Model training

Mark Wang & Paola Aleman

9/30/2019

# Instructions

For the following code to run smoothly it is necessary to have the folders that the line of code calls for.  
The main folder is called Forecast, which holds all R-markdowns. The subfolders for the Forecast folder are: Code, Data, Output. The subfolder Code holds scripts.  
The subfolder Data holds two more subfolders: Countries, Raw Data. The Countries subfolder [this is an optional subfolder] includes folders for each country to which we individually want to save data on. Raw Data folder only has the raw data on it.  
The subfolder Output includes a subfolder called Countries and this folder has folders for each country. Each country then has 2 additional folders: Plots, Tables.  
The Plots folder saves all graphs created in the script.  
The Tables folder saves all excel files with the performance indices.  
Note the models are label by c 1 k. which stands for c -> colombia, 1 -> club 6101 and k -> kitchen trash bags.  
It is necessary to change c 1 k whenever you’re analyzing a different set of item and club. This way graphs and models are saved into folders with their specific name.  
For easier substitution ctrl+f. In the first box type the name you want to substitute, such as c 1 k. and on the second box include the prefix you want to substitute with, suck as c 2 k. Then click the last All option.

# Setup

# Installing packages

# Define functions:

Three functions are defined in this section. firstly, performance\_index\_dtds function calculates the performance of prediction models on de-trended and de-seasonalized quantity data. Secondly, performance\_index\_raw function calculates the performance of prediction models on raw data. Thirdly, feature\_engineering function performs feature engineering to prepare for machine learning data.

# This scripts stores performance\_index function  
  
performance\_index\_dtds<-function(df\_Actual, pred\_name, pred\_value){  
 assign(pred\_name, exp(pred\_value+(retrend\_log-trend\_log[1])+reseasonal\_log))  
 df\_Actual[,colnames(df\_Actual)==pred\_name]<-NULL  
 df\_Actual<-cbind(df\_Actual, get(pred\_name)%>%as.numeric())  
 colnames(df\_Actual)[colnames(df\_Actual)=="get(pred\_name) %>% as.numeric()"]<-pred\_name  
 assign(paste0("MEAN\_", pred\_name, "\_total"),   
 mean(get(pred\_name))/df\_Actual%>%select("Actual")%>%data.matrix()%>%mean()-1,  
 envir = .GlobalEnv  
 )  
 assign(paste0("MEAN\_", pred\_name, "\_target"),  
 mean((get(pred\_name))[(len\_inv+1):(len\_inv+len\_dec)])/mean((df\_Actual%>%select("Actual")%>%data.matrix())[(len\_inv+1):(len\_inv+len\_dec)])-1,  
 envir=.GlobalEnv  
 )  
   
 assign(paste0("RMSE\_", pred\_name, "\_total" ),  
 RMSE(get(pred\_name),df\_Actual%>%select("Actual")%>%data.matrix()),  
 envir = .GlobalEnv)  
 assign(paste0("RMSE\_", pred\_name, "\_target" ),  
 RMSE((get(pred\_name)[(len\_inv+1):(len\_inv+len\_dec)]),(df\_Actual%>%select("Actual")%>%data.matrix())[(len\_inv+1):(len\_inv+len\_dec)]),  
 envir = .GlobalEnv)  
   
 assign(paste0("RMSE\_", pred\_name, "\_total" ),  
 RMSE(get(pred\_name),df\_Actual%>%select("Actual")%>%data.matrix()),  
 envir = .GlobalEnv)  
 assign(paste0("RMSE\_", pred\_name, "\_target" ),  
 RMSE((get(pred\_name)[(len\_inv+1):(len\_inv+len\_dec)]),(df\_Actual%>%select("Actual")%>%data.matrix())[(len\_inv+1):(len\_inv+len\_dec)]),  
 envir = .GlobalEnv)  
   
 assign(paste0("MAE\_", pred\_name, "\_total" ),  
 MAE(get(pred\_name),df\_Actual%>%select("Actual")%>%data.matrix()),  
 envir = .GlobalEnv)  
 assign(paste0("MAE\_", pred\_name, "\_target" ),  
 MAE((get(pred\_name)[(len\_inv+1):(len\_inv+len\_dec)]),(df\_Actual%>%select("Actual")%>%data.matrix())[(len\_inv+1):(len\_inv+len\_dec)]),  
 envir = .GlobalEnv)  
   
 assign(paste0("MPE\_", pred\_name, "\_total" ),  
 MPE(get(pred\_name),df\_Actual%>%select("Actual")%>%data.matrix()),  
 envir = .GlobalEnv)  
 assign(paste0("MPE\_", pred\_name, "\_target" ),  
 MPE((get(pred\_name)[(len\_inv+1):(len\_inv+len\_dec)]),(df\_Actual%>%select("Actual")%>%data.matrix())[(len\_inv+1):(len\_inv+len\_dec)]),  
 envir = .GlobalEnv)  
   
 assign(paste0("MAPE\_", pred\_name, "\_total" ),  
 MAPE(get(pred\_name),df\_Actual%>%select("Actual")%>%data.matrix()),  
 envir = .GlobalEnv)  
 assign(paste0("MAPE\_", pred\_name, "\_target" ),  
 MAPE((get(pred\_name)[(len\_inv+1):(len\_inv+len\_dec)]),(df\_Actual%>%select("Actual")%>%data.matrix())[(len\_inv+1):(len\_inv+len\_dec)]),  
 envir = .GlobalEnv)  
   
 assign(paste0("MASE\_", pred\_name, "\_total" ),  
 mase(get(pred\_name),df\_Actual%>%select("Actual")%>%data.matrix()),  
 envir = .GlobalEnv)  
 assign(paste0("MASE\_", pred\_name, "\_target" ),  
 mase((get(pred\_name)[(len\_inv+1):(len\_inv+len\_dec)]),(df\_Actual%>%select("Actual")%>%data.matrix())[(len\_inv+1):(len\_inv+len\_dec)]),  
 envir = .GlobalEnv)  
 return(df\_Actual)  
}  
  
performance\_index\_raw<-function(df\_Actual, pred\_name, pred\_value){  
 assign(pred\_name, exp(pred\_value))  
 df\_Actual[,colnames(df\_Actual)==pred\_name]<-NULL  
 df\_Actual<-cbind(df\_Actual, get(pred\_name)%>%as.numeric())  
 colnames(df\_Actual)[colnames(df\_Actual)=="get(pred\_name) %>% as.numeric()"]<-pred\_name  
 assign(paste0("MEAN\_", pred\_name, "\_total"),   
 mean(get(pred\_name))/df\_Actual%>%select("Actual")%>%data.matrix()%>%mean()-1,  
 envir = .GlobalEnv  
 )  
 assign(paste0("MEAN\_", pred\_name, "\_target"),  
 mean((get(pred\_name))[(len\_inv+1):(len\_inv+len\_dec)])/mean((df\_Actual%>%select("Actual")%>%data.matrix())[(len\_inv+1):(len\_inv+len\_dec)])-1,  
 envir=.GlobalEnv  
 )  
   
 assign(paste0("RMSE\_", pred\_name, "\_total" ),  
 RMSE(get(pred\_name),df\_Actual%>%select("Actual")%>%data.matrix()),  
 envir = .GlobalEnv)  
 assign(paste0("RMSE\_", pred\_name, "\_target" ),  
 RMSE((get(pred\_name)[(len\_inv+1):(len\_inv+len\_dec)]),(df\_Actual%>%select("Actual")%>%data.matrix())[(len\_inv+1):(len\_inv+len\_dec)]),  
 envir = .GlobalEnv)  
   
 assign(paste0("MAE\_", pred\_name, "\_total" ),  
 MAE(get(pred\_name),df\_Actual%>%select("Actual")%>%data.matrix()),  
 envir = .GlobalEnv)  
 assign(paste0("MAE\_", pred\_name, "\_target" ),  
 MAE((get(pred\_name)[(len\_inv+1):(len\_inv+len\_dec)]),(df\_Actual%>%select("Actual")%>%data.matrix())[(len\_inv+1):(len\_inv+len\_dec)]),  
 envir = .GlobalEnv)  
   
 assign(paste0("MPE\_", pred\_name, "\_total" ),  
 MPE(get(pred\_name),df\_Actual%>%select("Actual")%>%data.matrix()),  
 envir = .GlobalEnv)  
 assign(paste0("MPE\_", pred\_name, "\_target" ),  
 MPE((get(pred\_name)[(len\_inv+1):(len\_inv+len\_dec)]),(df\_Actual%>%select("Actual")%>%data.matrix())[(len\_inv+1):(len\_inv+len\_dec)]),  
 envir = .GlobalEnv)  
   
 assign(paste0("MAPE\_", pred\_name, "\_total" ),  
 MAPE(get(pred\_name),df\_Actual%>%select("Actual")%>%data.matrix()),  
 envir = .GlobalEnv)  
 assign(paste0("MAPE\_", pred\_name, "\_target" ),  
 MAPE((get(pred\_name)[(len\_inv+1):(len\_inv+len\_dec)]),(df\_Actual%>%select("Actual")%>%data.matrix())[(len\_inv+1):(len\_inv+len\_dec)]),  
 envir = .GlobalEnv)  
   
 assign(paste0("MASE\_", pred\_name, "\_total" ),  
 mase(get(pred\_name),df\_Actual%>%select("Actual")%>%data.matrix()),  
 envir = .GlobalEnv)  
 assign(paste0("MASE\_", pred\_name, "\_target" ),  
 mase((get(pred\_name)[(len\_inv+1):(len\_inv+len\_dec)]),(df\_Actual%>%select("Actual")%>%data.matrix())[(len\_inv+1):(len\_inv+len\_dec)]),  
 envir = .GlobalEnv)  
 return(df\_Actual)  
}  
feature\_engineering<-function(t\_v, feature\_list){  
 function\_names<-c("avg" , "diff")  
 function\_list<-list(function(x) roll\_meanr(x, n= 4), function(x) c(diff(x), NA))  
 weeks<-c(1, 4, 5, 8, 9, 12, 13, 17, 18, 22, 26, 52)  
 for (feature in feature\_list){   
 for (lag\_number in (weeks[weeks>=len\_dec+len\_inv])){  
 assign(paste0("lag", "\_", lag\_number, "\_", feature), lag((t\_v%>%select(feature))[[1]], lag\_number))  
 t\_v<-cbind(t\_v, get(paste0("lag", "\_", lag\_number, "\_", feature))%>%as.numeric())  
 colnames(t\_v)[colnames(t\_v)=="get(paste0(\"lag\", \"\_\", lag\_number, \"\_\", feature)) %>% as.numeric()"]<-paste0("lag", "\_", lag\_number, "\_", feature)  
 for (i in (1:length(function\_names))){  
 assign(paste0(function\_names[i], "\_", lag\_number, "\_", feature), lag(function\_list[[i]]((t\_v%>%select(feature))[[1]]), lag\_number))  
 t\_v<-cbind(t\_v, get(paste0(function\_names[i], "\_", lag\_number, "\_", feature))%>%as.numeric())  
 colnames(t\_v)[colnames(t\_v)=="get(paste0(function\_names[i], \"\_\", lag\_number, \"\_\", feature)) %>% "]<-paste0(function\_names[i], "\_", lag\_number, "\_", feature)  
 }  
 }  
 }  
 return(t\_v)  
}

# Importing Training Dataset

load("Data/Raw Data/trainingSet.RData")  
  
#Changing the format of variables  
##Changing "TransactionDate" from integer to date   
train$TransactionDate<- as.Date(train$TransactionDate%>%as.character(), "%Y%m%d")  
train$TransactionDate<- ymd(train$TransactionDate)  
  
##Changing "Club Number" from integer to Factor   
train$ClubNumber<- as.factor(train$ClubNumber)  
  
##Changing "item" from integer to Factor   
train$item<- as.factor(train$item)  
  
#Subsetting to country  
country\_set<-subset(train, Country == country)

## Dealing with Variables

##Changing Month into Factor  
country\_set$month<-as.factor(strftime(country\_set$TransactionDate, "%B"))  
#ClubNumber into Factor  
country\_set$ClubNumber<- as.factor(country\_set$ClubNumber)  
##Day of week  
country\_set$Day\_of\_week <- as.factor(strftime(country\_set$TransactionDate, "%A"))  
#Create Dummy Variables for month and weekday  
country\_set<- cbind(country\_set, fastDummies::dummy\_cols(country\_set$month))  
country\_set<- cbind(country\_set, fastDummies::dummy\_cols(country\_set$Day\_of\_week))  
  
#Take out unnecessary variables  
country\_set$.data<- NULL  
country\_set$.data<- NULL  
country\_set$Year<- NULL  
country\_set$.data\_February<- NULL  
country\_set$.data\_Monday <- NULL  
  
#Rename Month Dummy Variables   
colnames(country\_set)[colnames(country\_set)==".data\_January"] <- "Jan"  
colnames(country\_set)[colnames(country\_set)==".data\_March"] <- "Mar"  
colnames(country\_set)[colnames(country\_set)==".data\_April"] <- "Apr"  
colnames(country\_set)[colnames(country\_set)==".data\_May"] <- "May"  
colnames(country\_set)[colnames(country\_set)==".data\_June"] <- "June"  
colnames(country\_set)[colnames(country\_set)==".data\_July"] <- "July"  
colnames(country\_set)[colnames(country\_set)==".data\_August"] <- "Aug"  
colnames(country\_set)[colnames(country\_set)==".data\_September"] <- "Sep"  
colnames(country\_set)[colnames(country\_set)==".data\_October"] <- "Oct"  
colnames(country\_set)[colnames(country\_set)==".data\_November"] <- "Nov"  
colnames(country\_set)[colnames(country\_set)==".data\_December"] <- "Dec"  
  
#Rename Weekday dummy variables  
colnames(country\_set)[colnames(country\_set)==".data\_Tuesday"] <- "Tu"  
colnames(country\_set)[colnames(country\_set)==".data\_Thursday"] <- "Th"  
colnames(country\_set)[colnames(country\_set)==".data\_Wednesday"] <- "Wed"  
colnames(country\_set)[colnames(country\_set)==".data\_Friday"] <- "Fri"  
colnames(country\_set)[colnames(country\_set)==".data\_Saturday"] <- "Sat"  
colnames(country\_set)[colnames(country\_set)==".data\_Sunday"] <- "Sun"  
  
#Subsetting  
c1k<-country\_set[country\_set$Description==item & country\_set$ClubNumber==club,]  
  
c1k<-c1k[order(c1k$TransactionDate),]  
  
c1k$TransactionDate<- as.Date(c1k$TransactionDate)

# Summary statistics

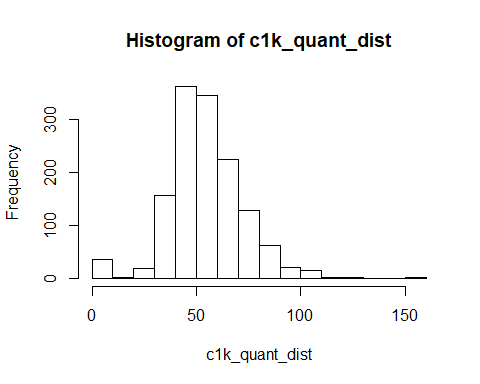
#Summary statistics  
  
# Time span  
## Beginning and Ending of dataset  
summary(c1k$TransactionDate)

## Min. 1st Qu. Median Mean 3rd Qu.   
## "2015-01-02" "2015-12-07" "2016-11-11" "2016-11-16" "2017-10-31"   
## Max.   
## "2018-10-05"

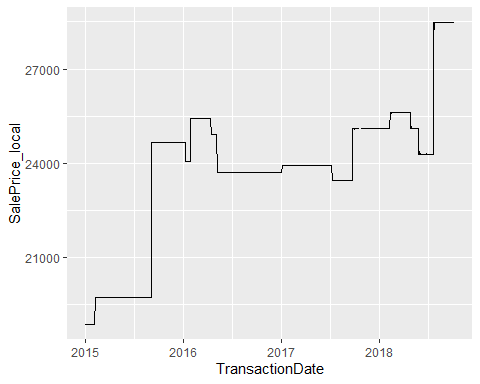
## Gaps  
c(min(c1k$TransactionDate):max(c1k$TransactionDate))[c(min(c1k$TransactionDate):max(c1k$TransactionDate))%in%c1k$TransactionDate==FALSE]%>%as.Date(origin="1970-1-1")

## [1] "2015-01-06" "2015-04-03" "2015-09-08" "2015-09-16" "2015-12-25"  
## [6] "2016-01-01" "2016-03-25" "2016-09-06" "2016-12-25" "2017-01-01"  
## [11] "2017-01-29" "2017-01-30" "2017-01-31" "2017-02-01" "2017-02-02"  
## [16] "2017-04-14" "2017-06-18" "2017-06-19" "2017-06-20" "2017-06-21"  
## [21] "2017-06-22" "2017-06-23" "2017-10-20" "2017-10-21" "2017-10-22"  
## [26] "2017-10-23" "2017-10-24" "2017-12-25" "2018-01-01" "2018-03-30"

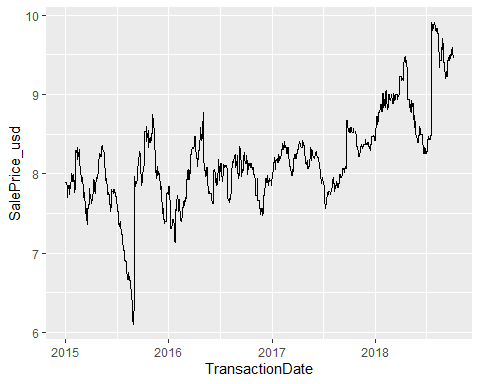
c1k\_quant\_dist<-c(c1k$quantity, rep(0, sum(c(min(c1k$TransactionDate):max(c1k$TransactionDate))%in%c1k$TransactionDate==FALSE)))  
hist(c1k\_quant\_dist, breaks = 20)



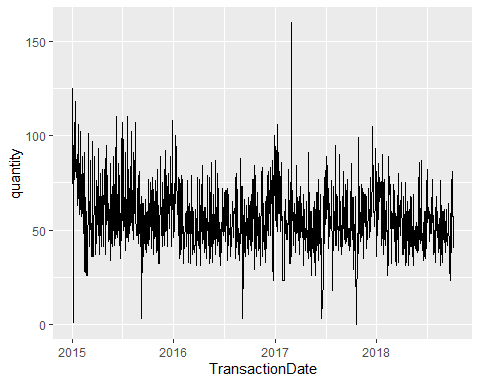
c1k$SalePrice\_local<- c1k$sales\_local/c1k$quantity  
c1k$SalePrice\_usd<- c1k$sales\_usd/c1k$quantity  
# cycles  
ggplot(data=c1k, aes(x=TransactionDate, y=SalePrice\_local))+geom\_line()



ggplot(data=c1k, aes(x=TransactionDate, y=SalePrice\_usd))+geom\_line()



ggplot(data=c1k, aes(x=TransactionDate, y=quantity))+geom\_line()



## Dropping unnecessary variables

#Drop unnecessary variables  
c1k$ClubNumber<-NULL  
c1k$Country<-NULL  
c1k$item<-NULL  
c1k$Description<-NULL  
c1k$Day\_of\_month<-NULL  
c1k$month<- NULL  
c1k$weekday<- NULL  
c1k$Day\_of\_week<- NULL  
c1k$Year<- NULL

# Loading the Categorical Sale

category<- read\_excel("Data/Book1.xlsx")  
#renaming variables  
colnames(category)[colnames(category)=="quantity"] <- "Category\_quantity"  
colnames(category)[colnames(category)=="salesusd"] <- "Category\_Sales\_usd"  
colnames(category)[colnames(category)=="sales"] <- "Category\_Sales\_local"  
#changing date formats  
category$TransactionDate<-ymd(category$TransactionDate)  
  
#subset to only club number 6101  
category\_club<- subset(category, ClubNumber == club)  
category\_club$ClubNumber<- NULL  
  
#Merge c1k with category   
c1k<- merge(c1k, category\_club)

# Aggregate data to the weekly level

c1k$TransactionYear<-year(c1k$TransactionDate) #Extract Transaction Year  
c1k$week\_number<-c1k$TransactionDate%>%date2ISOweek()%>%substr(7,8)  
c1k$week\_number\_year<-c1k$TransactionDate%>%date2ISOweek()%>%substr(1,4)  
  
#aggregation  
c1k\_week<-c1k%>%group\_by(week\_number\_year,week\_number)%>%summarise(  
 transaction\_avrg=mean(number\_transactions, na.rm = TRUE), members\_avrg=mean(number\_members, na.rm = TRUE), sales\_local\_avrg=mean(sales\_local, na.rm = TRUE),  
 sales\_usd\_avrg=mean(sales\_usd, na.rm = TRUE),  
 exchange\_rate\_avrg=mean(exchange\_rate, na.rm = TRUE),  
 category\_sales\_usd\_avrg=mean(Category\_Sales\_usd, na.rm = TRUE),  
 category\_sales\_local\_avrg=mean(Category\_Sales\_local, na.rm = TRUE),  
 category\_quantity\_avrg=mean(Category\_quantity, na.rm = TRUE),  
 quantity\_avrg=mean(quantity, na.rm = TRUE),  
 salePrice\_local\_avrg = mean(SalePrice\_local, na.rm = TRUE),  
 salePrice\_usd\_avrg = mean(SalePrice\_usd, na.rm = TRUE))  
  
c1k\_week$TransactionTime<-paste0(c1k\_week$week\_number\_year,"-W", c1k\_week$week\_number, "-3")%>%ISOweek2date()  
c1k\_week$week\_number<- as.numeric(c1k\_week$week\_number)  
c1k\_week$week\_number\_year<-as.numeric(c1k\_week$week\_number\_year)

# Identify Seasonality and create time series

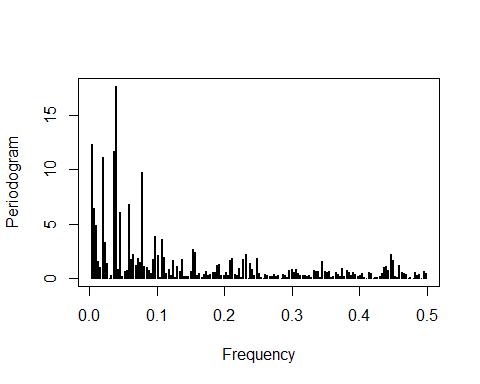
The periodogram function does not detect high frequency seasons accurately. To solve this problem, one of the built in functions need to be modified.  
run the following line in r:  
trace(spec.pgram, edit=TRUE)  
then change line 9 to:  
N <- N0 <- nrow(x) \* 128  
then click “save”  
the function is hence temporarily changed. Judgement is still needed to determine what type of seasonality it is capturing. In this case is year, half-year and three-month.  
The seasons detected are only rough estimates and we need to use knowledge about annual time series to determine the real seasons (semi-annual, annual etc.).Therefore, the identification of seasons here is not completely automated. Personal judgement is still needed.  
Firstly, 52.14, the number of weeks per year, is always included.  
Secondly, only periods shorter than 52.14 should be included, and their length should be adjusted to its closest whole-month length. For example, if 13.02 is reported is a strong period by Fourier transformation, 12.53 (three months) should be the season used in analyses.  
Thirdly, none period should be multiple of the other (otherwise Fourier regressor does not work). Therefore, half-year period is changed from 26.07 to 25.07 weeks. We believe better ways to deal with this problem exist.

trace(spec.pgram, edit=TRUE)

## Tracing function "spec.pgram" in package "stats"

## [1] "spec.pgram"

#detecting seasonality  
p <- periodogram(c1k\_week$quantity\_avrg[1:(nrow(c1k\_week)-len\_inv-len\_dec)])



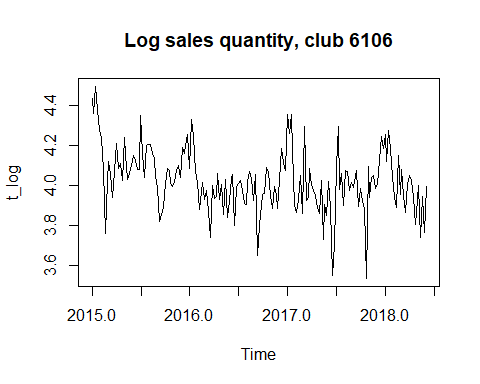
dd <- data.frame(freq=p$freq, spec=p$spec)  
order <- dd[order(-dd$spec),]  
top10 <- head(order, 10)  
   
# display the 3 highest "power" frequencies  
top10

## freq spec  
## 12 0.039087948 17.703496  
## 1 0.003257329 12.323585  
## 11 0.035830619 11.666099  
## 6 0.019543974 11.160643  
## 24 0.078175896 9.798765  
## 18 0.058631922 6.860350  
## 2 0.006514658 6.422977  
## 14 0.045602606 6.040181  
## 3 0.009771987 4.849362  
## 30 0.097719870 3.904339

# convert frequency to time periods  
time = 1/top10$freq  
  
order<-order[(1/order)<60,]  
top10 <- head(order, 10)  
time = 1/top10$freq  
  
season1<-25.07   
season2<-52.14   
season3<-13.14

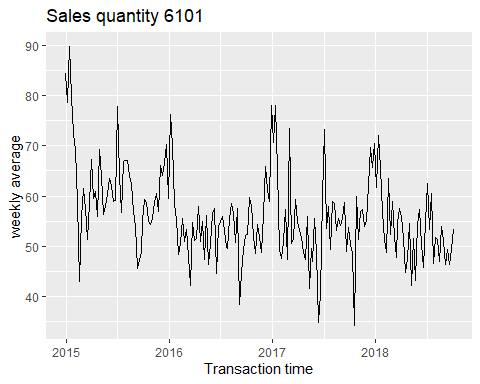
# Creating Time Series

#creating time series: log  
t\_log<-ts(log(c1k\_week$quantity\_avrg)[1:(nrow(c1k\_week)-len\_inv-len\_dec)], frequency = 52.14, start = c(c1k\_week$week\_number\_year[1], c1k\_week$week\_number[1]))   
  
plot(t\_log)+title("Log sales quantity, club 6106")



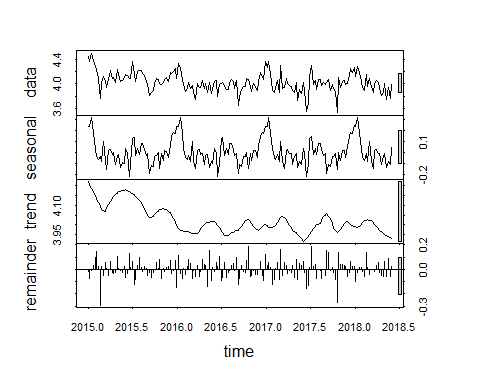
## integer(0)

ggplot(c1k\_week)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg))+xlab("Transaction time")+ylab("weekly average")+ggtitle(paste("Sales quantity", club))



## Deseasonalizing Methods

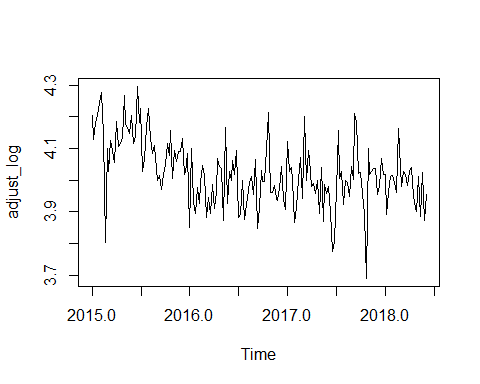
decompose\_log <- stl(t\_log, s.window = 13, t.window = 13)  
plot(decompose\_log)



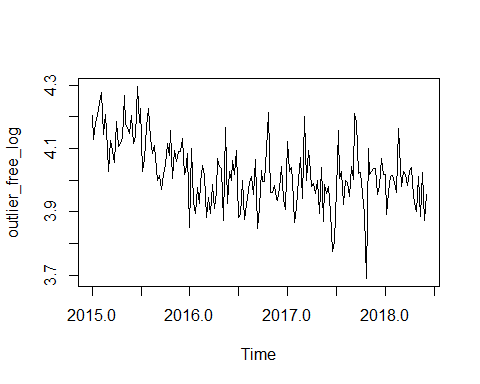
adjust\_log<- t\_log - decompose\_log$time.series[,1]

## Detrend after taking out the outliers

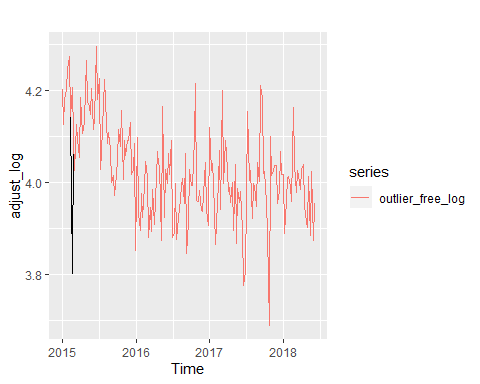
outlier\_free\_log<- tsclean(adjust\_log)  
trend\_log<- decompose\_log$time.series[, 2]  
detrend\_ts\_log <- outlier\_free\_log-(trend\_log - trend\_log[1])  
  
plot(adjust\_log)



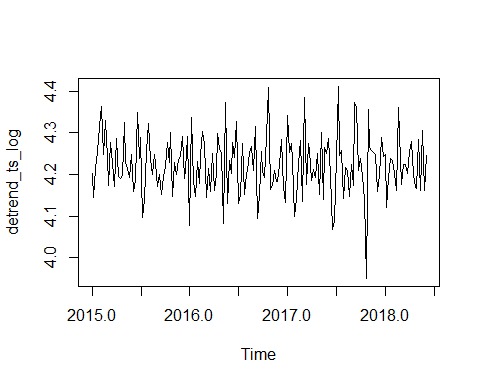
plot(outlier\_free\_log)



autoplot(adjust\_log)+autolayer(outlier\_free\_log)



plot(detrend\_ts\_log)

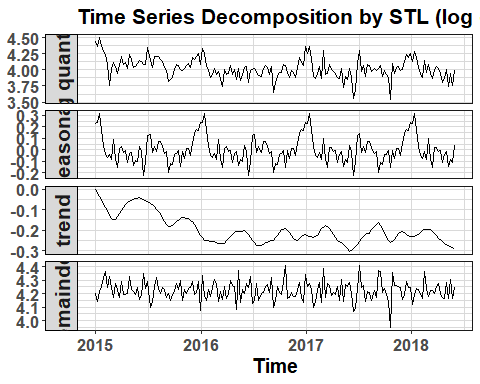


# Training and Validation Set: uni-variate methods

training\_log<-ts(detrend\_ts\_log[1:(nrow(c1k\_week)-len\_inv-len\_dec)], frequency = 52.14, start = c(c1k\_week$week\_number\_year[1], c1k\_week$week\_number[1]))   
  
# future trend and seasonality, which will be added back:  
  
trend\_fit\_log <- auto.arima(trend\_log)  
trend\_for\_log <- forecast(trend\_fit\_log, len\_inv+len\_dec)$mean  
retrend\_log<-trend\_for\_log  
reseasonal\_log<-forecast(decompose\_log$time.series[,1], len\_inv+len\_dec)$mean  
  
##validation set  
  
validation\_log<-ts(log(c1k\_week$quantity\_avrg)[(nrow(c1k\_week)-len\_inv-len\_dec+1):nrow(c1k\_week)], frequency = 52.14, end = c(c1k\_week$week\_number\_year[length(c1k\_week$week\_number\_year)], c1k\_week$week\_number[length(c1k\_week$week\_number\_year)]))  
  
validation<-ts(c1k\_week$quantity\_avrg[(nrow(c1k\_week)-len\_inv-len\_dec+1):nrow(c1k\_week)], frequency = 52.14, end = c(c1k\_week$week\_number\_year[length(c1k\_week$week\_number\_year)], c1k\_week$week\_number[length(c1k\_week$week\_number\_year)]))

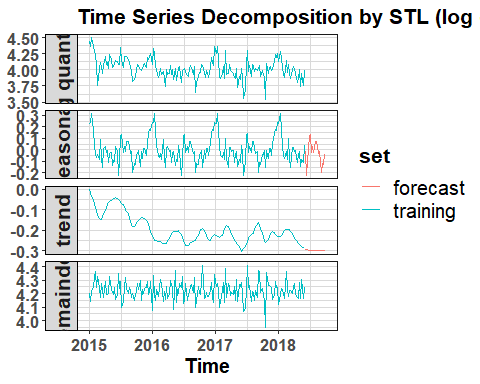
# Visualization of Trend, season and random

# data visualization  
theme\_ts <- theme(panel.border = element\_rect(fill = NA,   
 colour = "grey10"),  
 panel.background = element\_blank(),  
 panel.grid.minor = element\_line(colour = "grey85"),  
 panel.grid.major = element\_line(colour = "grey85"),  
 panel.grid.major.x = element\_line(colour = "grey85"),  
 axis.text = element\_text(size = 13, face = "bold"),  
 axis.title = element\_text(size = 15, face = "bold"),  
 plot.title = element\_text(size = 16, face = "bold"),  
 strip.text = element\_text(size = 16, face = "bold"),  
 strip.background = element\_rect(colour = "black"),  
 legend.text = element\_text(size = 15),  
 legend.title = element\_text(size = 16, face = "bold"),  
 legend.background = element\_rect(fill = "white"),  
 legend.key = element\_rect(fill = "white"))  
  
# Decomposition of log series  
  
decomp\_stl\_log<- data.table(Quant = c(t\_log, decompose\_log$time.series[, 1], decompose\_log$time.series[, 2]-decompose\_log$time.series[, 2][1], detrend\_ts\_log),  
 Date = rep(c1k\_week$TransactionTime[1:(nrow(c1k\_week)-len\_inv-len\_dec)], ncol(decompose\_log$time.series)+1),  
 Type = factor(rep(c("log quantity", colnames(decompose\_log$time.series)),  
 each = nrow(decompose\_log$time.series)),  
 levels = c("log quantity", colnames(decompose\_log$time.series))))  
  
ggplot(decomp\_stl\_log, aes(x = Date, y = Quant)) +  
 geom\_line() +   
 facet\_grid(Type ~ ., scales = "free\_y", switch = "y") +  
 labs(x = "Time", y = NULL,  
 title = "Time Series Decomposition by STL (log quantity)") +  
 theme\_ts



ggsave("Output/Countries/Colombia/Plots/decomp\_training\_log\_c1k.jpg", width = 6, height = 10)  
  
decomp\_stl\_training\_log<- decomp\_stl\_log  
decomp\_stl\_training\_log$set<-"training"  
  
decomp\_stl\_forecast\_log<-data.table(Quant=c(rep(NA, len\_dec+len\_inv), reseasonal\_log, retrend\_log-trend\_log[1], rep(NA, len\_dec+len\_inv)),   
 Date = rep(c1k\_week$TransactionTime[(nrow(c1k\_week)-len\_dec-len\_inv+1): nrow(c1k\_week)], ncol(decompose\_log$time.series)+1),   
 Type = factor(rep(c("log quantity", colnames(decompose\_log$time.series)),  
 each = len\_dec+len\_inv),  
 levels = c("log quantity", colnames(decompose\_log$time.series))))  
decomp\_stl\_forecast\_log$set<-"forecast"  
decomp\_stl\_combined\_log<-rbind(decomp\_stl\_training\_log, decomp\_stl\_forecast\_log)  
  
ggplot(decomp\_stl\_combined\_log, aes(x = Date, y = Quant, col=set)) +  
 geom\_line() +   
 facet\_grid(Type ~ ., scales = "free\_y", switch = "y") +  
 labs(x = "Time", y = NULL,  
 title = "Time Series Decomposition by STL (log quantity)") +  
 theme\_ts

## Warning: Removed 36 rows containing missing values (geom\_path).



ggsave("Output/Countries/Colombia/Plots/decomp\_combined\_log\_c1k.jpg", width = 8, height = 8)

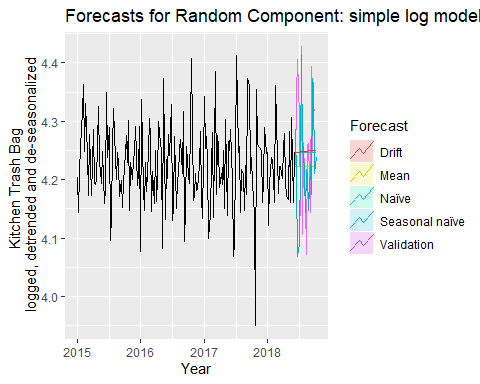
## Warning: Removed 36 rows containing missing values (geom\_path).

# Simple forecasting models

#Average Method   
Average\_Method<-meanf(training\_log, h = len\_dec+len\_inv)  
#Naive Method  
Naive\_Method<-naive(training\_log, h = len\_dec+len\_inv)  
#Seasonal Naive Method  
Seasonal\_Naive\_Method<-snaive(training\_log, h = len\_dec+len\_inv)

## Warning in lag.default(y, -lag): 'k' is not an integer

#Drift Method  
Drift\_Method<-rwf(training\_log, h = len\_dec+len\_inv, drift = TRUE)  
  
  
  
autoplot(training\_log) +  
 autolayer(validation\_log-(retrend\_log-trend\_log[1])-reseasonal\_log, series="Validation")+  
 autolayer(Average\_Method,  
 series="Mean", PI=FALSE) +  
 autolayer(Naive\_Method,  
 series="Naïve", PI=FALSE) +  
 autolayer(Seasonal\_Naive\_Method,  
 series="Seasonal naïve", PI=FALSE) +  
 autolayer(Drift\_Method,  
 series = "Drift", PI = FALSE)+  
 ggtitle("Forecasts for Random Component: simple log models") +  
 xlab("Year") + ylab( paste(item, "\n logged, detrended and de-seasonalized"))+  
 guides(colour=guide\_legend(title="Forecast"))



ggsave("Output/Countries/Colombia/Plots/simple\_log\_total\_c1k.jpg", width = 10, height = 5)

## Summarize performance

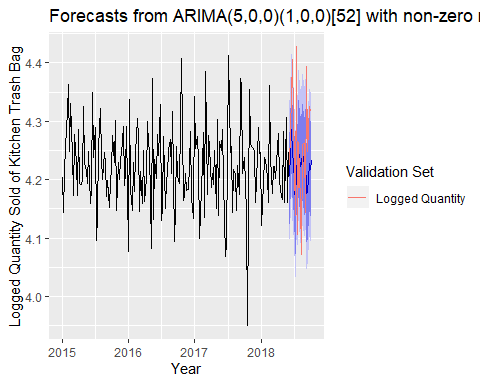
c1k\_week\_quantity\_models<-as.data.frame(validation)  
colnames(c1k\_week\_quantity\_models)<-"Actual"  
c1k\_week\_quantity\_models$TransactionYear<-c1k\_week$week\_number\_year[(nrow(c1k\_week)-len\_dec-len\_inv+1):nrow(c1k\_week)]  
c1k\_week\_quantity\_models$week\_number<-c1k\_week$week\_number[(nrow(c1k\_week)-len\_dec-len\_inv+1):nrow(c1k\_week)]  
c1k\_week\_quantity\_models$TransactionTime<-paste0(c1k\_week\_quantity\_models$TransactionYear,"-", c1k\_week\_quantity\_models$week\_number, "-3")%>%as.Date("%Y-%W-%w")  
  
c1k\_week\_quantity\_models<-performance\_index\_dtds(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "Average\_Method", pred\_value = Average\_Method$mean)  
  
c1k\_week\_quantity\_models<-performance\_index\_dtds(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "Naive\_Method", pred\_value = Naive\_Method$mean)  
  
c1k\_week\_quantity\_models<-performance\_index\_dtds(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "Seasonal\_Naive\_Method", pred\_value = Seasonal\_Naive\_Method$mean)  
  
c1k\_week\_quantity\_models<-performance\_index\_dtds(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "Drift\_Method", pred\_value = Drift\_Method$mean)  
  
# re-trended and re-seasonalized data: plots  
c1k\_week\_quantity\_models\_plot<-gather(c1k\_week\_quantity\_models%>%select(-c("Actual", "TransactionYear", "week\_number")), model, quantity, Average\_Method:Drift\_Method, factor\_key=TRUE)  
#for club 6107 run the following:  
#c1k\_week\_quantity\_models\_plot<-gather(c1k\_week\_quantity\_models%>%select(-c("Actual", "TransactionYear", "week\_number")), model, quantity, factor\_key=TRUE)  
c1k\_week\_quantity\_models\_plot$model<-as.character(c1k\_week\_quantity\_models\_plot$model)  
c1k\_week\_quantity\_models\_plot$quantity<-as.numeric(c1k\_week\_quantity\_models\_plot$quantity)  
  
c1k\_week\_int\_Actual<-c1k\_week%>%ungroup()%>%select(c("TransactionTime", "quantity\_avrg"))  
c1k\_week\_int\_Actual$model<-"Actual"  
c1k\_week\_int\_Actual$quantity<-as.numeric(c1k\_week\_int\_Actual$quantity\_avrg)  
c1k\_week\_int\_Actual<- c1k\_week\_int\_Actual[(nrow(c1k\_week)-len\_dec-len\_inv+1):nrow(c1k\_week),]  
  
  
c1k\_week\_int\_Actual$quantity\_avrg<-NULL  
  
c1k\_week\_quantity\_models\_plot<-rbind(as.data.frame(c1k\_week\_int\_Actual), as.data.frame(c1k\_week\_quantity\_models\_plot))

# Arima: auto.arima

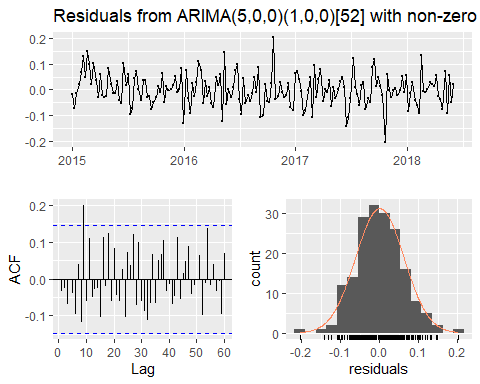
Simple\_Arima<- auto.arima(training\_log)  
summary(Simple\_Arima)

## Series: training\_log   
## ARIMA(5,0,0)(1,0,0)[52] with non-zero mean   
##   
## Coefficients:  
## ar1 ar2 ar3 ar4 ar5 sar1 mean  
## -0.1825 -0.1186 -0.2620 -0.2044 -0.2047 -0.3207 4.2232  
## s.e. 0.0732 0.0736 0.0713 0.0729 0.0740 0.0829 0.0019  
##   
## sigma^2 estimated as 0.00396: log likelihood=241.57  
## AIC=-467.15 AICc=-466.3 BIC=-441.65  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE  
## Training set 0.0008088031 0.06168521 0.04837278 -0.00268697 1.145589  
## MASE ACF1  
## Training set 0.5521407 -0.03315475

fc\_Simple\_Arima\_1<- forecast(Simple\_Arima, len\_dec+len\_inv, bootstrap = TRUE)  
  
#Graph the forecasted results   
autoplot(fc\_Simple\_Arima\_1) +  
 autolayer(log(validation)-(retrend\_log-trend\_log[1])-reseasonal\_log, series = "Logged Quantity")+  
 xlab("Year") + ylab(paste("Logged Quantity Sold of", item))+  
 guides(colour=guide\_legend(title="Validation Set"))



#Checking accuracy  
cr\_Simple\_Arima\_1<-checkresiduals(fc\_Simple\_Arima\_1)



##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(5,0,0)(1,0,0)[52] with non-zero mean  
## Q\* = 47.693, df = 29, p-value = 0.01581  
##   
## Model df: 7. Total lags used: 36

print(cr\_Simple\_Arima\_1$data.name)

## [1] "Residuals from ARIMA(5,0,0)(1,0,0)[52] with non-zero mean"

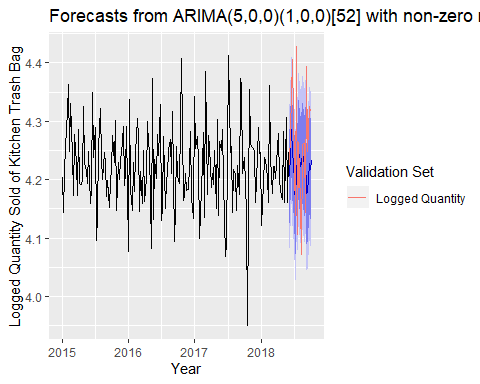
print(cr\_Simple\_Arima\_1$p.value)

## [1] 0.01581326

# Bring fc\_Simple\_Arima\_1$mean back through de-trending and de-seasonalizing  
c1k\_week\_quantity\_models<-performance\_index\_dtds(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "Simple\_Arima\_1", pred\_value = fc\_Simple\_Arima\_1$mean)

# ARIMA double season

#this arima model uses fourier and the three seasonality periods obtained above.  
  
Arima\_AIC <- auto.arima(training\_log)  
bestfit <- list(aicc=Arima\_AIC$aicc, i=0, j=0, fit=Arima\_AIC)  
  
fc\_ARIMA\_fourier<- forecast(Arima\_AIC, h = len\_dec+len\_inv)  
autoplot(fc\_ARIMA\_fourier) +  
 autolayer(log(validation)-(retrend\_log-trend\_log[1])-reseasonal\_log, series = "Logged Quantity")+  
 xlab("Year") + ylab(paste("Logged Quantity Sold of", item))+  
 guides(colour=guide\_legend(title="Validation Set"))



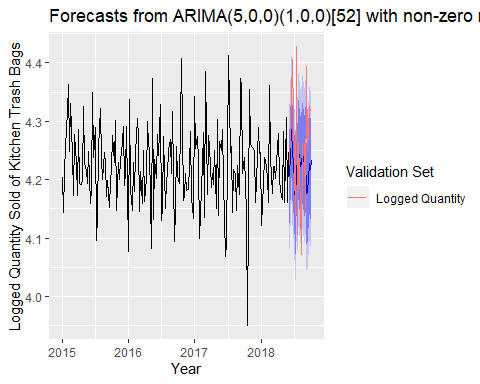
#choose the best model by AICc  
for(i in 1:3) {  
 for (j in 1:3){  
 z1 <- fourier(ts(training\_log, frequency= season1), K=i)  
 z2 <- fourier(ts(training\_log, frequency= season2), K=j)  
 Arima\_Seasons1\_2 <- auto.arima(training\_log, xreg=cbind(z1, z2), seasonal=F)  
 if(Arima\_Seasons1\_2$aicc < bestfit$aicc) {  
 bestfit <- list(aicc=Arima\_Seasons1\_2$aicc, i=i, j=j, Arima\_Seasons1\_2=Arima\_Seasons1\_2)  
 }  
 }  
}  
bestfit

## $aicc  
## [1] -466.3026  
##   
## $i  
## [1] 0  
##   
## $j  
## [1] 0  
##   
## $fit  
## Series: training\_log   
## ARIMA(5,0,0)(1,0,0)[52] with non-zero mean   
##   
## Coefficients:  
## ar1 ar2 ar3 ar4 ar5 sar1 mean  
## -0.1825 -0.1186 -0.2620 -0.2044 -0.2047 -0.3207 4.2232  
## s.e. 0.0732 0.0736 0.0713 0.0729 0.0740 0.0829 0.0019  
##   
## sigma^2 estimated as 0.00396: log likelihood=241.57  
## AIC=-467.15 AICc=-466.3 BIC=-441.65

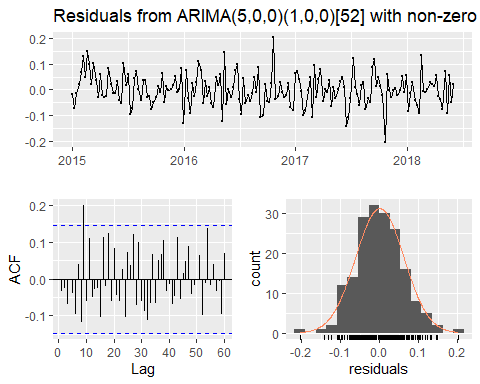
fc\_Arima\_Seasons1\_2 <- forecast(bestfit$fit,   
 xreg=cbind(  
 fourier(ts(training\_log, frequency=52), K=bestfit$i, h=len\_dec+len\_inv),  
 fourier(ts(training\_log, frequency=25), K=bestfit$j, h=len\_dec+len\_inv)))

## Warning in forecast.Arima(bestfit$fit, xreg =  
## cbind(fourier(ts(training\_log, : xreg not required by this model, ignoring  
## the provided regressors

fc\_Arima\_Seasons1\_2 <- forecast(bestfit$fit, h=len\_dec+len\_inv)  
   
autoplot(fc\_Arima\_Seasons1\_2) +  
 autolayer(validation\_log-(retrend\_log-trend\_log[1])-reseasonal\_log, series = "Logged Quantity")+  
 xlab("Year") + ylab("Logged Quantity Sold of Kitchen Trash Bags")+  
 guides(colour=guide\_legend(title="Validation Set"))



cr\_ARIMA\_seasons1\_2<-checkresiduals(fc\_Arima\_Seasons1\_2)



##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(5,0,0)(1,0,0)[52] with non-zero mean  
## Q\* = 47.693, df = 29, p-value = 0.01581  
##   
## Model df: 7. Total lags used: 36

print(cr\_ARIMA\_seasons1\_2$data.name)

## [1] "Residuals from ARIMA(5,0,0)(1,0,0)[52] with non-zero mean"

print(cr\_ARIMA\_seasons1\_2$p.value)

## [1] 0.01581326

# Bring fc\_ARIMA$mean back through de-trending and de-seasonalizing  
c1k\_week\_quantity\_models<-performance\_index\_dtds(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "Arima\_Seasons1\_2", pred\_value = fc\_Arima\_Seasons1\_2$mean)

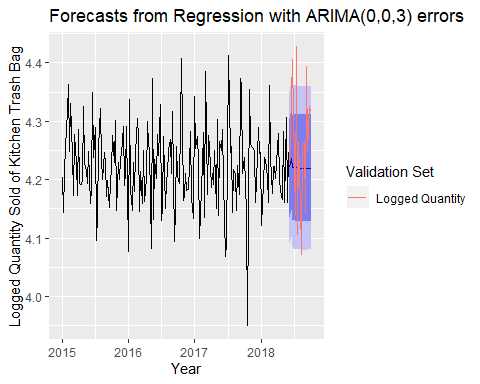
# ARIMA Model: ARIMA single season

This section uses grid to tune the K parameter to get the Fourier regressor

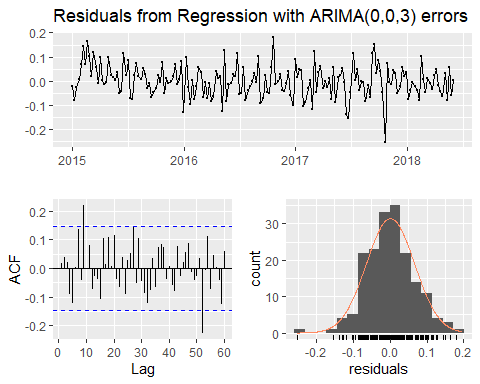
##Finding best fit model   
Arima\_Fourier\_AIC<-list(aicc=Inf)  
for(K in seq(25)) {  
 fit <- auto.arima(training\_log, xreg=fourier(training\_log, K=K),  
 seasonal=FALSE)  
 if(fit[["aicc"]] < Arima\_Fourier\_AIC[["aicc"]]) {  
 Arima\_Fourier\_AIC <- fit  
 bestK <- K  
 }  
}  
Arima\_Fourier\_AIC

## Series: training\_log   
## Regression with ARIMA(0,0,3) errors   
##   
## Coefficients:  
## ma1 ma2 ma3 intercept S1-52 C1-52  
## -0.2337 -0.2146 -0.2469 4.2228 0.0032 0.0014  
## s.e. 0.0738 0.0744 0.0662 0.0015 0.0025 0.0027  
##   
## sigma^2 estimated as 0.004387: log likelihood=234.68  
## AIC=-455.37 AICc=-454.71 BIC=-433.05

fc\_Arima\_Fourier\_AIC <- forecast(Arima\_Fourier\_AIC,xreg=fourier(training\_log, K=bestK, h=len\_dec+len\_inv))  
autoplot(fc\_Arima\_Fourier\_AIC) + autolayer(log(validation)-(retrend\_log-trend\_log[1])-reseasonal\_log, series = "Logged Quantity")+  
 xlab("Year") + ylab(paste("Logged Quantity Sold of", item))+  
 guides(colour=guide\_legend(title="Validation Set"))



cr\_ARIMA\_Fourier\_AIC<-checkresiduals(fc\_Arima\_Fourier\_AIC)



##   
## Ljung-Box test  
##   
## data: Residuals from Regression with ARIMA(0,0,3) errors  
## Q\* = 50.479, df = 30, p-value = 0.01105  
##   
## Model df: 6. Total lags used: 36

print(cr\_ARIMA\_Fourier\_AIC$data.name)

## [1] "Residuals from Regression with ARIMA(0,0,3) errors"

print(cr\_ARIMA\_seasons1\_2$p.value)

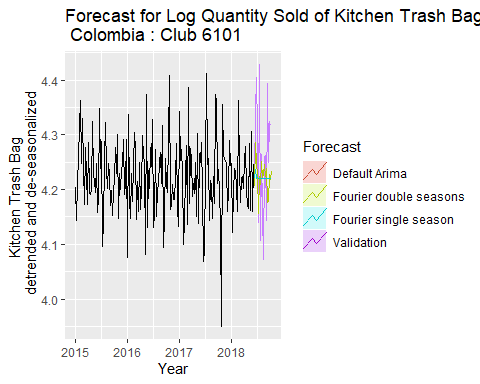
## [1] 0.01581326

# Bring fc\_Arima\_Fourier\_AIC$mean back through de-trending and de-seasonalizing  
c1k\_week\_quantity\_models<-performance\_index\_dtds(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "Arima\_Fourier\_AIC", pred\_value = fc\_Arima\_Fourier\_AIC$mean)

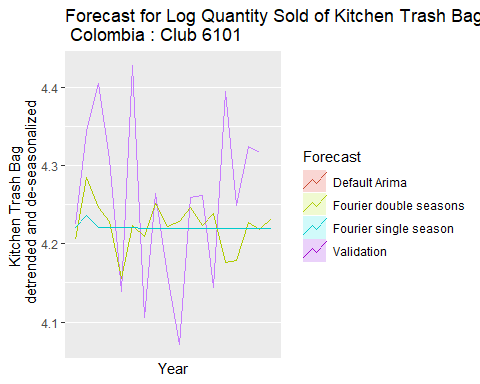
# Visualization of 3 ARIMA models

# before de-trending and de-seasoning  
autoplot(training\_log) +  
 autolayer(validation\_log-(retrend\_log-trend\_log[1])-reseasonal\_log, series="Validation", PI=FALSE)+  
 autolayer(fc\_Simple\_Arima\_1,  
 series="Default Arima", PI=FALSE) +  
 autolayer(fc\_Arima\_Seasons1\_2,  
 series="Fourier double seasons", PI=FALSE) +  
 autolayer(fc\_Arima\_Fourier\_AIC,  
 series="Fourier single season", PI=FALSE)+  
 ggtitle(paste("Forecast for Log Quantity Sold of", item, "\n", country, ": Club", club)) +  
 xlab("Year") + ylab(paste(item,"\n detrended and de-seasonalized"))+  
 guides(colour=guide\_legend(title="Forecast"))

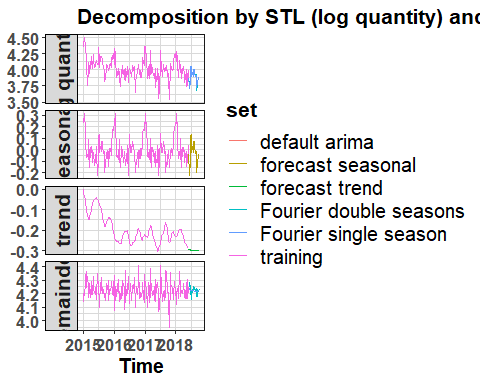
## Warning: Ignoring unknown parameters: PI



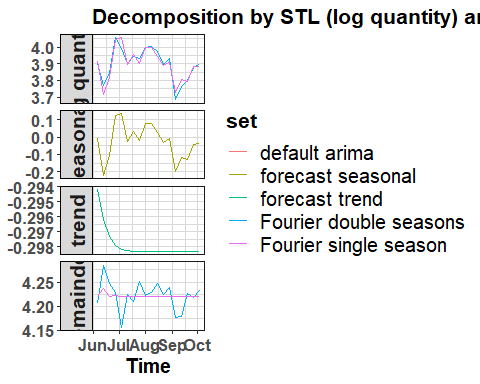
ggsave("Output/Countries/Colombia/Plots/ARIMA\_log\_total\_weekly\_c1k.jpg", width = 10, height = 5)  
  
autoplot(validation\_log-(retrend\_log-trend\_log[1])-reseasonal\_log, series="Validation")+  
 autolayer(fc\_Simple\_Arima\_1,  
 series="Default Arima", PI=FALSE) +  
 autolayer(fc\_Arima\_Seasons1\_2,  
 series="Fourier double seasons", PI=FALSE) +  
 autolayer(fc\_Arima\_Fourier\_AIC,  
 series="Fourier single season", PI=FALSE)+  
 ggtitle(paste("Forecast for Log Quantity Sold of", item, "\n", country, ": Club", club)) +  
 xlab("Year") + ylab(paste(item,"\n detrended and de-seasonalized"))+  
 guides(colour=guide\_legend(title="Forecast"))



ggsave("Output/Countries/Colombia/Plots/ARIMA\_log\_total\_weekly1\_c1k.jpg", width = 10, height = 5)  
  
decomp\_stl<- data.table(Quant = c(t\_log, as.numeric(decompose\_log$time.series)),  
 Date = rep(c1k\_week$TransactionTime[1:(nrow(c1k\_week)-len\_dec-len\_inv)], ncol(decompose\_log$time.series)+1),  
 Type = factor(rep(c("original data", colnames(decompose\_log$time.series)),  
 each = nrow(decompose\_log$time.series)),  
 levels = c("original data", colnames(decompose\_log$time.series))))  
  
decomp\_ARIMA\_forecast\_log<-data.table(Quant=c(fc\_Simple\_Arima\_1$mean+reseasonal\_log+(retrend\_log-trend\_log[1]), fc\_Arima\_Seasons1\_2$mean+reseasonal\_log+(retrend\_log-trend\_log[1]), fc\_Arima\_Fourier\_AIC$mean+reseasonal\_log+(retrend\_log-trend\_log[1]), reseasonal\_log, (retrend\_log-trend\_log[1]), fc\_Simple\_Arima\_1$mean, fc\_Arima\_Seasons1\_2$mean, fc\_Arima\_Fourier\_AIC$mean),   
 Date = rep(c1k\_week$TransactionTime[(nrow(c1k\_week)-len\_dec-len\_inv+1): nrow(c1k\_week)], ncol(decompose\_log$time.series)+5),   
 Type = factor(rep(c(rep("log quantity", 3), colnames(decompose\_log$time.series)[1:2], rep(colnames(decompose\_log$time.series)[3], 3)),  
 each = len\_dec+len\_inv),  
 levels = c("log quantity", colnames(decompose\_log$time.series))))  
decomp\_ARIMA\_forecast\_log$set<-rep(c("default arima", "Fourier double seasons", "Fourier single season", "forecast seasonal", "forecast trend", "default arima", "Fourier double seasons", "Fourier single season"), each=len\_dec+len\_inv)  
  
decomp\_ARIMA\_combined\_log<-rbind(decomp\_stl\_training\_log, decomp\_ARIMA\_forecast\_log)  
  
  
ggplot(decomp\_ARIMA\_combined\_log, aes(x = Date, y = Quant, col=set)) +  
 geom\_line() +   
 facet\_grid(Type ~ ., scales = "free\_y", switch = "y") +  
 labs(x = "Time", y = NULL,  
 title = "Decomposition by STL (log quantity) and ARIMA predictions") +  
 theme\_ts



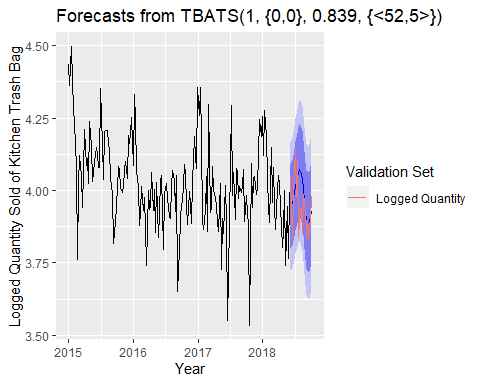
ggsave("Output/Countries/Colombia/Plots/ARIMA\_log\_grid\_weekly\_c1k.jpg", width = 10, height = 5)  
  
  
ggplot(decomp\_ARIMA\_combined\_log[decomp\_ARIMA\_combined\_log$Date>=(max(decomp\_ARIMA\_combined\_log$Date, na.rm = TRUE)-119),], aes(x = Date, y = Quant, col=set)) +  
 geom\_line() + facet\_grid(Type ~ ., scales = "free\_y", switch = "y") +  
 labs(x = "Time", y = NULL,  
 title = "Decomposition by STL (log quantity) and ARIMA predictions") +  
 theme\_ts



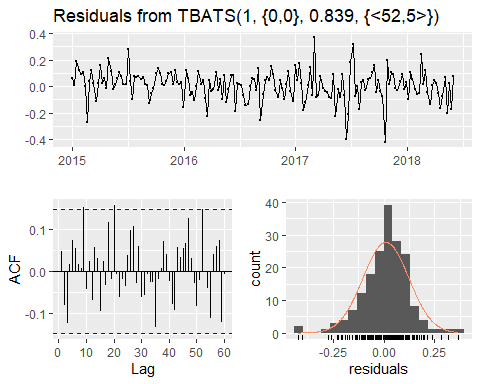
ggsave("Output/Countries/Colombia/Plots/ARIMA\_log\_grid\_weekly1\_c1k.jpg", width = 10, height = 5)

# TBATS

# Uses a combination of Fourier terms with an exponential smoothing state space model and a Box-Cox transformation. Seasonality is allowed to change slowly over time.  
  
#Feed Raw Data to this model [with seasonality, trend and outliers]  
fit\_TBATS\_Raw\_1<- tbats(t\_log, use.box.cox = NULL, use.trend = TRUE, use.damped.trend = NULL, seasonal.periods = 52, use.arma.errors = TRUE, biasadj = TRUE)  
  
  
fc\_TBATS\_Raw\_1<- forecast(fit\_TBATS\_Raw\_1, h=len\_dec+len\_inv, bootstrap = TRUE) #TBATS(1, {0,0}, 0.915, {<52,5>})  
autoplot(fc\_TBATS\_Raw\_1) + autolayer(log(validation), series = "Logged Quantity")+  
 xlab("Year") + ylab(paste("Logged Quantity Sold of", item))+  
 guides(colour=guide\_legend(title="Validation Set"))

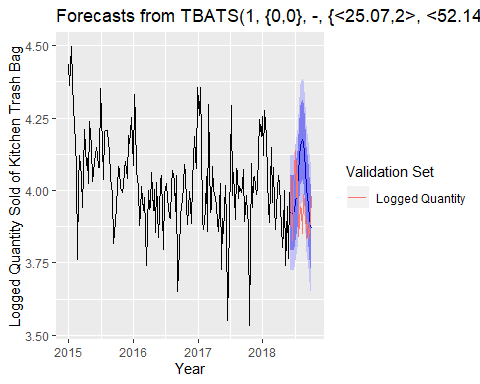


checkresiduals(fc\_TBATS\_Raw\_1)

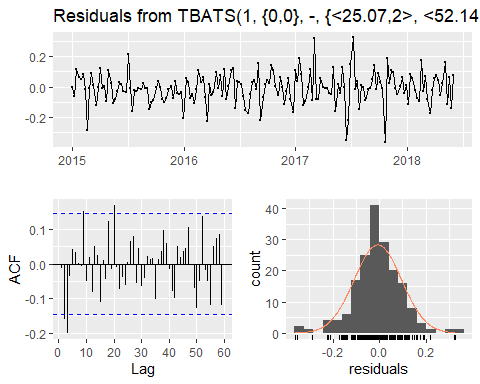


##   
## Ljung-Box test  
##   
## data: Residuals from TBATS(1, {0,0}, 0.839, {<52,5>})  
## Q\* = 35.404, df = 19, p-value = 0.01248  
##   
## Model df: 17. Total lags used: 36

c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "fc\_TBATS\_Raw\_1", pred\_value = fc\_TBATS\_Raw\_1$mean)  
  
#Feed Raw Data to this model [with seasonality, trend and outliers] [2 seasonal periods]  
fit\_TBATS\_Season1\_2<- tbats(t\_log, use.box.cox = NULL, use.trend = NULL, use.damped.trend = NULL, seasonal.periods = c(season2,season1), use.arma.errors = TRUE, biasadj = TRUE)  
  
fc\_TBATS\_Season1\_2<- forecast(fit\_TBATS\_Season1\_2, h=len\_dec+len\_inv, bootstrap = TRUE)   
autoplot(fc\_TBATS\_Season1\_2) + autolayer(log(validation), series = "Logged Quantity")+  
 xlab("Year") + ylab(paste("Logged Quantity Sold of", item))+  
 guides(colour=guide\_legend(title="Validation Set"))

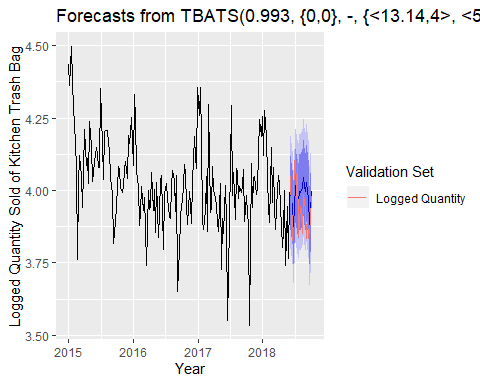


checkresiduals(fc\_TBATS\_Season1\_2)

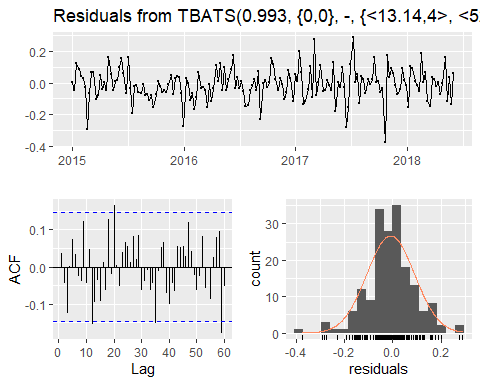


##   
## Ljung-Box test  
##   
## data: Residuals from TBATS(1, {0,0}, -, {<25.07,2>, <52.14,5>})  
## Q\* = 41.218, df = 16, p-value = 0.0005157  
##   
## Model df: 20. Total lags used: 36

c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "fc\_TBATS\_Season1\_2", pred\_value = fc\_TBATS\_Season1\_2$mean)  
  
## TBATS Model with top 3 seasonal periods   
fc\_TBATS\_Season2\_3<- forecast(tbats(t\_log, use.box.cox = NULL, use.trend = NULL, use.damped.trend = NULL, seasonal.periods = c(season2,season3), use.arma.errors = TRUE, biasadj = TRUE),h=len\_dec+len\_inv)   
autoplot(fc\_TBATS\_Season2\_3)+ autolayer(log(validation), series = "Logged Quantity")+  
 xlab("Year") + ylab(paste("Logged Quantity Sold of", item))+  
 guides(colour=guide\_legend(title="Validation Set"))

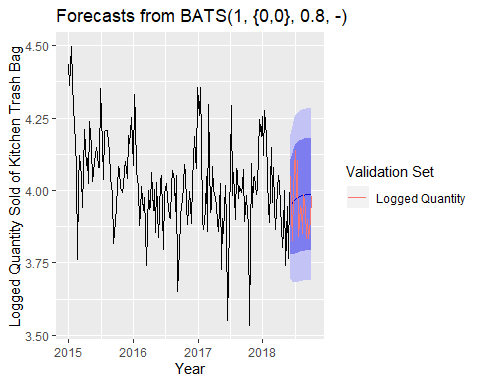


checkresiduals(fc\_TBATS\_Season2\_3)

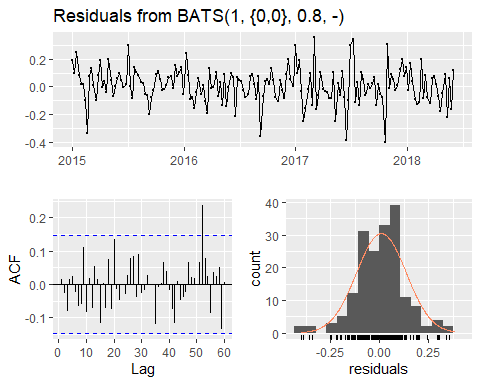


##   
## Ljung-Box test  
##   
## data: Residuals from TBATS(0.993, {0,0}, -, {<13.14,4>, <52.14,5>})  
## Q\* = 40.359, df = 11, p-value = 3.106e-05  
##   
## Model df: 25. Total lags used: 36

c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "fc\_TBATS\_Season2\_3", pred\_value = fc\_TBATS\_Season2\_3$mean)  
  
## TBATS Model with top 2 and 3 seasonal periods   
fc\_TBATS\_Season1\_3 <- forecast(tbats(t\_log, seasonal.periods=c(season1,season3)), h=len\_dec+len\_inv)   
autoplot(fc\_TBATS\_Season1\_3)+ autolayer(log(validation), series = "Logged Quantity")+  
 xlab("Year") + ylab(paste("Logged Quantity Sold of", item))+  
 guides(colour=guide\_legend(title="Validation Set"))

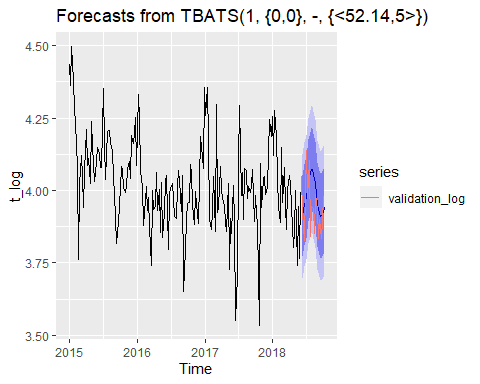


checkresiduals(fc\_TBATS\_Season1\_3)

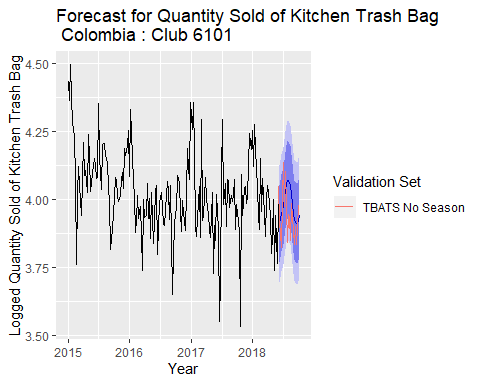


##   
## Ljung-Box test  
##   
## data: Residuals from BATS(1, {0,0}, 0.8, -)  
## Q\* = 27.732, df = 31, p-value = 0.635  
##   
## Model df: 5. Total lags used: 36

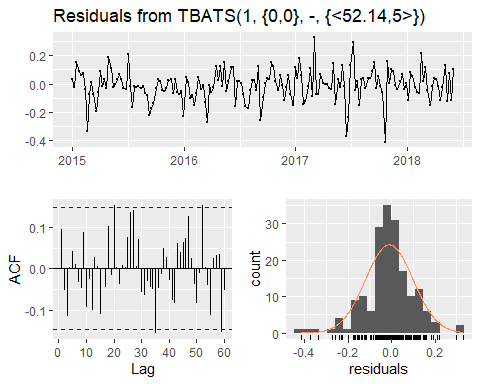
c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "fc\_TBATS\_Season1\_3", pred\_value = fc\_TBATS\_Season1\_3$mean)  
  
## really simple tbats  
fit\_TBATS\_NoSeason<- tbats(t\_log)  
fc\_TBATS\_NoSeason<- forecast(fit\_TBATS\_NoSeason, h=len\_dec+len\_inv)  
autoplot(fc\_TBATS\_NoSeason) + autolayer(validation\_log)



autoplot(fc\_TBATS\_NoSeason)+ autolayer(log(validation), series = "TBATS No Season")+  
 xlab("Year") + ylab(paste("Logged Quantity Sold of", item))+  
 ggtitle(paste("Forecast for Quantity Sold of", item, "\n", country, ": Club", club)) +  
 guides(colour=guide\_legend(title="Validation Set"))



checkresiduals(fc\_TBATS\_NoSeason)



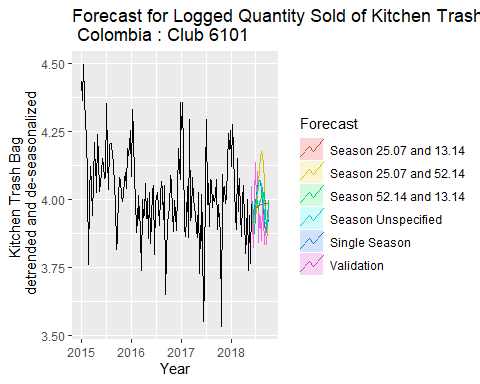
##   
## Ljung-Box test  
##   
## data: Residuals from TBATS(1, {0,0}, -, {<52.14,5>})  
## Q\* = 40.259, df = 22, p-value = 0.01008  
##   
## Model df: 14. Total lags used: 36

c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "fc\_TBATS\_NoSeason", pred\_value = fc\_TBATS\_NoSeason$mean)

# visualization of 5 TBATS models

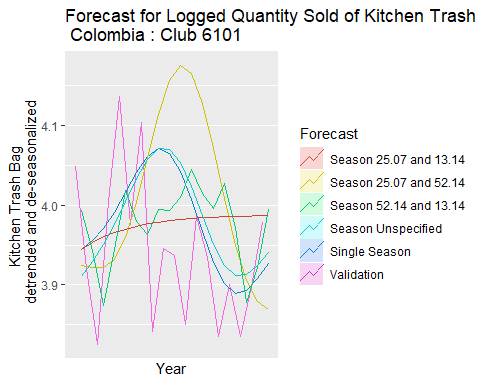
autoplot(t\_log) +  
 autolayer(validation\_log, series="Validation", PI=FALSE)+  
 autolayer(fc\_TBATS\_Raw\_1,  
 series="Single Season", PI=FALSE) +  
 autolayer(fc\_TBATS\_Season1\_2,  
 series=paste("Season", season1, "and", season2), PI=FALSE) +  
 autolayer(fc\_TBATS\_Season2\_3,  
 series=paste("Season", season2, "and", season3), PI=FALSE) +  
 autolayer(fc\_TBATS\_Season1\_3,  
 series=paste( "Season", season1, "and", season3), PI=FALSE) +  
 autolayer(fc\_TBATS\_NoSeason,  
 series="Season Unspecified", PI=FALSE) +  
 ggtitle(paste("Forecast for Logged Quantity Sold of", item, "\n", country, ": Club", club)) +  
 xlab("Year") + ylab(paste(item, "\n detrended and de-seasonalized"))+  
 guides(colour=guide\_legend(title="Forecast"))

## Warning: Ignoring unknown parameters: PI



ggsave("Output/Countries/Colombia/Plots/tbats\_log\_total\_weekly\_c1k.jpg", width = 10, height = 5)  
  
autoplot(validation\_log, series="Validation", PI=FALSE)+  
 autolayer(fc\_TBATS\_Raw\_1,  
 series="Single Season", PI=FALSE) +  
 autolayer(fc\_TBATS\_Season1\_2,  
 series=paste("Season", season1, "and", season2), PI=FALSE) +  
 autolayer(fc\_TBATS\_Season2\_3,  
 series=paste("Season", season2, "and", season3), PI=FALSE) +  
 autolayer(fc\_TBATS\_Season1\_3,  
 series=paste("Season", season1, "and", season3), PI=FALSE) +  
 autolayer(fc\_TBATS\_NoSeason,  
 series="Season Unspecified", PI=FALSE) +  
 ggtitle(paste("Forecast for Logged Quantity Sold of", item, "\n", country, ": Club", club)) +  
 xlab("Year") + ylab(paste(item, "\n detrended and de-seasonalized"))+  
 guides(colour=guide\_legend(title="Forecast"))

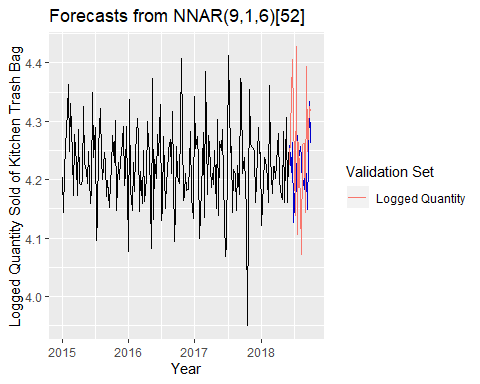
## Warning: Ignoring unknown parameters: PI



ggsave("Output/Countries/Colombia/Plots/tbats\_log\_total\_weekly1\_c1k.jpg", width = 10, height = 5)

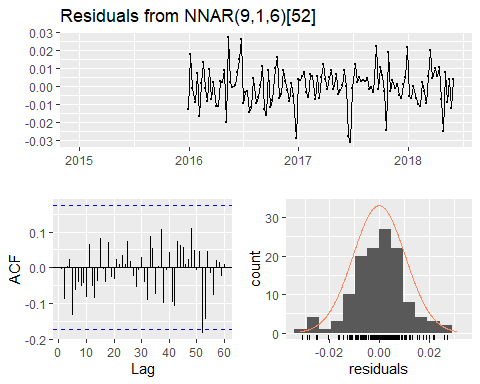
# Neural network

#Neural Networks (input detrneded and de-seasonalized data)  
#NNAR(p,P,k)m -> p = lagged inputs, P = equivalent to ARIMA(p,0,0)(P,0,0)m, k = nods in the single hidden layer   
fit\_NN\_1<- nnetar(training\_log, lambda = "auto")  
fc\_NN\_1<- forecast(fit\_NN\_1, h=len\_dec+len\_inv)  
autoplot(fc\_NN\_1) + autolayer(log(validation)-(retrend\_log-trend\_log[1])-reseasonal\_log, series = "Logged Quantity")+  
 xlab("Year") + ylab(paste("Logged Quantity Sold of", item))+  
 guides(colour=guide\_legend(title="Validation Set"))

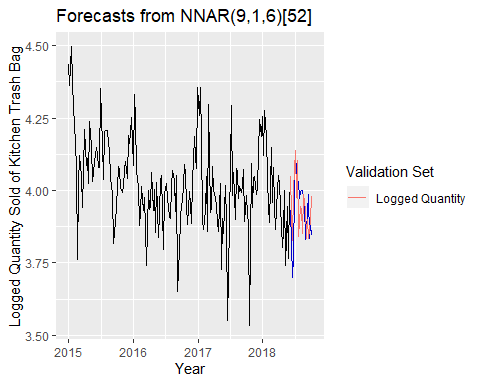


checkresiduals(fc\_NN\_1)

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

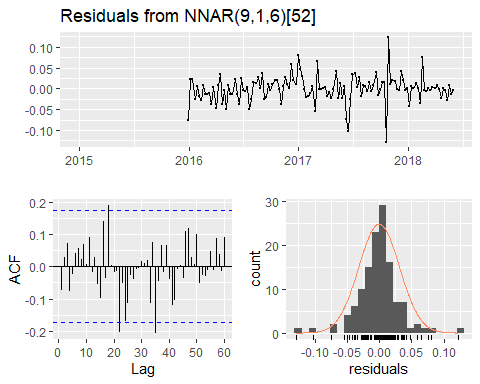


c1k\_week\_quantity\_models<-performance\_index\_dtds(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "fc\_NN\_1", pred\_value = fc\_NN\_1$mean)  
  
# Neural networks (input raw data)  
fit\_NN\_Raw<- nnetar(t\_log, lambda = "auto")  
fc\_NN\_Raw<- forecast(fit\_NN\_Raw, h=len\_dec+len\_inv)  
autoplot(fc\_NN\_Raw) + autolayer(log(validation), series = "Logged Quantity")+  
 xlab("Year") + ylab(paste("Logged Quantity Sold of", item))+  
 guides(colour=guide\_legend(title="Validation Set"))



checkresiduals(fc\_NN\_Raw)

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

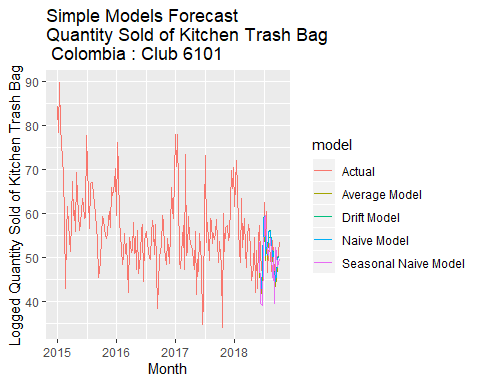


c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "fc\_NN\_Raw", pred\_value = fc\_NN\_Raw$mean)

# Combination of Uni-variate Forecasts

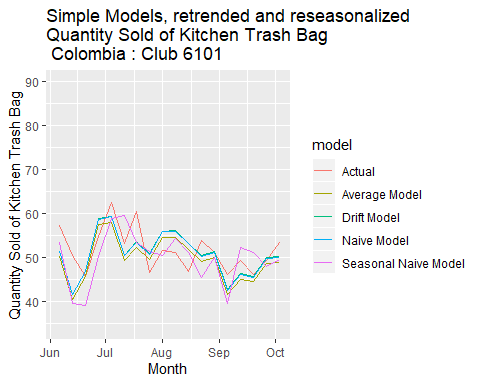
# Summarize performance of current models

tsmodels<-c("Average\_Method","Naive\_Method","Seasonal\_Naive\_Method","Drift\_Method","Simple\_Arima\_1","Arima\_Seasons1\_2", "Arima\_Fourier\_AIC","fc\_TBATS\_Raw\_1","fc\_TBATS\_Season1\_2","fc\_TBATS\_Season2\_3","fc\_TBATS\_Season1\_3","fc\_TBATS\_NoSeason","fc\_NN\_1","fc\_NN\_Raw", "Combination\_uni")  
indices<-c("MEAN", "RMSE", "MAE", "MPE", "MAPE", "MASE")  
performance\_week\_total<-matrix(nrow = length(tsmodels), ncol = length(indices))  
for (i in 1:length(tsmodels)){  
 for (j in 1:length(indices)){  
 performance\_week\_total[i, j]=get(paste0(indices[j], "\_", tsmodels[i], "\_total"))  
 }  
}  
performance\_week\_total<-as.data.frame(performance\_week\_total)  
colnames(performance\_week\_total)<-indices  
rownames(performance\_week\_total)<-tsmodels  
write.csv(x = performance\_week\_total, file = "Output/Countries/Colombia/Tables/performance\_week\_total\_c1k.csv")  
  
performance\_week\_target<-matrix(nrow = length(tsmodels), ncol = length(indices))  
for (i in 1:length(tsmodels)){  
 for (j in 1:length(indices)){  
 performance\_week\_target[i, j]=get(paste0(indices[j], "\_", tsmodels[i], "\_target"))  
 }  
}  
performance\_week\_target<-as.data.frame(performance\_week\_target)  
colnames(performance\_week\_target)<-indices  
rownames(performance\_week\_target)<-tsmodels  
write.csv(x = performance\_week\_total, file = "Output/Countries/Colombia/Tables/performance\_week\_target\_c1k.csv")  
# re-trended and re-seasonalized plots  
  
c1k\_week\_quantity\_models\_plot<-gather(c1k\_week\_quantity\_models%>%select(-c("Actual", "TransactionYear", "week\_number")), model, quantity\_avrg, Average\_Method:Combination\_uni, factor\_key=TRUE)  
c1k\_week\_quantity\_models\_plot$TransactionTime<-as.Date(c1k\_week\_quantity\_models\_plot$TransactionTime)  
c1k\_week\_quantity\_models\_plot$model<-as.character(c1k\_week\_quantity\_models\_plot$model)  
c1k\_week\_quantity\_models\_plot$quantity\_avrg<-as.numeric(c1k\_week\_quantity\_models\_plot$quantity\_avrg)  
  
c1k\_week\_int\_Actual<-c1k\_week%>%ungroup()%>%select(c("TransactionTime", "quantity\_avrg"))  
c1k\_week\_int\_Actual$model<-"Actual"  
c1k\_week\_int\_Actual$TransactionTime<-as.Date(c1k\_week\_int\_Actual$TransactionTime)  
c1k\_week\_int\_Actual$quantity\_avrg<-as.numeric(c1k\_week\_int\_Actual$quantity\_avrg)  
c1k\_week\_quantity\_models\_plot<-rbind(c1k\_week\_int\_Actual, c1k\_week\_quantity\_models\_plot)  
  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Average\_Method"]<-"Average Model"  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Naive\_Method"]<-"Naive Model"  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Seasonal\_Naive\_Method"]<-"Seasonal Naive Model"  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Drift\_Method"]<-"Drift Model"  
  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Simple\_Arima\_1"]<-"Default ARIMA"  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Arima\_Seasons1\_2"]<-"Double Season ARIMA"  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Arima\_Fourier\_AIC"]<-"Single Season ARIMA"  
  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="fc\_TBATS\_Raw\_1"]<-"TBATS model 1 with Raw Data"  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="fc\_TBATS\_Season1\_2"]<-paste("TBATS model 2 with Seasons", season1, "and", season2)  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="fc\_TBATS\_Season2\_3"]<-paste("TBATS model 2 with Seasons", season2, "and", season3)  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="fc\_TBATS\_Season1\_3"]<-paste("TBATS model 2 with Seasons", season1, "and", season3)  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="fc\_TBATS\_NoSeason"]<-"TBATS No Season"  
  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="fc\_NN\_1"]<-"Neural Networks De-trended and De-seasonalized"  
  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="fc\_NN\_Raw"]<-"Neural Networks on Raw Data"  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Combination\_uni"]<- "Combination Models Univariate"  
# plots: simple models  
c1k\_week\_quantity\_models\_plot\_log<-c1k\_week\_quantity\_models\_plot[(c1k\_week\_quantity\_models\_plot$model=="Actual" |c1k\_week\_quantity\_models\_plot$model=="Average Model"  
 |c1k\_week\_quantity\_models\_plot$model=="Naive Model"  
 |c1k\_week\_quantity\_models\_plot$model=="Seasonal Naive Model"  
 |c1k\_week\_quantity\_models\_plot$model=="Drift Model"),]  
ggplot(data=c1k\_week\_quantity\_models\_plot\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+ggtitle(paste("Simple Models Forecast","\nQuantity Sold of", item, "\n",country, ": Club", club))+ xlab("Month") + ylab(paste("Logged Quantity Sold of", item))



ggsave("Output/Countries/Colombia/Plots/simple\_log\_weekly\_pred\_c1k.jpg", width = 7, height = 4)  
  
ggplot(data=c1k\_week\_quantity\_models\_plot\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+xlim(max(c1k\_week$TransactionTime, na.rm = TRUE)-120, max(c1k\_week$TransactionTime, na.rm = TRUE))+ggtitle(paste("Simple Models, retrended and reseasonalized","\nQuantity Sold of", item, "\n",country, ": Club", club)) + xlab("Month") + ylab(paste("Quantity Sold of", item))

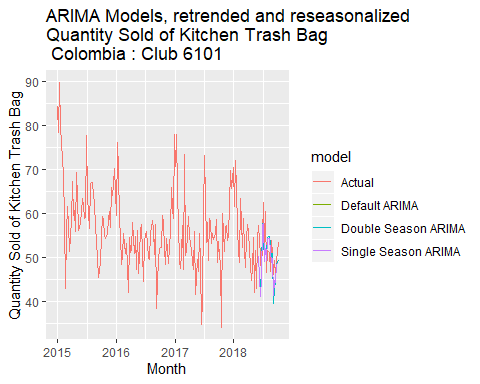
## Warning: Removed 179 rows containing missing values (geom\_path).



ggsave("Output/Countries/Colombia/Plots/simple\_log\_weekly\_pred1\_c1k.jpg", width = 7, height = 4)

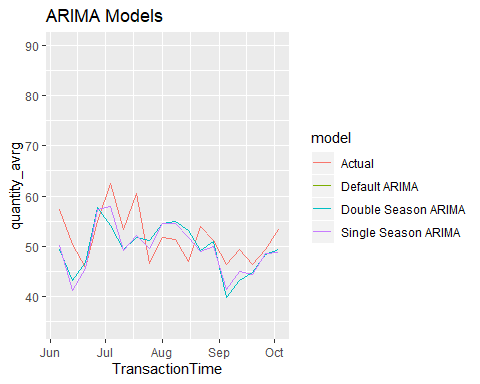
## Warning: Removed 179 rows containing missing values (geom\_path).

# plots: ARMIA models   
c1k\_week\_quantity\_models\_plot\_ARIMA\_log<-c1k\_week\_quantity\_models\_plot[(c1k\_week\_quantity\_models\_plot$model=="Actual"  
|c1k\_week\_quantity\_models\_plot$model=="Default ARIMA" |c1k\_week\_quantity\_models\_plot$model=="Double Season ARIMA"  
 |c1k\_week\_quantity\_models\_plot$model=="Single Season ARIMA"),]  
ggplot(data=c1k\_week\_quantity\_models\_plot\_ARIMA\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+ggtitle(paste("ARIMA Models, retrended and reseasonalized","\nQuantity Sold of", item, "\n",country, ": Club", club)) + xlab("Month") + ylab(paste("Quantity Sold of", item))



ggsave("Output/Countries/Colombia/Plots/ARIMA\_log\_weekly\_pred\_c1k.jpg", width = 8, height = 4)  
  
ggplot(data=c1k\_week\_quantity\_models\_plot\_ARIMA\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+xlim(max(c1k\_week$TransactionTime, na.rm = TRUE)-120, max(c1k\_week$TransactionTime, na.rm = TRUE))+ggtitle("ARIMA Models")

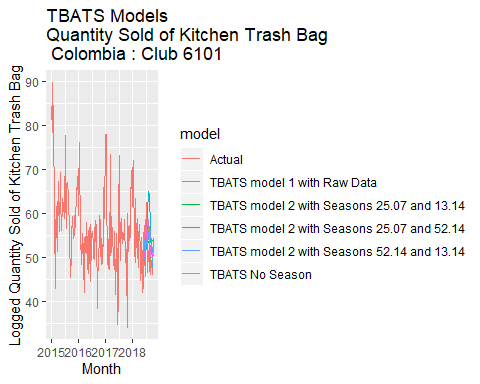
## Warning: Removed 179 rows containing missing values (geom\_path).



ggsave("Output/Countries/Colombia/Plots/ARIMA\_log\_weekly\_pred1\_c1k.jpg", width = 8, height = 4)

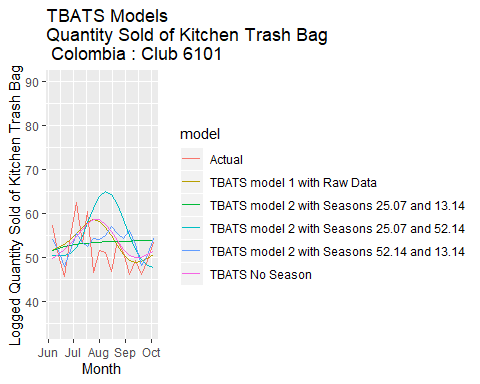
## Warning: Removed 179 rows containing missing values (geom\_path).

# plots: TBATS models  
c1k\_week\_quantity\_models\_plot\_TBATS\_log<-c1k\_week\_quantity\_models\_plot[(c1k\_week\_quantity\_models\_plot$model=="Actual"  
|c1k\_week\_quantity\_models\_plot$model=="TBATS model 1 with Raw Data" |c1k\_week\_quantity\_models\_plot$model==paste("TBATS model 2 with Seasons", season1, "and", season2)  
|c1k\_week\_quantity\_models\_plot$model==paste("TBATS model 2 with Seasons", season2, "and", season3)  
|c1k\_week\_quantity\_models\_plot$model==paste("TBATS model 2 with Seasons", season1, "and", season3)  
|c1k\_week\_quantity\_models\_plot$model=="TBATS No Season"),]  
ggplot(data=c1k\_week\_quantity\_models\_plot\_TBATS\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+ggtitle(paste("TBATS Models","\nQuantity Sold of", item, "\n",country, ": Club", club)) + xlab("Month") + ylab(paste("Logged Quantity Sold of", item))



ggsave("Output/Countries/Colombia/Plots/TBATS\_log\_weekly\_pred\_c1k.jpg", width = 8, height = 4)  
ggplot(data=c1k\_week\_quantity\_models\_plot\_TBATS\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+xlim(max(c1k\_week$TransactionTime, na.rm = TRUE)-120, max(c1k\_week$TransactionTime, na.rm = TRUE))+ggtitle(paste("TBATS Models","\nQuantity Sold of", item, "\n",country, ": Club", club)) + xlab("Month") + ylab(paste("Logged Quantity Sold of", item))

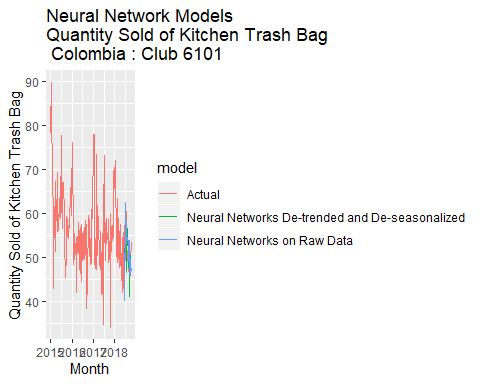
## Warning: Removed 179 rows containing missing values (geom\_path).



ggsave("Output/Countries/Colombia/Plots/TBATS\_log\_weekly\_pred1\_c1k.jpg", width = 5, height = 4)

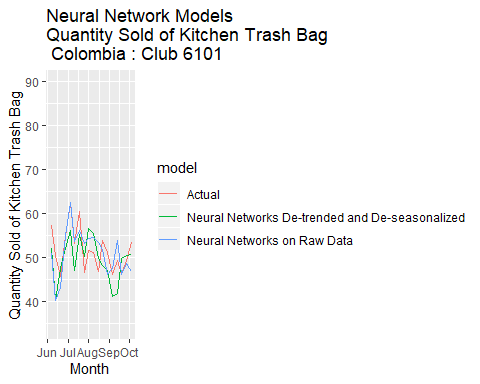
## Warning: Removed 179 rows containing missing values (geom\_path).

# plots: neural network  
c1k\_week\_quantity\_models\_plot\_nn\_log<-c1k\_week\_quantity\_models\_plot[(c1k\_week\_quantity\_models\_plot$model=="Actual"  
|c1k\_week\_quantity\_models\_plot$model=="Neural Networks De-trended and De-seasonalized"  
|c1k\_week\_quantity\_models\_plot$model=="Neural Networks on Raw Data"),]  
ggplot(data=c1k\_week\_quantity\_models\_plot\_nn\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+ggtitle(paste("Neural Network Models","\nQuantity Sold of", item, "\n",country, ": Club", club)) + xlab("Month") + ylab(paste("Quantity Sold of", item))



ggsave("Output/Countries/Colombia/Plots/nn\_log\_weekly\_pred\_c1k.jpg", width = 8, height = 4)  
ggplot(data=c1k\_week\_quantity\_models\_plot\_nn\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+xlim(max(c1k\_week$TransactionTime, na.rm = TRUE)-120, max(c1k\_week$TransactionTime, na.rm = TRUE))+ggtitle(paste("Neural Network Models","\nQuantity Sold of", item, "\n",country, ": Club", club)) + xlab("Month") + ylab(paste("Quantity Sold of", item))

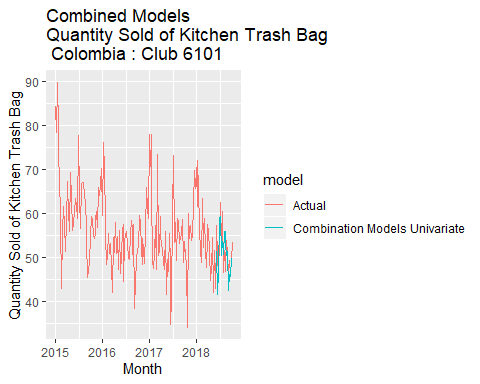
## Warning: Removed 179 rows containing missing values (geom\_path).



ggsave("Output/Countries/Colombia/Plots/nn\_log\_weekly\_pred1\_c1k.jpg", width = 6, height = 5)

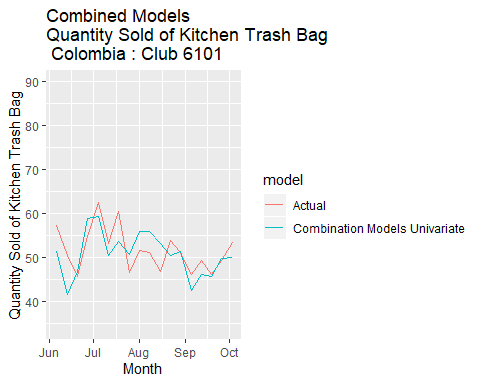
## Warning: Removed 179 rows containing missing values (geom\_path).

# plots: combined models   
c1k\_week\_quantity\_models\_plot\_combined\_log<-c1k\_week\_quantity\_models\_plot[(c1k\_week\_quantity\_models\_plot$model=="Actual"  
|c1k\_week\_quantity\_models\_plot$model=="Combination Models Univariate"),]  
ggplot(data=c1k\_week\_quantity\_models\_plot\_combined\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+ggtitle(paste("Combined Models","\nQuantity Sold of", item, "\n",country, ": Club", club)) + xlab("Month") + ylab(paste("Quantity Sold of", item))



ggsave("Output/Countries/Colombia/Plots/combined\_log\_weekly\_pred\_c1k.jpg", width = 8, height = 4)  
ggplot(data=c1k\_week\_quantity\_models\_plot\_combined\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+xlim(max(c1k\_week$TransactionTime, na.rm = TRUE)-120, max(c1k\_week$TransactionTime, na.rm = TRUE))+ggtitle(paste("Combined Models","\nQuantity Sold of", item, "\n",country, ": Club", club)) + xlab("Month") + ylab(paste("Quantity Sold of", item))

## Warning: Removed 179 rows containing missing values (geom\_path).



ggsave("Output/Countries/Colombia/Plots/combined\_log\_weekly\_pred1\_c1k.jpg", width = 6, height = 5)

## Warning: Removed 179 rows containing missing values (geom\_path).

# Machine learning setup

# This part of the code is based on https://rpubs.com/mattBrown88/TimeSeriesMachineLearning  
# feature engineering   
c1k\_week$month<-month(c1k\_week$TransactionTime)  
c1k\_week$month<-as.factor(c1k\_week$month)  
c1k\_week$log1p\_transaction\_avrg<-log1p(c1k\_week$transaction\_avrg) #  
c1k\_week$log1p\_members\_avrg<-log1p(c1k\_week$members\_avrg) #  
c1k\_week$log1p\_sales\_local\_avrg<-log1p(c1k\_week$sales\_local\_avrg) #  
c1k\_week$log1p\_sales\_usd\_avrg<-log1p(c1k\_week$sales\_usd\_avrg) #  
c1k\_week$log1p\_category\_sales\_local\_avrg<-log1p(c1k\_week$category\_sales\_local\_avrg) #  
c1k\_week$log1p\_quantity\_avrg<-log1p(c1k\_week$quantity\_avrg) #  
c1k\_week$log1p\_category\_sales\_local\_avrg<- log1p(c1k\_week$category\_sales\_local\_avrg)  
c1k\_week$log1p\_category\_sales\_usd\_avrg<-log1p(c1k\_week$category\_sales\_usd\_avrg)  
c1k\_week$log1p\_category\_quantity\_avrg<- log1p(c1k\_week$category\_quantity\_avrg)  
c1k\_week$log1p\_salePrice\_local\_avrg<- log1p(c1k\_week$salePrice\_local\_avrg)  
c1k\_week$log1p\_salePrice\_usd\_avrg<- log1p(c1k\_week$salePrice\_usd\_avrg)  
  
t\_v\_c1k<-c1k\_week  
t\_v\_c1k<- t\_v\_c1k[order(t\_v\_c1k$TransactionTime),]%>%ungroup()  
t\_v\_c1k<- feature\_engineering(t\_v\_c1k, c("quantity\_avrg", "transaction\_avrg", "members\_avrg", "sales\_local\_avrg", "exchange\_rate\_avrg", "sales\_usd\_avrg", "category\_sales\_local\_avrg", "category\_sales\_usd\_avrg", "category\_quantity\_avrg","salePrice\_local\_avrg","salePrice\_usd\_avrg", "log1p\_quantity\_avrg","log1p\_category\_sales\_local\_avrg","log1p\_salePrice\_local\_avrg","log1p\_salePrice\_usd\_avrg"))  
# creating training matrix  
t\_v\_c1k<- fastDummies::dummy\_cols(t\_v\_c1k, select\_columns = c("week\_number\_year", "week\_number", "month"))  
t\_v\_c1k$month<-NULL  
t\_v\_c1k$week\_number\_year<-NULL  
t\_v\_c1k$week\_number<-NULL  
  
train\_c1k<-t\_v\_c1k[1:(nrow(t\_v\_c1k)-len\_dec-len\_inv),]  
test\_c1k<-t\_v\_c1k[(nrow(t\_v\_c1k)-len\_dec-len\_inv+1):nrow(t\_v\_c1k),]  
c1k\_week\_quantity\_models\_train<-data.frame(Actual=train\_c1k$quantity\_avrg, TransactionTime=train\_c1k$TransactionTime)  
c1k\_week\_quantity\_models\_train<-c1k\_week\_quantity\_models\_train[rowSums(is.na(train\_c1k))==0,]

# XGB model

trainc1k\_XGB<-train\_c1k%>%select(-c("transaction\_avrg", "members\_avrg", "sales\_local\_avrg", "exchange\_rate\_avrg", "sales\_usd\_avrg", "category\_sales\_local\_avrg", "category\_sales\_usd\_avrg", "category\_quantity\_avrg","salePrice\_local\_avrg","salePrice\_usd\_avrg", "log1p\_quantity\_avrg", "log1p\_transaction\_avrg", "log1p\_members\_avrg", "log1p\_sales\_local\_avrg", "log1p\_sales\_usd\_avrg", "log1p\_category\_sales\_local\_avrg","log1p\_category\_sales\_usd\_avrg", "log1p\_category\_quantity\_avrg","log1p\_salePrice\_local\_avrg","log1p\_salePrice\_usd\_avrg","TransactionTime"))  
testc1k\_XGB<-test\_c1k%>%select(-c("transaction\_avrg", "members\_avrg", "sales\_local\_avrg", "exchange\_rate\_avrg", "sales\_usd\_avrg", "category\_sales\_local\_avrg", "category\_sales\_usd\_avrg", "category\_quantity\_avrg","salePrice\_local\_avrg","salePrice\_usd\_avrg", "log1p\_quantity\_avrg", "log1p\_transaction\_avrg", "log1p\_members\_avrg", "log1p\_sales\_local\_avrg", "log1p\_sales\_usd\_avrg", "log1p\_category\_sales\_local\_avrg","log1p\_category\_sales\_usd\_avrg", "log1p\_category\_quantity\_avrg","log1p\_salePrice\_local\_avrg","log1p\_salePrice\_usd\_avrg","TransactionTime"))  
  
trainTask <- makeRegrTask(data = trainc1k\_XGB, target = "quantity\_avrg")  
testTask <- makeRegrTask(data = testc1k\_XGB, target = "quantity\_avrg")  
  
xgb\_learner <- makeLearner(  
 "regr.xgboost",  
 predict.type = "response",  
 par.vals = list(  
 objective = "reg:squarederror",  
 eval\_metric = "rmse",  
 nrounds = 200  
 )  
)

## Warning in makeParam(id = id, type = "numeric", learner.param = TRUE, lower = lower, : NA used as a default value for learner parameter missing.  
## ParamHelpers uses NA as a special value for dependent parameters.

# Create a model  
xgb\_model <- mlr::train(xgb\_learner, task = trainTask)  
  
xgb\_params <- makeParamSet(  
 # The number of trees in the model (each one built sequentially)  
 makeIntegerParam("nrounds", lower = 100, upper = 500),  
 # number of splits in each tree  
 makeIntegerParam("max\_depth", lower = 1, upper = 10),  
 # "shrinkage" - prevents overfitting  
 makeNumericParam("eta", lower = .1, upper = .5),  
 # L2 regularization - prevents overfitting  
 makeNumericParam("lambda", lower = -1, upper = 0, trafo = function(x) 10^x)  
)  
control <- makeTuneControlRandom(maxit = 1)  
resample\_desc <- makeResampleDesc("CV", iters = 10)  
tuned\_params <- tuneParams(  
 learner = xgb\_learner,  
 task = trainTask,  
 resampling = resample\_desc,  
 par.set = xgb\_params,  
 control = control  
)

## [Tune] Started tuning learner regr.xgboost for parameter set:

## Type len Def Constr Req Tunable Trafo  
## nrounds integer - - 100 to 500 - TRUE -  
## max\_depth integer - - 1 to 10 - TRUE -  
## eta numeric - - 0.1 to 0.5 - TRUE -  
## lambda numeric - - -1 to 0 - TRUE Y

## With control class: TuneControlRandom

## Imputation value: Inf

## [Tune-x] 1: nrounds=212; max\_depth=1; eta=0.417; lambda=0.294

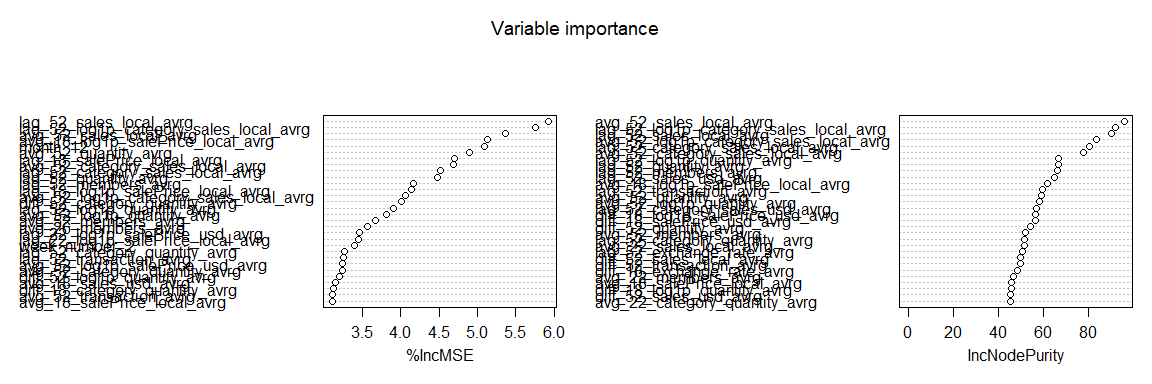
## [Tune-y] 1: mse.test.mean=57.6360687; time: 0.1 min

## [Tune] Result: nrounds=212; max\_depth=1; eta=0.417; lambda=0.294 : mse.test.mean=57.6360687

xgb\_tuned\_learner <- setHyperPars(  
 learner = xgb\_learner,  
 par.vals = tuned\_params$x  
)  
xgb\_model <- mlr::train(xgb\_tuned\_learner, trainTask)  
XGBoost\_pred <- predict(xgb\_model ,testTask)  
XGBoost\_pred\_train <- predict(xgb\_model ,trainTask)  
  
c1k\_week\_quantity\_models\_train<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models\_train, pred\_name = "XGBoost\_train", pred\_value = log(XGBoost\_pred\_train$data$response[rowSums(is.na(trainc1k\_XGB))==0]))  
c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "XGBoost", pred\_value = log(XGBoost\_pred$data$response))

# Random Forest 1

trainc1k\_RF<-train\_c1k%>%select(-c("transaction\_avrg", "members\_avrg", "sales\_local\_avrg", "exchange\_rate\_avrg", "sales\_usd\_avrg", "category\_sales\_local\_avrg", "category\_sales\_usd\_avrg", "category\_quantity\_avrg","salePrice\_local\_avrg","salePrice\_usd\_avrg", "log1p\_quantity\_avrg", "log1p\_transaction\_avrg", "log1p\_members\_avrg", "log1p\_sales\_local\_avrg", "log1p\_sales\_usd\_avrg", "log1p\_category\_sales\_local\_avrg","log1p\_category\_sales\_usd\_avrg", "log1p\_category\_quantity\_avrg","log1p\_salePrice\_local\_avrg","log1p\_salePrice\_usd\_avrg","TransactionTime"))  
testc1k\_RF<-test\_c1k%>%select(-c("transaction\_avrg", "members\_avrg", "sales\_local\_avrg", "exchange\_rate\_avrg", "sales\_usd\_avrg", "category\_sales\_local\_avrg", "category\_sales\_usd\_avrg", "category\_quantity\_avrg","salePrice\_local\_avrg","salePrice\_usd\_avrg", "log1p\_quantity\_avrg", "log1p\_transaction\_avrg", "log1p\_members\_avrg", "log1p\_sales\_local\_avrg", "log1p\_sales\_usd\_avrg", "log1p\_category\_sales\_local\_avrg","log1p\_category\_sales\_usd\_avrg", "log1p\_category\_quantity\_avrg","log1p\_salePrice\_local\_avrg","log1p\_salePrice\_usd\_avrg","TransactionTime"))  
  
RF\_1 <- randomForest(quantity\_avrg ~. , data = trainc1k\_RF,  
 ntree = 1000, mtry = 3, nodesize = 5, importance = TRUE, na.action = na.omit)  
  
varImpPlot(RF\_1, main = "Variable importance")



RF1\_pred<-predict(RF\_1, testc1k\_RF)  
  
RF1\_pred\_training<- predict(RF\_1, trainc1k\_RF)   
  
c1k\_week\_quantity\_models\_train<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models\_train, pred\_name = "RF1\_train", pred\_value = log(RF1\_pred\_training[rowSums(is.na(trainc1k\_XGB))==0]))  
c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "RF\_1", pred\_value = log(RF1\_pred))

# Random Forest 2

#Defining the Control  
trControl<- trainControl(method = "cv", number = 10, search = "grid")  
metric <- "RMSE"  
seed<- set.seed(156230)

## Step 1 Run a Default model

rf\_default<- caret::train(quantity\_avrg~ .  
 , data = trainc1k\_RF  
 , method = "rf", metric = "RMSE", trControl = trControl, na.action=na.exclude)  
rf\_default

## Random Forest   
##   
## 179 samples  
## 249 predictors  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 111, 112, 112, 111, 111, 112, ...   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared MAE   
## 2 6.683692 0.2527522 5.084373  
## 125 6.489195 0.2920607 5.010122  
## 249 6.527729 0.2862338 5.013442  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 125.

#RMSE was used to select the optimal model using the smallest value.

## Step 2 Search best mtry

#mtry is the number of variables available for splitting at each tree node  
#our train\_c1k\_RF has 250 total variables  
tuneGrid<- expand.grid(.mtry = seq(1, ncol(trainc1k\_RF), by=5))  
rf\_mtry<- caret::train(quantity\_avrg ~ .,   
 data = trainc1k\_RF,   
 method = "rf", metric = "RMSE", tuneGrid = tuneGrid, trControl = trControl, importance = TRUE, na.action=na.exclude)  
rf\_mtry

## Random Forest   
##   
## 179 samples  
## 249 predictors  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 112, 112, 112, 111, 112, 111, ...   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared MAE   
## 1 6.883028 0.3130084 5.172350  
## 6 6.463001 0.3361164 4.816446  
## 11 6.459938 0.3209486 4.874654  
## 16 6.450559 0.3220682 4.832082  
## 21 6.436809 0.3157919 4.837000  
## 26 6.394562 0.3199599 4.811466  
## 31 6.471042 0.3013551 4.913096  
## 36 6.476990 0.3037334 4.919074  
## 41 6.458852 0.3100378 4.903166  
## 46 6.431114 0.3116912 4.897221  
## 51 6.460485 0.3038405 4.900504  
## 56 6.439244 0.3018084 4.883869  
## 61 6.434038 0.3075800 4.919431  
## 66 6.485084 0.2943143 4.936775  
## 71 6.508754 0.2907438 4.959824  
## 76 6.491397 0.2929488 4.957963  
## 81 6.509473 0.2960098 4.960651  
## 86 6.470837 0.2988569 4.959631  
## 91 6.519334 0.2903668 4.970590  
## 96 6.502878 0.2937188 4.949699  
## 101 6.469234 0.3044884 4.940436  
## 106 6.513771 0.2933617 4.989416  
## 111 6.535130 0.2867969 4.986336  
## 116 6.528252 0.2860714 5.018130  
## 121 6.518558 0.2925833 5.035651  
## 126 6.517571 0.2863877 4.998616  
## 131 6.529244 0.2832787 5.000177  
## 136 6.556507 0.2778367 5.001433  
## 141 6.473572 0.2993046 4.944888  
## 146 6.502480 0.2875476 4.999082  
## 151 6.539472 0.2832881 5.018569  
## 156 6.544013 0.2867437 4.994949  
## 161 6.537175 0.2843534 5.010671  
## 166 6.530579 0.2897839 5.010901  
## 171 6.545182 0.2832081 5.038563  
## 176 6.530205 0.2830025 4.997330  
## 181 6.585599 0.2734805 5.065926  
## 186 6.528108 0.2931565 5.005990  
## 191 6.558788 0.2783870 5.021307  
## 196 6.531378 0.2872316 5.012974  
## 201 6.564261 0.2753188 5.035333  
## 206 6.571322 0.2794229 5.031339  
## 211 6.558794 0.2799183 5.012777  
## 216 6.558049 0.2778867 5.029900  
## 221 6.612389 0.2703709 5.070618  
## 226 6.566822 0.2799427 5.044617  
## 231 6.558948 0.2827003 5.010274  
## 236 6.542242 0.2788520 4.996055  
## 241 6.562306 0.2756142 5.055933  
## 246 6.534192 0.2790388 5.026686  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 26.

#RMSE was used to select the optimal model using the smallest value.  
  
best\_mtry<- rf\_mtry$bestTune$mtry #store the best value for mtry  
min(rf\_mtry$results$RMSE)

## [1] 6.394562

## Step 3 Search Best Maxnodes

store\_maxnode<- list() # create a list to find the optimal max of nodes  
tuneGrid<- expand.grid(.mtry= best\_mtry)  
for (maxnodes in c(1, 2, 3, 4, 5, 7, 8, 9, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70)){  
 set.seed(156230)  
 rf\_maxnode<- caret::train(quantity\_avrg ~ .,   
 data = trainc1k\_RF,   
 method = "rf", metric = "RMSE", tuneGrid = tuneGrid, trControl = trControl, importance = TRUE, maxnodes = maxnodes, nodesize = 4, na.action = na.exclude)  
 current\_iteration<- toString(maxnodes)  
 store\_maxnode[[current\_iteration]]<- rf\_maxnode  
}

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =  
## trainInfo, : There were missing values in resampled performance measures.

results\_node<- resamples(store\_maxnode)  
results\_node<-summary(results\_node)  
nnode\_optimal<-results\_node$models[results\_node$statistics$RMSE%>%as.data.frame()%>%select("Mean")%>%as.matrix()%>%as.numeric()%>%which.min()]%>%as.numeric()

## Step 4 Search the best ntrees

store\_maxtrees <- list()  
for (ntree in c(10, 20, 30, 40, 50, 100 , 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700)) {  
 set.seed(156230)  
 rf\_maxtrees <- caret::train(quantity\_avrg ~ .,   
 data = trainc1k\_RF,  
 method = "rf",  
 metric = "RMSE",  
 tuneGrid = tuneGrid,  
 trControl = trControl,  
 importance = TRUE,  
 maxnodes = nnode\_optimal,  
 ntree = ntree,  
 na.action = na.exclude)  
 key <- toString(ntree)  
 store\_maxtrees[[key]] <- rf\_maxtrees  
}  
results\_tree <- resamples(store\_maxtrees)  
results\_tree<-summary(results\_tree)  
ntrees\_optimal<-results\_tree$models[results\_tree$statistics$RMSE%>%as.data.frame()%>%select("Mean")%>%as.matrix()%>%as.numeric()%>%which.min()]%>%as.numeric()

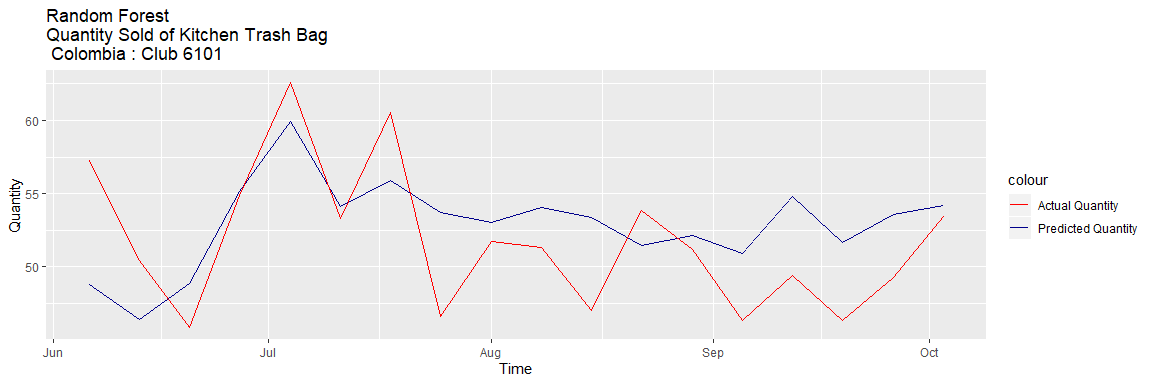
## Step 5 Run model with the best specifications found above

fit\_rf<- caret::train(quantity\_avrg ~ .,   
 data = trainc1k\_RF,  
 method = "rf",  
 metric = "RMSE",  
 tuneGrid = tuneGrid,  
 trControl = trControl,  
 importance = TRUE,  
 maxnodes = nnode\_optimal,  
 ntree = ntrees\_optimal,  
 na.action = na.exclude)  
  
fit\_rf

## Random Forest   
##   
## 179 samples  
## 249 predictors  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 112, 112, 112, 112, 112, 112, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 6.31946 0.3343251 4.922425  
##   
## Tuning parameter 'mtry' was held constant at a value of 26

## Step 6 Evaluate Model

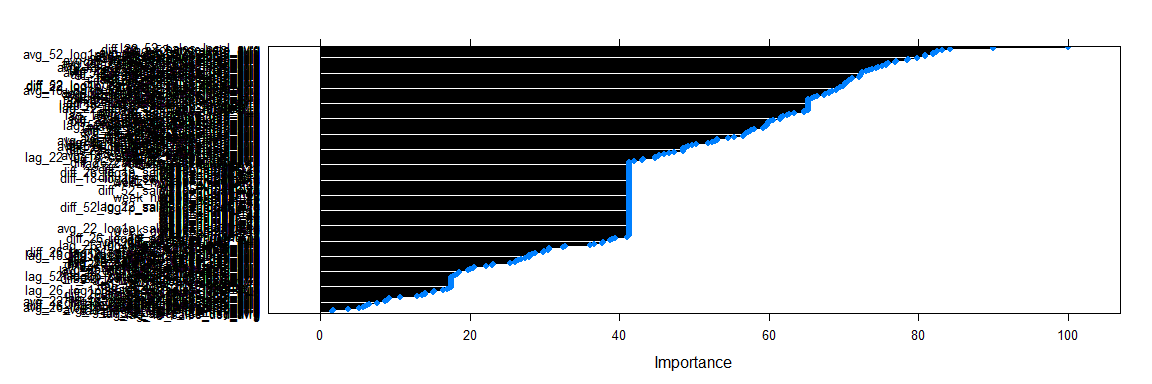
pred\_rf2<- predict(fit\_rf,test\_c1k, predict.all = TRUE)  
pred\_rf2\_train<- predict(fit\_rf,train\_c1k, predict.all = FALSE)  
pred\_rf2\_train<-merge(data.frame(week=1:nrow(train\_c1k)), data.frame(week=pred\_rf2\_train%>%names(), pred=pred\_rf2\_train), all.x = TRUE)  
  
# Visualize Results  
  
pred\_rf2\_dt<- data.table(Predicted\_Quantity = pred\_rf2, TransactionTime = test\_c1k$TransactionTime)  
  
ggplot() +  
 geom\_line(pred\_rf2\_dt, mapping = aes(TransactionTime, Predicted\_Quantity,color = "Predicted Quantity"))+  
 geom\_line(test\_c1k, mapping = aes(TransactionTime, quantity\_avrg, color = "Actual Quantity"))+  
 labs(x = "Time", y = "Quantity", title = paste("Random Forest","\nQuantity Sold of", item, "\n",country, ": Club", club)) +  
 scale\_color\_manual(values = c("Predicted Quantity" = 'darkblue', "Actual Quantity" = 'red'))



varImp(fit\_rf)

## rf variable importance  
##   
## only 20 most important variables shown (out of 249)  
##   
## Overall  
## lag\_52\_sales\_local\_avrg 100.00  
## diff\_22\_exchange\_rate\_avrg 90.00  
## lag\_52\_members\_avrg 84.22  
## avg\_26\_log1p\_quantity\_avrg 83.09  
## avg\_52\_log1p\_quantity\_avrg 82.49  
## avg\_52\_log1p\_category\_sales\_local\_avrg 82.33  
## avg\_18\_salePrice\_local\_avrg 82.01  
## diff\_22\_transaction\_avrg 81.89  
## avg\_52\_salePrice\_local\_avrg 80.90  
## avg\_26\_salePrice\_usd\_avrg 80.84  
## diff\_52\_transaction\_avrg 79.75  
## avg\_26\_log1p\_salePrice\_usd\_avrg 78.65  
## diff\_18\_category\_quantity\_avrg 78.48  
## avg\_18\_members\_avrg 76.99  
## avg\_22\_category\_sales\_local\_avrg 76.87  
## lag\_52\_sales\_usd\_avrg 75.97  
## lag\_22\_category\_sales\_local\_avrg 75.68  
## avg\_18\_log1p\_salePrice\_local\_avrg 75.64  
## lag\_22\_exchange\_rate\_avrg 75.19  
## lag\_26\_quantity\_avrg 74.60

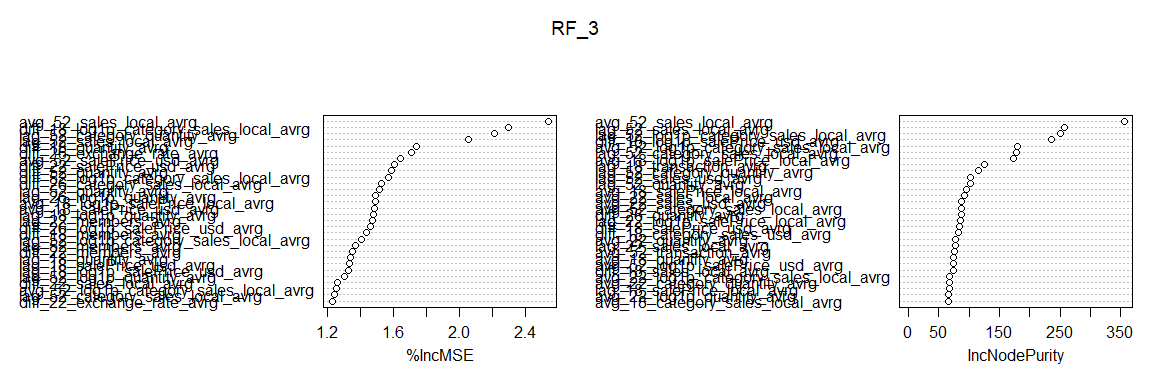
plot(varImp(fit\_rf)) #variable of importance



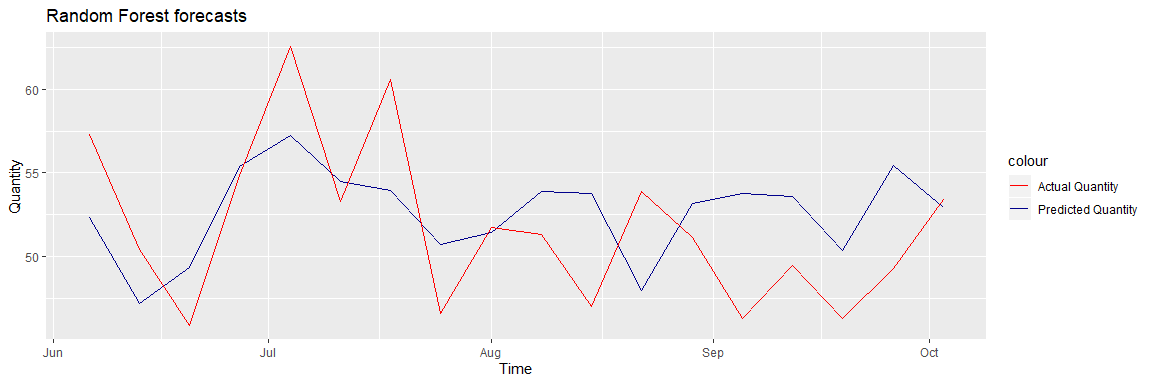
c1k\_week\_quantity\_models\_train<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models\_train, pred\_name = "RF2\_train", pred\_value = log(pred\_rf2\_train$pred[rowSums(is.na(trainc1k\_XGB))==0]))  
  
c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "RF\_2", pred\_value = log(pred\_rf2))

# Random Forest 3

RF\_3<- randomForest(quantity\_avrg ~ .,   
 data = trainc1k\_RF,  
 mtry = best\_mtry,  
 importance = TRUE,  
 maxnodes = nnode\_optimal,  
 ntree = ntrees\_optimal,  
 na.action = na.exclude)  
   
varImpPlot(RF\_3)



pred\_rf3 <- predict(RF\_3, test\_c1k)  
pred\_rf3<- data.table(Predicted\_Quantity = pred\_rf3, TransactionTime = test\_c1k$TransactionTime)  
pred\_rf3\_train <- predict(RF\_3, train\_c1k)  
  
ggplot() +  
 geom\_line(pred\_rf3, mapping = aes(TransactionTime, Predicted\_Quantity,color = "Predicted Quantity"))+  
 geom\_line(test\_c1k, mapping = aes(TransactionTime, quantity\_avrg, color = "Actual Quantity"))+  
 labs(x = "Time", y = "Quantity", title = "Random Forest forecasts") +  
 scale\_color\_manual(values = c("Predicted Quantity" = 'darkblue', "Actual Quantity" = 'red'))



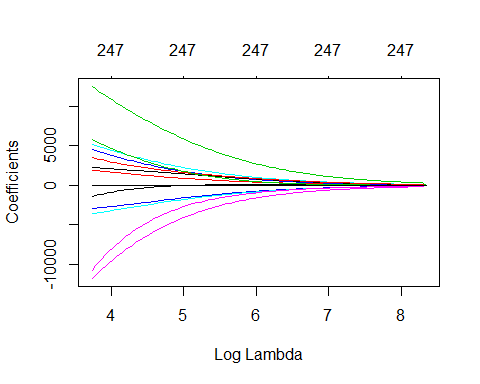
c1k\_week\_quantity\_models\_train<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models\_train, pred\_name = "RF3\_train", pred\_value = log(pred\_rf3\_train[rowSums(is.na(trainc1k\_XGB))==0]))  
c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "RF\_3", pred\_value = log(pred\_rf3$Predicted\_Quantity))

# Linear regressions: prepare datasets

#GLMNET does not use the formula method (y ~ x). So prior to modeling we need to create our feature and target set.  
  
train\_c1k\_GLMNET<-drop\_na(train\_c1k%>%select(-c("transaction\_avrg", "members\_avrg", "sales\_local\_avrg", "exchange\_rate\_avrg", "sales\_usd\_avrg", "category\_sales\_local\_avrg", "category\_sales\_usd\_avrg", "category\_quantity\_avrg", "salePrice\_local\_avrg","salePrice\_usd\_avrg", "log1p\_quantity\_avrg", "log1p\_transaction\_avrg", "log1p\_members\_avrg", "log1p\_sales\_local\_avrg", "log1p\_sales\_usd\_avrg", "log1p\_category\_sales\_local\_avrg","log1p\_category\_sales\_usd\_avrg", "log1p\_category\_quantity\_avrg","log1p\_salePrice\_local\_avrg","log1p\_salePrice\_usd\_avrg", "TransactionTime")))  
test\_c1k\_GLMNET<-drop\_na(test\_c1k%>%select(-c("transaction\_avrg", "members\_avrg", "sales\_local\_avrg", "exchange\_rate\_avrg", "sales\_usd\_avrg", "category\_sales\_local\_avrg", "category\_sales\_usd\_avrg", "category\_quantity\_avrg", "salePrice\_local\_avrg","salePrice\_usd\_avrg", "log1p\_quantity\_avrg", "log1p\_transaction\_avrg", "log1p\_members\_avrg", "log1p\_sales\_local\_avrg", "log1p\_sales\_usd\_avrg", "log1p\_category\_sales\_local\_avrg","log1p\_category\_sales\_usd\_avrg", "log1p\_category\_quantity\_avrg","log1p\_salePrice\_local\_avrg","log1p\_salePrice\_usd\_avrg", "TransactionTime")))  
  
y\_train<- data.matrix(train\_c1k\_GLMNET["quantity\_avrg"])  
x\_train<-data.matrix(subset(train\_c1k\_GLMNET, select=-c(quantity\_avrg)))  
  
y\_test<- data.matrix(test\_c1k\_GLMNET["quantity\_avrg"])  
x\_test<-data.matrix(subset(test\_c1k\_GLMNET, select=-c(quantity\_avrg)))

## Ridge

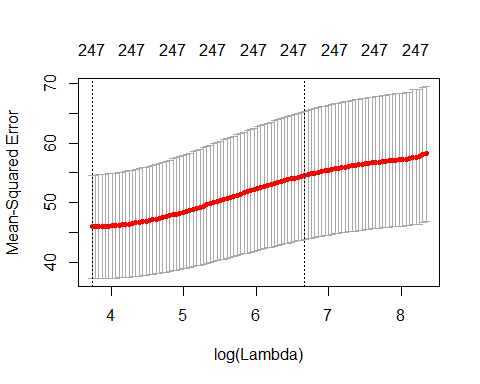
#Ridge is performed with alpha=0  
  
ridge <- glmnet(x\_train,y\_train,alpha = 0)  
  
plot(ridge, xvar = "lambda")



ridge$lambda %>% head()

## [1] 4198.643 4007.808 3825.647 3651.765 3485.787 3327.352

#Tuning to find the right value for lamda   
ridge\_cv <- cv.glmnet(x\_train,y\_train,alpha = 0)  
plot(ridge\_cv)



# as we constrain our coefficients with log ( λ ) ≥ 0 penalty, the MSE rises considerably. The numbers at the top of the plot (38) just refer to the number of variables in the model. Ridge regression does not force any variables to exactly zero so all features will remain in the model   
  
#The first and second vertical dashed lines represent the λ value with the minimum MSE and the largest λ value within one standard error of the minimum MSE.   
#extract our minimum and one standard error MSE and λ values  
min(ridge\_cv$cvm) #minimum MSE

## [1] 45.90906

ridge\_cv$lambda.min #lambda for this minimum MSE

## [1] 41.98643

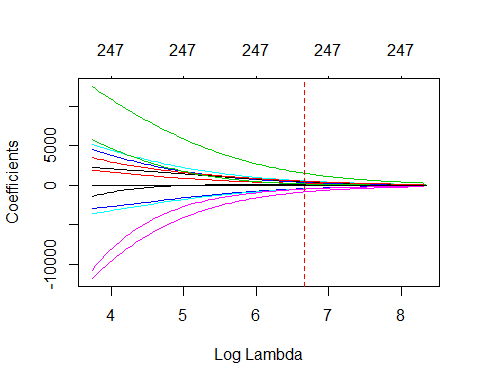
ridge\_cv$cvm[ridge\_cv$lambda == ridge\_cv$lambda.1se] # 1 st.error of min MSE

## [1] 54.56092

ridge\_cv$lambda.1se # lambda for this MSE

## [1] 786.749

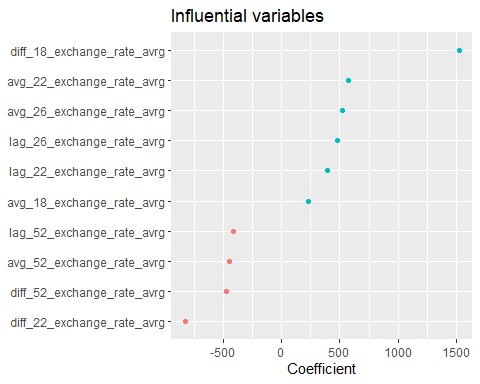
ridge\_min <- glmnet(x\_train,y\_train, alpha = 0)  
plot(ridge\_min, xvar = "lambda")  
abline(v = log(ridge\_cv$lambda.1se), col = "red", lty = "dashed")



#Most Influential Feautures to predict accuracy  
coef(ridge\_cv, s = "lambda.1se") %>%  
 tidy() %>%  
 filter(row != "(Intercept)") %>%  
 top\_n(10, wt = abs(value)) %>%  
 ggplot(aes(value, reorder(row, value), color = value > 0)) +  
 geom\_point(show.legend = FALSE) +  
 ggtitle("Influential variables") +  
 xlab("Coefficient") +  
 ylab(NULL)

## Warning: 'tidy.dgCMatrix' is deprecated.  
## See help("Deprecated")

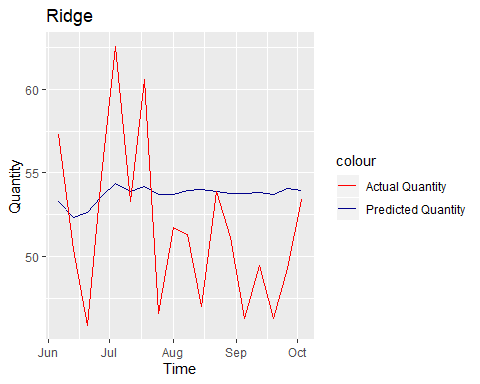
## Warning: 'tidy.dgTMatrix' is deprecated.  
## See help("Deprecated")



min(ridge\_cv$cvm)

## [1] 45.90906

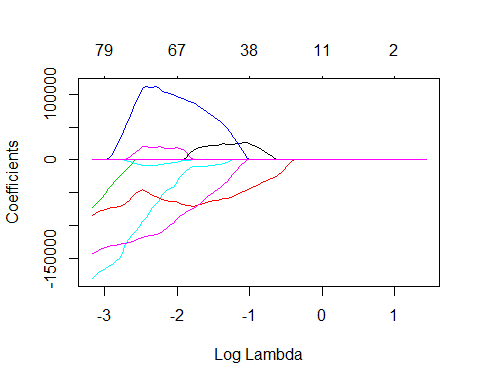
#Ridge model will retian all variables. Therefore, a ridge model is good only if we believe that we need to retain all features in the model yet reduce the noise that less influential variable smay create and minimize collinearity. Ridge doesn't perform feature selection   
##PREDICTING  
ridge\_pred<- predict(ridge\_cv, s=ridge\_cv$lambda.1se, x\_test, type = "response")  
ridge\_pred\_train<-predict(ridge\_cv, s=ridge\_cv$lambda.1se, x\_train, type = "response")  
ridge\_pred\_train<-merge(data.frame(week=1:nrow(train\_c1k)), data.frame(week=ridge\_pred\_train%>%rownames()%>%as.numeric(), pred=ridge\_pred\_train), all.x = TRUE)  
colnames(ridge\_pred\_train)<-c("week", "pred")  
#Graph  
pred\_ridge<- data.frame(Predicted\_Quantity = ridge\_pred, TransactionTime = test\_c1k$TransactionTime)  
colnames(pred\_ridge)<-c("Predicted\_Quantity", "TransactionTime")  
ggplot() +  
 geom\_line(pred\_ridge, mapping = aes(TransactionTime, Predicted\_Quantity, color = "Predicted Quantity"))+  
 geom\_line(test\_c1k, mapping = aes(TransactionTime, quantity\_avrg, color = "Actual Quantity"))+  
 labs(x = "Time", y = "Quantity", title = "Ridge") + scale\_color\_manual(values = c("Predicted Quantity" = 'darkblue', "Actual Quantity" = 'red'))



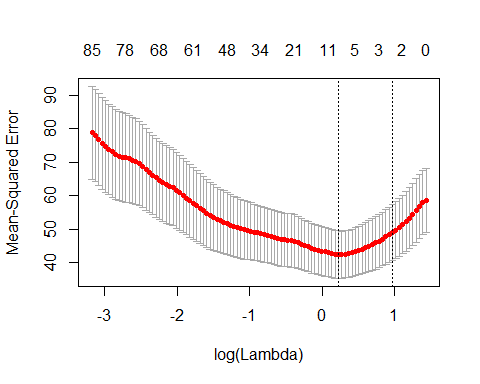
c1k\_week\_quantity\_models\_train<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models\_train, pred\_name = "Ridge\_train", pred\_value = log((ridge\_pred\_train[rowSums(is.na(trainc1k\_XGB))==0,])%>%select(pred)%>%unlist()))  
  
c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "Ridge", pred\_value = log(ridge\_pred))

## Lasso

lasso <- glmnet(x\_train,y\_train,alpha = 1)  
  
plot(lasso, xvar = "lambda")



# when log(λ)=− 3 all 8 variables are in the model, when log(λ)=−1 2 variables are retained  
   
#Tuning to find the right value for lambda   
lasso\_cv <- cv.glmnet(x\_train,y\_train,alpha = 1)  
plot(lasso\_cv) #We see that we can minimize our MSE by applying approximately −9≤log(λ)≤−4



#extract our minimum and one standard error MSE and λ values  
min(lasso\_cv$cvm) #minimum MSE

## [1] 42.44072

lasso\_cv$lambda.min #lambda for this minimum MSE

## [1] 1.252727

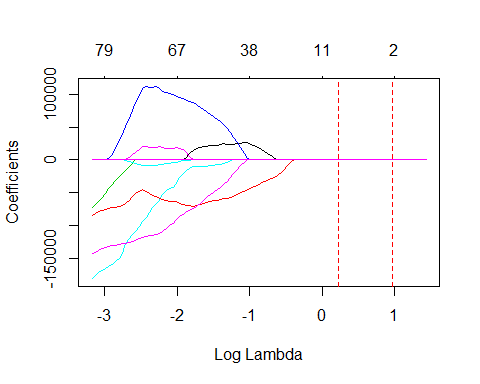
lasso\_cv$cvm[lasso\_cv$lambda == lasso\_cv$lambda.1se] # 1 st.error of min MSE

## [1] 49.05256

lasso\_cv$lambda.1se # lambda for this MSE

## [1] 2.63687

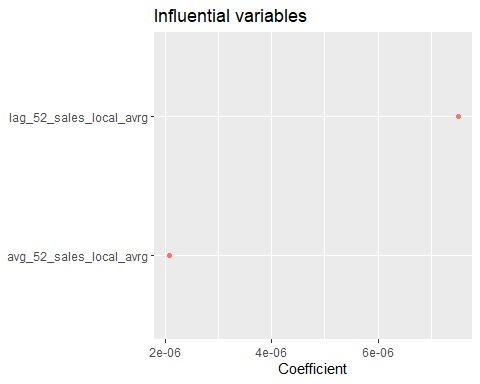
lasso\_min <- glmnet(x\_train,y\_train, alpha = 1)  
plot(lasso\_min, xvar = "lambda")  
abline(v = log(lasso\_cv$lambda.min), col = "red", lty = "dashed")  
abline(v = log(lasso\_cv$lambda.1se), col = "red", lty = "dashed")



#Most Influential Feautures to predict accuracy  
coef(lasso\_cv, s = "lambda.1se") %>%  
 tidy() %>%  
 filter(row != "(Intercept)") %>%  
 ggplot(aes(value, reorder(row, value), color = value > 0)) +  
 geom\_point(show.legend = FALSE) +  
 ggtitle("Influential variables") +  
 xlab("Coefficient") +  
 ylab(NULL) # The 3 most influential variables are "Exchange Rate" "Sales\_Local" and "Lag Quantity"

## Warning: 'tidy.dgCMatrix' is deprecated.  
## See help("Deprecated")

## Warning: 'tidy.dgTMatrix' is deprecated.  
## See help("Deprecated")



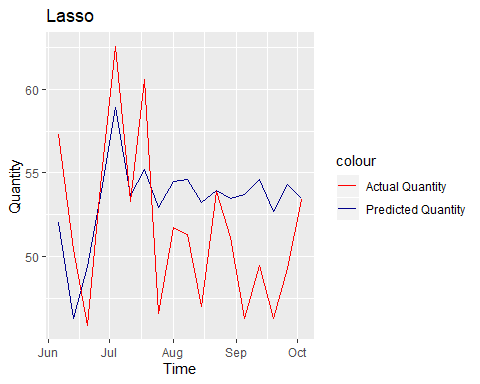
#Minimum RIDGE MSE  
min(ridge\_cv$cvm)

## [1] 45.90906

#minimum LASSO MSE   
min(lasso\_cv$cvm)

## [1] 42.44072

##PREDICTING  
lasso\_pred<- predict(lasso, s=lasso\_cv$lambda.min, x\_test, type = "response")  
lasso\_pred\_train<-predict(lasso, s=lasso\_cv$lambda.min, x\_train, type = "response")  
results\_train<-merge(data.frame(week=1:nrow(train\_c1k)), data.frame(week=lasso\_pred\_train%>%rownames()%>%as.numeric(), pred=lasso\_pred\_train), all.x = TRUE)  
colnames(results\_train)<-c("week", "pred")  
#Graph  
pred\_lasso<- data.table(Predicted\_Quantity = lasso\_pred, TransactionTime = test\_c1k$TransactionTime)  
colnames(pred\_lasso)<-c("Predicted\_Quantity", "TransactionTime")  
ggplot() +  
 geom\_line(pred\_lasso, mapping = aes(TransactionTime, Predicted\_Quantity, color = "Predicted Quantity"))+  
 geom\_line(test\_c1k, mapping = aes(TransactionTime, quantity\_avrg, color = "Actual Quantity"))+  
 labs(x = "Time", y = "Quantity", title = "Lasso") + scale\_color\_manual(values = c("Predicted Quantity" = 'darkblue', "Actual Quantity" = 'red'))



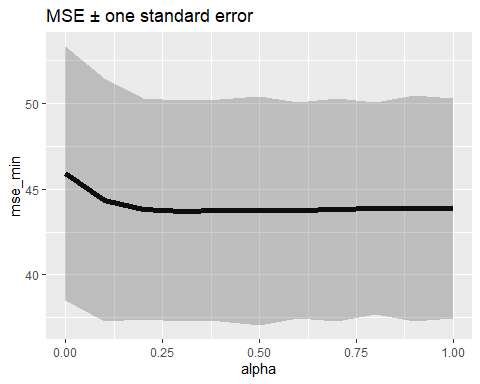
c1k\_week\_quantity\_models\_train<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models\_train, pred\_name = "Lasso\_train", pred\_value = log(lasso\_pred\_train[rowSums(is.na(trainc1k\_XGB))==0]))  
  
c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "Lasso", pred\_value = log(lasso\_pred))

## Elastic Net

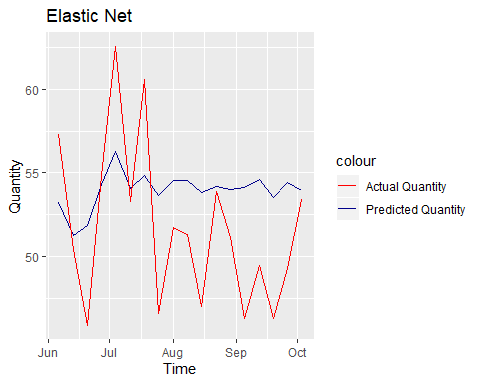
#http://uc-r.github.io/regularized\_regression   
  
#TUNING  
  
#Tune λ and the alpha parameters.  
# maintain the same folds across all models  
fold\_id <- sample(1:10, size = length(y\_train), replace=TRUE)  
  
# search across a range of alphas  
tuning\_grid <- tibble::tibble(  
 alpha = seq(0, 1, by = .1),  
 mse\_min = NA,  
 mse\_1se = NA,  
 lambda\_min = NA,  
 lambda\_1se = NA  
)  
  
#Now we can iterate over each alpha value, apply a CV elastic net, and extract the minimum and one standard error MSE values and their respective λ values.  
for(i in seq\_along(tuning\_grid$alpha)) {  
   
 # fit CV model for each alpha value  
 fit <- cv.glmnet(x\_train, y\_train, alpha = tuning\_grid$alpha[i], foldid = fold\_id)  
   
 # extract MSE and lambda values  
 tuning\_grid$mse\_min[i] <- fit$cvm[fit$lambda == fit$lambda.min]  
 tuning\_grid$mse\_1se[i] <- fit$cvm[fit$lambda == fit$lambda.1se]  
 tuning\_grid$lambda\_min[i] <- fit$lambda.min  
 tuning\_grid$lambda\_1se[i] <- fit$lambda.1se  
}  
  
tuning\_grid

## # A tibble: 11 x 5  
## alpha mse\_min mse\_1se lambda\_min lambda\_1se  
## <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 0 45.9 53.3 42.0 494.   
## 2 0.1 44.4 51.5 8.24 21.9   
## 3 0.2 43.8 50.3 4.74 10.9   
## 4 0.3 43.7 50.2 3.31 7.64  
## 5 0.4 43.8 50.2 2.85 6.01  
## 6 0.5 43.7 50.4 2.28 5.03  
## 7 0.6 43.8 50.1 1.99 4.20  
## 8 0.7 43.8 50.3 1.71 3.77  
## 9 0.8 43.9 50.1 1.49 3.30  
## 10 0.9 43.9 50.5 1.39 3.07  
## 11 1 43.9 50.3 1.25 2.76

#plot the MSE   
elastic\_tuning<-tuning\_grid %>%mutate(se = mse\_1se - mse\_min)  
  
elastic\_tuning%>%  
 ggplot(aes(alpha, mse\_min)) +  
 geom\_line(size = 2) +  
 geom\_ribbon(aes(ymax = mse\_min + se, ymin = mse\_min - se), alpha = .25) +  
 ggtitle("MSE ± one standard error")



alpha\_optimal<-(elastic\_tuning%>%as.data.frame()%>%select("alpha"))[which.min(elastic\_tuning%>%as.data.frame()%>%select("mse\_1se")%>%unlist()%>%as.numeric()),1]  
#advantage of the elastic net model is that it enables effective regularization via the ridge penalty with the feature selection characteristics of the lasso penalty  
  
  
elastic\_cv<-cv.glmnet(x\_train,y\_train,alpha = alpha\_optimal)  
  
elastic\_pred<- predict(elastic\_cv, s=elastic\_cv$lambda.1se, x\_test, type = "response")  
elastic\_pred\_train<- predict(elastic\_cv, s=elastic\_cv$lambda.1se, x\_train, type = "response")  
results\_train<-merge(data.frame(week=1:nrow(train\_c1k)), data.frame(week=elastic\_pred\_train%>%rownames()%>%as.numeric(), pred=elastic\_pred\_train), all.x = TRUE)  
colnames(results\_train)<-c("week", "pred")  
#Graph  
pred\_elastic\_cv<- data.table(Predicted\_Quantity = elastic\_pred, TransactionTime = test\_c1k$TransactionTime)  
colnames(pred\_elastic\_cv)<-c("Predicted\_Quantity", "TransactionTime")  
  
ggplot() +  
 geom\_line(pred\_elastic\_cv, mapping = aes(TransactionTime, Predicted\_Quantity, color = "Predicted Quantity"))+  
 geom\_line(test\_c1k, mapping = aes(TransactionTime, quantity\_avrg, color = "Actual Quantity"))+  
 labs(x = "Time", y = "Quantity", title = "Elastic Net") + scale\_color\_manual(values = c("Predicted Quantity" = 'darkblue', "Actual Quantity" = 'red'))



c1k\_week\_quantity\_models\_train<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models\_train, pred\_name = "elastic\_train", pred\_value = log(elastic\_pred\_train[rowSums(is.na(trainc1k\_XGB))==0]))  
  
c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "elastic", pred\_value = log(elastic\_pred))

# Combination of models: ml

# Best two models with total prediction  
tsmodels<-c("Average\_Method","Naive\_Method","Seasonal\_Naive\_Method","Drift\_Method","Simple\_Arima\_1","Arima\_Seasons1\_2", "Arima\_Fourier\_AIC","fc\_TBATS\_Raw\_1","fc\_TBATS\_Season1\_2","fc\_TBATS\_Season2\_3","fc\_TBATS\_Season1\_3","fc\_TBATS\_NoSeason","fc\_NN\_1","fc\_NN\_Raw")  
tsmodels\_ml<-c("XGBoost", "RF\_1", "RF\_2","RF\_3", "Ridge", "Lasso", "elastic")  
best\_two\_total\_ml<-tsmodels\_ml[rank(mget(paste0("RMSE\_", tsmodels\_ml, "\_total"))%>%as.numeric())<=2]  
# Best two models with target  
best\_two\_target\_ml<-tsmodels\_ml[rank(mget(paste0("RMSE\_", tsmodels\_ml, "\_target"))%>%as.numeric())<=2]  
#   
Combination\_ml<-(1/4\*(c1k\_week\_quantity\_models%>%select(best\_two\_total\_ml[1])+c1k\_week\_quantity\_models%>%select(best\_two\_total\_ml[2])+c1k\_week\_quantity\_models%>%select(best\_two\_target\_ml[1])+c1k\_week\_quantity\_models%>%select(best\_two\_target\_ml[2])))%>%unlist()  
  
c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "Combination\_ml", pred\_value = log(Combination\_ml))

# Combining best traditional and machine learning models

# Best two models with total prediction  
best\_two\_total<-tsmodels[rank(mget(paste0("RMSE\_", tsmodels, "\_total"))%>%as.numeric())<=2]  
#   
Combination\_uni\_ml<-(1/2\*(c1k\_week\_quantity\_models%>%select(best\_two\_total\_ml[1])+c1k\_week\_quantity\_models%>%select(best\_two\_total\_ml[2])+c1k\_week\_quantity\_models%>%select(best\_two\_total[1])+c1k\_week\_quantity\_models%>%select(best\_two\_total[2])))%>%unlist()  
  
c1k\_week\_quantity\_models<-performance\_index\_raw(df\_Actual = c1k\_week\_quantity\_models, pred\_name = "Combination\_uni\_ml", pred\_value = log(Combination\_uni\_ml))  
best\_two\_total

## [1] "Naive\_Method" "Drift\_Method"

tsmodels<-c("Average\_Method","Naive\_Method","Seasonal\_Naive\_Method","Drift\_Method","Simple\_Arima\_1","Arima\_Seasons1\_2", "Arima\_Fourier\_AIC","fc\_TBATS\_Raw\_1","fc\_TBATS\_Season1\_2","fc\_TBATS\_Season2\_3","fc\_TBATS\_Season1\_3","fc\_TBATS\_NoSeason","fc\_NN\_1","fc\_NN\_Raw", "Combination\_uni")  
  
best\_two\_total\_ml

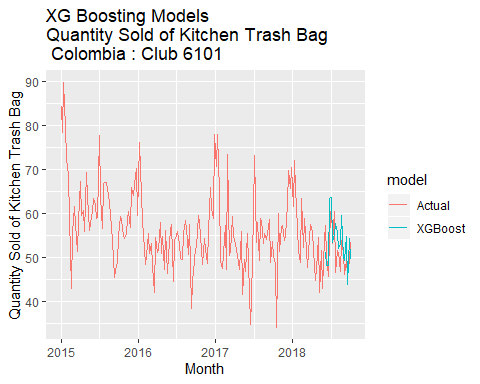
## [1] "RF\_1" "RF\_2"

# Summarize performance of current models

tsmodels\_ml<-c("XGBoost", "RF\_1", "RF\_2","RF\_3", "Ridge", "Lasso", "elastic", "Combination\_ml", "Combination\_uni\_ml")  
indices<-c("MEAN", "RMSE", "MAE", "MPE", "MAPE", "MASE")  
performance\_week\_total\_ml<-matrix(nrow = length(tsmodels\_ml), ncol = length(indices))  
for (i in 1:length(tsmodels\_ml)){  
 for (j in 1:length(indices)){  
 performance\_week\_total\_ml[i, j]=get(paste0(indices[j], "\_", tsmodels\_ml[i], "\_total"))  
 }  
}  
performance\_week\_total\_ml<-as.data.frame(performance\_week\_total\_ml)  
colnames(performance\_week\_total\_ml)<-indices  
rownames(performance\_week\_total\_ml)<-tsmodels\_ml  
write.csv(x = performance\_week\_total\_ml, file = "Output/Countries/Colombia/Tables/performance\_week\_total\_ml\_c1k.csv")  
  
performance\_week\_target\_ml<-matrix(nrow = length(tsmodels\_ml), ncol = length(indices))  
for (i in 1:length(tsmodels\_ml)){  
 for (j in 1:length(indices)){  
 performance\_week\_target\_ml[i, j]=get(paste0(indices[j], "\_", tsmodels\_ml[i], "\_target"))  
 }  
}  
performance\_week\_target\_ml<-as.data.frame(performance\_week\_target\_ml)  
colnames(performance\_week\_target\_ml)<-indices  
rownames(performance\_week\_target\_ml)<-tsmodels\_ml  
write.csv(x = performance\_week\_target\_ml, file = "Output/Countries/Colombia/Tables/performance\_week\_target\_ml\_c1k.csv")  
  
tsmodels\_ml\_train<-c("XGBoost\_train", "RF1\_train", "RF2\_train","RF3\_train", "Ridge\_train", "Lasso\_train", "elastic\_train")  
indices<-c("MEAN", "RMSE", "MAE", "MPE", "MAPE", "MASE")  
performance\_week\_total\_ml\_train<-matrix(nrow = length(tsmodels\_ml\_train), ncol = length(indices))  
for (i in 1:length(tsmodels\_ml\_train)){  
 for (j in 1:length(indices)){  
 performance\_week\_total\_ml\_train[i, j]=get(paste0(indices[j], "\_", tsmodels\_ml\_train[i], "\_total"))  
 }  
}  
performance\_week\_total\_ml\_train<-as.data.frame(performance\_week\_total\_ml\_train)  
colnames(performance\_week\_total\_ml\_train)<-indices  
rownames(performance\_week\_total\_ml\_train)<-tsmodels\_ml\_train  
write.csv(x = performance\_week\_total\_ml\_train, file = "Output/Countries/Colombia/Tables/performance\_week\_total\_ml\_train\_c1k.csv")

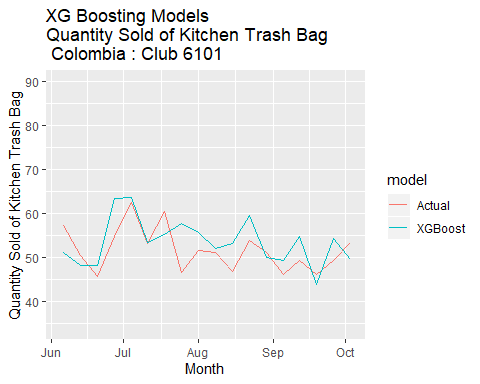
# Visualization of machine learning models

# re-trended and re-seasonalized plots  
  
c1k\_week\_quantity\_models\_plot<-gather(c1k\_week\_quantity\_models%>%select(-c("Actual", "TransactionYear", "week\_number")), model, quantity\_avrg, Average\_Method:Combination\_uni\_ml, factor\_key=TRUE)  
c1k\_week\_quantity\_models\_plot$TransactionTime<-as.Date(c1k\_week\_quantity\_models\_plot$TransactionTime)  
c1k\_week\_quantity\_models\_plot$model<-as.character(c1k\_week\_quantity\_models\_plot$model)  
c1k\_week\_quantity\_models\_plot$quantity\_avrg<-as.numeric(c1k\_week\_quantity\_models\_plot$quantity\_avrg)  
  
c1k\_week\_int\_Actual<-c1k\_week%>%ungroup()%>%select(c("TransactionTime", "quantity\_avrg"))  
c1k\_week\_int\_Actual$model<-"Actual"  
c1k\_week\_int\_Actual$TransactionTime<-as.Date(c1k\_week\_int\_Actual$TransactionTime)  
c1k\_week\_int\_Actual$quantity\_avrg<-as.numeric(c1k\_week\_int\_Actual$quantity\_avrg)  
c1k\_week\_quantity\_models\_plot<-rbind(c1k\_week\_int\_Actual, c1k\_week\_quantity\_models\_plot)  
  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Average\_Method"]<-"Average Model"  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Naive\_Method"]<-"Naive Model"  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Seasonal\_Naive\_Method"]<-"Seasonal Naive Model"  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Drift\_Method"]<-"Drift Model"  
  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Simple\_Arima\_1"]<-"Default ARIMA"  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Arima\_Seasons1\_2"]<-"Double Season ARIMA"  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="Arima\_Fourier\_AIC"]<-"Single Season ARIMA"  
  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="fc\_TBATS\_Raw\_1"]<-"TBATS model 1 with Raw Data"  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="fc\_TBATS\_Season1\_2"]<-paste("TBATS model 2 with Seasons", season1, "and", season2)  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="fc\_TBATS\_Season2\_3"]<-paste("TBATS model 2 with Seasons", season2, "and", season3)  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="fc\_TBATS\_Season1\_3"]<-paste("TBATS model 2 with Seasons", season1, "and", season3)  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="fc\_TBATS\_NoSeason"]<-"TBATS No Season"  
  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="fc\_NN\_1"]<-"Neural Networks De-trended and De-seasonalized"  
c1k\_week\_quantity\_models\_plot$model[c1k\_week\_quantity\_models\_plot$model=="fc\_NN\_Raw"]<-"Neural Networks on Raw Data"  
  
# plots: XGBoost models   
c1k\_week\_quantity\_models\_plot\_XGBoost\_log<-c1k\_week\_quantity\_models\_plot[(c1k\_week\_quantity\_models\_plot$model=="Actual"  
|c1k\_week\_quantity\_models\_plot$model=="XGBoost"),]  
ggplot(data=c1k\_week\_quantity\_models\_plot\_XGBoost\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+ggtitle(paste("XG Boosting Models","\nQuantity Sold of", item, "\n",country, ": Club", club))+ xlab("Month") + ylab(paste("Quantity Sold of", item))



ggsave("Output/Countries/Colombia/Plots/XGBoost\_log\_weekly\_pred\_c1k.jpg", width = 8, height = 4)   
ggplot(data=c1k\_week\_quantity\_models\_plot\_XGBoost\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+xlim(max(c1k\_week$TransactionTime, na.rm = TRUE)-120, max(c1k\_week$TransactionTime, na.rm = TRUE))+ggtitle(paste("XG Boosting Models","\nQuantity Sold of", item, "\n",country, ": Club", club))+ xlab("Month") + ylab(paste("Quantity Sold of", item))

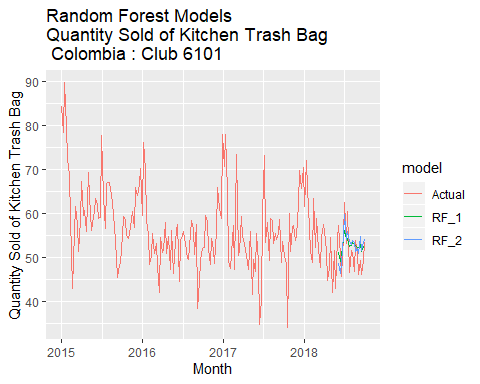
## Warning: Removed 179 rows containing missing values (geom\_path).



ggsave("Output/Countries/Colombia/Plots/XGBoost\_log\_weekly\_pred1.jpg", width = 8, height = 4)

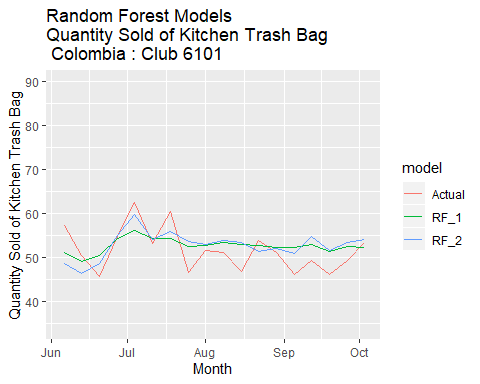
## Warning: Removed 179 rows containing missing values (geom\_path).

# plots: RF models   
c1k\_week\_quantity\_models\_plot\_RF\_log<-c1k\_week\_quantity\_models\_plot[(c1k\_week\_quantity\_models\_plot$model=="Actual"  
|c1k\_week\_quantity\_models\_plot$model=="RF\_1"  
|c1k\_week\_quantity\_models\_plot$model=="RF\_2"),]  
ggplot(data=c1k\_week\_quantity\_models\_plot\_RF\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+ggtitle(paste("Random Forest Models","\nQuantity Sold of", item, "\n",country, ": Club", club))+ xlab("Month") + ylab(paste("Quantity Sold of", item))



ggsave("Output/Countries/Colombia/Plots/RF\_log\_weekly\_pred\_c1k.jpg", width = 8, height = 4)  
ggplot(data=c1k\_week\_quantity\_models\_plot\_RF\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+xlim(max(c1k\_week$TransactionTime, na.rm = TRUE)-120, max(c1k\_week$TransactionTime, na.rm = TRUE))+ggtitle(paste("Random Forest Models","\nQuantity Sold of", item, "\n",country, ": Club", club))+ xlab("Month") + ylab(paste("Quantity Sold of", item))

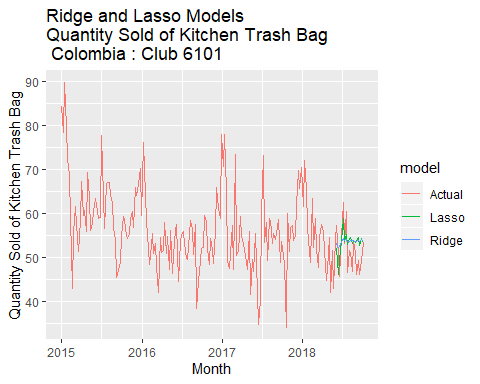
## Warning: Removed 179 rows containing missing values (geom\_path).



ggsave("Output/Countries/Colombia/Plots/RF\_log\_weekly\_pred1\_c1k.jpg", width = 8, height = 4)

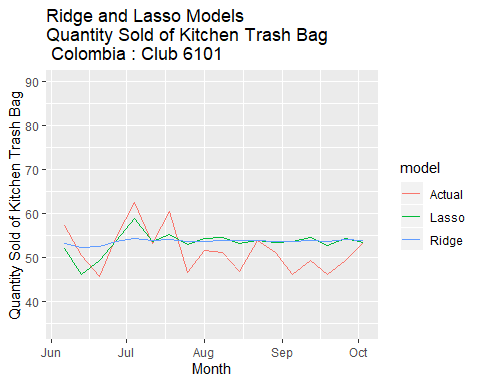
## Warning: Removed 179 rows containing missing values (geom\_path).

# plots: Ridge and LASSO models   
c1k\_week\_quantity\_models\_plot\_RidgeLASSO\_log<-c1k\_week\_quantity\_models\_plot[(c1k\_week\_quantity\_models\_plot$model=="Actual"  
|c1k\_week\_quantity\_models\_plot$model=="Ridge"  
|c1k\_week\_quantity\_models\_plot$model=="Lasso"),]  
ggplot(data=c1k\_week\_quantity\_models\_plot\_RidgeLASSO\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+ggtitle(paste("Ridge and Lasso Models","\nQuantity Sold of", item, "\n",country, ": Club", club))+ xlab("Month") + ylab(paste("Quantity Sold of", item))



ggsave("Output/Countries/Colombia/Plots/RidgeLASSO\_log\_weekly\_pred\_c1k.jpg", width = 8, height = 4)  
ggplot(data=c1k\_week\_quantity\_models\_plot\_RidgeLASSO\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+ggtitle(paste("Ridge and Lasso Models","\nQuantity Sold of", item, "\n",country, ": Club", club))+ xlab("Month") + ylab(paste("Quantity Sold of", item))+xlim(max(c1k\_week$TransactionTime, na.rm = TRUE)-120, max(c1k\_week$TransactionTime, na.rm = TRUE))

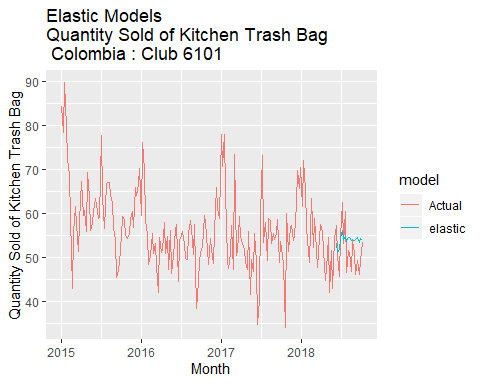
## Warning: Removed 179 rows containing missing values (geom\_path).



ggsave("Output/Countries/Colombia/Plots/RidgeLASSO\_log\_weekly\_pred1\_c1k.jpg", width = 8, height = 4)

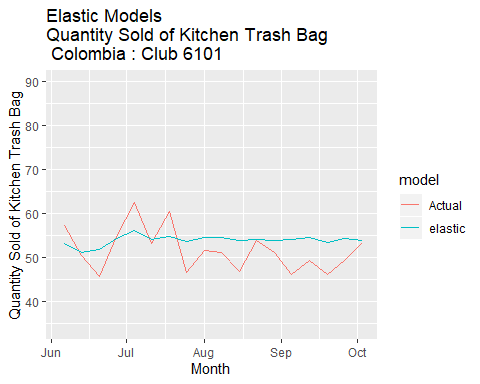
## Warning: Removed 179 rows containing missing values (geom\_path).

# plots: elastic models   
c1k\_week\_quantity\_models\_plot\_elastic\_log<-c1k\_week\_quantity\_models\_plot[(c1k\_week\_quantity\_models\_plot$model=="Actual"  
|c1k\_week\_quantity\_models\_plot$model=="elastic"),]  
ggplot(data=c1k\_week\_quantity\_models\_plot\_elastic\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+ggtitle(paste("Elastic Models","\nQuantity Sold of", item, "\n",country, ": Club", club))+ xlab("Month") + ylab(paste("Quantity Sold of", item))



ggsave("Output/Countries/Colombia/Plots/elastic\_log\_weekly\_pred\_c1k.jpg", width = 8, height = 4)  
ggplot(data=c1k\_week\_quantity\_models\_plot\_elastic\_log)+geom\_line(aes(x=TransactionTime, y=quantity\_avrg, col=model))+ggtitle(paste("Elastic Models","\nQuantity Sold of", item, "\n",country, ": Club", club))+ xlab("Month") + ylab(paste("Quantity Sold of", item))+xlim(max(c1k\_week$TransactionTime, na.rm = TRUE)-120, max(c1k\_week$TransactionTime, na.rm = TRUE))

## Warning: Removed 179 rows containing missing values (geom\_path).



ggsave("Output/Countries/Colombia/Plots/elastic\_log\_weekly\_pred1\_c1k.jpg", width = 8, height = 4)

## Warning: Removed 179 rows containing missing values (geom\_path).

save.image(paste0("Output/Countries/Colombia/", club, ".RData"))