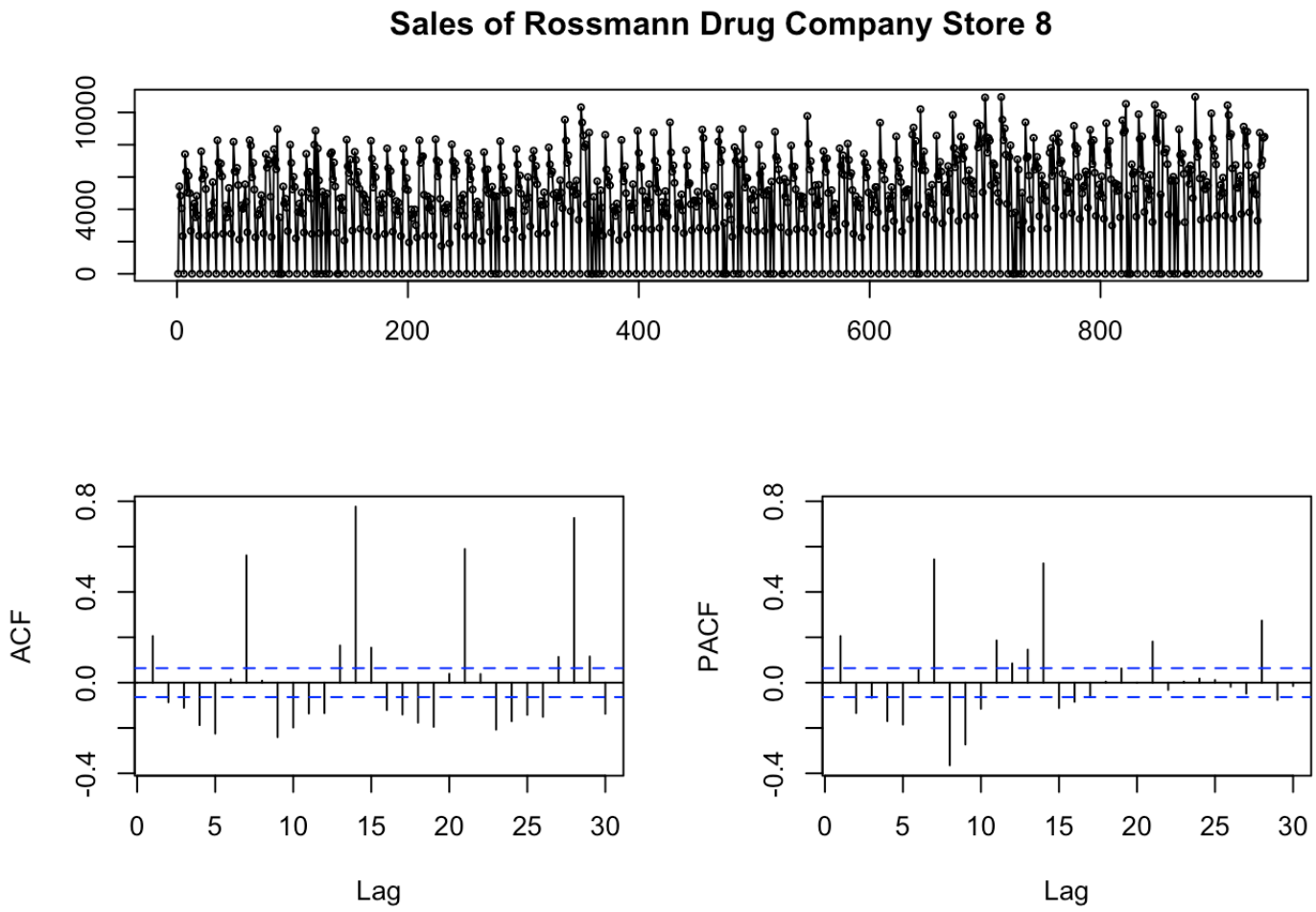
```
#ggseasonplot(train[,2], col = rainbow(15), year.labels = TRUE)
ggAcf(train_w[,2], main='Sales of Rossmann Drug Company', col ='red')
```

```
## Warning: Ignoring unknown parameters: main
```

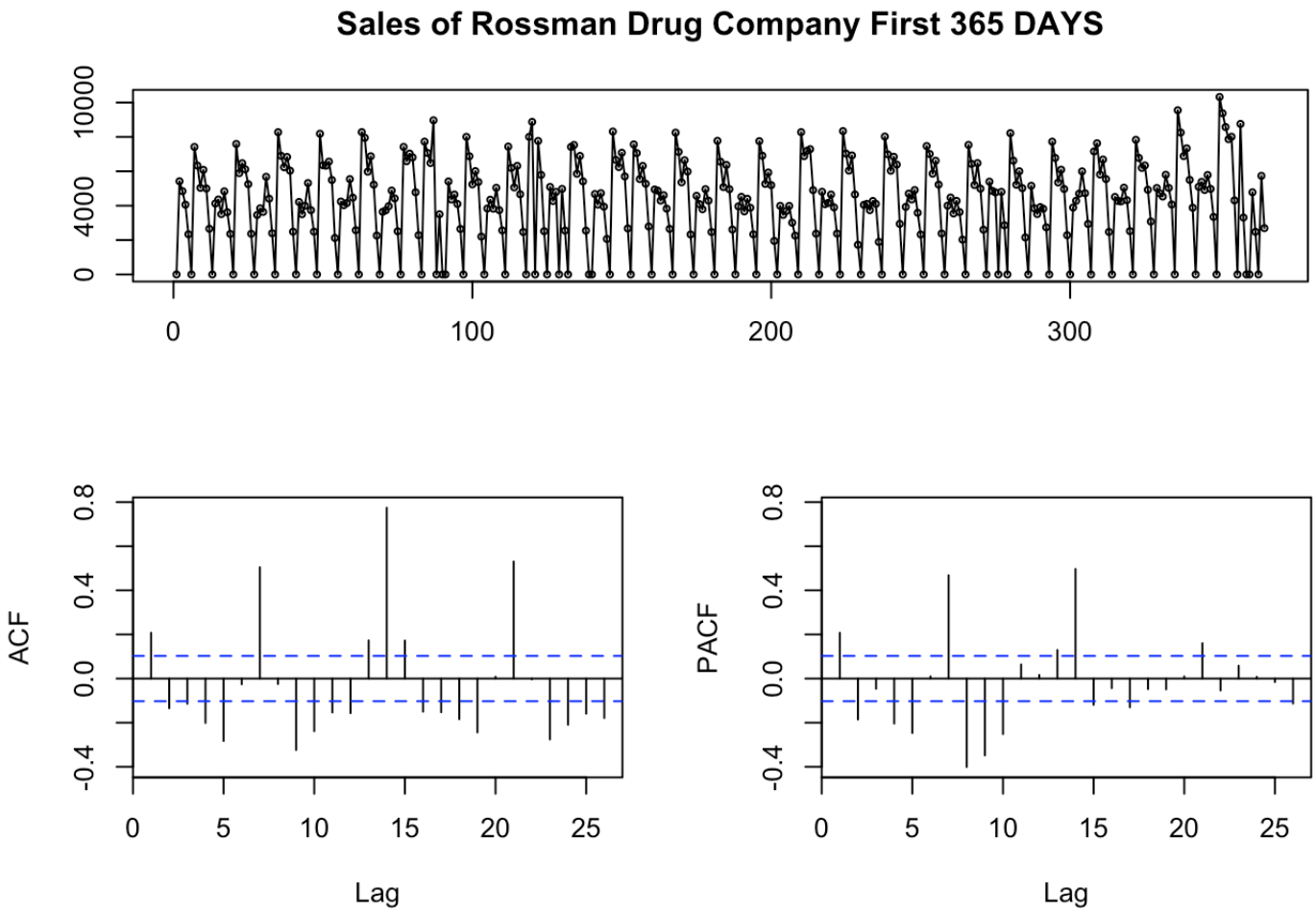


so the data is not stationary.

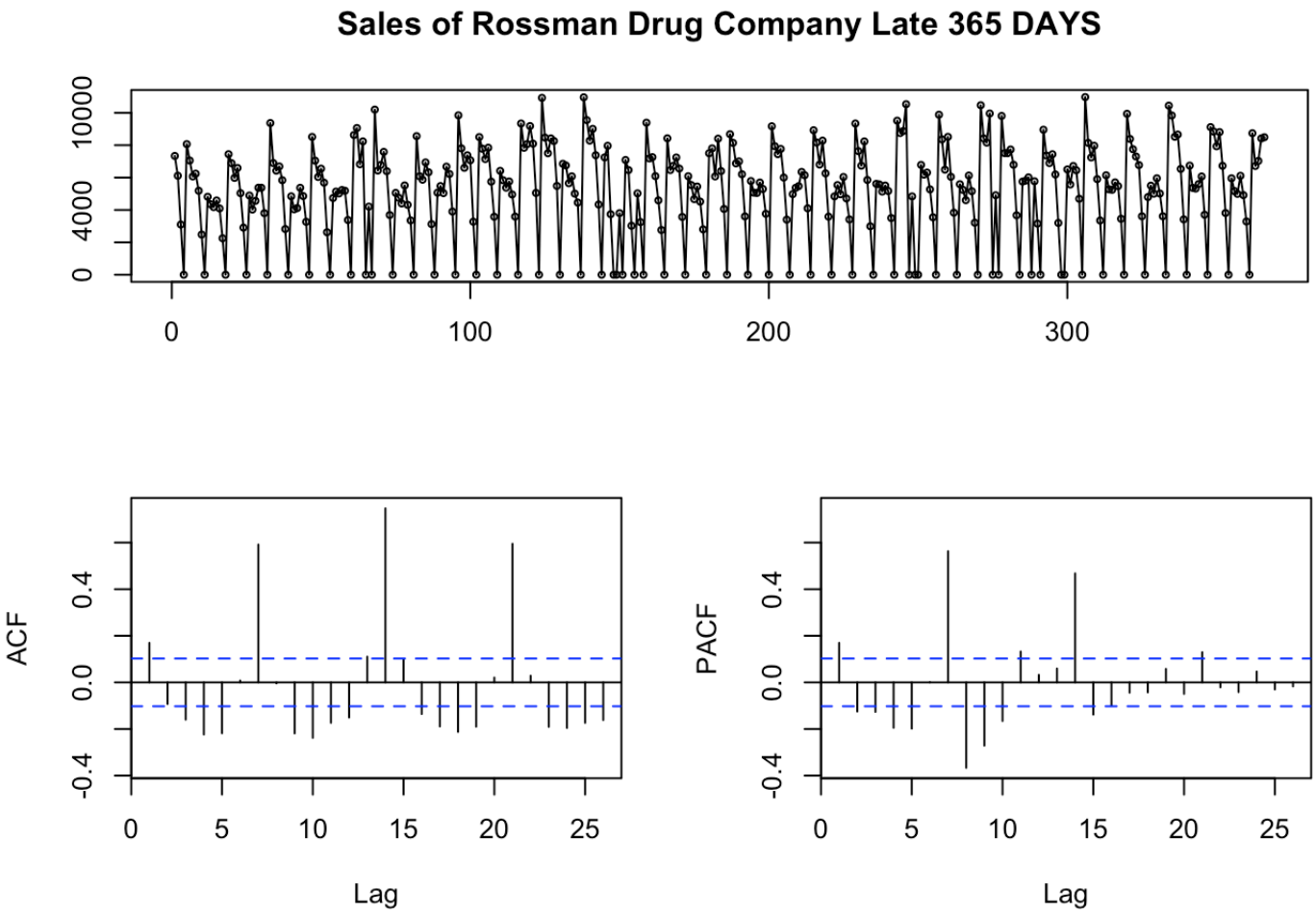
```
tsdisplay(store_8[,2], main = 'Sales of Rossmann Drug Company Store 8')
```




```
tsdisplay(store_8[1:365,2], main= 'Sales of Rossman Drug Company First 365 DAYS')
```




```
tsdisplay(store_8[577:942,2], main= 'Sales of Rossman Drug Company Late 365 DAYS')
```




```
#plot(store_8[,2], ylab = 'Sales', xlab = "Year", main = 'Sales of Rossmann Drug Comp
any')
```

ACF shows the data is not stationary. There is a baseline value but couldn't see clear trend. It also has strong seasonality. no trend observed. The maginitude of the data also remain the same. There are significnat auto correlation at lag 7, and lag 14.

also, from the first 365 days obsevation to the last 365 days observation, the magnitude of the data shows some increasing. and use auto boxCox lambda choose, it gives 0.657 as our lambda value for weekly frequency. In Conclusion, the variance is not stable. we need box cox transformation to stablize the variance.

```
BoxCox.lambda(train_w[,2])
```

```
## [1] 0.2045416
```

KPSS Test for stationarity

```
kpss.test(train_w[,2])
```

```
## Warning in kpss.test(train_w[, 2]): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  train_w[, 2]
## KPSS Level = 2.8265, Truncation lag parameter = 6, p-value = 0.01
```

```
adf.test(train_w[,2])
```

```
## Warning in adf.test(train_w[, 2]): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  train_w[, 2]
## Dickey-Fuller = -17.007, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

for kpss test, the null hypothesis is that the data are stationary and non-seasonal. the p value is 0.01 for train data set which is smaller than 0.05 which we could reject null hypothesis which means the data is not stationary and nonseasonal. although ADF test gives us small p value which we reject the null hypothesis that the data is stationary.

if we do first order differencing

Differencing and EACF

```
ndiffs(train_w[,2])
```

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

```
nsdiffs(train_w[,2])
```

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

```
kpss.test(diff(train_w[,2],d = 1))
```

```
## Warning in kpss.test(diff(train_w[, 2], d = 1)): p-value greater than
## printed p-value
```

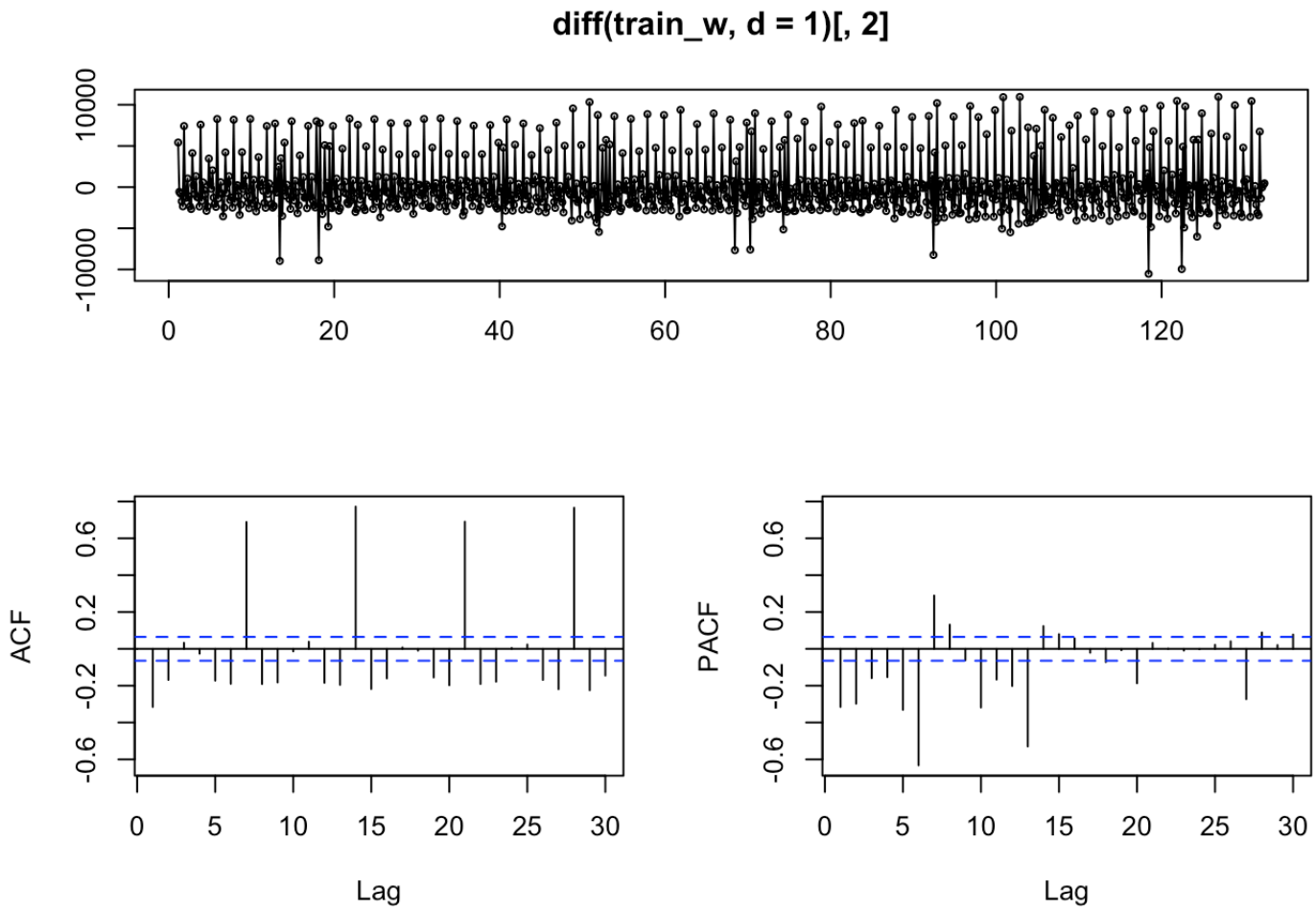
```
##
##  KPSS Test for Level Stationarity
##
## data:  diff(train_w[, 2], d = 1)
## KPSS Level = 0.013299, Truncation lag parameter = 6, p-value = 0.1
```

```
adf.test(diff(train_w[,2],d = 1))
```

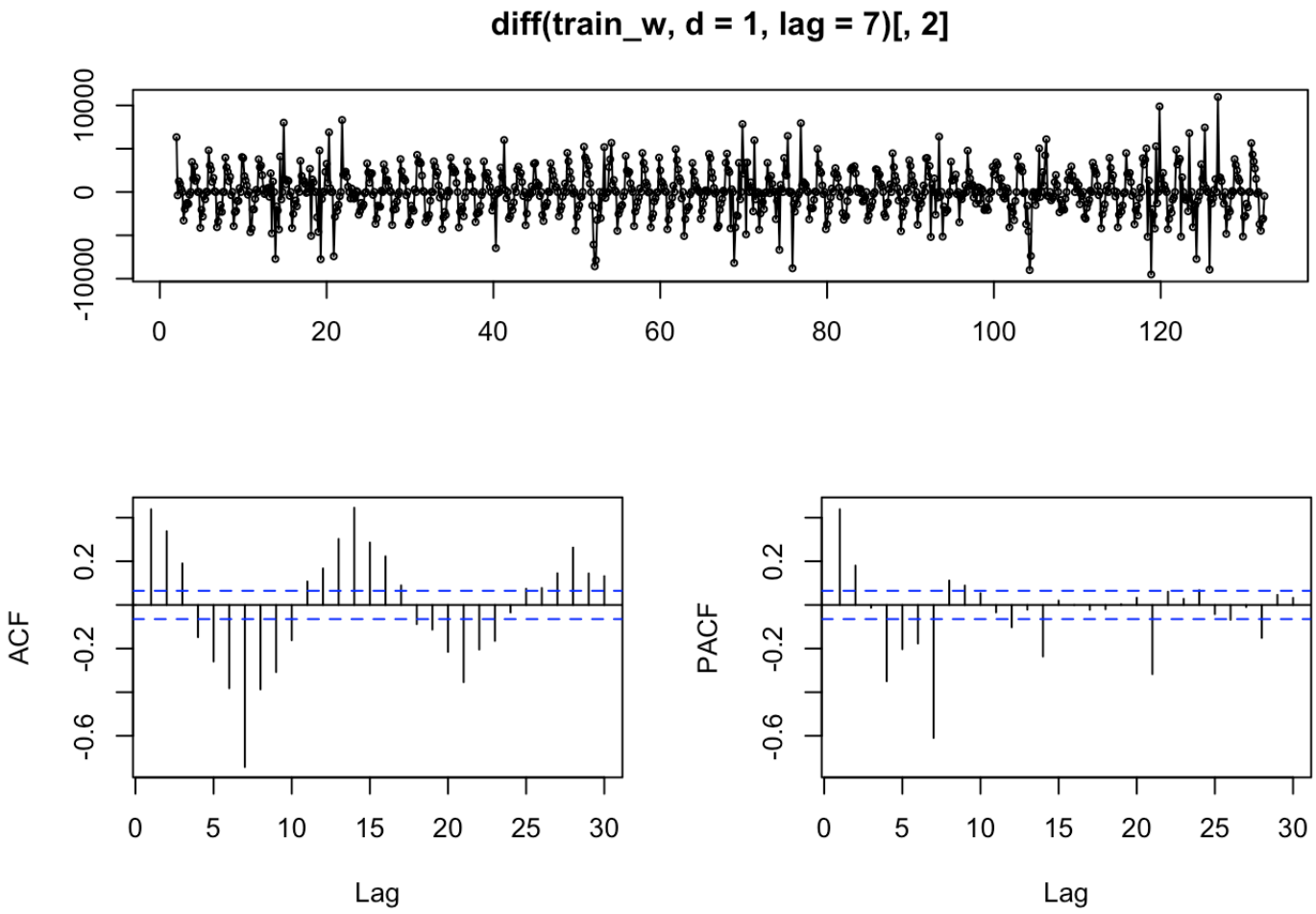
```
## Warning in adf.test(diff(train_w[, 2], d = 1)): p-value smaller than
## printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(train_w[, 2], d = 1)
## Dickey-Fuller = -17.003, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

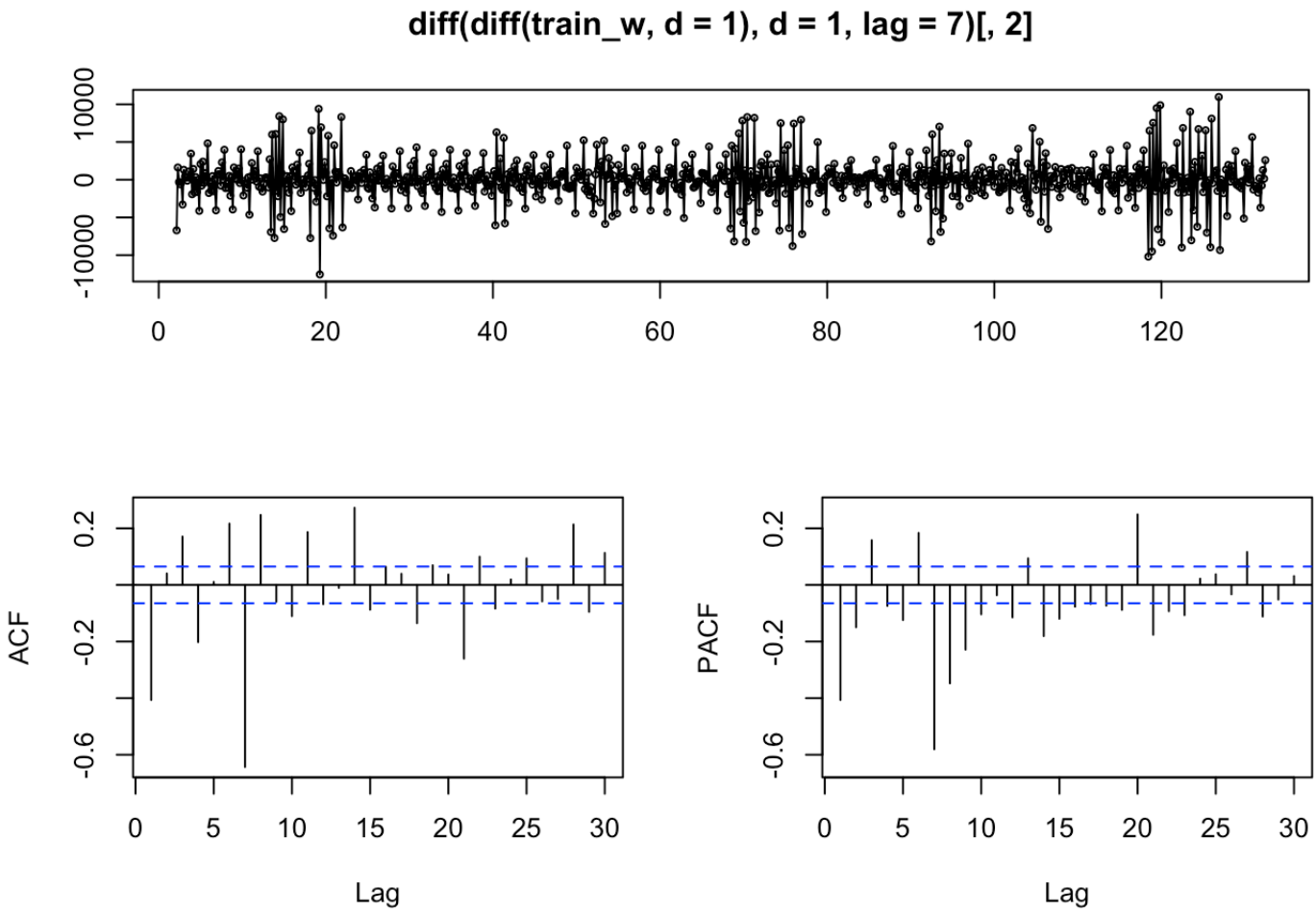
```
tsdisplay(diff(train_w,d=1)[,2])
```



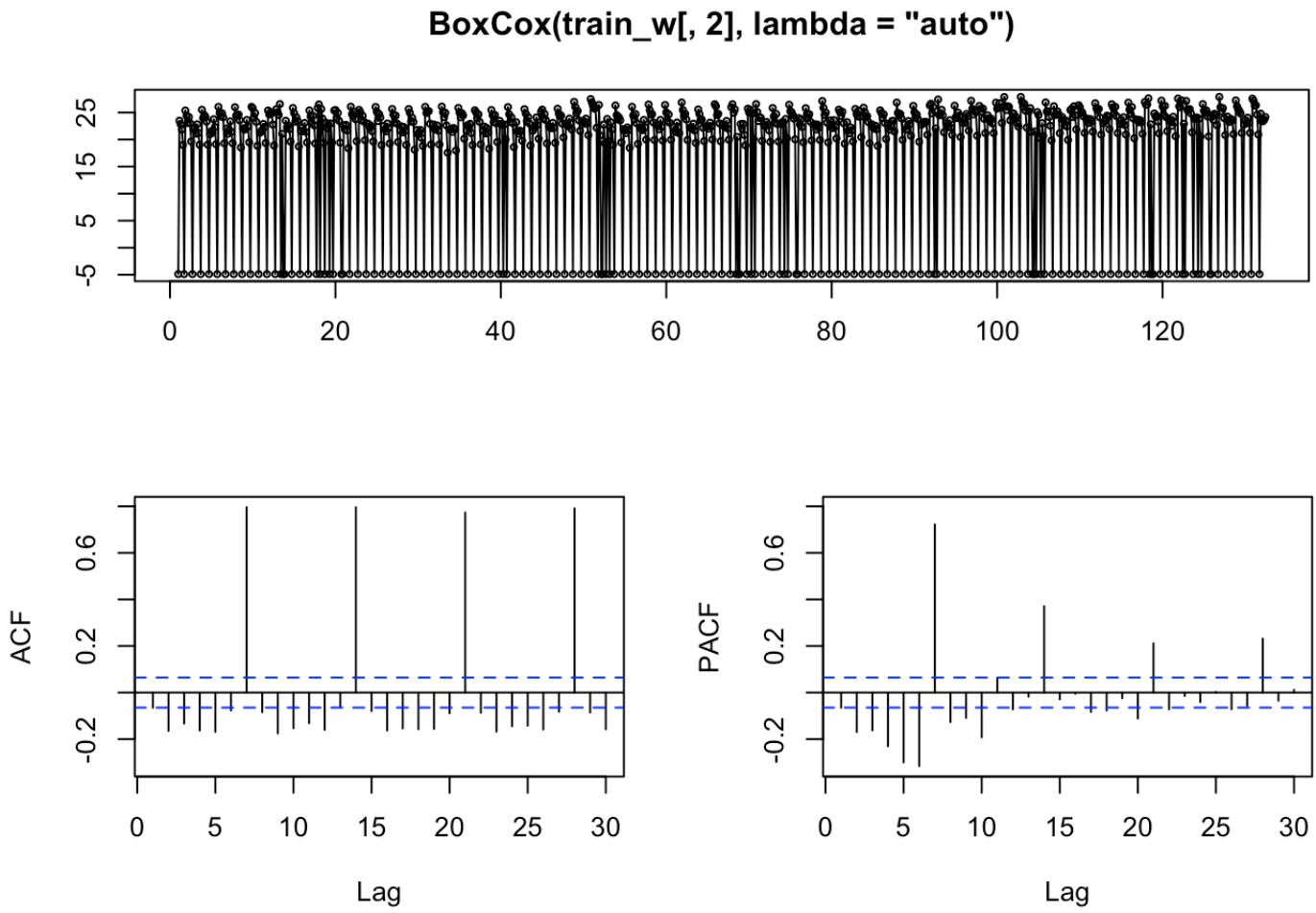

```
tsdisplay(diff(train_w,d=1,lag = 7)[,2])
```




```
tsdisplay(diff(diff(train_w,d = 1), d= 1,lag = 7)[,2])
```




```
#box_lambda transformation

lbd = BoxCox.lambda(train_w[,2])

tsdisplay(BoxCox(train_w[,2], lambda = 'auto'))
```




```
train_t<-BoxCox(train_w[,2], lambda = lbd)
test_t <-BoxCox(test_w[,2], lambda = lbd)

kpss.test(train_t) # non stationary with smaller P
```

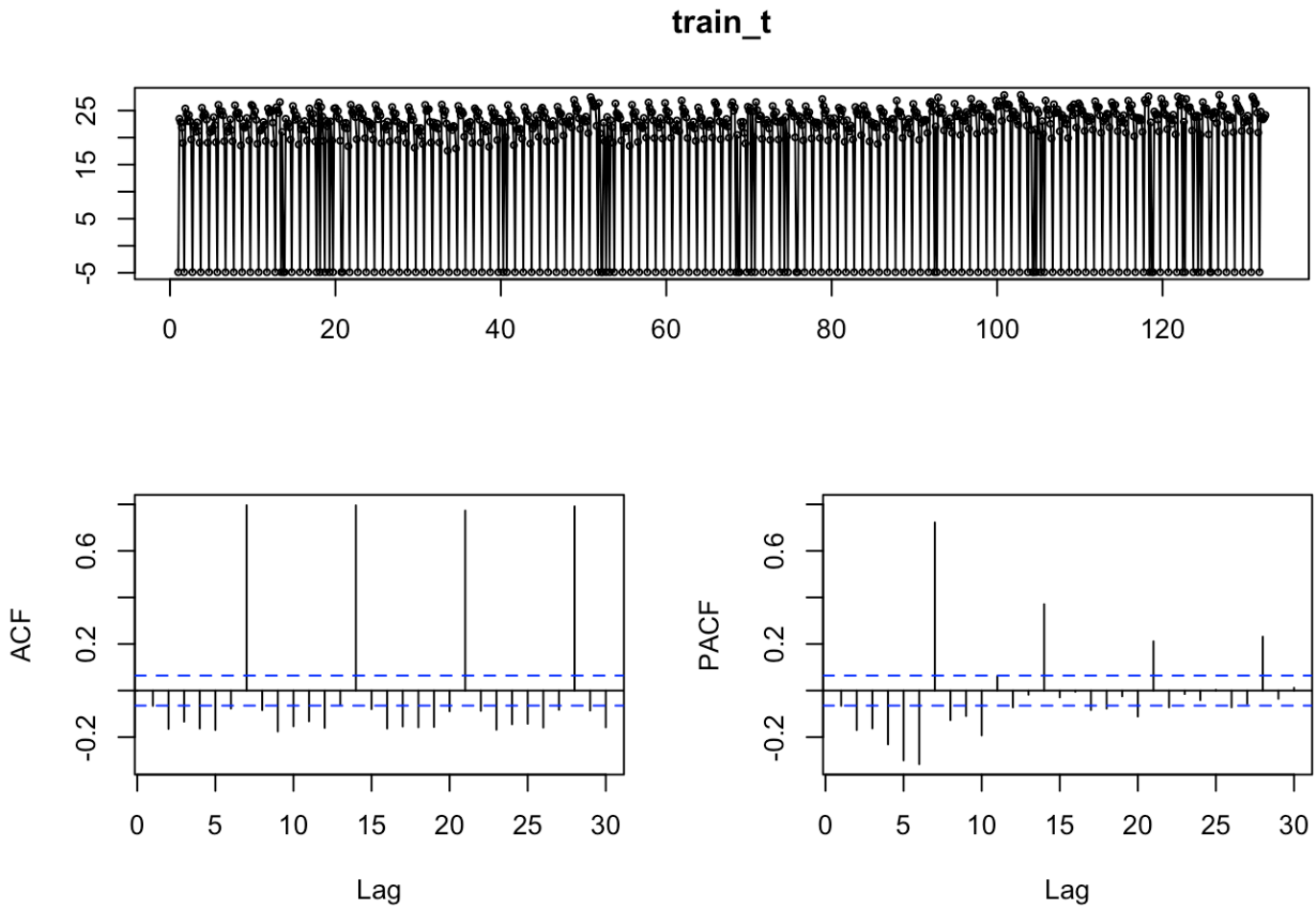
```
##
##  KPSS Test for Level Stationarity
##
## data:  train_t
## KPSS Level = 0.61263, Truncation lag parameter = 6, p-value =
## 0.02149
```

```
adf.test(train_t) # trend stationary
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  train_t
## Dickey-Fuller = -11.396, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```
tsdisplay(train_t)
```




```
# choice 1 seasonal differencing with lag 7
kpss.test(diff(train_w[,2],d = 1, lag = 7))
```

```
## Warning in kpss.test(diff(train_w[, 2], d = 1, lag = 7)): p-value greater
## than printed p-value
```

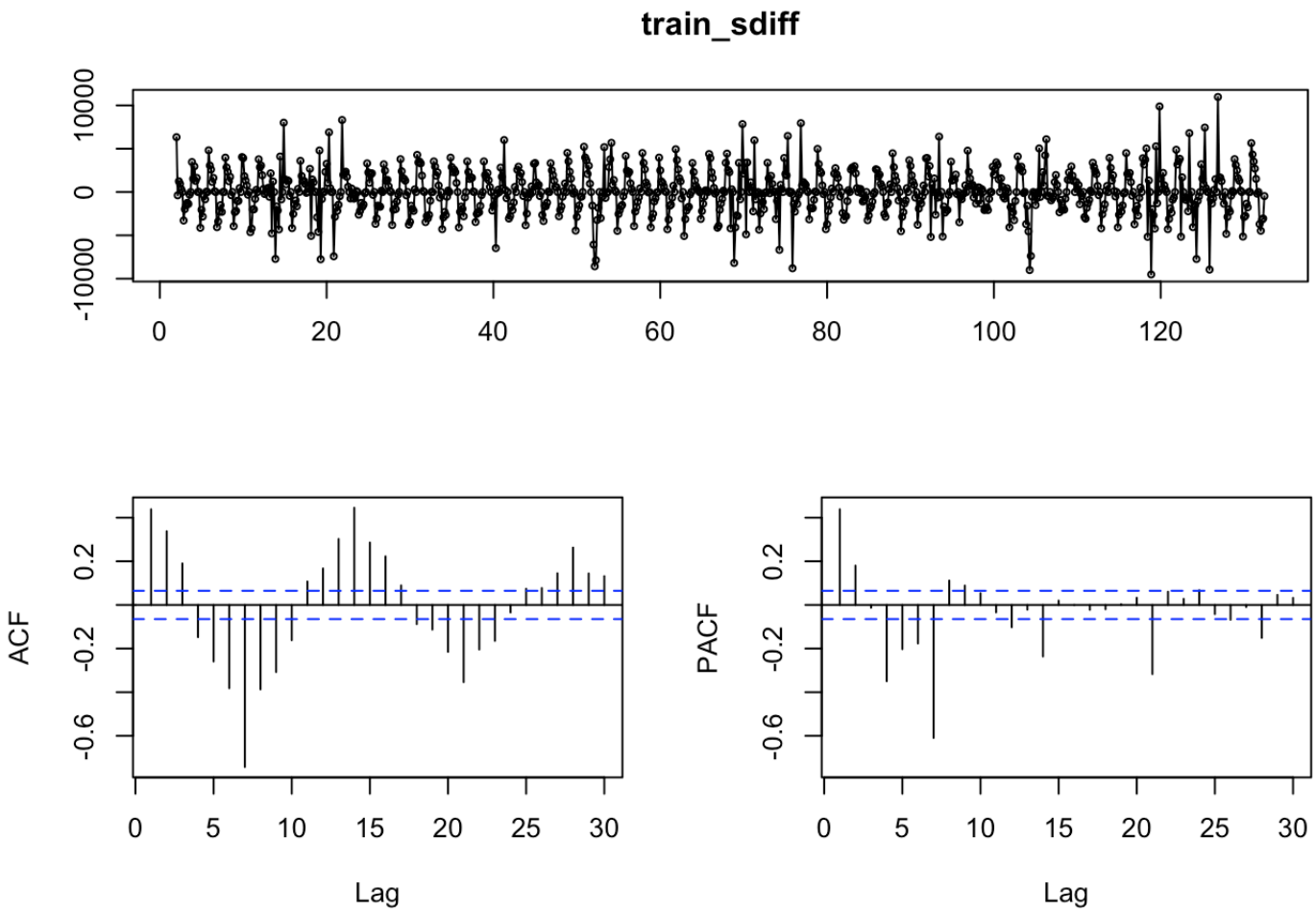
```
##
##  KPSS Test for Level Stationarity
##
## data:  diff(train_w[, 2], d = 1, lag = 7)
## KPSS Level = 0.0039852, Truncation lag parameter = 6, p-value =
## 0.1
```

```
adf.test(diff(train_w[,2],d = 1, lag = 7))
```

```
## Warning in adf.test(diff(train_w[, 2], d = 1, lag = 7)): p-value smaller
## than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(train_w[, 2], d = 1, lag = 7)
## Dickey-Fuller = -12.642, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```
train_sdiff <- diff(train_w[,2],d = 1, lag = 7)
test_sdiff <- diff(train_w[,2],d = 1, lag = 7)

tsdisplay(train_sdiff) # acf exponentially decaying or sinusoidal and has a hard cutt
- off in the PACF at lag 7 try AR model
```




```
eacf(train_sdiff, ar.max = 7, ma.max = 7)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7
## 0 x x x x x x x x
## 1 x x x x o o x x
## 2 x x x x x o x x
## 3 o x x x o o x x
## 4 x x x x o x x x
## 5 x o x x o o x x
## 6 x x x x x x x x
## 7 x x x o x o x o
```

```
# possible choice for model ARMA(3,0), ARMA(1,4) ARMA(3,4),ARMA(4,4)  set up 4 as max
imum

#choice 2 first order differencing
kpss.test(diff(train_t,d = 1))
```

```
## Warning in kpss.test(diff(train_t, d = 1)): p-value greater than printed p-
## value
```

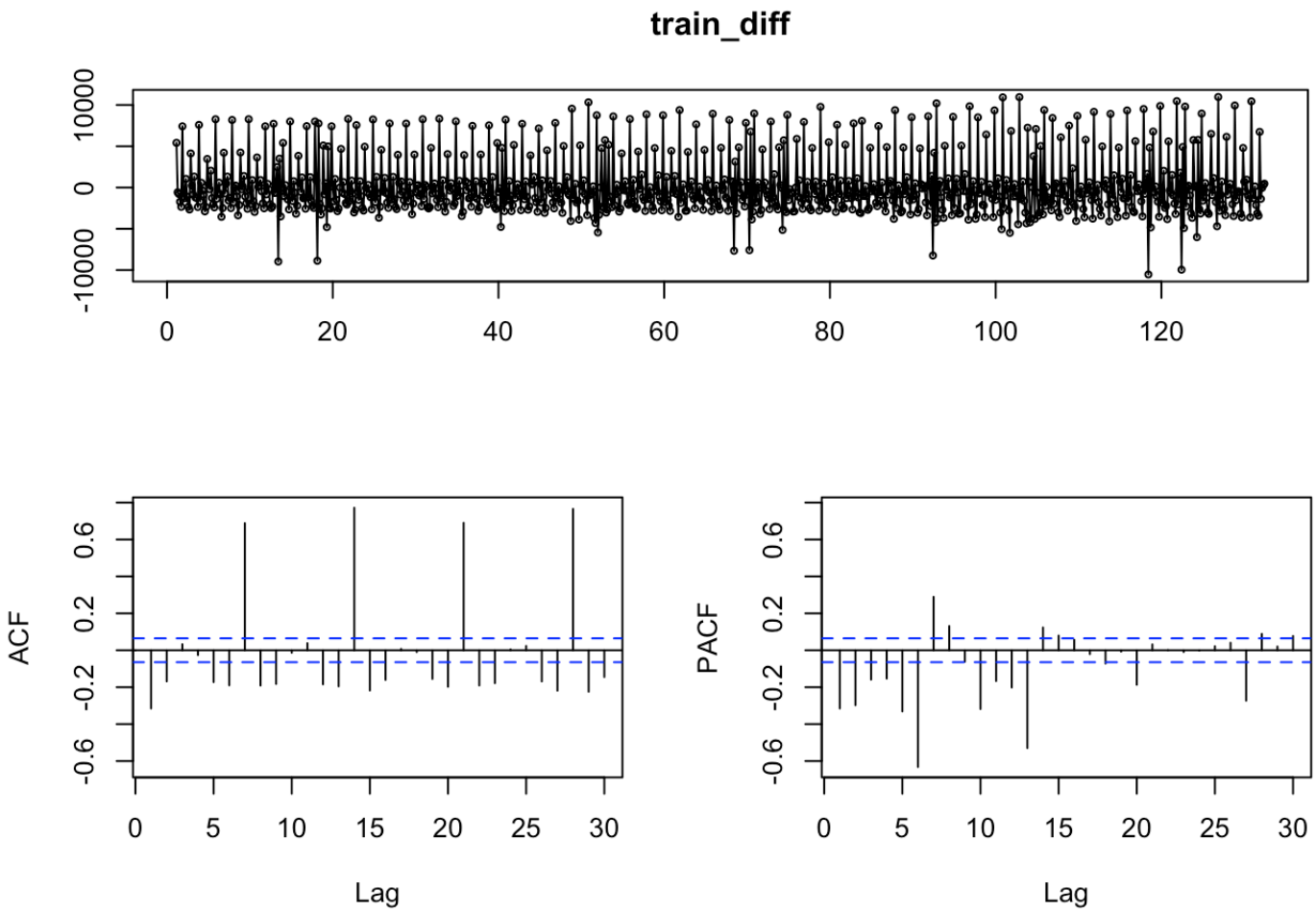
```
##
##  KPSS Test for Level Stationarity
##
## data:  diff(train_t, d = 1)
## KPSS Level = 0.039418, Truncation lag parameter = 6, p-value = 0.1
```

```
adf.test(diff(train_t,d = 1))
```

```
## Warning in adf.test(diff(train_t, d = 1)): p-value smaller than printed p-
## value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(train_t, d = 1)
## Dickey-Fuller = -15.647, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```
train_diff <- diff(train_w[,2],d = 1)
test_diff <- diff(train_w[,2],d = 1)

tsdisplay(train_diff)
```




```
eacf(train_diff,ar.max = 4, ma.max = 4)
```

```
## AR/MA
##   0 1 2 3 4
## 0 x x o o x
## 1 x x x o o
## 2 x x o o o
## 3 x x x o o
## 4 x x o x o
```

```
## possible choices for model MA(2), MA(3) ARMA(2,2) ARMA(2,3) ARMA(2,4) ARMA(4,2) ARMA(4,4)

#choice 3  box transform with first order seasonal differencing

kpss.test(diff(train_t,d = 1, lag = 7))
```

```
## Warning in kpss.test(diff(train_t, d = 1, lag = 7)): p-value greater than
## printed p-value
```

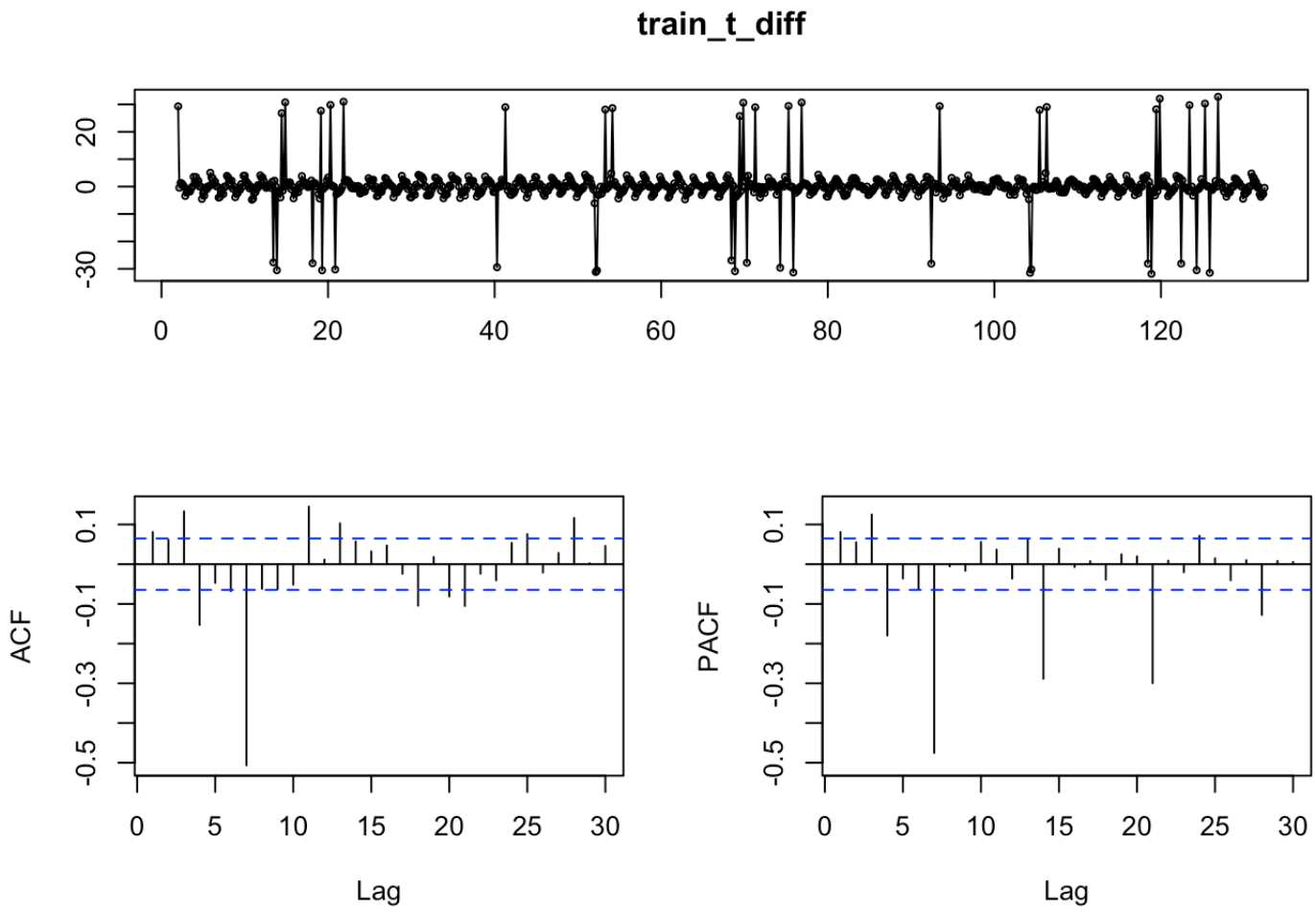
```
##
##  KPSS Test for Level Stationarity
##
## data:  diff(train_t, d = 1, lag = 7)
## KPSS Level = 0.0070569, Truncation lag parameter = 6, p-value =
## 0.1
```

```
adf.test(diff(train_t,d = 1, lag = 7))
```

```
## Warning in adf.test(diff(train_t, d = 1, lag = 7)): p-value smaller than
## printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(train_t, d = 1, lag = 7)
## Dickey-Fuller = -12.812, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```
train_t_diff <- diff(train_t,d = 1, lag = 7)
test_t_diff <- diff(test_t,d = 1, lag = 7)

tsdisplay(train_t_diff) # some extrem value show up, maybe due to the 0 sales
```



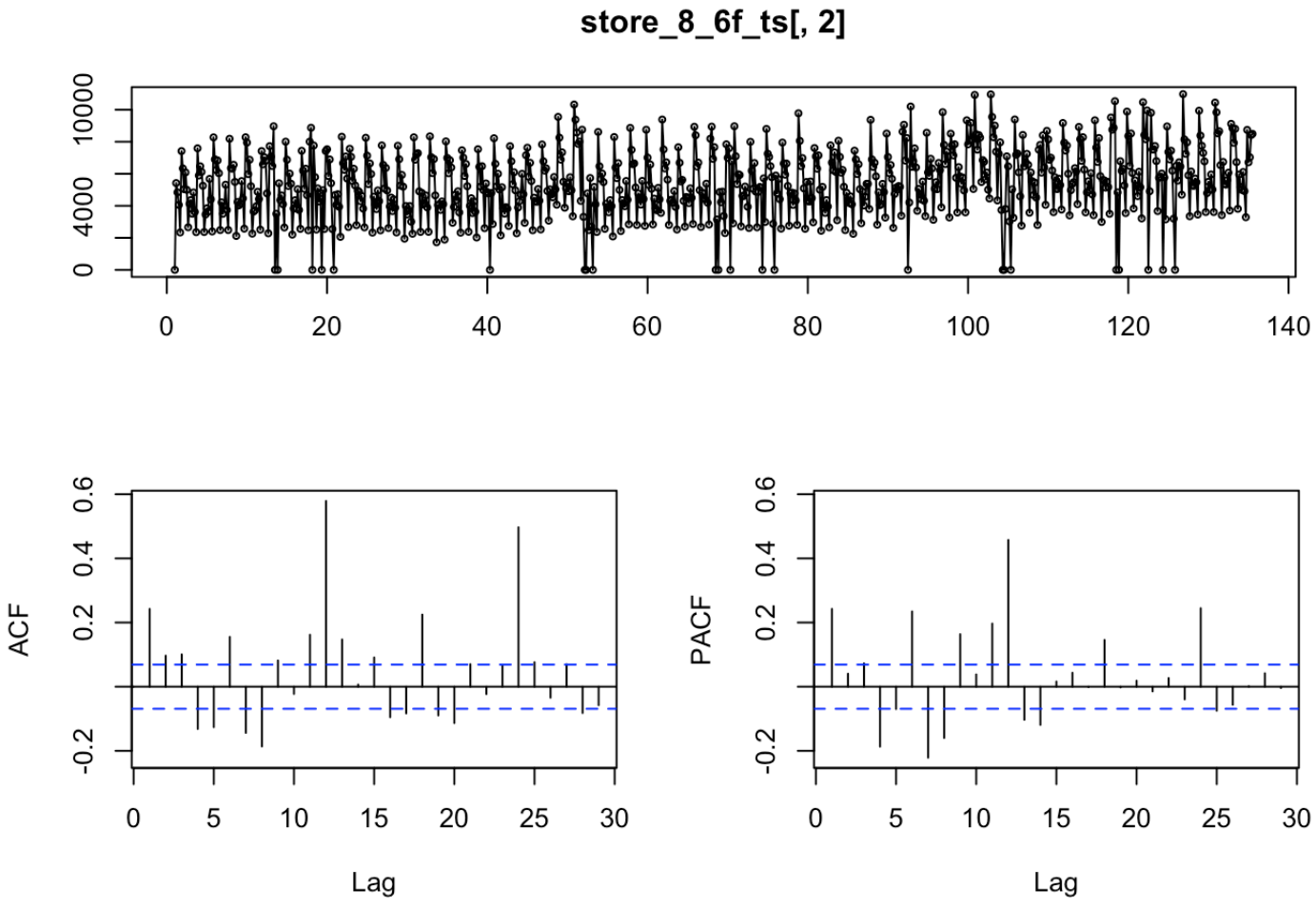

```
eacf(train_t_diff,ar.max = 4, ma.max = 4)
```

```
## AR/MA
##   0 1 2 3 4
## 0 x o x x o
## 1 x o x x o
## 2 x x x x o
## 3 x x x o o
## 4 x x x x o
```

```
##  possible choice models  MA(1), MA(4), ARMA(1,1),  ARMA(1,4), ARMA(4,4)

#choice 4 for first seasonal differencing and first order differencing

eacf(diff(train_t_diff,d = 1),ar.max = 4, ma.max = 4)
```

```
## AR/MA
##   0 1 2 3 4
## 0 x o x x x
## 1 x x x x o
## 2 x x x o o
## 3 x x x x o
## 4 x x x x x
```

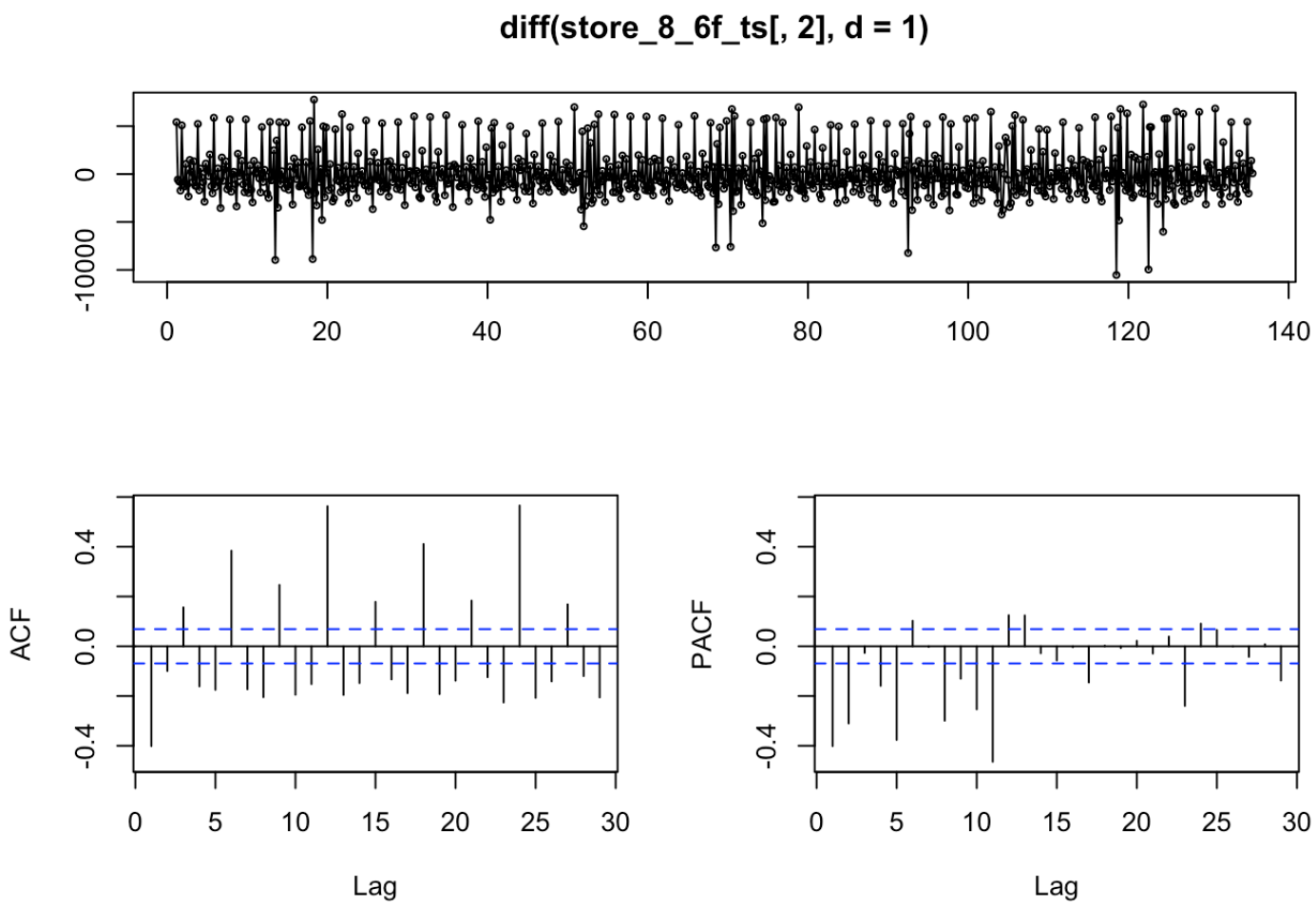
```
#Another way, we could remove sunday 0 sales and set up frq = 6

store_8_6f <- store_8[which(store_8$DayOfWeek!=7),]

store_8_6f_ts<-ts(store_8_6f, f = 6)

tsdisplay(store_8_6f_ts[,2])
```



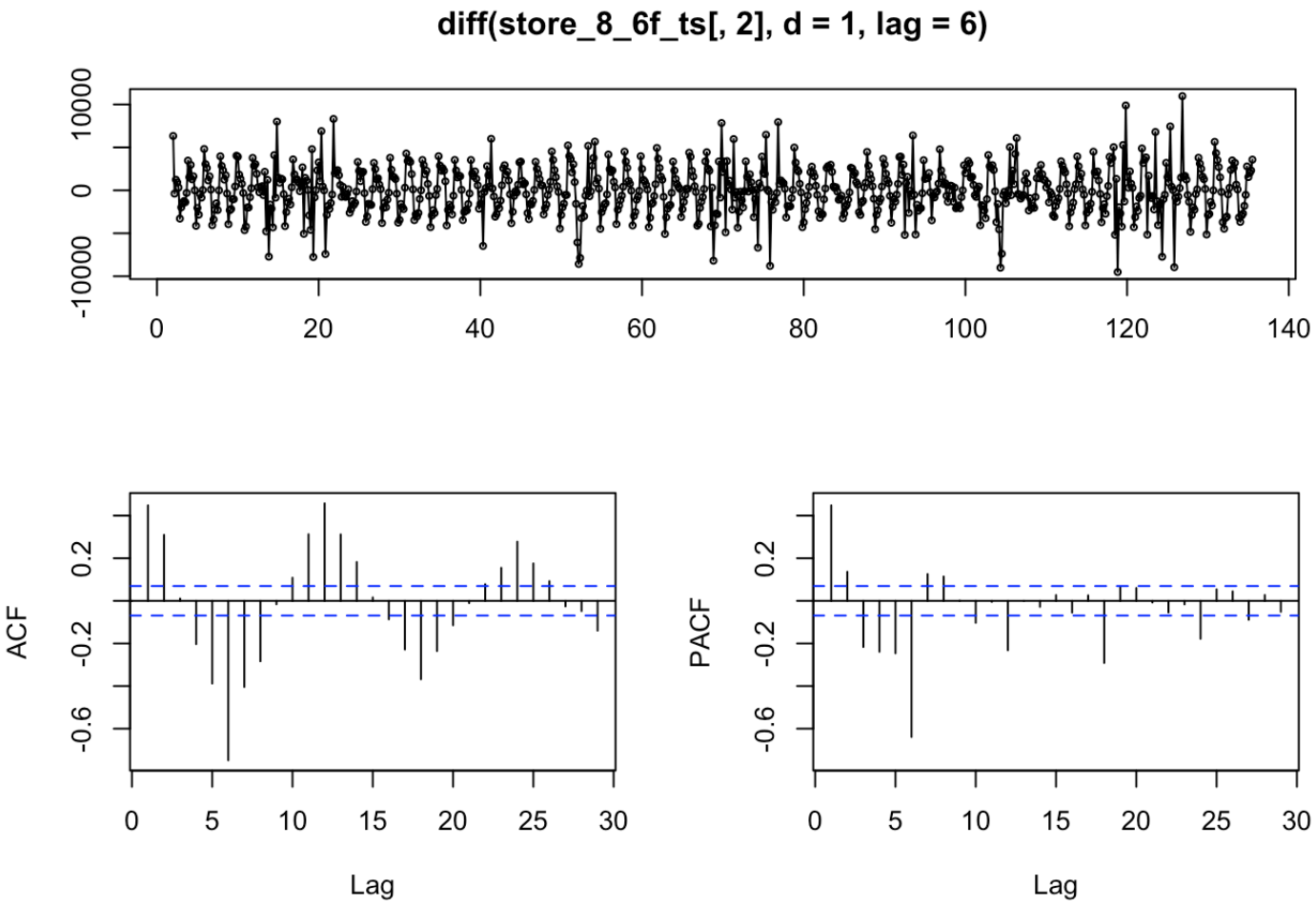

```
BoxCox.lambda(store_8_6f_ts)
```

```
## [1] 0.115733
```

```
tsdisplay(diff(store_8_6f_ts[,2],d=1))
```




```
tsdisplay(diff(store_8_6f_ts[,2],d=1, lag = 6))
```



If we apply first order differencing on lag 7, KPSS test pvalue is larger than 0.1 we accept null hypothesis that data is stationary, we could not reject the null hypothesis, Acf test p value is smaller than 0.05, we reject the null hypothsis and the data is staionary.

another way to look at it is our store is closed on 7, so if we remove 0 sales for 7 , will the data look better?

we tried to use f =6 and removed all the sales of Sunday, but the stationary test and plot didn't have too much difference. since f = 7 is a more general case for other store type. we will go with original data set.

try different modesl

```
result = data.frame()
#choice 1 seasonal differecing
bestfit_arima1 <-list(aicc = Inf)
bestrmse1 <- list(rmse= NA)
# choice 2
bestfit_arima2 <-list(aicc = Inf)
bestfit_rmse2 <- list(rmse= NA)
# Choice 3
bestfit_arima3 <-list(aicc = Inf)
bestfit_rmse3 <- list(rmse= NA)

p=q=0
for (p in 0:4 )
{ for (q in 0:4){
  fit <- Arima (train_sdiff, order = c(p,0,q),
                include.drift = TRUE)
  if (fit$aicc < bestfit_arima1$aicc)
    bestfit_arima1<-fit
  if (fit$rmse <-  accuracy(bestfit_arima1)[2])
    bestfit_rmse <-accuracy(bestfit_arima1)[2]
    else break;
  }
}

bestfit_arima1$aicc
```

```
## [1] 16286.53
```

```
bestfit_arima1
```

```
## Series: train_sdiff
## ARIMA(3,0,4) with drift
##
## Coefficients:
##           ar1      ar2      ar3     ma1     ma2     ma3     ma4  intercept
##       -1.5846  -1.2417  -0.4362  2.2425  2.7875  2.2265  0.9832    66.3685
## s.e.   0.0312   0.0466   0.0320  0.0130  0.0308  0.0336  0.0145   251.8706
##         drift
##       -0.1026
## s.e.   0.4767
##
## sigma^2 estimated as 3122837:  log likelihood=-8133.14
## AIC=16286.29   AICc=16286.53   BIC=16334.46
```

```
accuracy(bestfit_arima1)
```

```
##                     ME     RMSE     MAE MPE MAPE     MASE         ACF1
## Training set -4.37894 1758.433 1212.57 NaN  Inf 0.353787 -0.007004042
```

```
bestfit_rmse
```

```
## [1] 1758.433
```

```
result= cbind('choic1 with seasonal differencing ARMA(3,4)', bestfit_arima1$aicc,accu
racy(bestfit_arima1)[2])
colnames(result) <- c('Model', 'AICC', 'RMSE')


## choice 2
p=q=0
for (p in 1:4 )
{ for (q in 0:4){
  try(
  fit <- Arima (train_diff, order = c(p,0,q))
  )
  if (fit$aicc < bestfit_arima2$aicc)
    bestfit_arima2<-fit
    else break;
  }
}

bestfit_arima2
```

```
## Series: train_diff
## ARIMA(1,0,3) with non-zero mean
##
## Coefficients:
##           ar1      ma1      ma2      ma3    mean
##       -0.7220  -0.0366  -0.8836  -0.0798  1.5483
## s.e.   0.0571   0.0680   0.0287   0.0469  0.3781
##
## sigma^2 estimated as 6589043:  log likelihood=-8528.69
## AIC=17069.38   AICc=17069.47   BIC=17098.32
```

```
result_temp= cbind('choic2 with first order differencing ARMA(1,3)', bestfit_arima2$a
icc,accuracy(bestfit_arima2)[2])
result = rbind(result, result_temp)



## choice 3

for (p in 0:2 )
{ for (q in 0:2){
    for (P in 0:2){
      for (Q in 0:2){
        try(
        fit <-Arima (train_w[,2], order = c(p,0,q), seasonal = list(order =c(0,1,0),
period =7),
                     include.drift = TRUE, lambda = 'auto',method = 'ML')
        )
        if (fit$aicc < bestfit_arima3$aicc)
          bestfit_arima3 <- fit
          else break

      }
    }
  }
}

bestfit_arima3
```

```
## Series: train_w[, 2]
## ARIMA(2,0,2)(0,1,0)[7] with drift
## Box Cox transformation: lambda= 0.2045416
##
## Coefficients:
##          ar1      ar2      ma1     ma2   drift
##       1.1522  -0.7127  -1.2610  1.0000  0.0063
## s.e.  0.0240   0.0236   0.0046  0.0058  0.0383
##
## sigma^2 estimated as 38.03:  log likelihood=-2961.69
## AIC=5935.38   AICc=5935.47   BIC=5964.28
```

```
result_temp<- cbind('BoxCox Transform with Seasonal differencing ARMA(0,0,0)(0,1,1)[7
]', bestfit_arima3$aicc,accuracy(bestfit_arima3)[2])
result = rbind(result, result_temp)

bestfit_arima_auto<-Arima (train_w[,2], order = c(1,0,0), seasonal = list(order =c(1,
1,0), period =7),
                include.drift = TRUE, lambda = 'auto',method = 'ML')
bestfit_arima_auto$aicc
```

```
## [1] 5805.176
```

```
accuracy(bestfit_arima_auto)
```

```
##                     ME     RMSE      MAE MPE MAPE      MASE       ACF1
## Training set 193.7008 2062.694 1305.924 NaN  Inf 0.7286638 0.09133815
```

```
result_temp<- cbind('auto arima (1,0,0)(1,1,0)', bestfit_arima_auto$aicc,accuracy(bes
tfit_arima_auto)[2])
result = rbind(result, result_temp)



#choice 4 for first seasonal differencing and first order differencing
bestfit_arima4 <-list(aicc = Inf)
bestfit_rmse4 <- list(rmse= NA)
for (p in 0:2 )
{ for (q in 0:2){
    for (P in 0:2){
      for (Q in 0:2){
        try(
        fit <-Arima (train_w[,2], order = c(p,0,q), seasonal = list(order =c(P,1,Q),
period =7),
                     include.drift = TRUE, lambda = 'auto',method = 'ML')
        )
        if (fit$aicc < bestfit_arima4$aicc)
          bestfit_arima4 <- fit
          else break

      }
    }
  }
}
```

```
## Error in optim(init[mask], armafn, method = optim.method, hessian = TRUE,  :
##   non-finite finite-difference value [1]
```

```
bestfit_arima4
```

```
## Series: train_w[, 2]
## ARIMA(0,0,0)(0,1,1)[7] with drift
## Box Cox transformation: lambda= 0.2045416
##
## Coefficients:
##          sma1   drift
##       -1.0000  0.0016
## s.e.   0.0236  0.0006
##
## sigma^2 estimated as 22.46:  log likelihood=-2734.98
## AIC=5475.97   AICc=5475.99   BIC=5490.42
```

```
result_temp<- cbind('BoxCox Transform with first order and Seasonal differencing ARMA
(0,1,1)(1,1,2)[7]', bestfit_arima4$aicc,accuracy(bestfit_arima4)[2])
result = rbind(result, result_temp)


#choice 5 for first seasonal differencing and first order differencing
bestfit_arima5 <-list(aicc = Inf)
bestfit_rmse5 <- list(rmse= NA)
for (p in 0:1 )
{ for (q in 0:1){
    for (P in 0:1){
      for (Q in 0:1){
        try(
        fit <-Arima (train_w[,2], order = c(p,1,q), seasonal = list(order =c(P,0,Q),
period =7),
                     include.drift = TRUE, lambda = 'auto',method = 'ML')
        )
        if (fit$aicc < bestfit_arima5$aicc)
          bestfit_arima5 <- fit
          else break

      }
    }
  }
}
bestfit_arima5
```

```
## Series: train_w[, 2]
## ARIMA(0,1,1)(1,0,1)[7] with drift
## Box Cox transformation: lambda= 0.2045416
##
## Coefficients:
##          ma1  sar1     sma1   drift
##       -1e+00     1  -0.9884  0.0015
## s.e.   9e-04     0   0.0032  0.0007
##
## sigma^2 estimated as 22.73:  log likelihood=-2758.74
## AIC=5527.47   AICc=5527.54   BIC=5551.59
```

```
result_temp<- cbind('BoxCox Transform with first order differencing ARMA(0,1,1)(1,0,1
)[7]', bestfit_arima5$aicc,accuracy(bestfit_arima5)[2])
result = rbind(result, result_temp)

bestfit_arima6 <-Arima (train_w[,2], order = c(1,0,1), seasonal = list(order =c(1,1,1
), period =7),
                include.drift = TRUE, lambda = 'auto',method = 'ML')

bestfit_arima_auto
```

```
## Series: train_w[, 2]
## ARIMA(1,0,0)(1,1,0)[7] with drift
## Box Cox transformation: lambda= 0.2045416
##
## Coefficients:
##          ar1     sar1   drift
##       0.0518  -0.5116  0.0045
## s.e.  0.0335   0.0286  0.0190
##
## sigma^2 estimated as 33.3:  log likelihood=-2898.57
## AIC=5805.13   AICc=5805.18   BIC=5824.4
```

```
bestfit_arima6
```

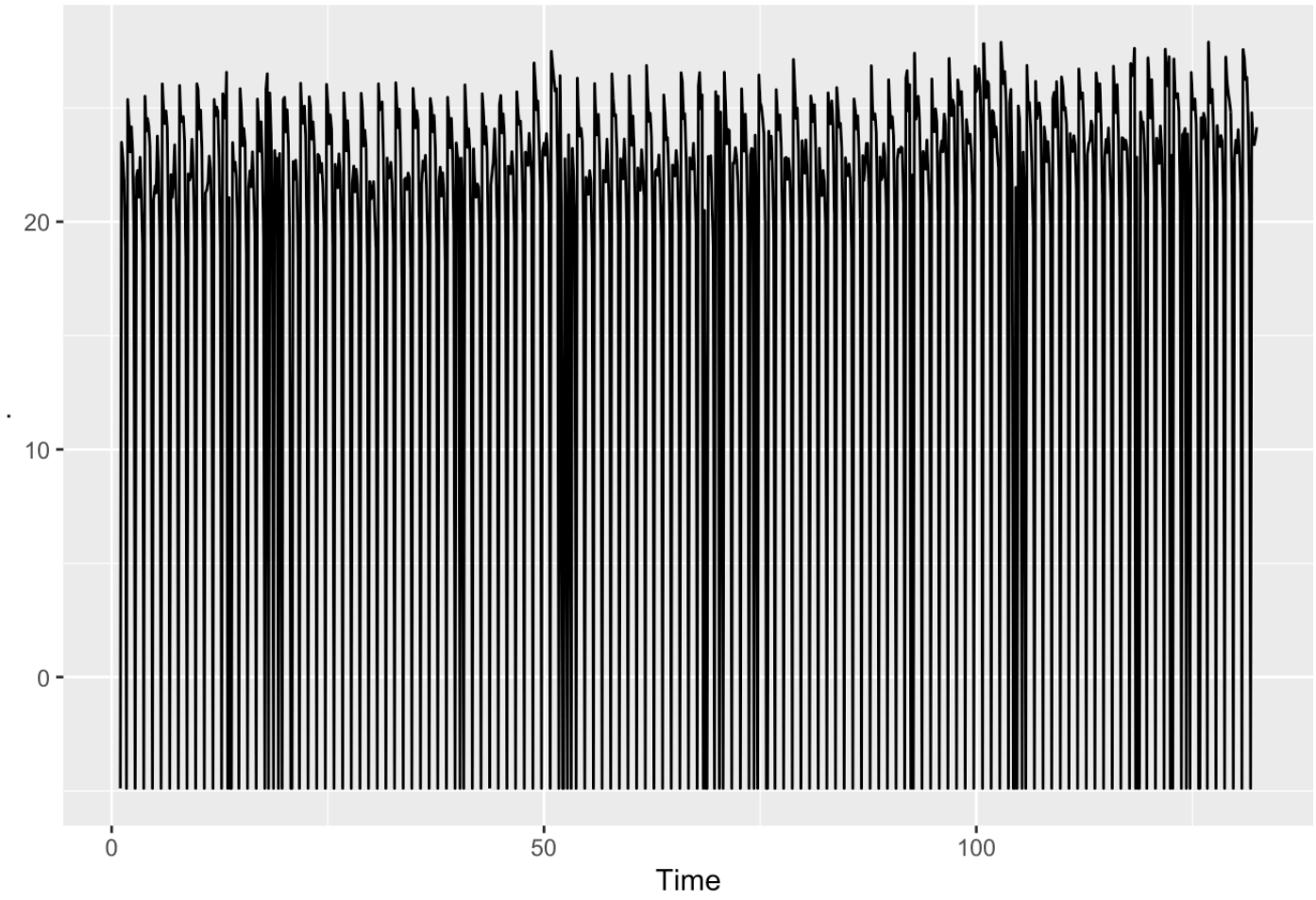
```
## Series: train_w[, 2]
## ARIMA(1,0,1)(1,1,1)[7] with drift
## Box Cox transformation: lambda= 0.2045416
##
## Coefficients:
##          ar1      ma1     sar1     sma1   drift
##       0.4923  -0.4137  -0.0178  -0.9998  0.0016
## s.e.  0.2455   0.2563   0.0338   0.0530  0.0007
##
## sigma^2 estimated as 22.35:  log likelihood=-2731.24
## AIC=5474.49   AICc=5474.58   BIC=5503.39
```

```
result_temp<- cbind('boxcox transform with seasonal differencing ARIMA (1,0,1)(1,1,1)
', bestfit_arima6$aicc,accuracy(bestfit_arima6)[2])
result = rbind(result, result_temp)

result
```

```
##      Model
## [1,] "choic1 with seasonal differencing ARMA(3,4)"
## [2,] "choic2 with first order differencing ARMA(1,3)"
## [3,] "BoxCox Transform with Seasonal differencing ARMA(0,0,0)(0,1,1)[7]"
## [4,] "auto arima (1,0,0)(1,1,0)"
## [5,] "BoxCox Transform with first order and Seasonal differencing ARMA(0,1,1)(1,1,
2)[7]"
## [6,] "BoxCox Transform with first order differencing ARMA(0,1,1)(1,0,1)[7]"
## [7,] "boxcox transform with seasonal differencing ARIMA (1,0,1)(1,1,1)"
##      AICC               RMSE
## [1,] "16286.5299636318" "1758.43320044006"
## [2,] "17069.468986031"  "2559.92834140316"
## [3,] "5935.46937492017" "2663.44094437545"
## [4,] "5805.17606676936" "2062.6935460107"
## [5,] "5475.99279264714" "1747.22148233029"
## [6,] "5527.53854731872" "1785.25615966751"
## [7,] "5474.57962273049" "1711.579971031"
```

Auto ARIMA

```
train_w[,2] %>% BoxCox(lambda = 'auto') %>% autoplot()
```




```
Arima_auto=auto.arima(train_w[,2], lambda= 'auto', approximation = TRUE, seasonal = T
RUE)
summary(Arima_auto)
```

```
## Series: train_w[, 2]
## ARIMA(1,0,0)(1,1,0)[7] with drift
## Box Cox transformation: lambda= 0.2045416
##
## Coefficients:
##          ar1     sar1   drift
##       0.0518  -0.5116  0.0045
## s.e.  0.0335   0.0286  0.0190
##
## sigma^2 estimated as 33.3:  log likelihood=-2898.57
## AIC=5805.13   AICc=5805.18   BIC=5824.4
##
## Training set error measures:
##                     ME     RMSE      MAE MPE MAPE      MASE     ACF1
## Training set 193.6749 2062.697 1305.926 NaN  Inf 0.7286649 0.091337
```

```
arimaorder(Arima_auto)
```

```
##         p         d         q         P         D         Q Frequency
##         1         0         0         1         1         0         7
```

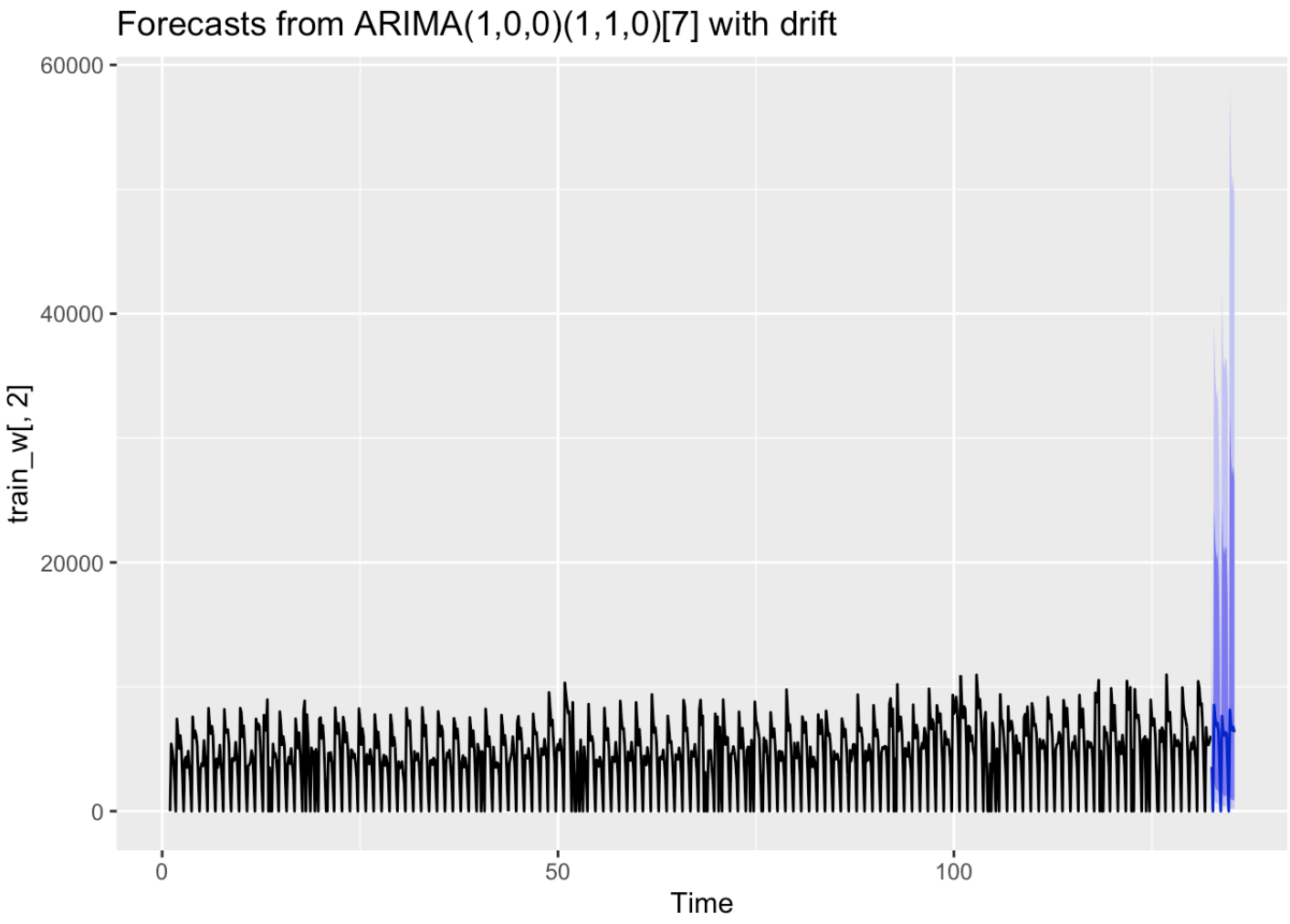
```
m1=auto.arima(train_w[,2], lambda= 'auto')

summary(m1)
```

```
## Series: train_w[, 2]
## ARIMA(1,0,0)(1,1,0)[7] with drift
## Box Cox transformation: lambda= 0.2045416
##
## Coefficients:
##          ar1     sar1   drift
##       0.0518  -0.5116  0.0045
## s.e.  0.0335   0.0286  0.0190
##
## sigma^2 estimated as 33.3:  log likelihood=-2898.57
## AIC=5805.13   AICc=5805.18   BIC=5824.4
##
## Training set error measures:
##                     ME     RMSE      MAE MPE MAPE      MASE     ACF1
## Training set 193.6749 2062.697 1305.926 NaN  Inf 0.7286649 0.091337
```

```
arimaorder(m1)
```

```
##         p         d         q         P         D         Q Frequency
##         1         0         0         1         1         0         7
```

```
fc_auto<-forecast(Arima_auto,h = 21,level=c(80,95) )
fc_80 <-forecast(Arima_auto,h = 21,level=80)
fc_95<-forecast(Arima_auto,h = 21,level=95)
autoplot(fc_auto)
```



Comparison between auto Arima and bestfit_arima6

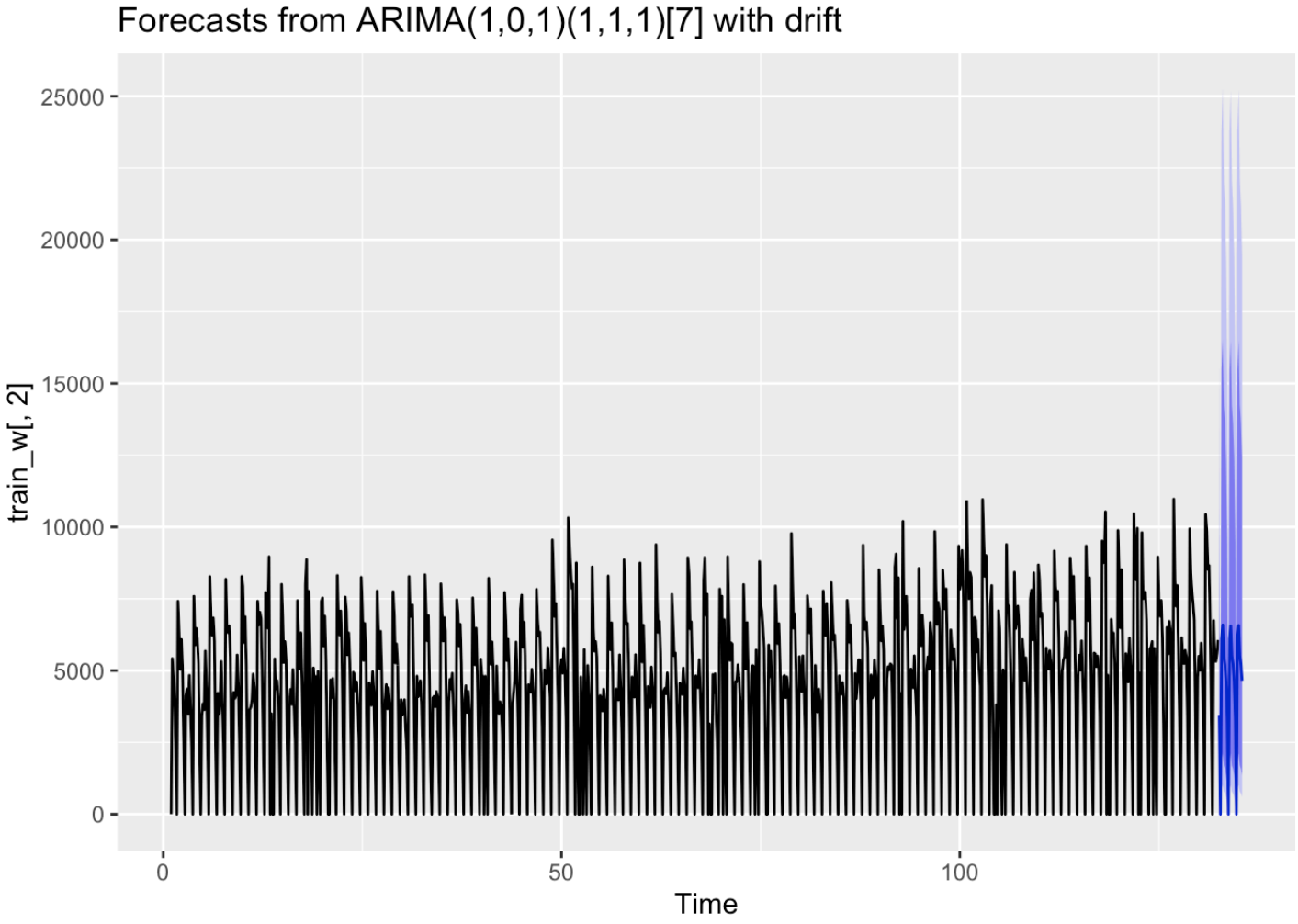
```
# Comparison between auto Arima and bestfit_arima6

accuracy(bestfit_arima6)
```

```
##                      ME    RMSE      MAE MPE MAPE      MASE      ACF1
## Training set 642.6458 1711.58 1156.843 NaN  Inf 0.6454813 0.1597024
```

```
accuracy(Arima_auto)
```

```
##                      ME     RMSE      MAE MPE MAPE      MASE     ACF1
## Training set 193.6749 2062.697 1305.926 NaN  Inf 0.7286649 0.091337
```

```
fc_arima6<-forecast(bestfit_arima6, h = 21,level=c(80,95))
autoplot(fc_arima6)
```



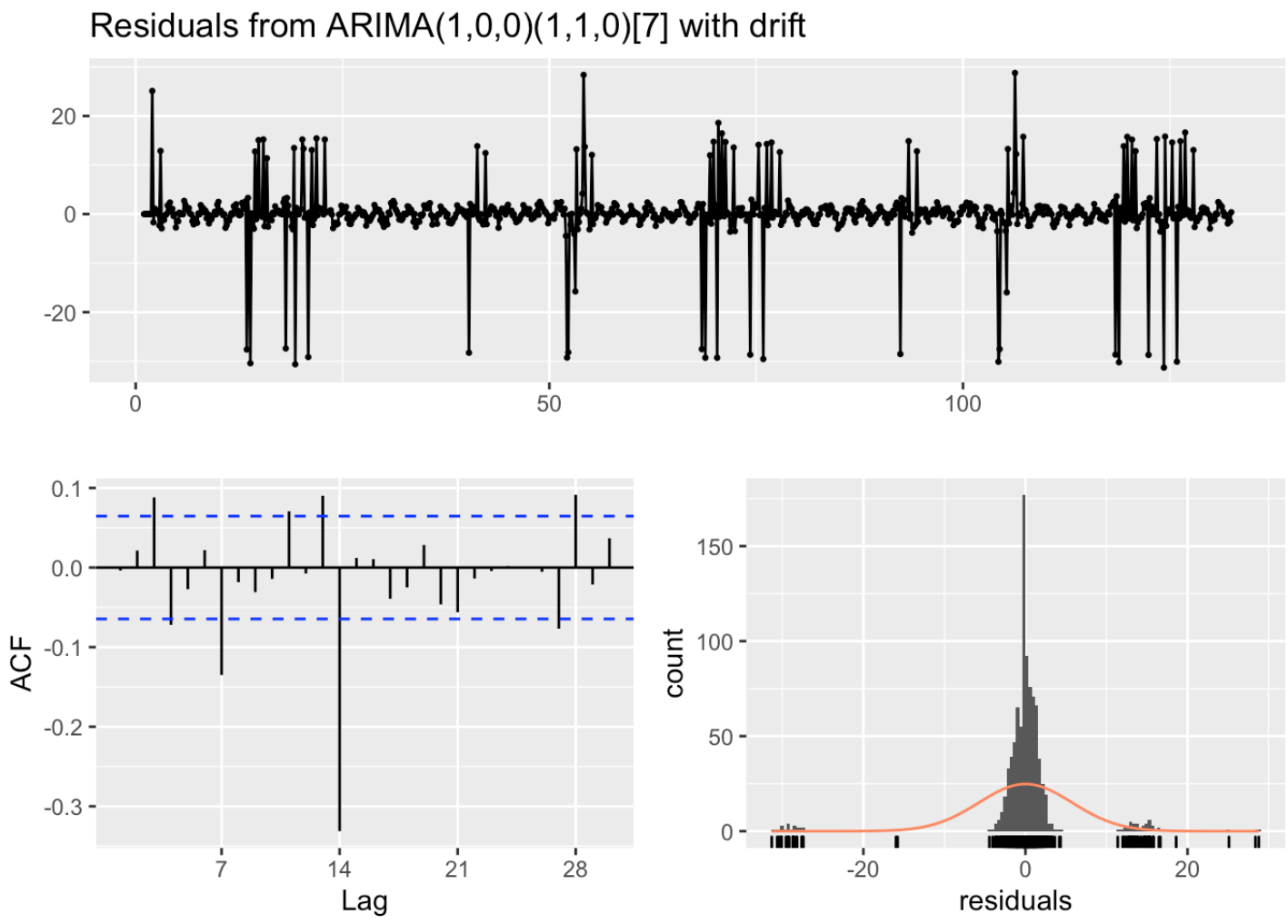

```
accuracy(fc_auto,test_w[,2])
```

```
##                     ME     RMSE       MAE  MPE MAPE      MASE      ACF1
## Training set 193.6749 2062.697 1305.9261  NaN  Inf 0.7286649 0.0913370
## Test set     211.1674 1022.575  786.8524 -Inf  Inf 0.4390384 0.6070192
##              Theil's U
## Training set        NA
## Test set             0
```

```
accuracy(fc_arima6,test_w[,2])
```

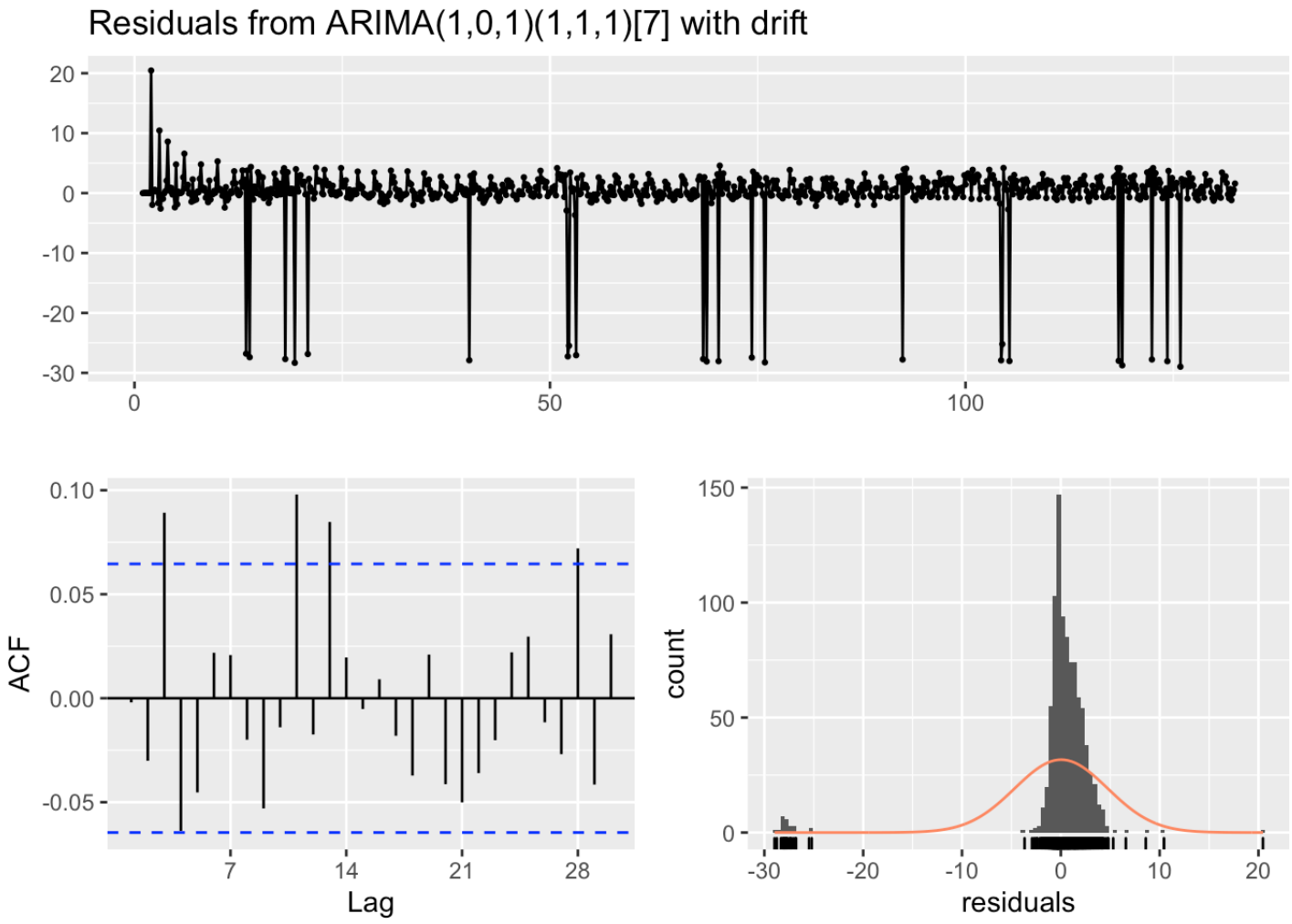
```
##                     ME     RMSE       MAE  MPE MAPE      MASE      ACF1
## Training set  642.6458 1711.580 1156.843  NaN  Inf 0.6454813 0.1597024
## Test set     1164.1599 1889.326 1367.790 -Inf  Inf 0.7631829 0.4926323
##              Theil's U
## Training set        NA
## Test set             0
```

```
checkresiduals(Arima_auto)
```



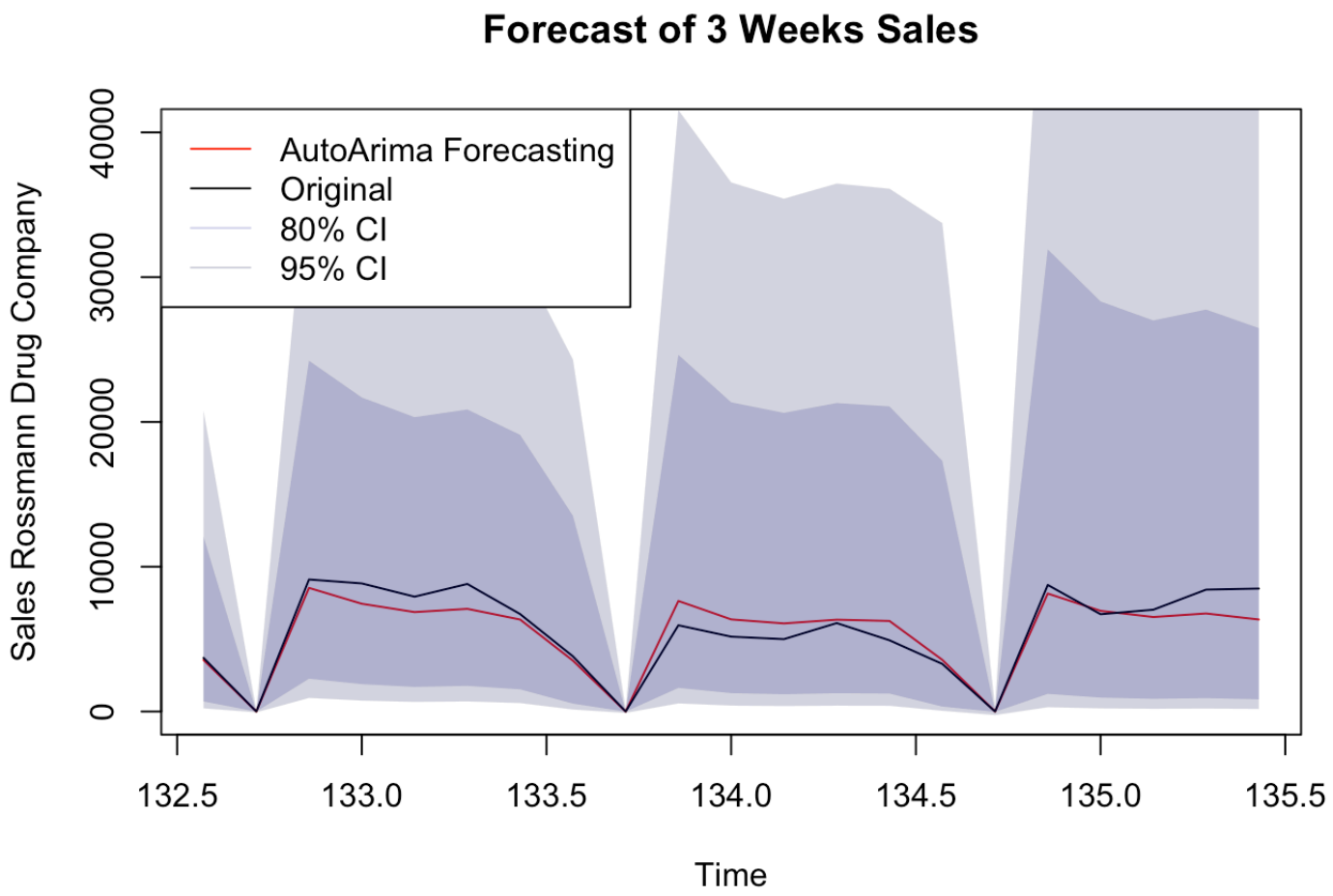
```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(1,1,0)[7] with drift
## Q* = 147.09, df = 11, p-value < 2.2e-16
##
## Model df: 3.   Total lags used: 14
```

```
checkresiduals(bestfit_arima6)
```

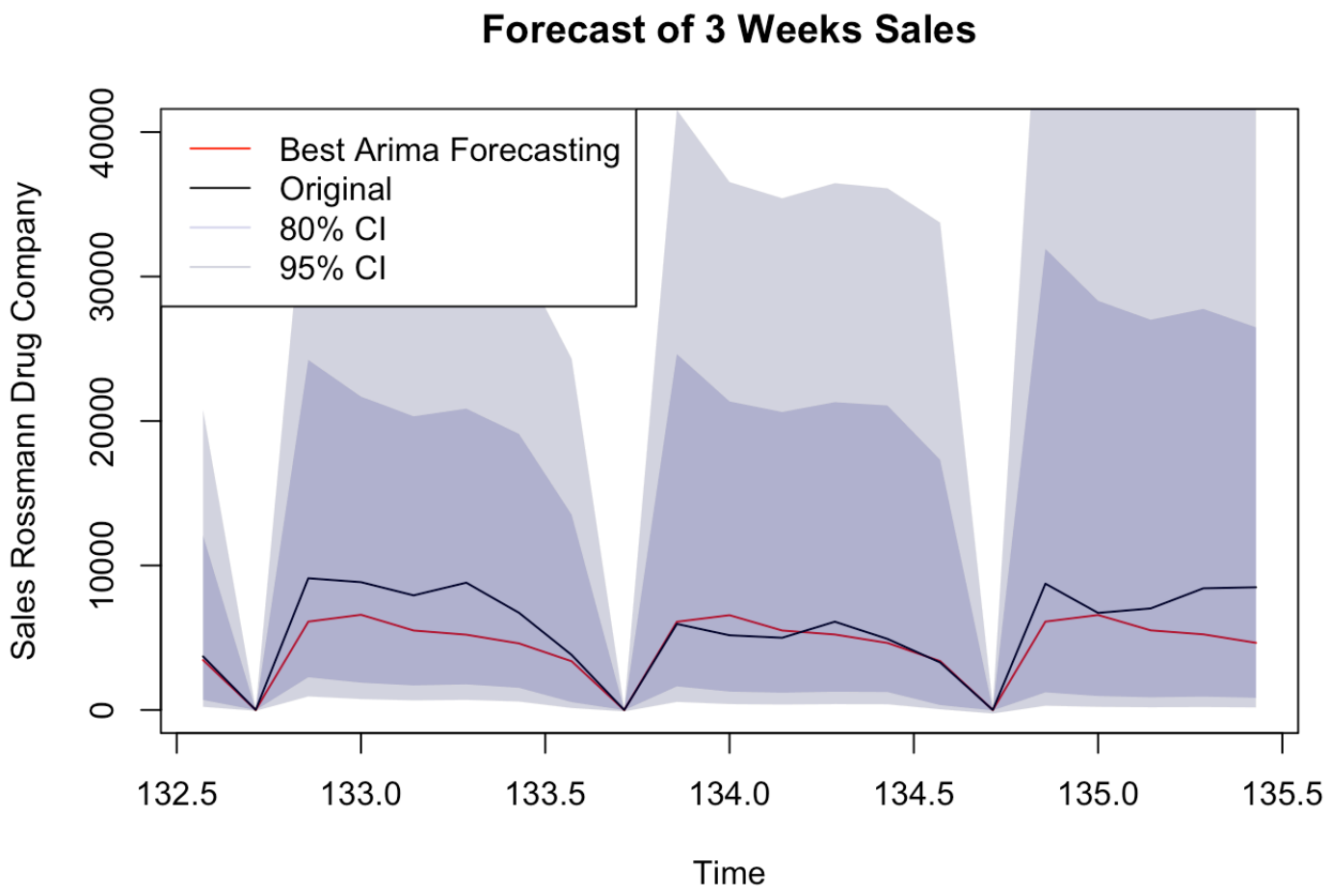



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1)(1,1,1)[7] with drift
## Q* = 34.227, df = 9, p-value = 8.149e-05
##
## Model df: 5.   Total lags used: 14
```

```
plot(fc_auto$mean, ylim = c(0,40000), ylab = 'Sales Rossmann Drug Company', col = 2,
main = 'Forecast of 3 Weeks Sales')
lines(test_w[,2], col=1, type = 'l')
polygon(c(time(test_w[,2]),rev(time(test_w[,2]))), c(fc_80$upper,rev(fc_80$lower)),
# create %80 interval
   col=rgb(0,0,0.6,0.2), border=FALSE)
polygon(c(time(test_w[,2]),rev(time(test_w[,2]))), c(fc_95$upper,rev(fc_95$lower)),
# create %95 interval
   col=rgb(0,0,0.3,0.2), border=FALSE)
legend('topleft', legend = c('AutoArima Forecasting', 'Original', '80% CI', '95% CI')
, col=c(2, 1,rgb(0,0,0.6,0.2),rgb(0,0,0.3,0.2)), lty = 1)
```

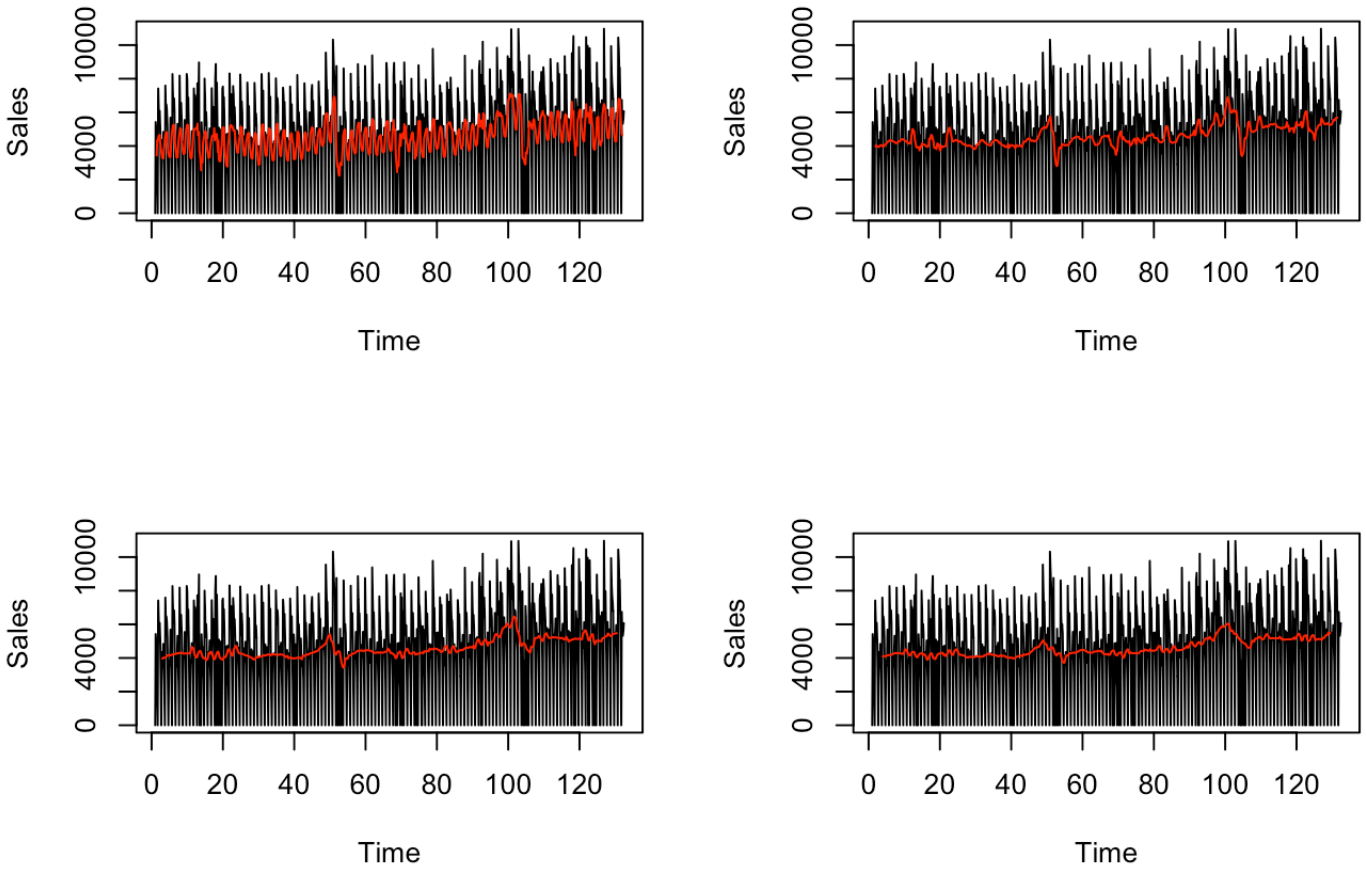


```
plot(fc_arima6$mean, ylim = c(0,40000), ylab = 'Sales Rossmann Drug Company', col = 2
, main = 'Forecast of 3 Weeks Sales')
lines(test_w[,2], col=1, type = 'l')
polygon(c(time(test_w[,2]),rev(time(test_w[,2]))), c(fc_80$upper,rev(fc_80$lower)),
# create %80 interval
   col=rgb(0,0,0.6,0.2), border=FALSE)
polygon(c(time(test_w[,2]),rev(time(test_w[,2]))), c(fc_95$upper,rev(fc_95$lower)),
# create %95 interval
   col=rgb(0,0,0.3,0.2), border=FALSE)
legend('topleft', legend = c('Best Arima Forecasting', 'Original', '80% CI', '95% CI'
) , col=c(2, 1,rgb(0,0,0.6,0.2),rgb(0,0,0.3,0.2)), lty = 1)
```

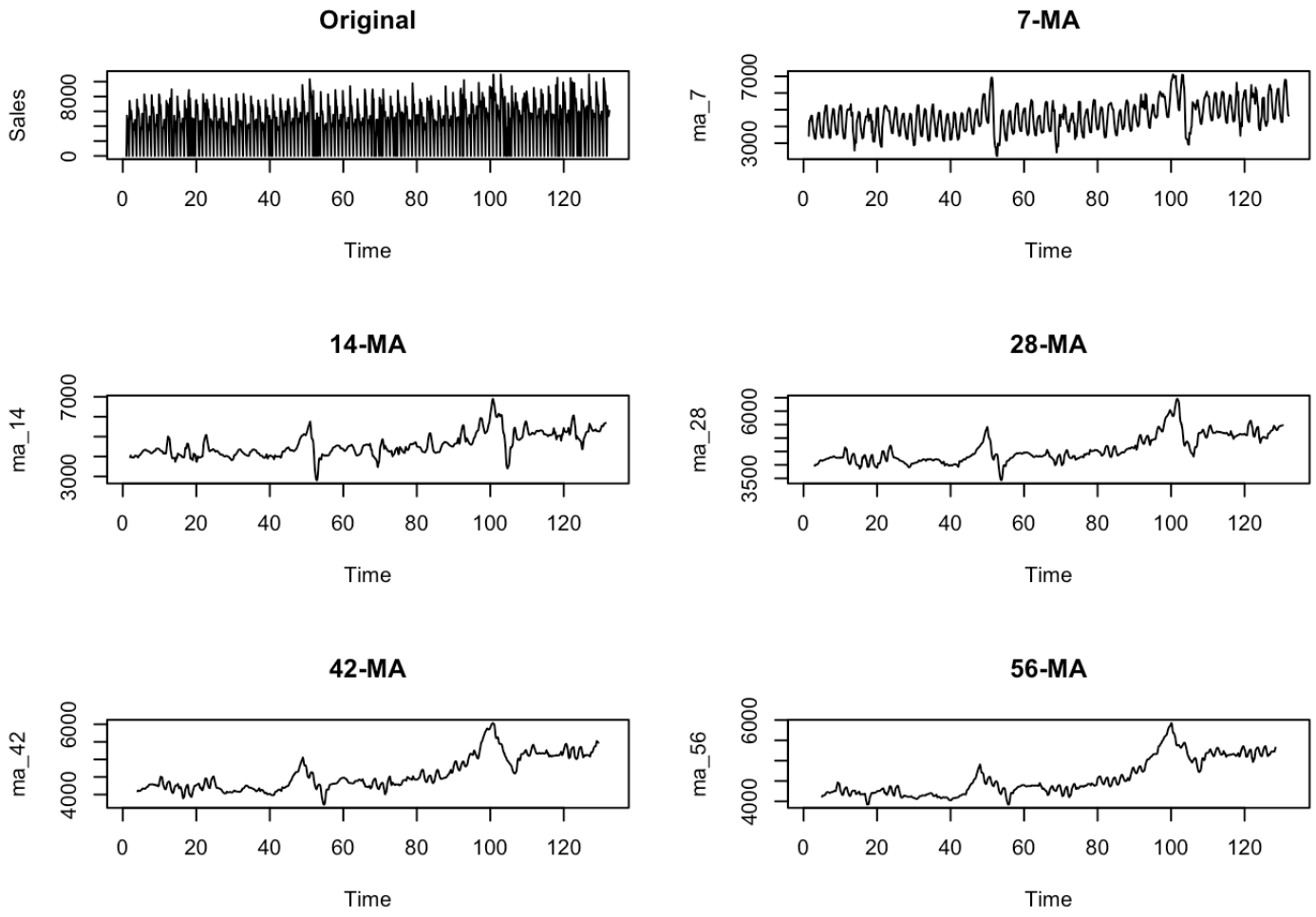


Moving Average Smoothing

```
ma_7<-ma(train_w[,2],7)
ma_14 <-ma(train_w[,2],14)
ma_28 <-ma(train_w[,2],28)
ma_42 <-ma(train_w[,2],42)
ma_56<-ma(train_w[,2],56)

par(mfrow = c(2,2))
plot(train_w[,2], ylab = 'Sales')
points(ma_7, main = '7-MA', type = 'l',col = 2)

plot(train_w[,2], ylab = 'Sales')
points(ma_14, main = '14-MA', type = 'l',col = 2)

plot(train_w[,2], ylab = 'Sales')
points(ma_28, main = '28-MA', type = 'l',col = 2)

plot(train_w[,2], ylab = 'Sales')
points(ma_42, main = '42-MA', type = 'l',col = 2)
```

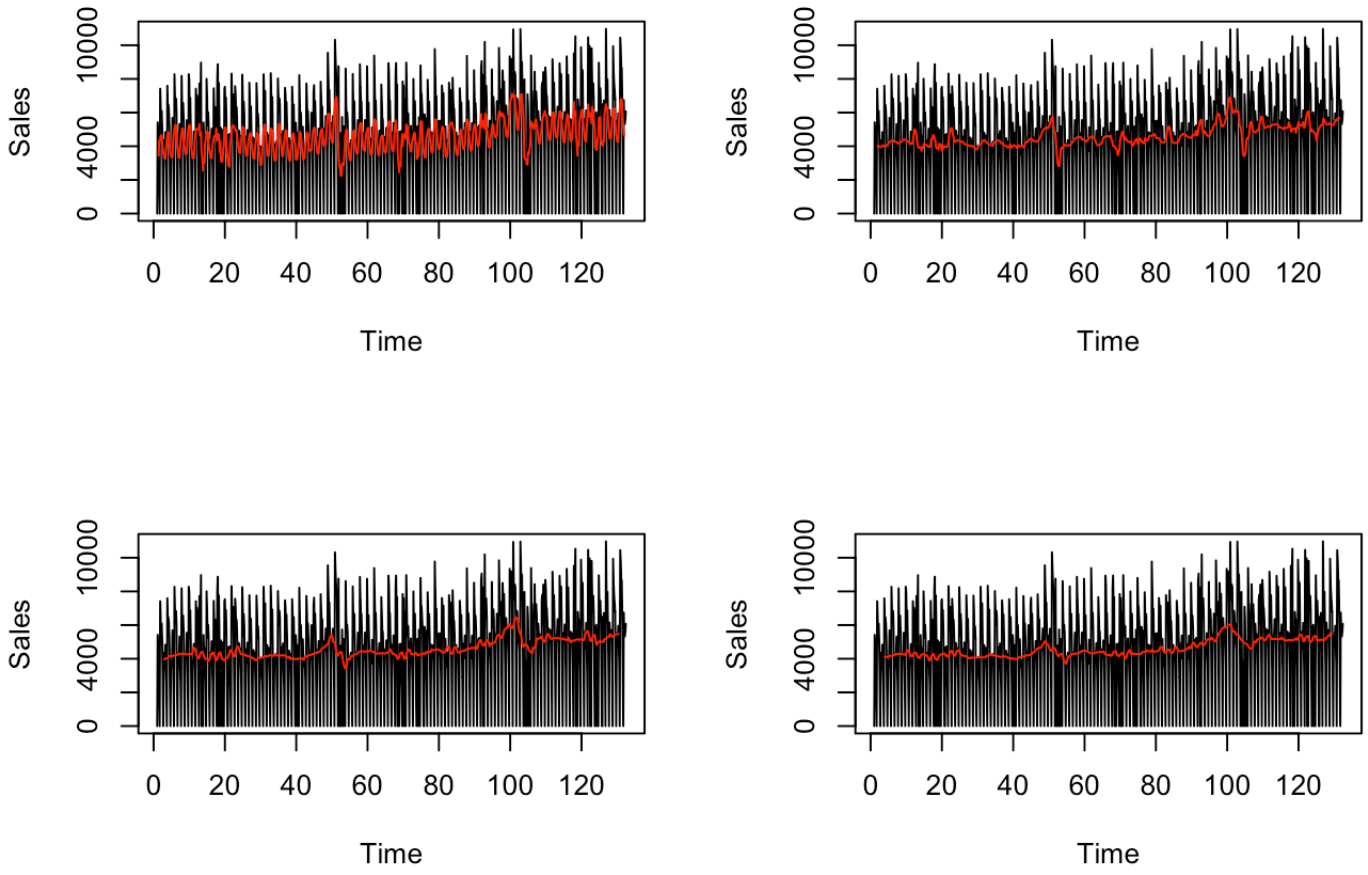


```
par(mfrow = c(3,2))
plot(train_w[,2], main = 'Original', ylab = 'Sales')
plot(ma_7, main = '7-MA')
plot(ma_14, main = '14-MA')
plot(ma_28, main = '28-MA')
plot(ma_42, main = '42-MA')
plot(ma_56, main = '56-MA')
```

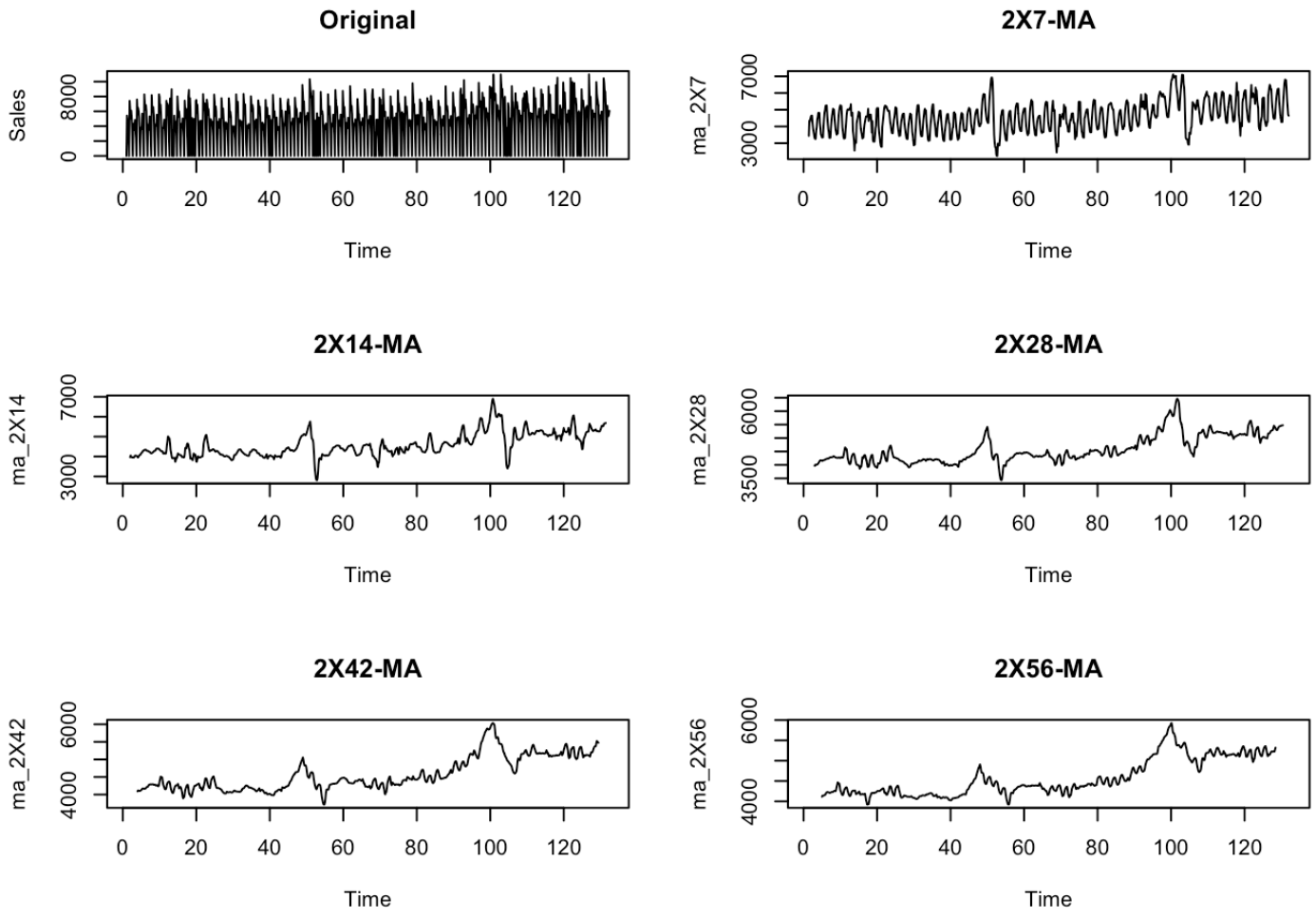


Weighted Moving Average

```
ma_2X7<-ma(train_w[,2],order = 7, centre = TRUE)

ma_2X14 <-ma(train_w[,2],14,centre = TRUE)
ma_2X28 <-ma(train_w[,2],28,centre = TRUE)
ma_2X42 <-ma(train_w[,2],42,centre = TRUE)
ma_2X56<-ma(train_w[,2],56,centre = TRUE)

par(mfrow = c(2,2))
plot(train_w[,2], ylab = 'Sales')
points(ma_2X7, main = '7-MA', type = 'l',col = 2)

plot(train_w[,2], ylab = 'Sales')
points(ma_2X14, main = '14-MA', type = 'l',col = 2)

plot(train_w[,2], ylab = 'Sales')
points(ma_2X28, main = '28-MA', type = 'l',col = 2)

plot(train_w[,2], ylab = 'Sales')
points(ma_2X42, main = '42-MA', type = 'l',col = 2)
```



```
par(mfrow = c(3,2))
plot(train_w[,2], main = 'Original', ylab = 'Sales')
plot(ma_2X7, main = '2X7-MA')
plot(ma_2X14, main = '2X14-MA')
plot(ma_2X28, main = '2X28-MA')
plot(ma_2X42, main = '2X42-MA')
plot(ma_2X56, main = '2X56-MA')
```



Holt-Winter Model

here we are using additive Holt_Winter's Model. Since our data contains 0 values so additive seasonality are used here.

```
hw_model= hw(train_w[,2],h=21, damped = TRUE, seasonal='additive')
summary(hw_model)
```

```
##
```

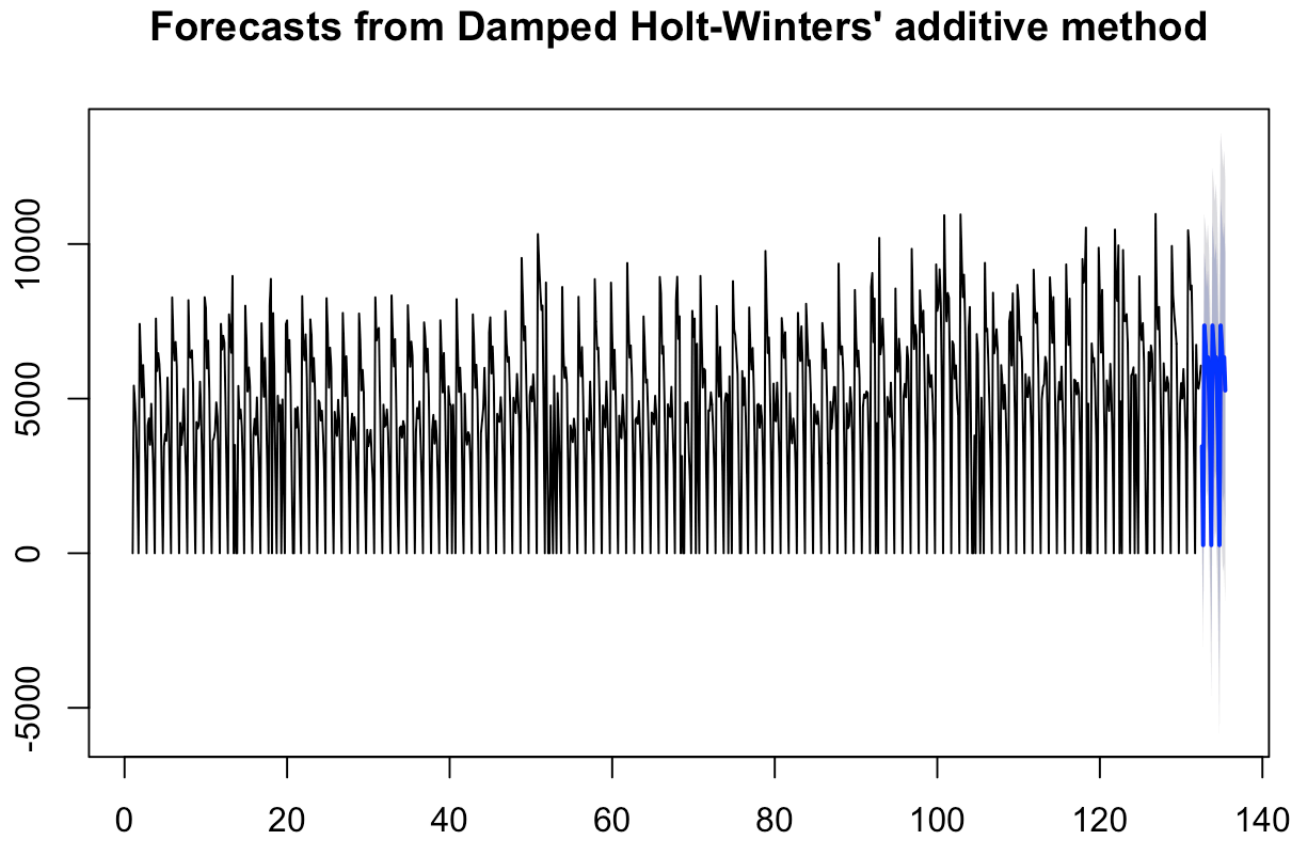
```
## Forecast method: Damped Holt-Winters' additive method
##
## Model Information:
## Damped Holt-Winters' additive method
##
## Call:
##  hw(y = train_w[, 2], h = 21, seasonal = "additive", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.4485
##     beta  = 1e-04
##     gamma = 0.0156
##     phi   = 0.8986
##
##   Initial states:
##     l = 2933.5121
##     b = 123.4475
##     s = 2131.94 -4332.449 -1609.358 331.3822 1421.986 802.2309
##           1254.268
##
##   sigma:  1540.189
##
##      AIC     AICc      BIC
## 19819.82 19820.23 19882.56
##
## Error measures:
##                     ME     RMSE      MAE MPE MAPE     MASE       ACF1
## Training set 2.441195 1530.122 998.9087 NaN  Inf 0.557359 0.01242899
##
## Forecasts:
##          Point Forecast      Lo 80     Hi 80      Lo 95     Hi 95
## 132.5714      3457.1089  1483.2772  5430.941   438.3939  6475.824
## 132.7143       265.1005 -1898.2205  2428.422 -3043.4134  3573.614
## 132.8571      7362.8000  5025.2401  9700.360  3787.8106 10937.789
## 133.0000      6698.9706  4199.2369  9198.704  2875.9578 10521.983
## 133.1429      5915.3285  3263.2767  8567.380  1859.3653  9971.292
## 133.2857      6335.0948  3538.9736  9131.216  2058.7964 10611.393
## 133.4286      5264.9687  2331.8146  8198.123   779.0966  9750.841
## 133.5714      3456.8061   383.6830  6529.929 -1243.1300  8156.742
## 133.7143       264.8284 -2933.5303  3463.187 -4626.6391  5156.296
## 133.8571      7362.5555  4043.6633 10681.448  2286.7479 12438.363
## 134.0000      6698.7509  3263.5344 10133.967  1445.0407 11952.461
## 134.1429      5915.1311  2367.3871  9462.875   489.3249 11340.937
## 134.2857      6334.9174  2678.0938  9991.741   742.2883 11927.547
## 134.4286      5264.8093  1502.0552  9027.563  -489.8265 11019.445
## 134.5714      3456.6628  -416.2959  7329.622 -2466.5165  9379.842
## 134.7143       264.6997 -3708.4498  4237.849 -5811.7081  6341.108
```

```
## 134.8571      7362.4398  3291.5569 11433.323  1136.5616 13588.318
## 135.0000      6698.6470  2532.3158 10864.978   326.7933 13070.501
## 135.1429      5915.0377  1655.3905 10174.685  -599.5305 12429.606
## 135.2857      6334.8335  1983.8659 10685.801  -319.3972 12989.064
## 135.4286      5264.7339   824.3190  9705.149 -1526.2947 12055.762
```

```
hw_model$model
```

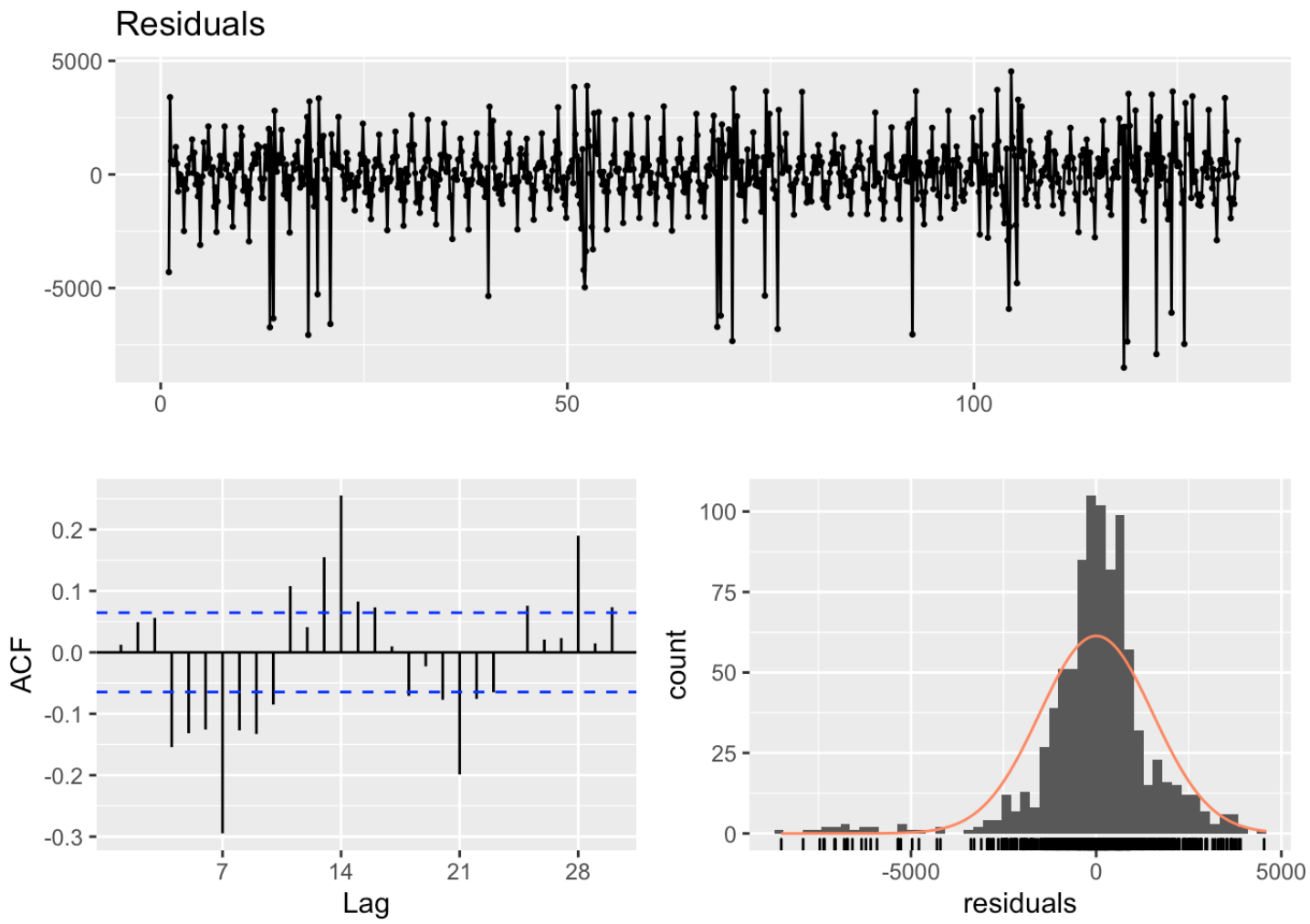
```
## Damped Holt-Winters' additive method
##
## Call:
##  hw(y = train_w[, 2], h = 21, seasonal = "additive", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.4485
##     beta  = 1e-04
##     gamma = 0.0156
##     phi   = 0.8986
##
##   Initial states:
##     l = 2933.5121
##     b = 123.4475
##     s = 2131.94 -4332.449 -1609.358 331.3822 1421.986 802.2309
##           1254.268
##
##   sigma:  1540.189
##
##      AIC     AICc      BIC
## 19819.82 19820.23 19882.56
```

```
plot(hw_model)
```




```
checkresiduals(hw_model$residuals)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```



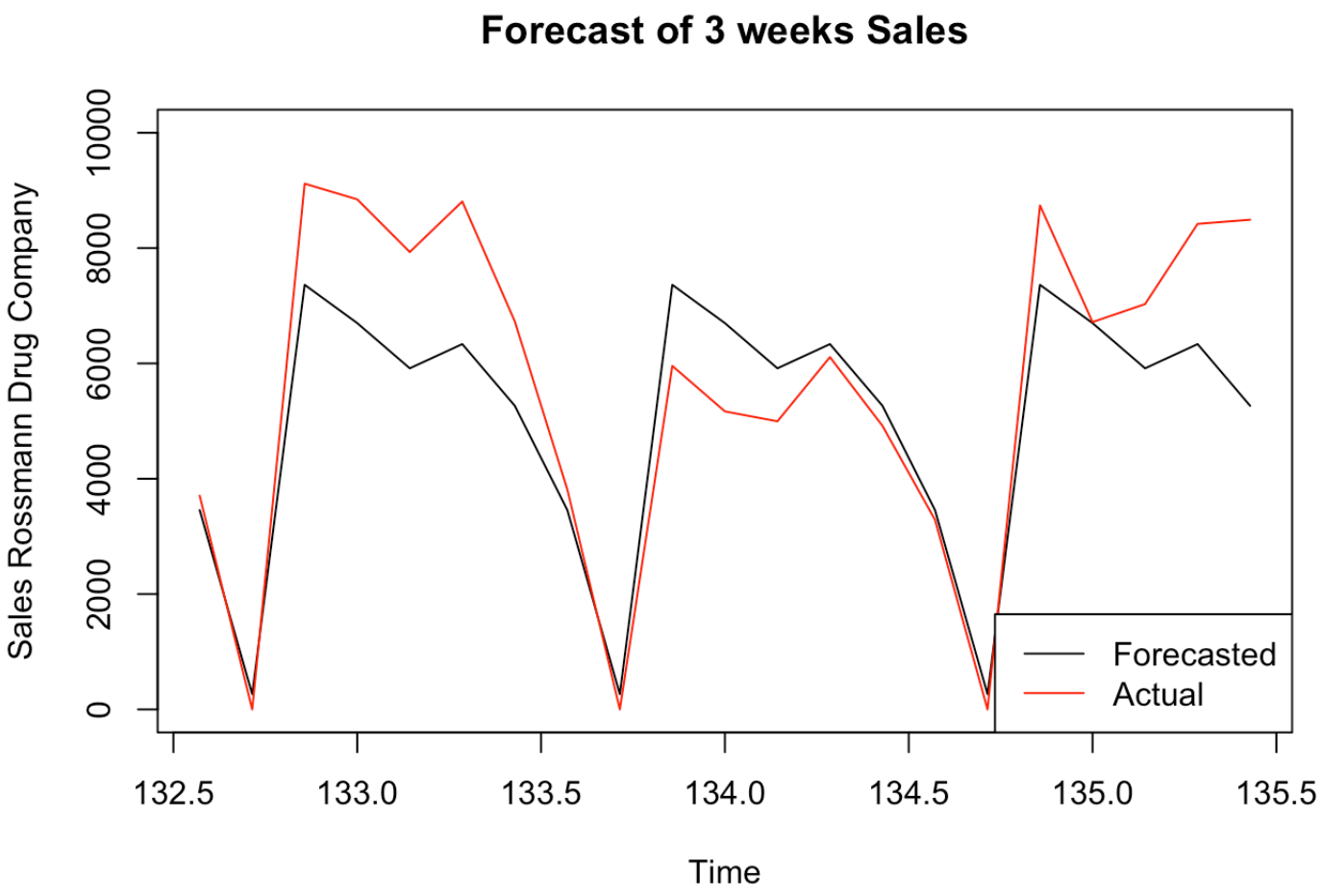

```
hw_acc<- accuracy(hw_model$mean, test_w[,2])
accuracy(hw_model)
```

```
##                      ME     RMSE      MAE MPE MAPE     MASE       ACF1
## Training set 2.441195 1530.122 998.9087 NaN  Inf 0.557359 0.01242899
```

```
hw_acc
```

```
##                 ME     RMSE      MAE  MPE MAPE      ACF1 Theil's U
## Test set 613.6559 1442.077 1127.318 -Inf  Inf 0.6570323         0
```

```
plot(hw_model$mean, ylim = c(0,10000), ylab = 'Sales Rossmann Drug Company', col = 1,
main = 'Forecast of 3 weeks Sales')
points(test_w[,2], col='2', type = 'l')
legend ('bottomright',legend = c('Forecasted','Actual'),col =c( 1,2),lty= 1)
```




```
hw_model_no_damped= hw(train_w[,2],h=21, damped = FALSE, seasonal='additive')
summary(hw_model_no_damped)
```

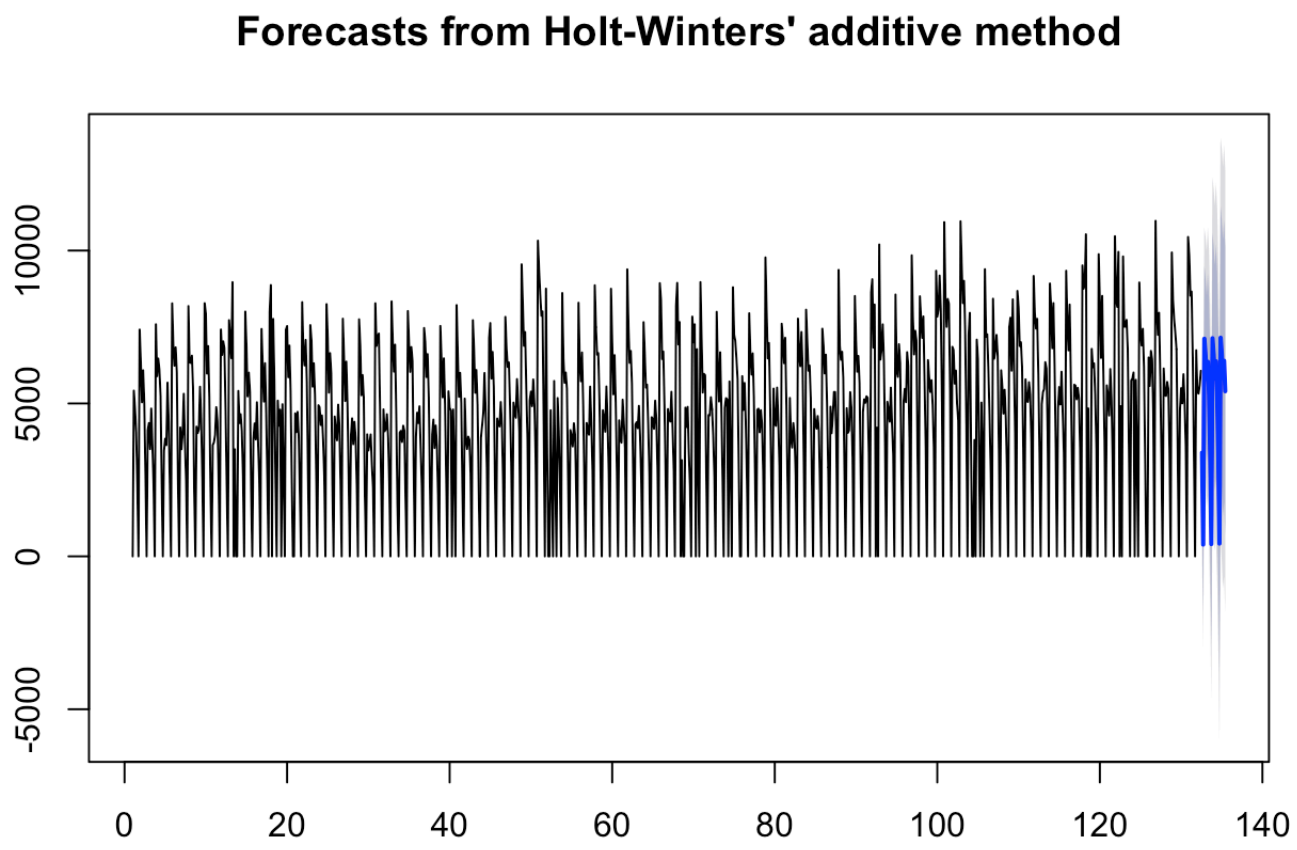
```
##
## Forecast method: Holt-Winters' additive method
##
## Model Information:
## Holt-Winters' additive method
##
## Call:
##  hw(y = train_w[, 2], h = 21, seasonal = "additive", damped = FALSE)
##
##   Smoothing parameters:
##     alpha = 0.4623
##     beta  = 0.002
##     gamma = 2e-04
##
##   Initial states:
##     l = 3245.8519
```

```
##     b = 132.6898
##     s = 2117.873 -4608.712 -1603.409 360.8331 1356.795 805.153
##           1571.467
##
##   sigma:  1539.965
##
##      AIC     AICc      BIC
## 19818.57 19818.91 19876.47
##
## Error measures:
##                      ME     RMSE      MAE MPE MAPE      MASE        ACF1
## Training set -70.02558 1530.741 988.9754 NaN  Inf 0.5518165 0.001825674
##
## Forecasts:
##          Point Forecast      Lo 80     Hi 80       Lo 95     Hi 95
## 132.5714      3389.2400  1415.6954  5362.785   370.96400  6407.516
## 132.7143       386.4292 -1789.5133  2562.372 -2941.38769  3714.246
## 132.8571      7114.8668  4752.2627  9477.471  3501.57565 10728.158
## 133.0000      6570.0718  4033.0602  9107.083  2690.04737 10450.096
## 133.1429      5807.3118  3105.7666  8508.857  1675.65503  9938.969
## 133.2857      6360.6623  3502.7461  9218.578  1989.85661 10731.468
## 133.4286      5366.9221  2359.5185  8374.326   767.49522  9966.349
## 133.5714      3405.8223   254.7316  6556.913 -1413.35497  8225.000
## 133.7143       403.0114 -2886.5423  3692.565 -4627.92687  5433.950
## 133.8571      7131.4491  3707.9271 10554.971  1895.62396 12367.274
## 134.0000      6586.6540  3033.1456 10140.162  1152.03183 12021.276
## 134.1429      5823.8940  2143.9548  9503.833   195.91263 11451.875
## 134.2857      6377.2445  2574.0713 10180.418   560.79301 12193.696
## 134.4286      5383.5044  1459.9887  9307.020  -616.99506 11384.004
## 134.5714      3422.4045  -618.9034  7463.712 -2758.24256  9603.052
## 134.7143       419.5937 -3737.0232  4576.211 -5937.40329  6776.591
## 134.8571      7148.0313  2878.3090 11417.754   618.05453 13678.008
## 135.0000      6603.2363  2222.4377 10984.035   -96.61697 13303.089
## 135.1429      5840.4763  1350.4766 10330.476 -1026.38568 12707.338
## 135.2857      6393.8268  1796.3640 10991.290  -637.38585 13425.039
## 135.4286      5400.0866   696.7762 10103.397 -1793.00599 12593.179
```

```
hw_model_no_damped$model
```

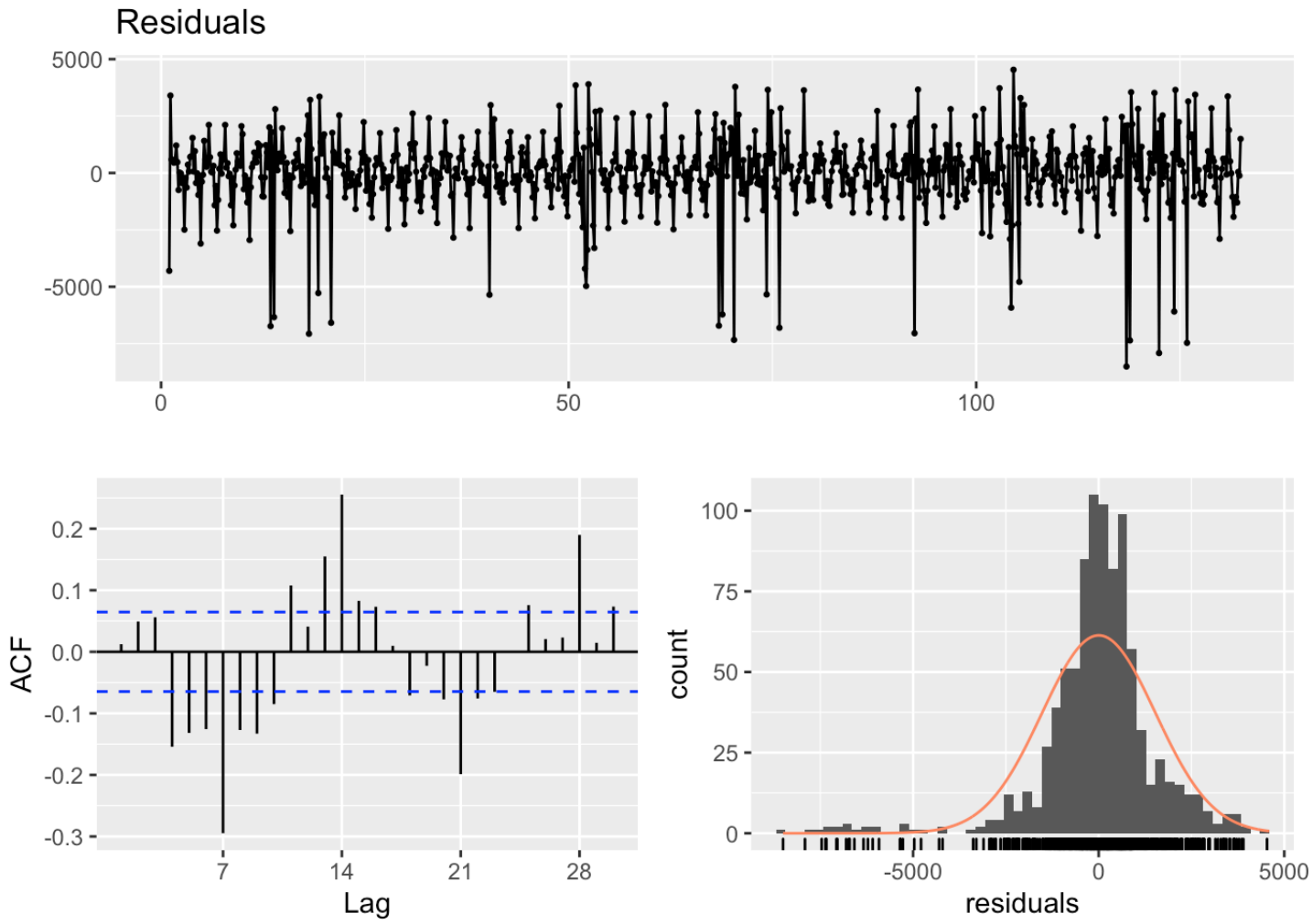
```
## Holt-Winters' additive method
##
## Call:
##  hw(y = train_w[, 2], h = 21, seasonal = "additive", damped = FALSE)
##
##   Smoothing parameters:
##     alpha = 0.4623
##     beta  = 0.002
##     gamma = 2e-04
##
##   Initial states:
##     l = 3245.8519
##     b = 132.6898
##     s = 2117.873 -4608.712 -1603.409 360.8331 1356.795 805.153
##           1571.467
##
##   sigma:  1539.965
##
##      AIC     AICc      BIC
## 19818.57 19818.91 19876.47
```

```
plot(hw_model_no_damped)
```



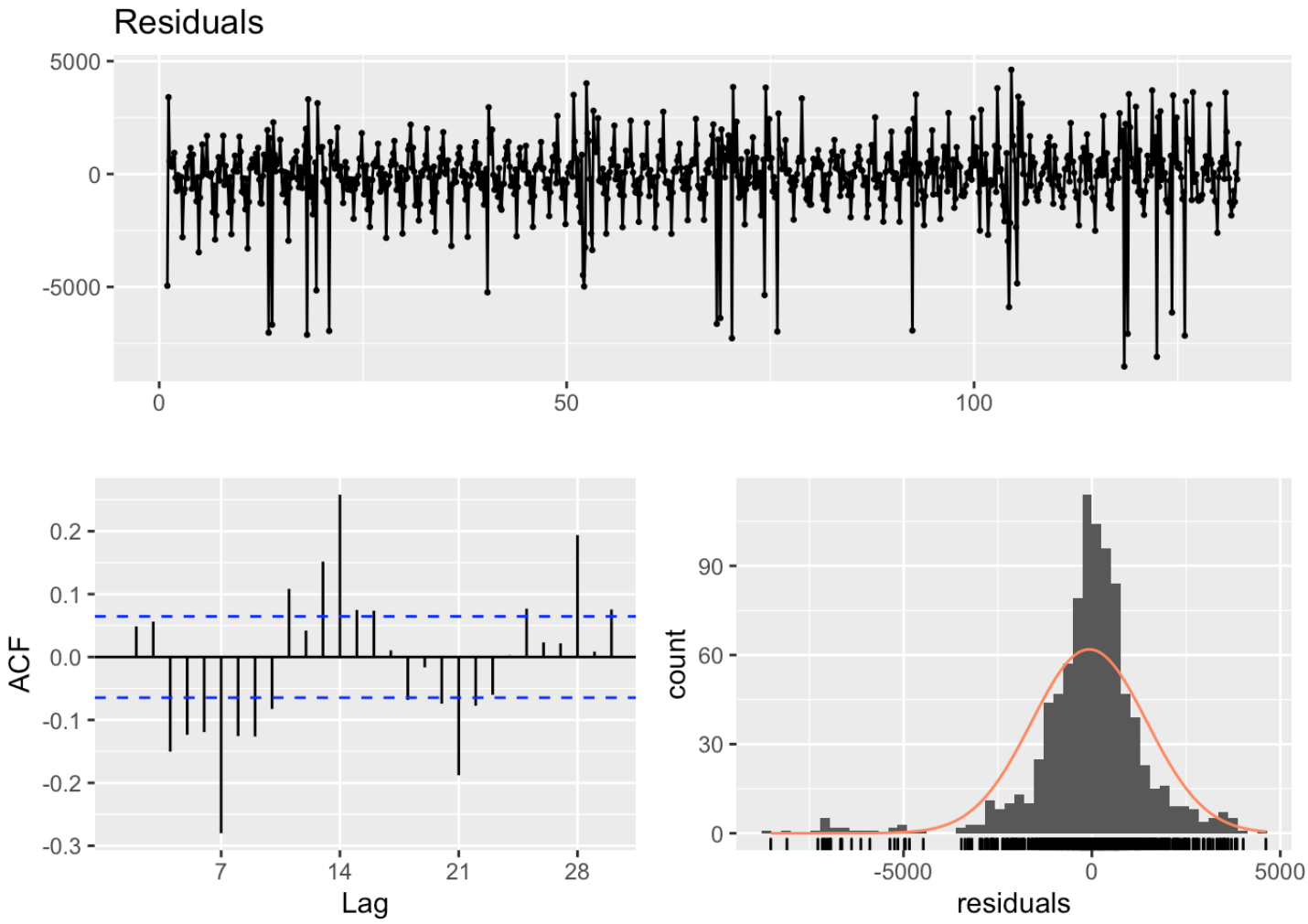

```
checkresiduals(hw_model$residuals)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```




```
checkresiduals(hw_model_no_damped$residuals)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```




```
hw_acc_no_damped<- accuracy(hw_model_no_damped$mean, test_w[,2])
accuracy(hw_model_no_damped)
```

```
##                       ME     RMSE      MAE MPE MAPE      MASE        ACF1
## Training set -70.02558 1530.741 988.9754 NaN  Inf 0.5518165 0.001825674
```

From the residual, we still coud see some lags have higher autocorrelation. The residual are not pure white noise.

and from the Aicc and accurancy, the damped hw model slightly better than no damped ones.

For the smoothing parameter, ϕ = 0.9018 which approximatedly equal to 1 .This larger ϕ means the trend line is changing more rapidly. The smaller gamma γ=0.012 means the seasonal component hardly changed over time. and the smaller beta which is e-04 indicates that the slope component hardly changes over time.

TBATS

TBATS for long time series data. Allow for annual seasonality as well as weekly seasonality. In that case, a multiple seasonal model such as TBATS is useful. For our daily data, we think it might have a weekly pattern as well as an annual patter. More people come to store on Weekends.

```
#data$Sale%>% mstl() %>% autoplot() + xlab ('daily') # mstl is for decomposition
```

transoform the data using multi

942 data points. leave 21 for forecasting. so 921 for training

```
train_tbats <-store_8[1:920,] # split data for training/testing
test_tbats<-store_8[921:941,]

m_tbats<-tbats(train_tbats[,2],seasonal.period = c(7,365.25))
m_tbats$AIC
```

```
## [1] 19554.96
```

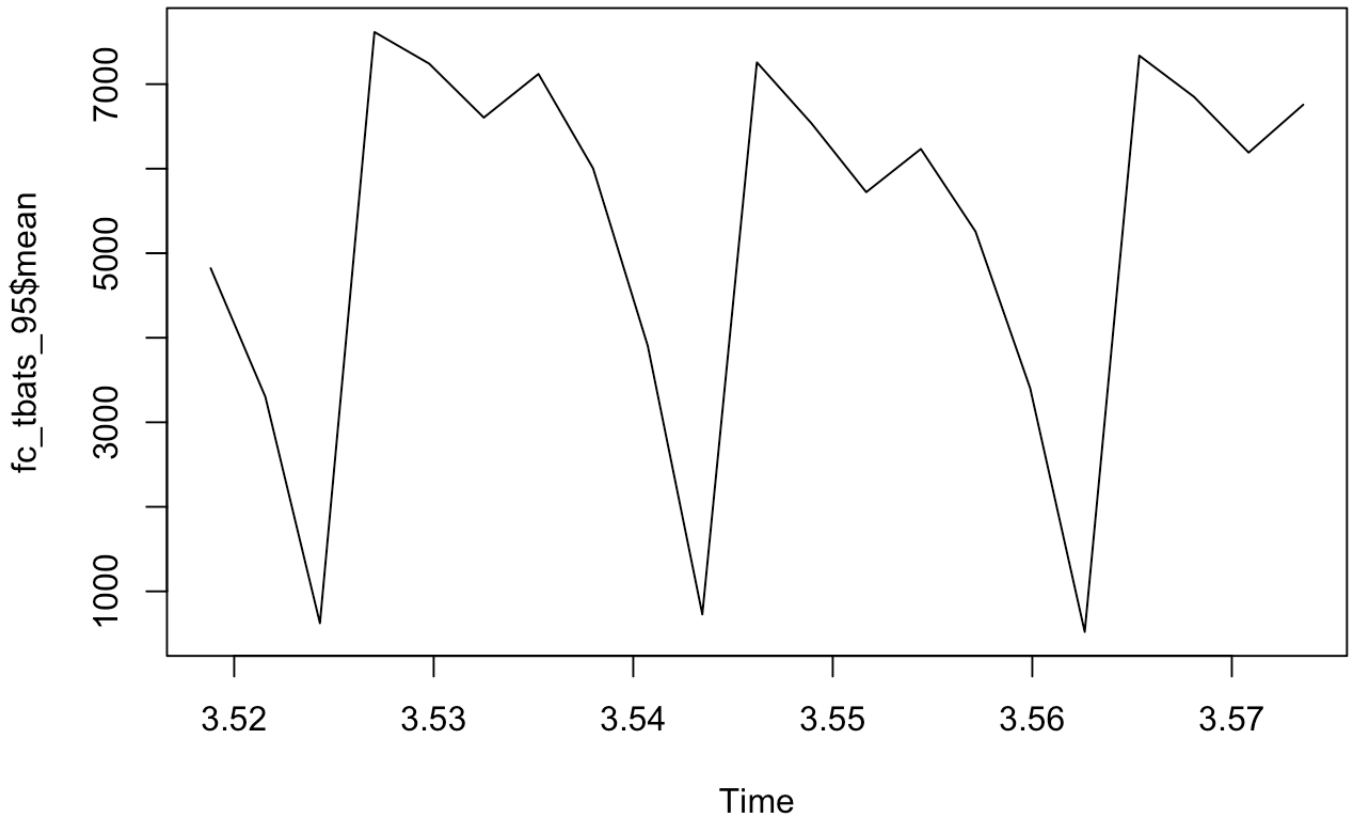
```
Arima_auto$aic
```

```
## [1] 5805.132
```

```
fc_tbats_80 <- forecast(m_tbats, h = 21, level =80)
fc_tbats_95 <- forecast(m_tbats, h = 21, level =95)
fc_tbats_80$mean
```

```
## Multi-Seasonal Time Series:
## Start: 3.518823
## Seasonal Periods: 7 365.25
## Data:
##  [1] 4822.0451 3302.3498  623.1985 7615.8396 7242.2371 6604.2279 7120.5428
##  [8] 6003.8507 3906.1418  727.9862 7257.1136 6534.6473 5722.2077 6233.2348
## [15] 5257.3242 3404.6282  520.2749 7337.0669 6850.1880 6190.0697 6756.9580
```

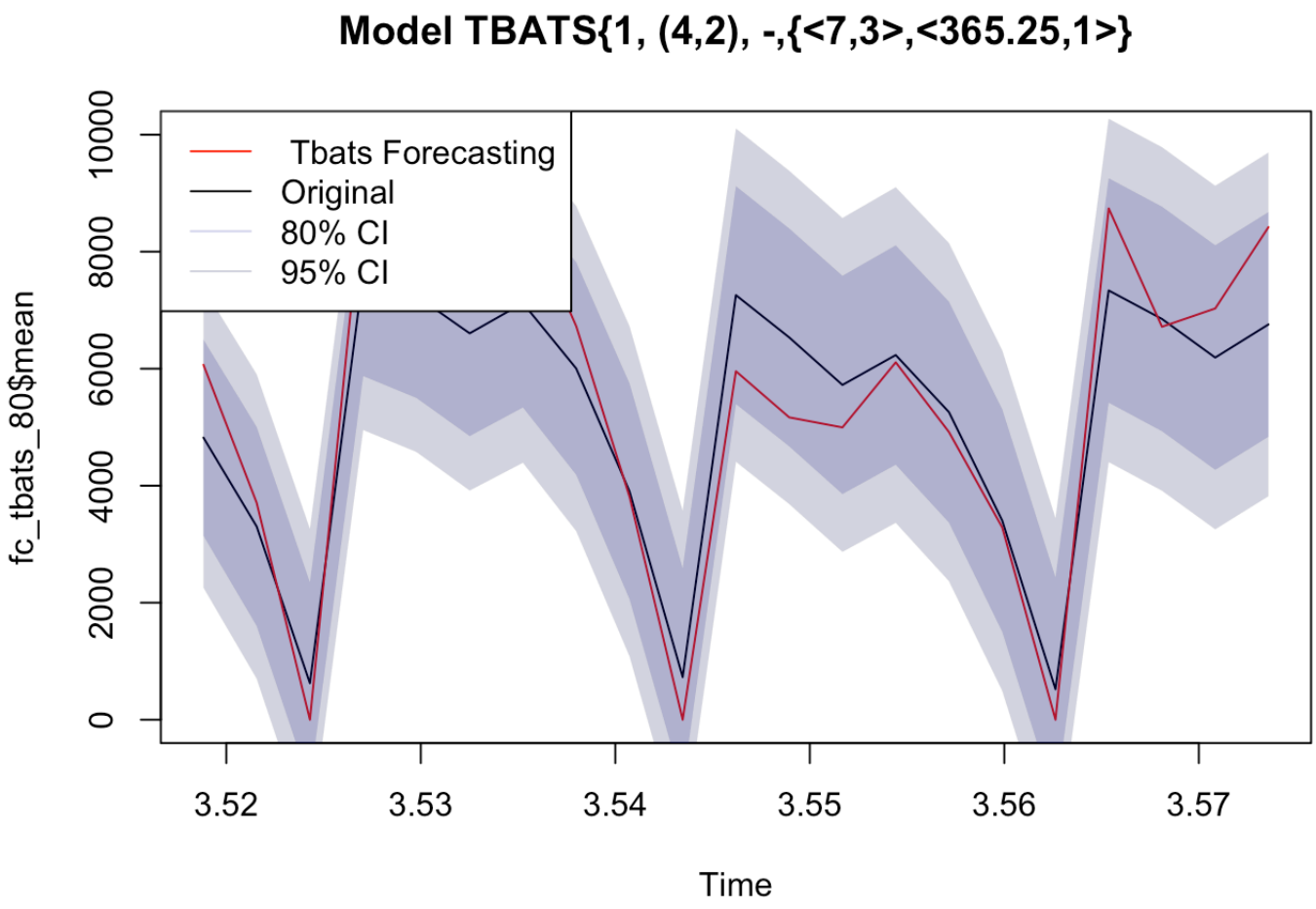
```
plot(fc_tbats_95$mean)
```




```
summary(m_tbats)
```

```
##                   Length Class  Mode
## lambda                 0  -none- NULL
## alpha                  1  -none- numeric
## beta                   0  -none- NULL
## damping.parameter      0  -none- NULL
## gamma.one.values       2  -none- numeric
## gamma.two.values       2  -none- numeric
## ar.coefficients        4  -none- numeric
## ma.coefficients        2  -none- numeric
## likelihood             1  -none- numeric
## optim.return.code      1  -none- numeric
## variance               1  -none- numeric
## AIC                    1  -none- numeric
## parameters             2  -none- list
## seed.states           25  -none- numeric
## fitted.values        920  -none- numeric
## errors               920  -none- numeric
## x                  23000  -none- numeric
## seasonal.periods       2  -none- numeric
## k.vector               2  -none- numeric
## y                    920  -none- numeric
## p                      1  -none- numeric
## q                      1  -none- numeric
## call                   3  -none- call
## series                 1  -none- character
## method                 1  -none- character
```

```
plot(fc_tbats_80$mean, col = 1, main ='Model TBATS{1, (4,2), -,{<7,3>,<365.25,1>}', y
lim = c(0,10000))
points(time(fc_tbats_80$mean),test_tbats[,2], col=2, type ='l')
polygon(c(time(fc_tbats_80$mean),rev(time(fc_tbats_80$mean))), c(fc_tbats_80$upper,re
v(fc_tbats_80$lower)),  # create %80 interval
   col=rgb(0,0,0.6,0.2), border=FALSE)
polygon(c(time(fc_tbats_80$mean),rev(time(fc_tbats_80$mean))), c(fc_tbats_95$upper,re
v(fc_tbats_95$lower)),  # create %95 interval
   col=rgb(0,0,0.3,0.2), border=FALSE)
legend('topleft', legend = c(' Tbats Forecasting', 'Original', '80% CI', '95% CI') ,
col=c(2, 1,rgb(0,0,0.6,0.2),rgb(0,0,0.3,0.2)), lty = 1)
```



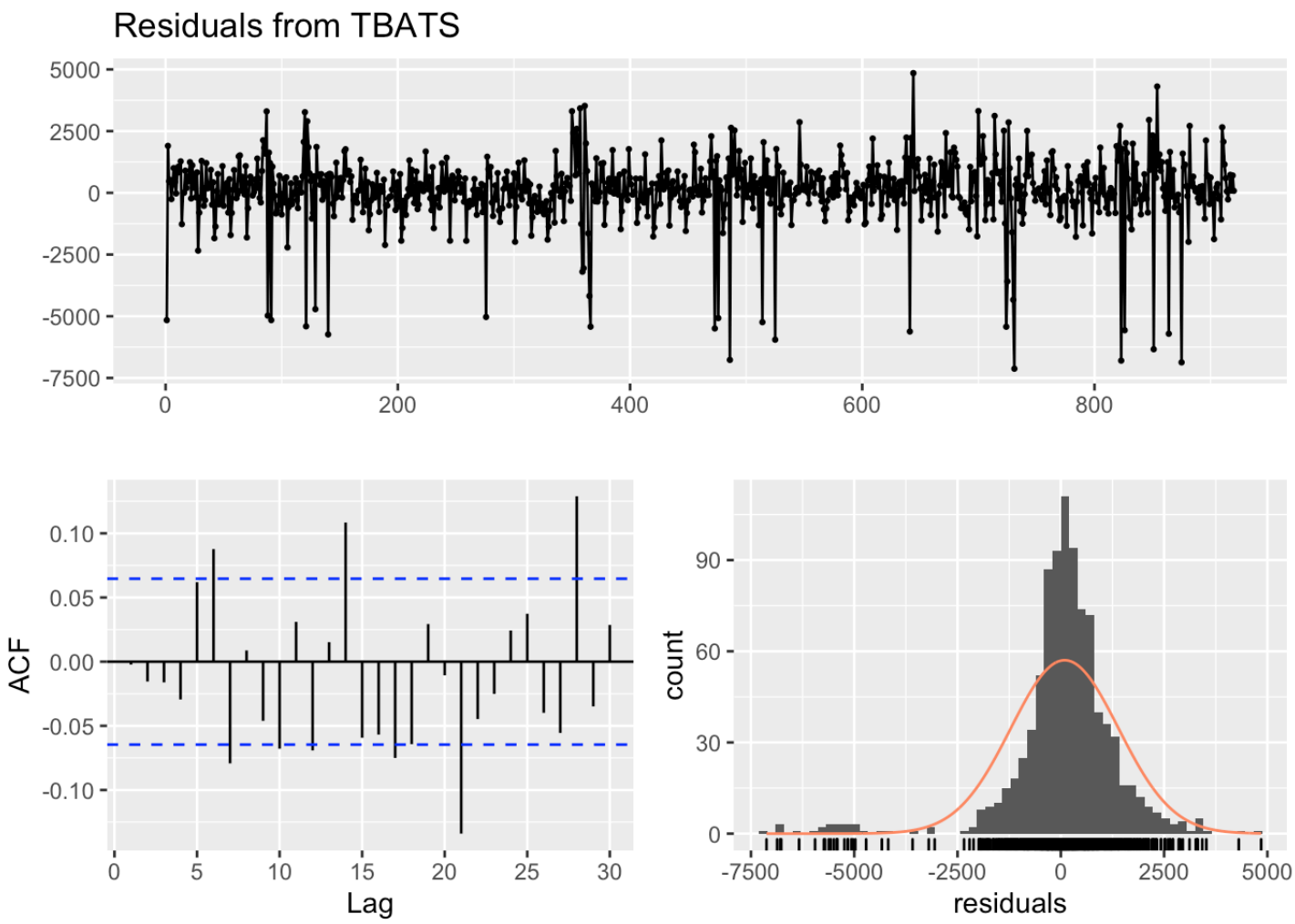

```
accuracy(m_tbats)
```

```
##                     ME     RMSE      MAE MPE MAPE      MASE         ACF1
## Training set 92.77539 1308.165 826.233 NaN  Inf 0.3435096 -0.002250673
```

```
accuracy(fc_tbats_80$mean,test_tbats[,2] )
```

```
##                ME     RMSE      MAE  MPE MAPE
## Test set 301.1365 1036.342 879.5123 -Inf  Inf
```

```
checkresiduals(m_tbats)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from TBATS
## Q* = 119.08, df = 3, p-value < 2.2e-16
##
## Model df: 36.   Total lags used: 39
```

Model TBATS{1, (4,2), -,{<7,3>,<365.25,1>} AIC: 19554.96 auto arima AIC 5805.132

Model shows 1 as box-cox transformation which means no box-cox transofrmation. {4,2} shows ARIMA error, - no damping parapeter, seasonal period = 7, fourier terms = 3, seasonal period = 365.25, terms = 1 p value is small, reject null, residual aren't independent from each other.

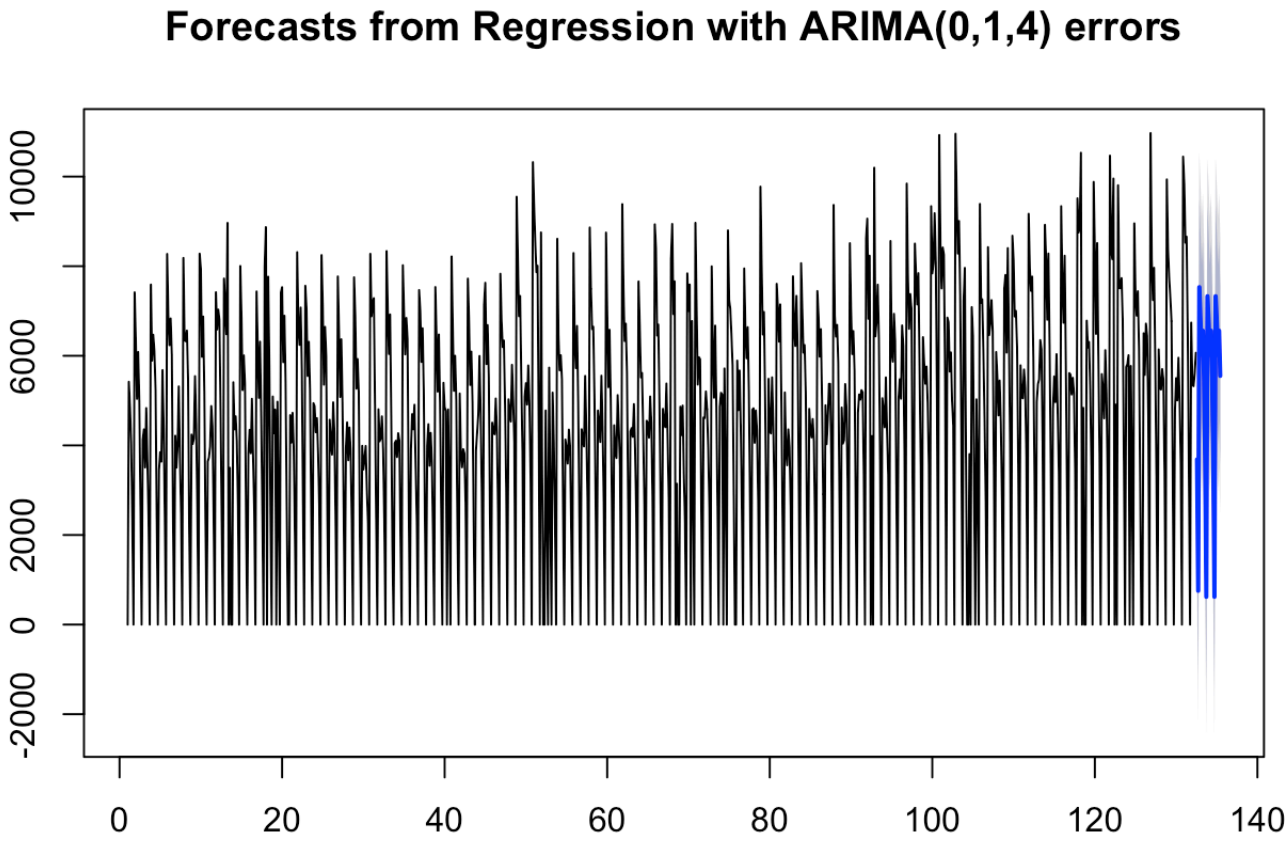
Fourier Term

use fourier terms and the arima model by minimizing the AICc

```
library(forecast)
bestfit <-list(aicc = Inf)
for (i in 1:3)
{
  fit = auto.arima(train_w[,2], xreg = fourier(train_w[,2], K = i), seasonal = FALSE)
  if (fit$aicc < bestfit$aicc)
    bestfit <-fit
    else break;
}

bestfit
```

```
## Series: train_w[, 2]
## Regression with ARIMA(0,1,4) errors
##
## Coefficients:
##           ma1      ma2      ma3      ma4       S1-7       C1-7       S2-7
##       -0.7048  -0.0153  -0.0516  -0.2107  2167.9863  -322.9430  1183.5617
## s.e.   0.0328   0.0387   0.0400   0.0293    70.5966    70.5753    53.6025
##           C2-7      S3-7       C3-7
##       1302.4426  500.4847  1142.1039
## s.e.    53.6464   56.3053    56.3941
##
## sigma^2 estimated as 1968557:  log likelihood=-7968.63
## AIC=15959.25   AICc=15959.55   BIC=16012.32
```

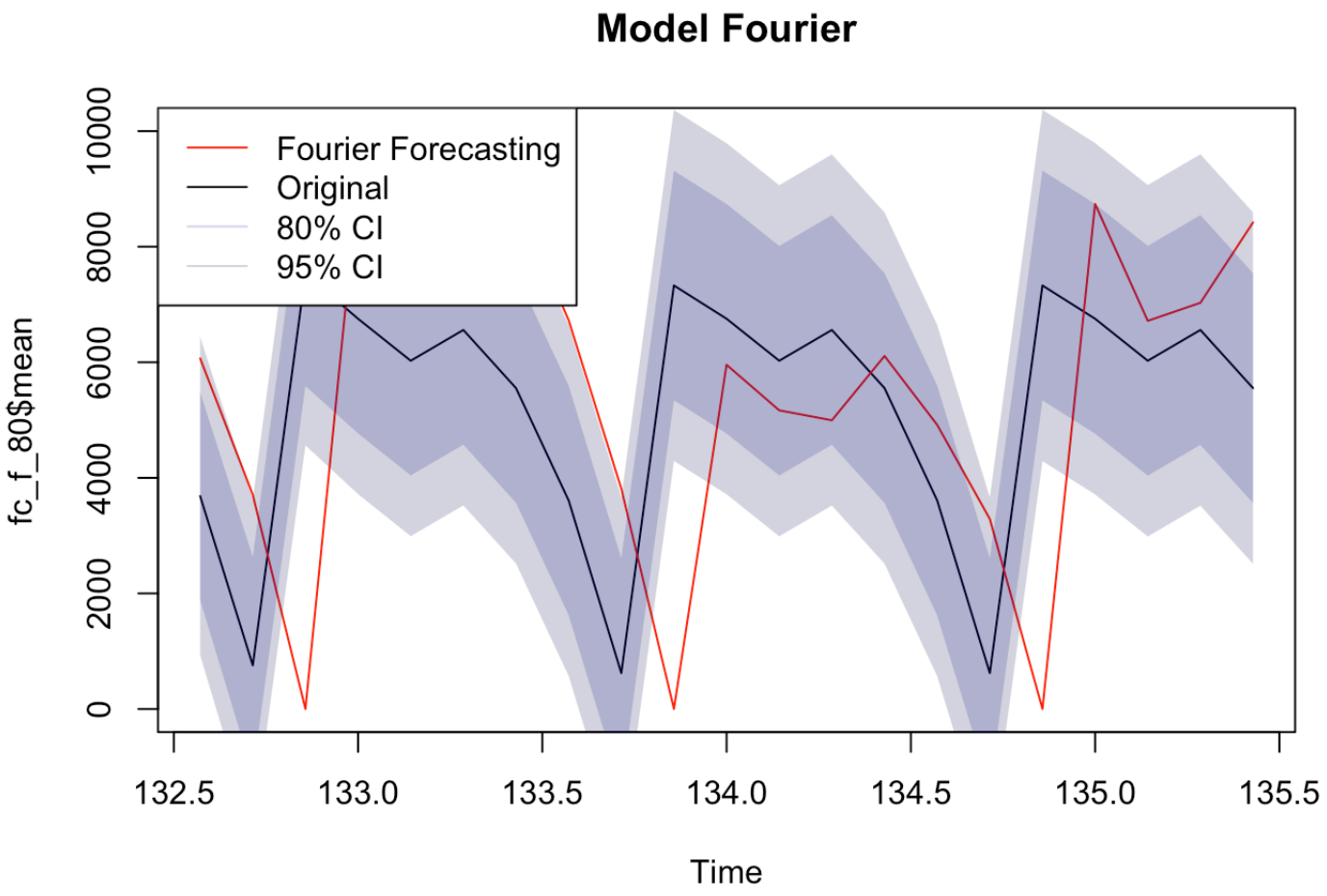
```
fc_f <-forecast(bestfit, xreg = fourier(train_w[,2], K = 3, h =21))
plot(fc_f)
```



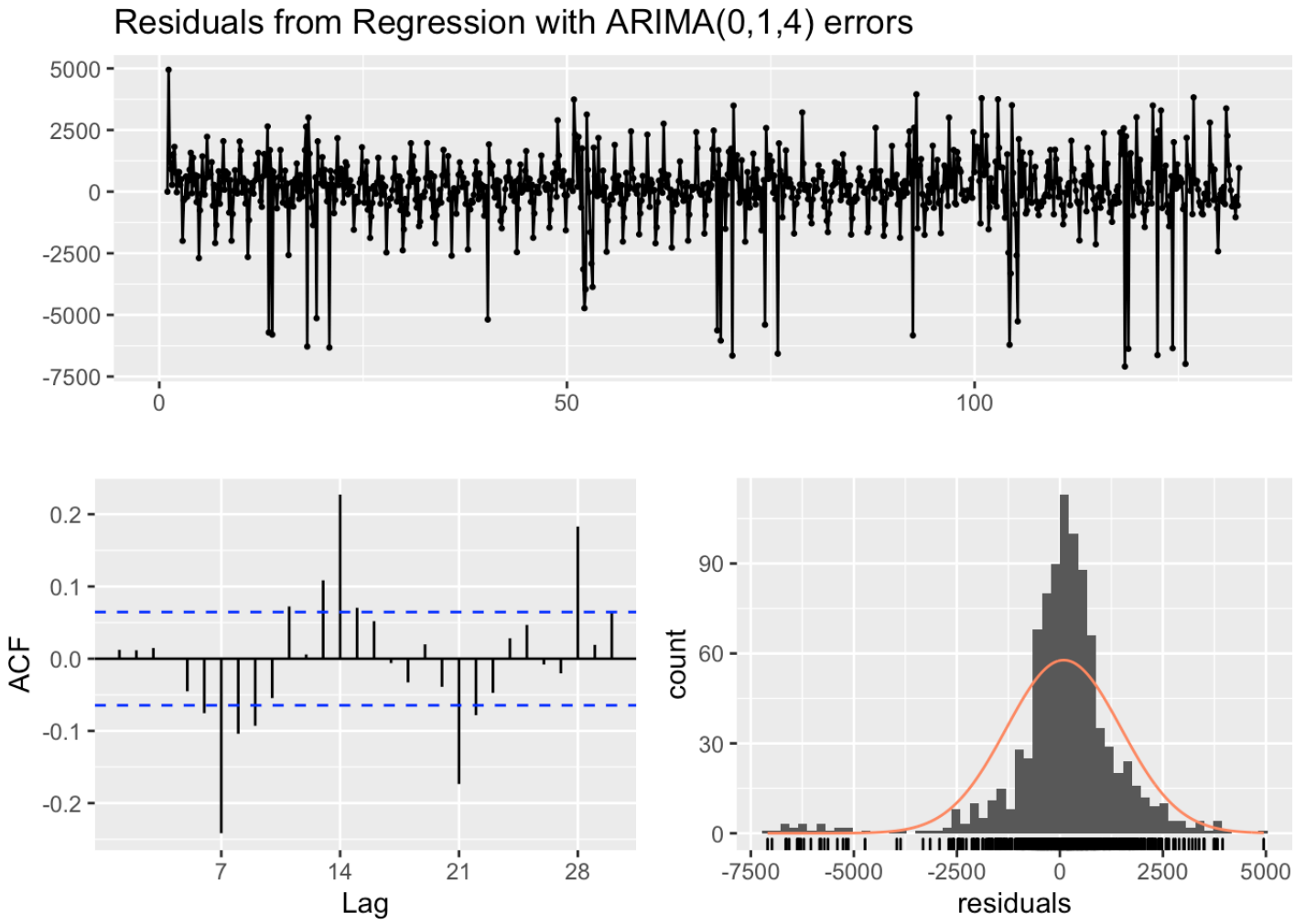

```
fc_f_80 <- forecast(bestfit, xreg = fourier(train_w[,2], K = 3, h =21), level =80)

fc_f_95 <- forecast(bestfit, xreg = fourier(train_w[,2], K = 3,  h = 21), level =95)
fc_f_80$mean
```

```
## Time Series:
## Start = c(132, 5)
## End = c(135, 4)
## Frequency = 7
##  [1] 3684.8473  755.0181 7530.6542 6752.8574 6026.2896 6559.1145 5552.6250
##  [8] 3608.6745  620.7785 7328.5940 6752.8574 6026.2896 6559.1145 5552.6250
## [15] 3608.6745  620.7785 7328.5940 6752.8574 6026.2896 6559.1145 5552.6250
```

```
plot(fc_f_80$mean, col = 1, main ='Model Fourier', ylim = c(0,10000))
points(time(fc_f_80$mean),test_tbats[,2], col=2, type ='l')
polygon(c(time(fc_f_80$mean),rev(time(fc_f_80$mean))), c(fc_f_80$upper,rev(fc_f_80$lo
wer)),  # create %80 interval
   col=rgb(0,0,0.6,0.2), border=FALSE)
polygon(c(time(fc_f_80$mean),rev(time(fc_f_80$mean))), c(fc_f_95$upper,rev(fc_f_95$lo
wer)),  # create %95 interval
   col=rgb(0,0,0.3,0.2), border=FALSE)
legend('topleft', legend = c('Fourier Forecasting', 'Original', '80% CI', '95% CI') ,
col=c(2, 1,rgb(0,0,0.6,0.2),rgb(0,0,0.3,0.2)), lty = 1)
```




```
checkresiduals(bestfit)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,4) errors
## Q* = 147.27, df = 4, p-value < 2.2e-16
##
## Model df: 10.   Total lags used: 14
```

```
accuracy(fc_f)
```

```
##                     ME     RMSE      MAE MPE MAPE      MASE       ACF1
## Training set 93.44877 1394.649 888.6706 NaN  Inf 0.4958497 0.0122485
```

```
accuracy(fc_f, test_w[,2])
```

```
##                     ME     RMSE       MAE  MPE MAPE      MASE      ACF1
## Training set  93.44877 1394.649  888.6706  NaN  Inf 0.4958497 0.0122485
## Test set     429.60605 1373.435 1136.9001 -Inf  Inf 0.6343538 0.6305069
##              Theil's U
## Training set        NA
## Test set             0
```

```
bestfit
```

```
## Series: train_w[, 2]
## Regression with ARIMA(0,1,4) errors
##
## Coefficients:
##           ma1      ma2      ma3      ma4       S1-7       C1-7       S2-7
##       -0.7048  -0.0153  -0.0516  -0.2107  2167.9863  -322.9430  1183.5617
## s.e.   0.0328   0.0387   0.0400   0.0293    70.5966    70.5753    53.6025
##            C2-7      S3-7       C3-7
##       1302.4426  500.4847  1142.1039
## s.e.    53.6464   56.3053    56.3941
##
## sigma^2 estimated as 1968557:  log likelihood=-7968.63
## AIC=15959.25   AICc=15959.55   BIC=16012.32
```

The best model have 3 pairs of Fourier terms with nt process is an Arima (0,1,4). Total number of degrees of freedom is 10 (3*2 -seasonality, 4 ARMA)