Biostat 625 Final Project draft2

Haisheng Xu (haisheng)

12/13/2021

Data Cleaning

```
data2 = read.csv("Liquor_Items.csv")
data3 = na.omit(data2)

data_cleaner = data3[data3$Bottles.Sold>=730,]
data_cleaner$Category = as.factor(data_cleaner$Category)
data_cleaner$Vendor.Number = as.factor(data_cleaner$Vendor.Number)


data_index = read.csv("Covid_index.csv")


data_index = merge(x = data_cleaner, y = data_index, by = "Item.Number", all = TRUE)
data_index = (na.omit(data_index))
data_index = data_index[-12]
data_index$PopularityC = ifelse(data_index$Covid_index >= 0, 1, 0)




data_p = data_index[,-c(10,12,13)]
data_p$Popularity = ifelse(data_p$Popularity == "Popular", 1, 0)

data_c = data_index[,-c(11,12)]
```

Data Correlation

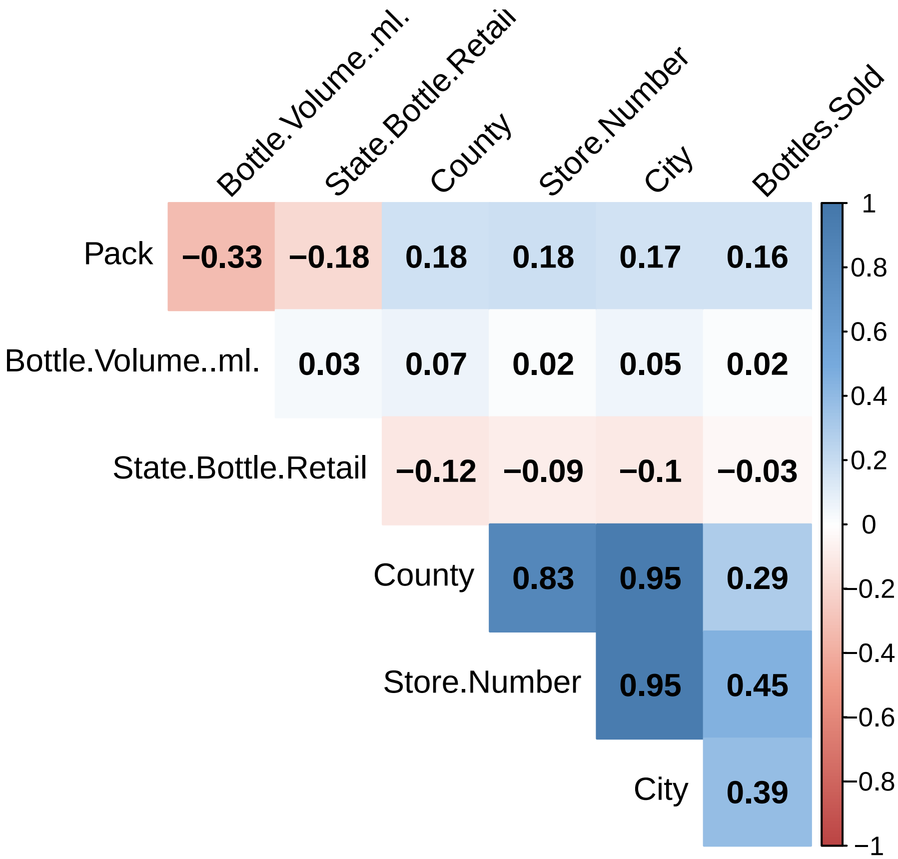
```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.0.4
```

```
## corrplot 0.84 loaded
```

```
#data3$Store.Number

M <- cor(data3[,c(4:10)])

# corrplot(M, method="color")

col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
```

```
corrplot(M, method="color", col=col(200),
         type="upper",
         addCoef.col = "black", # Add coefficient of correlation
         tl.col="black", tl.srt=45, #Text label color and rotation
         # Combine with significance
          insig = "blank",
         # hide correlation coefficient on the principal diagonal
         diag=FALSE
         )
```




```
library(caTools)
data_used = data_p
set.seed(111111)
split = sample.split(data_used$Popularity, SplitRatio = 0.60)

training_set = subset(data_used, split == TRUE)
test_set = subset(data_used, split == FALSE)


data_used = data_c
set.seed(111111)
split = sample.split(data_used$Popularity, SplitRatio = 0.60)

training_setc = subset(data_used, split == TRUE)
test_setc = subset(data_used, split == FALSE)
```

```r
library(e1071)
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```r
library(ResourceSelection)
```

```
## ResourceSelection 0.3-5   2019-07-22
```

```r
library(ggplot2)


full = glm(Popularity~.-Item.Number, data = training_set, family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
nullmodel = glm(Popularity~1, data = training_set, family=binomial)
n=nrow(training_set)
fit_step = step(nullmodel,scope=list(lower= nullmodel,
upper=full),direction="both",k=log(n))
```

```
## Start:  AIC=1903.18
## Popularity ~ 1
##
##                        Df Deviance    AIC
## + Store.Number          1   1077.2 1092.0
## + City                  1   1205.5 1220.4
## + County                1   1228.7 1243.5
## + State.Bottle.Retail   1   1815.3 1830.2
## + Pack                  1   1818.9 1833.8
## <none>                      1895.8 1903.2
## + Bottle.Volume..ml.    1   1895.6 1910.5
## + Category             49   1674.1 2045.8
## + Vendor.Number        98   1726.2 2462.1
##
## Step:  AIC=1092.03
## Popularity ~ Store.Number
##
##                        Df Deviance    AIC
## + Pack                  1  1014.60 1036.9
## + State.Bottle.Retail   1  1028.12 1050.4
## <none>                     1077.17 1092.0
## + Bottle.Volume..ml.    1  1071.11 1093.4
## + City                  1  1074.22 1096.5
## + County                1  1077.00 1099.3
## + Category             49   937.54 1316.6
## + Vendor.Number        98   963.07 1706.4
## - Store.Number          1  1895.75 1903.2
##
## Step:  AIC=1036.9
## Popularity ~ Store.Number + Pack
##
##                        Df Deviance     AIC
## + Bottle.Volume..ml.    1   962.39  992.12
```

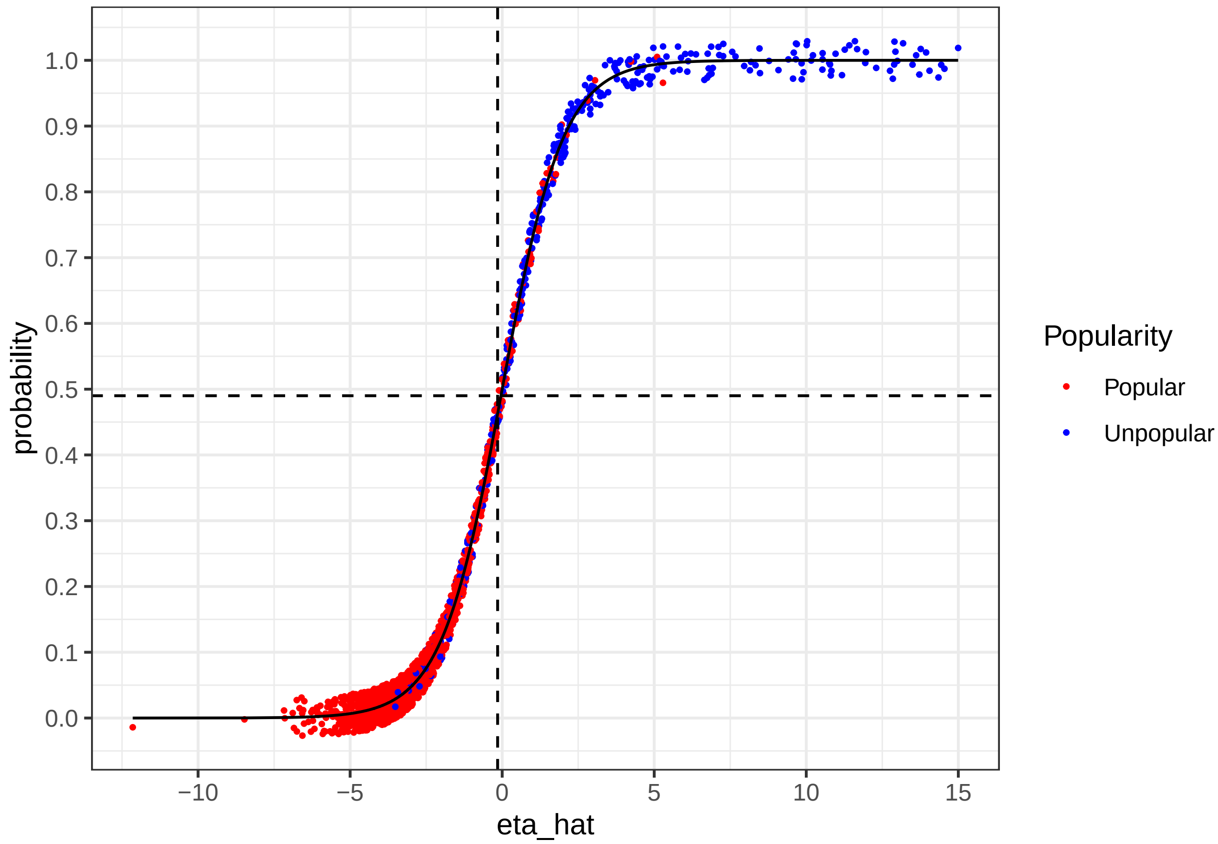
```
## + State.Bottle.Retail  1   998.91 1028.64
## <none>                     1014.60 1036.90
## + County                1  1011.28 1041.01
## + City                  1  1014.60 1044.33
## - Pack                  1  1077.17 1092.03
## + Category             49   890.61 1277.13
## + Vendor.Number        98   911.86 1662.60
## - Store.Number          1  1818.92 1833.79
##
## Step:  AIC=992.12
## Popularity ~ Store.Number + Pack + Bottle.Volume..ml.
##
##                        Df Deviance     AIC
## + State.Bottle.Retail  1   935.67  972.84
## <none>                     962.39  992.12
## + County                1   961.38  998.55
## + City                  1   961.83  999.00
## - Bottle.Volume..ml.    1  1014.60 1036.90
## - Pack                  1  1071.11 1093.41
## + Category             49   845.50 1239.45
## + Vendor.Number        98   869.34 1627.52
## - Store.Number          1  1788.83 1811.12
##
## Step:  AIC=972.84
## Popularity ~ Store.Number + Pack + Bottle.Volume..ml. + State.Bottle.Retail
##
##                        Df Deviance     AIC
## <none>                     935.67  972.84
## + City                  1   932.78  977.38
## + County                1   935.67  980.27
## - State.Bottle.Retail  1   962.39  992.12
## - Bottle.Volume..ml.    1   998.91 1028.64
## - Pack                  1  1008.30 1038.03
## + Category             49   812.80 1214.19
## + Vendor.Number        98   855.82 1621.42
## - Store.Number          1  1750.79 1780.53
```

```r
glm1 = summary(fit_step)$coefficients
hoslem.test(fit_step$y, fit_step$fitted.values,g=10)
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  fit_step$y, fit_step$fitted.values
## X-squared = 9.1248, df = 8, p-value = 0.3319
```

```r
etahat_fit = predict(fit_step, type = "link")
pb_fit = predict(fit_step, type = "response")

ggplot(training_set,aes(x= etahat_fit,y= pb_fit))+
geom_point(aes(color=factor(Popularity)),position=position_jitter(height=0.03,width=0),size=0.5)+
geom_line(aes(x= etahat_fit,y=pb_fit))+
labs(x="eta_hat",y="probability")+
scale_color_manual(values=c("red","blue"),name="Popularity",labels=c("Popular","Unpopular"))+
geom_hline(yintercept=0.49,linetype="dashed")+
```

```
geom_vline(xintercept=-0.15,linetype="dashed")+
scale_y_continuous(breaks=seq(0,1,by=0.1))+theme_bw()
```




```
pihat_test = predict(fit_step,newdata=test_set,
type="response")
threshold = 0.49
predicted_category =
factor(ifelse(pihat_test>threshold, 1,0) )
cm11 = confusionMatrix(data= predicted_category,reference= as.factor(as.numeric(test_set$Popularity)))
cm11
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 811 100
##          1  37 180
##
##                Accuracy : 0.8785
##                  95% CI : (0.858, 0.897)
##     No Information Rate : 0.7518
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6481
##
##  Mcnemar's Test P-Value : 1.177e-07
```

```
##
##             Sensitivity : 0.9564
##             Specificity : 0.6429
##          Pos Pred Value : 0.8902
##          Neg Pred Value : 0.8295
##              Prevalence : 0.7518
##          Detection Rate : 0.7190
##    Detection Prevalence : 0.8076
##       Balanced Accuracy : 0.7996
##
##        'Positive' Class : 0
##
```

```
full = glm(PopularityC~.-Item.Number, data = training_setc, family=binomial)
nullmodel = glm(PopularityC~1, data = training_setc, family=binomial)
n=nrow(training_setc)
fit_step = step(nullmodel,scope=list(lower= nullmodel,
upper=full),direction="both",k=log(n))
```

```
## Start:  AIC=2319.45
## PopularityC ~ 1
##
##                          Df Deviance    AIC
## + State.Bottle.Retail   1   2297.1 2312.0
## <none>                      2312.0 2319.4
## + Bottles.Sold          1   2309.4 2324.2
## + Store.Number          1   2310.4 2325.3
## + County                1   2310.6 2325.4
## + Pack                  1   2310.7 2325.6
## + Bottle.Volume..ml.    1   2310.8 2325.7
## + City                  1   2311.3 2326.2
## + Category             49   2196.2 2567.8
## + Vendor.Number       105   2089.3 2877.2
##
## Step:  AIC=2311.96
## PopularityC ~ State.Bottle.Retail
##
##                          Df Deviance    AIC
## <none>                      2297.1 2312.0
## + Bottle.Volume..ml.    1   2293.4 2315.7
## + Bottles.Sold          1   2295.7 2317.9
## + Store.Number          1   2296.2 2318.5
## + County                1   2296.5 2318.8
## + Pack                  1   2296.9 2319.2
## + City                  1   2296.9 2319.2
## - State.Bottle.Retail   1   2312.0 2319.4
## + Category             49   2188.5 2567.6
## + Vendor.Number       105   2081.0 2876.4
```

```
glm2 = summary(fit_step)$coefficients

#Only one variable significant: Retailed

#hoslem.test(fit_step$y, fit_step$fitted.values,g=10)
```

```r
#
# etahat_fit = predict(fit_step, type = "link")
# pb_fit = predict(fit_step, type = "response")
#
# ggplot(training_setc,aes(x= etahat_fit,y= pb_fit))+
# geom_point(aes(color=factor(PopularityC)),position=position_jitter(height=0.03,width=0),size=0.5)+
# geom_line(aes(x= etahat_fit,y=pb_fit))+
# labs(x="eta_hat",y="probability")+
# scale_color_manual(values=c("red","blue"),name="PopularityC",labels=c("Popular","Unpopular"))+
# geom_hline(yintercept=0.49,linetype="dashed")+
# geom_vline(xintercept=-0.15,linetype="dashed")+
# scale_y_continuous(breaks=seq(0,1,by=0.1))+theme_bw()
#
# #
# #
# pihat_test = predict(fit_step,newdata=test_setc,
# type="response")
# threshold = 0.49
# predicted_category =
# factor(ifelse(pihat_test>threshold, 1,0) )
# confusionMatrix(data= predicted_category,reference= as.factor(as.numeric(test_set$Popularity)))
```

```r
#Random Forest Cannot more than 53 levels

# Decision Tree
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.0.5
```

```r
library(rpart.plot)
```

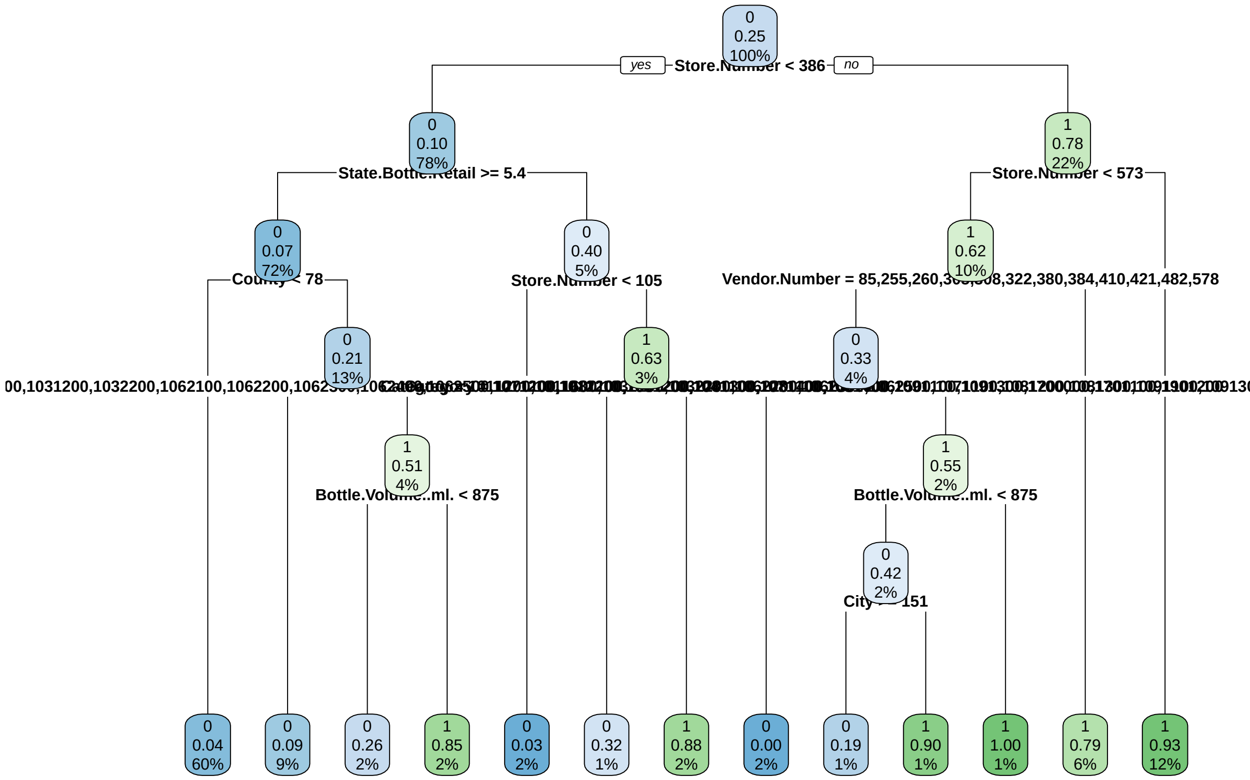
```
## Warning: package 'rpart.plot' was built under R version 4.0.5
```

```r
library(dplyr)
```

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

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

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

```r
library(ggplot2)

fit <- rpart(Popularity~.-Item.Number, data = training_set, method = 'class')
rpart.plot(fit,cex = 0.5)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```

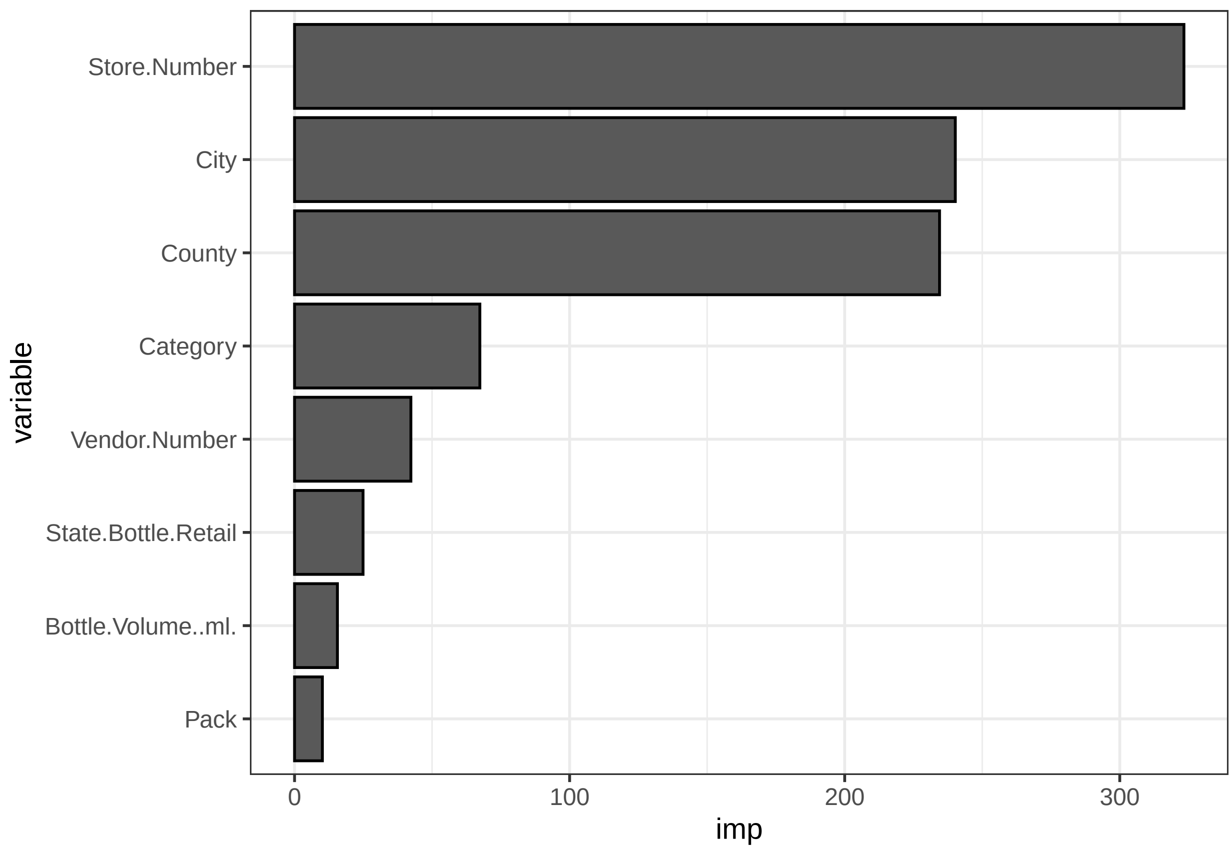



```
predict_unseen <-predict(fit, test_set, type = 'class')
table_mat <- table(test_set$Popularity, predict_unseen)
cm12 = confusionMatrix(table_mat)
cm12
```

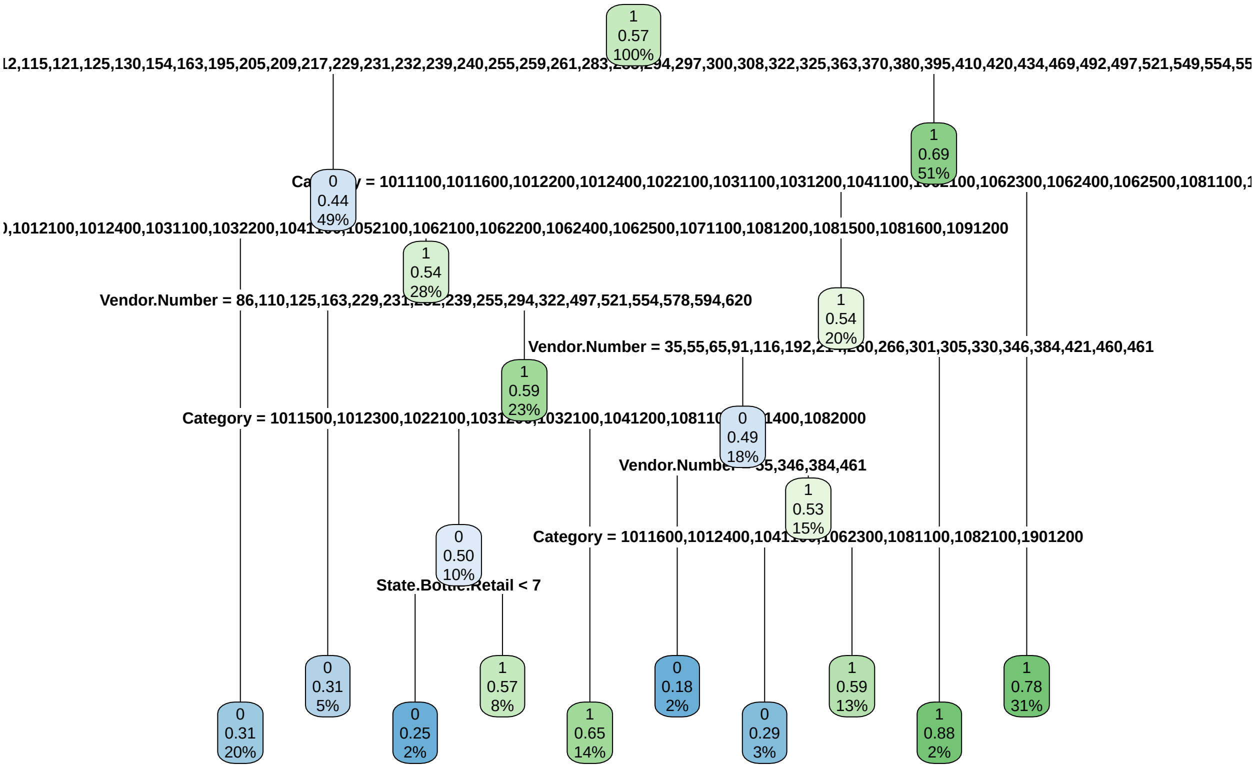
```
## Confusion Matrix and Statistics
##
##    predict_unseen
##       0   1
##   0 791  57
##   1  89 191
##
##                 Accuracy : 0.8706
##                   95% CI : (0.8496, 0.8896)
##      No Information Rate : 0.7801
##      P-Value [Acc > NIR] : 4.927e-15
##
##                    Kappa : 0.6394
##
##   Mcnemar's Test P-Value : 0.0103
##
##              Sensitivity : 0.8989
##              Specificity : 0.7702
##           Pos Pred Value : 0.9328
##           Neg Pred Value : 0.6821
##               Prevalence : 0.7801
##           Detection Rate : 0.7012
```

```
##    Detection Prevalence : 0.7518
##       Balanced Accuracy : 0.8345
##
##        'Positive' Class : 0
##
```

```r
# Importance of variables
importance = data.frame(imp = fit$variable.importance)
df2 <- importance %>%
  tibble::rownames_to_column() %>%
  dplyr::rename("variable" = rowname) %>%
  dplyr::arrange(imp) %>%
  dplyr::mutate(variable = forcats::fct_inorder(variable))

ggplot2::ggplot(df2) +
  geom_col(aes(x = variable, y = imp),
           col = "black", show.legend = F) +
  coord_flip() +
  scale_fill_grey() +
  theme_bw()
```




```r
#Random Forest Cannot more than 53 levels

# Decision Tree

fit <- rpart(PopularityC~.-Item.Number, data = training_setc, method = 'class')
rpart.plot(fit,cex = 0.5)
```

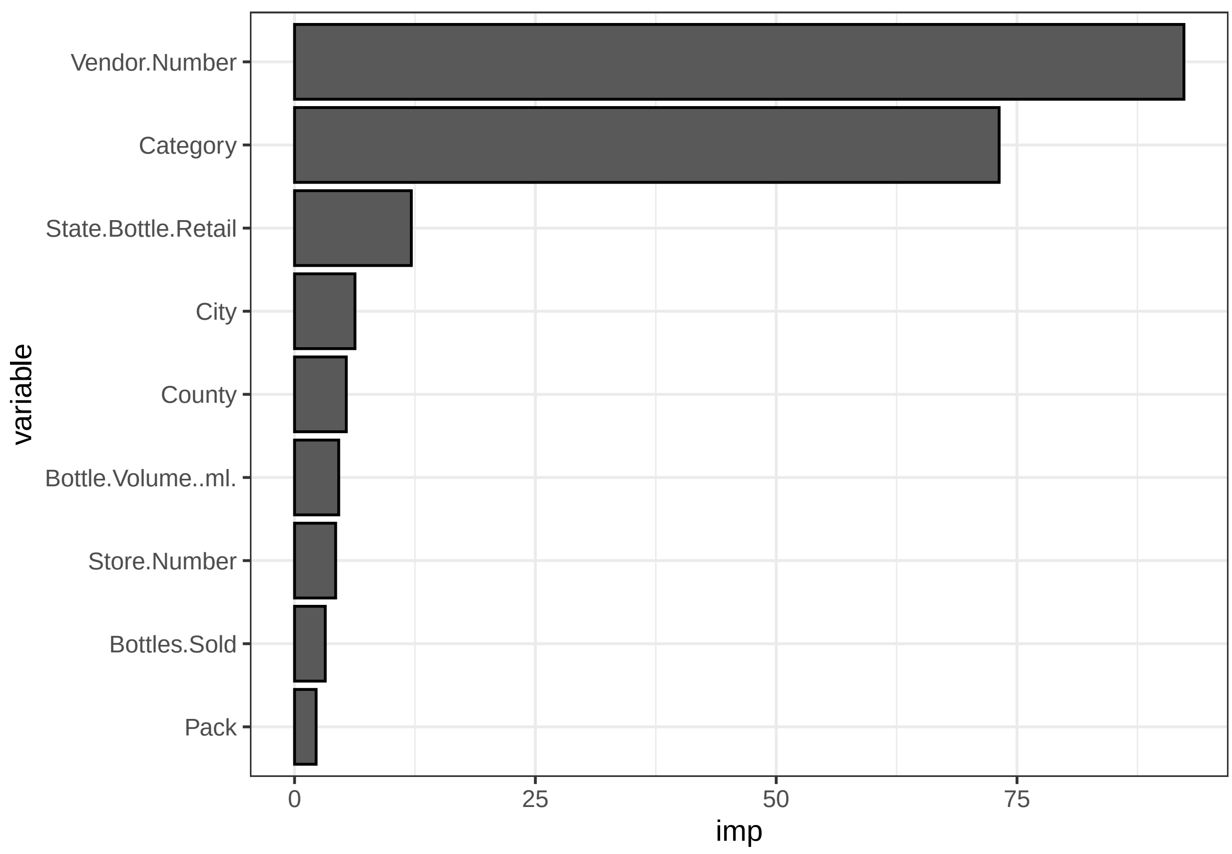


```r
predict_unseen <-predict(fit, test_setc, type = 'class')
table_mat <- table(test_setc$PopularityC, predict_unseen)
cm22 = confusionMatrix(table_mat)
cm22
```

```
## Confusion Matrix and Statistics
##
##    predict_unseen
##       0   1
##   0 190 296
##   1 179 463
##
##                Accuracy : 0.5789
##                  95% CI : (0.5495, 0.6079)
##     No Information Rate : 0.6729
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1155
##
##  Mcnemar's Test P-Value : 1.024e-07
##
##             Sensitivity : 0.5149
##             Specificity : 0.6100
##          Pos Pred Value : 0.3909
##          Neg Pred Value : 0.7212
##              Prevalence : 0.3271
##          Detection Rate : 0.1684
```

```
##    Detection Prevalence : 0.4309
##       Balanced Accuracy : 0.5625
##
##        'Positive' Class : 0
##
```

```r
# Importance of variables
importance = data.frame(imp = fit$variable.importance)
df2 <- importance %>%
  tibble::rownames_to_column() %>%
  dplyr::rename("variable" = rowname) %>%
  dplyr::arrange(imp) %>%
  dplyr::mutate(variable = forcats::fct_inorder(variable))

ggplot2::ggplot(df2) +
  geom_col(aes(x = variable, y = imp),
           col = "black", show.legend = F) +
  coord_flip() +
  scale_fill_grey() +
  theme_bw()
```




```r
# too many levels
```

```r
library(caTools)
library(e1071)
library(caret)
```

```r
classifier = svm(formula = Popularity~.-Item.Number,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'linear')

summary(classifier)
```

```
##
## Call:
## svm(formula = Popularity ~ . - Item.Number, data = training_set,
##     type = "C-classification", kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  450
##
##  ( 237 213 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
y_pred = predict(classifier, newdata = test_set[,-10])
cm = table(test_set$Popularity, y_pred)
cm13 = confusionMatrix(cm)
cm13
```

```
## Confusion Matrix and Statistics
##
##    y_pred
##       0   1
##   0 812  36
##   1  98 182
##
##                Accuracy : 0.8812
##                  95% CI : (0.8609, 0.8995)
##     No Information Rate : 0.8067
##     P-Value [Acc > NIR] : 1.330e-11
##
##                   Kappa : 0.6562
##
##  Mcnemar's Test P-Value : 1.367e-07
##
##             Sensitivity : 0.8923
##             Specificity : 0.8349
##          Pos Pred Value : 0.9575
##          Neg Pred Value : 0.6500
##              Prevalence : 0.8067
##          Detection Rate : 0.7199
```

```
##    Detection Prevalence : 0.7518
##       Balanced Accuracy : 0.8636
##
##        'Positive' Class : 0
##
```

```r
# Cannot plot for more than 2 predictors
```

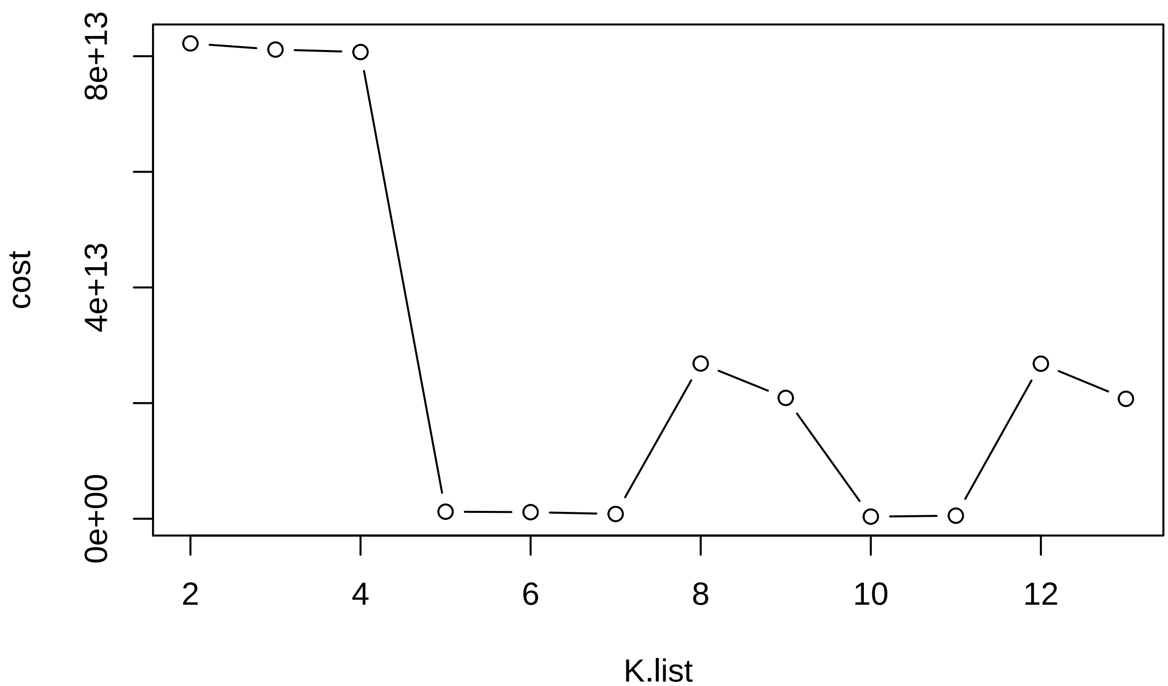
```r
library(caTools)
library(e1071)
library(caret)

classifier = svm(formula = PopularityC~.-Item.Number,
                 data = training_setc,
                 type = 'C-classification',
                 kernel = 'linear')


y_pred = predict(classifier, newdata = test_setc[,-11])
cm = table(test_setc$PopularityC, y_pred)
cm23 = confusionMatrix(cm)
cm23
```

```
## Confusion Matrix and Statistics
##
##    y_pred
##       0   1
##   0 168 318
##   1 168 474
##
##                Accuracy : 0.5691
##                  95% CI : (0.5397, 0.5983)
##     No Information Rate : 0.7021
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0873
##
##  Mcnemar's Test P-Value : 1.392e-11
##
##             Sensitivity : 0.5000
##             Specificity : 0.5985
##          Pos Pred Value : 0.3457
##          Neg Pred Value : 0.7383
##              Prevalence : 0.2979
##          Detection Rate : 0.1489
##    Detection Prevalence : 0.4309
##       Balanced Accuracy : 0.5492
##
##        'Positive' Class : 0
##
```

```r
# Cannot plot for more than 2 predictors
```

```r
library(e1071)
library(caTools)
library(class)
```

```r
n = dim(training_set)[1]
set.seed(1111)
# Using the elbow method to decide the number of clusters.
K.list = 2:13
cost= rep(NA, length(K.list))
for (i in 1:length(K.list)){
  K.i = K.list[i]
  mu.i = training_set[sample(1:n, size=K.i, replace = FALSE), ]
  km.i <- kmeans(training_set, centers=mu.i)
  cost[i] = km.i$tot.withinss
}
# Plot the elbow curve
plot(K.list, cost, type='b')
```




```r
classifier_knn <- knn(train = training_set[,-1],
                      test = test_set[,-1],
                      cl = training_set$Popularity,
                      k = 5)
#classifier_knn
cm <- table(test_set$Popularity, classifier_knn)
cm
```

```
##    classifier_knn
##       0   1
##   0 813  35
##   1 122 158
```

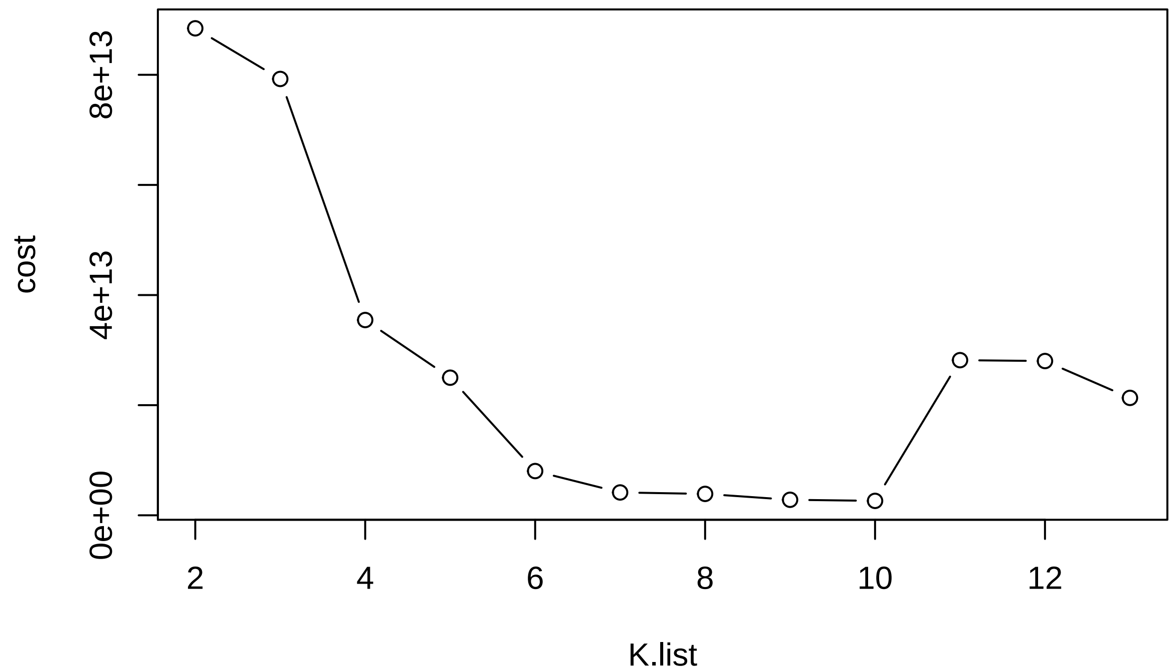
```r
misClassError <- mean(classifier_knn != test_set$Popularity)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 0.860815602836879"
```

```r
cm14 = confusionMatrix(cm)
cm14
```

```
## Confusion Matrix and Statistics
##
##    classifier_knn
##       0   1
##   0 813  35
##   1 122 158
##
##                Accuracy : 0.8608
##                  95% CI : (0.8392, 0.8805)
##     No Information Rate : 0.8289
##     P-Value [Acc > NIR] : 0.002037
##
##                   Kappa : 0.5838
##
##  Mcnemar's Test P-Value : 6.717e-12
##
##             Sensitivity : 0.8695
##             Specificity : 0.8187
##          Pos Pred Value : 0.9587
##          Neg Pred Value : 0.5643
##              Prevalence : 0.8289
##          Detection Rate : 0.7207
##    Detection Prevalence : 0.7518
##       Balanced Accuracy : 0.8441
##
##        'Positive' Class : 0
##
```

```r
n = dim(training_setc)[1]
set.seed(1111)
# Using the elbow method to decide the number of clusters.
K.list = 2:13
cost= rep(NA, length(K.list))
for (i in 1:length(K.list)){
  K.i = K.list[i]
  mu.i = training_setc[sample(1:n, size=K.i, replace = FALSE), ]
  km.i <- kmeans(training_setc, centers=mu.i)
  cost[i] = km.i$tot.withinss
}
# Plot the elbow curve
plot(K.list, cost, type='b')
```




```
classifier_knn <- knn(train = training_setc[,-1],
                      test = test_setc[,-1],
                      cl = training_setc$PopularityC,
                      k = 6)
#classifier_knn
cm <- table(test_setc$PopularityC, classifier_knn)
cm
```

```
##    classifier_knn
##       0   1
##   0 215 271
##   1 222 420
```

```
misClassError <- mean(classifier_knn != test_set$Popularity)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 0.414007092198582"
```

```
cm24 = confusionMatrix(cm)
cm24
```

```
## Confusion Matrix and Statistics
##
##    classifier_knn
##       0   1
##   0 215 271
##   1 222 420
##
```

```
##                Accuracy : 0.5629
##                  95% CI : (0.5334, 0.5921)
##     No Information Rate : 0.6126
##     P-Value [Acc > NIR] : 0.99970
##
##                   Kappa : 0.0978
##
##  Mcnemar's Test P-Value : 0.03063
##
##             Sensitivity : 0.4920
##             Specificity : 0.6078
##          Pos Pred Value : 0.4424
##          Neg Pred Value : 0.6542
##              Prevalence : 0.3874
##          Detection Rate : 0.1906
##    Detection Prevalence : 0.4309
##       Balanced Accuracy : 0.5499
##
##        'Positive' Class : 0
##
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  3.0.5     v purrr   0.3.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```
library(caret)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.0.5
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-3
```

```
cv.lasso = cv.glmnet(model.matrix(Popularity~.-Item.Number-Vendor.Number, training_set)[,-1], training_

lasso = glmnet(model.matrix(Popularity~.-Item.Number-Vendor.Number, training_set)[,-1], training_set$Po

las1 = coef(lasso)


x.test <- model.matrix(Popularity~.-Item.Number-Vendor.Number, test_set)[,-1]
probabilities <- lasso %>% predict(newx = x.test)
```

```r
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)

# Model accuracy
observed.classes <- test_set$Popularity
mean(predicted.classes == observed.classes)
```

```
## [1] 0.8803191
```

```r
cm = table(predicted.classes, observed.classes)
cm15 = confusionMatrix(cm)
cm15
```

```
## Confusion Matrix and Statistics
##
##                  observed.classes
## predicted.classes   0   1
##                 0 827 114
##                 1  21 166
##
##                Accuracy : 0.8803
##                  95% CI : (0.8599, 0.8987)
##     No Information Rate : 0.7518
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6392
##
##  Mcnemar's Test P-Value : 2.412e-15
##
##             Sensitivity : 0.9752
##             Specificity : 0.5929
##          Pos Pred Value : 0.8789
##          Neg Pred Value : 0.8877
##              Prevalence : 0.7518
##          Detection Rate : 0.7332
##    Detection Prevalence : 0.8342
##       Balanced Accuracy : 0.7840
##
##        'Positive' Class : 0
##
```

```r
cv.lasso = cv.glmnet(model.matrix(PopularityC~.-Item.Number, training_setc)[,-1], training_setc$Populari

lasso = glmnet(model.matrix(PopularityC~.-Item.Number, training_setc)[,-1], training_setc$PopularityC,

las2 = coef(lasso)


x.test <- model.matrix(PopularityC~.-Item.Number, test_setc)[,-1]
probabilities <- lasso %>% predict(newx = x.test)
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)

# Model accuracy
observed.classes <- test_setc$PopularityC
mean(predicted.classes == observed.classes)
```

```
## [1] 0.4991135
```

```
cm = table(predicted.classes, observed.classes)
cm25 = confusionMatrix(cm)
cm25
```

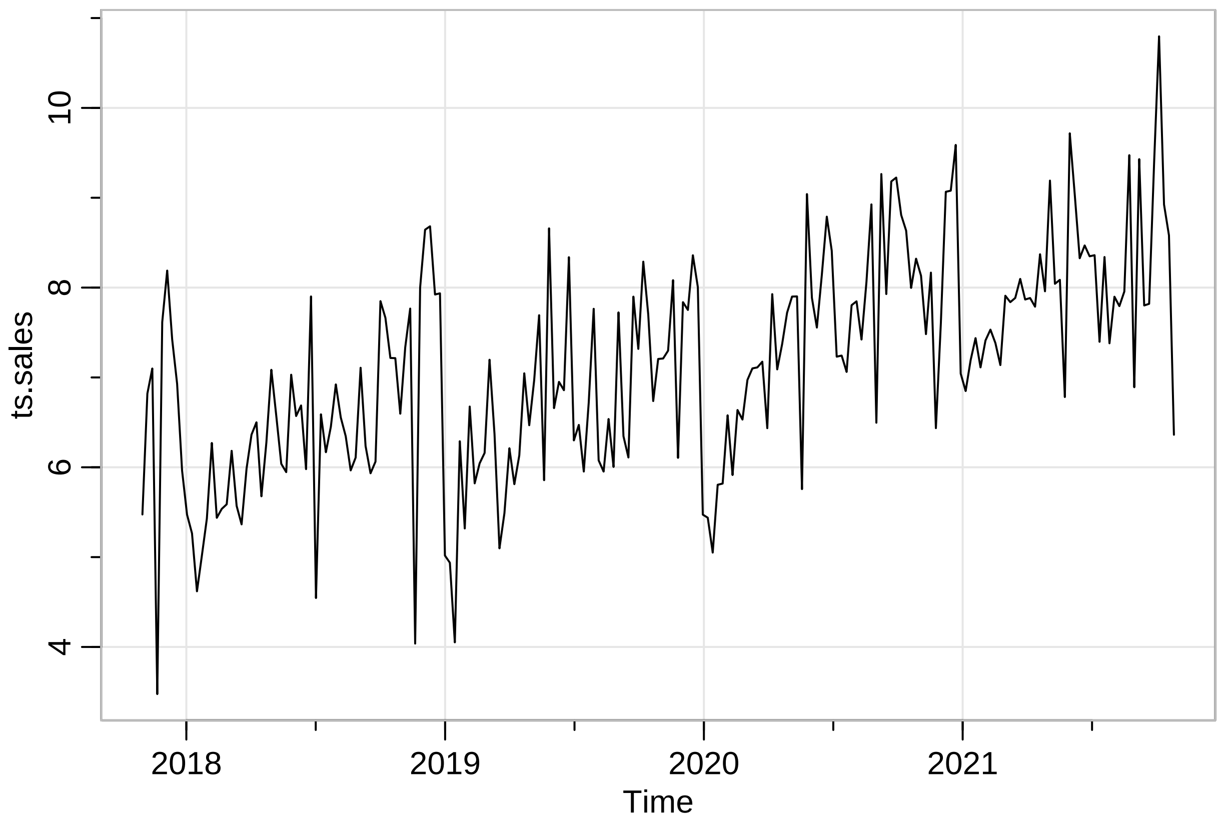
```
## Confusion Matrix and Statistics
##
##                  observed.classes
## predicted.classes   0   1
##                 0 395 474
##                 1  91 168
##
##                Accuracy : 0.4991
##                  95% CI : (0.4695, 0.5287)
##     No Information Rate : 0.5691
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0679
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8128
##             Specificity : 0.2617
##          Pos Pred Value : 0.4545
##          Neg Pred Value : 0.6486
##              Prevalence : 0.4309
##          Detection Rate : 0.3502
##    Detection Prevalence : 0.7704
##       Balanced Accuracy : 0.5372
##
##        'Positive' Class : 0
##
```

Time series

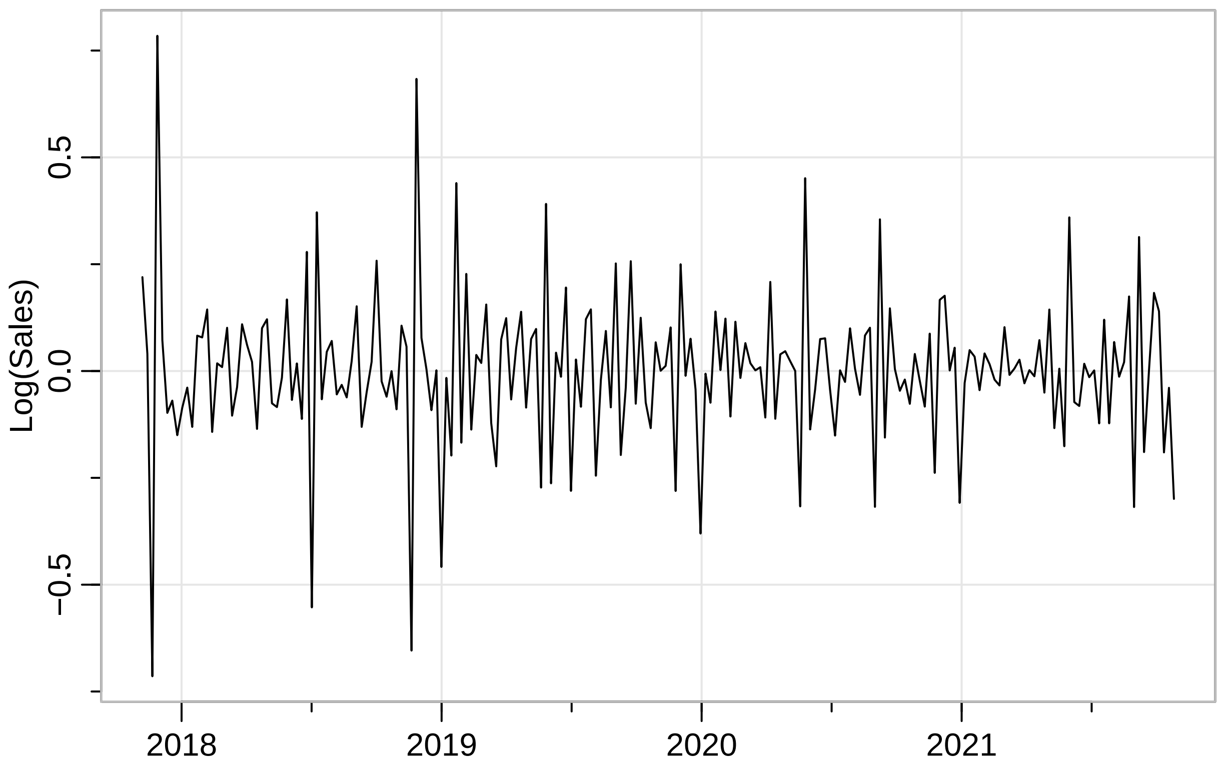
```
# Without considering seasonality in a short time period
library(lubridate)
```

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

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

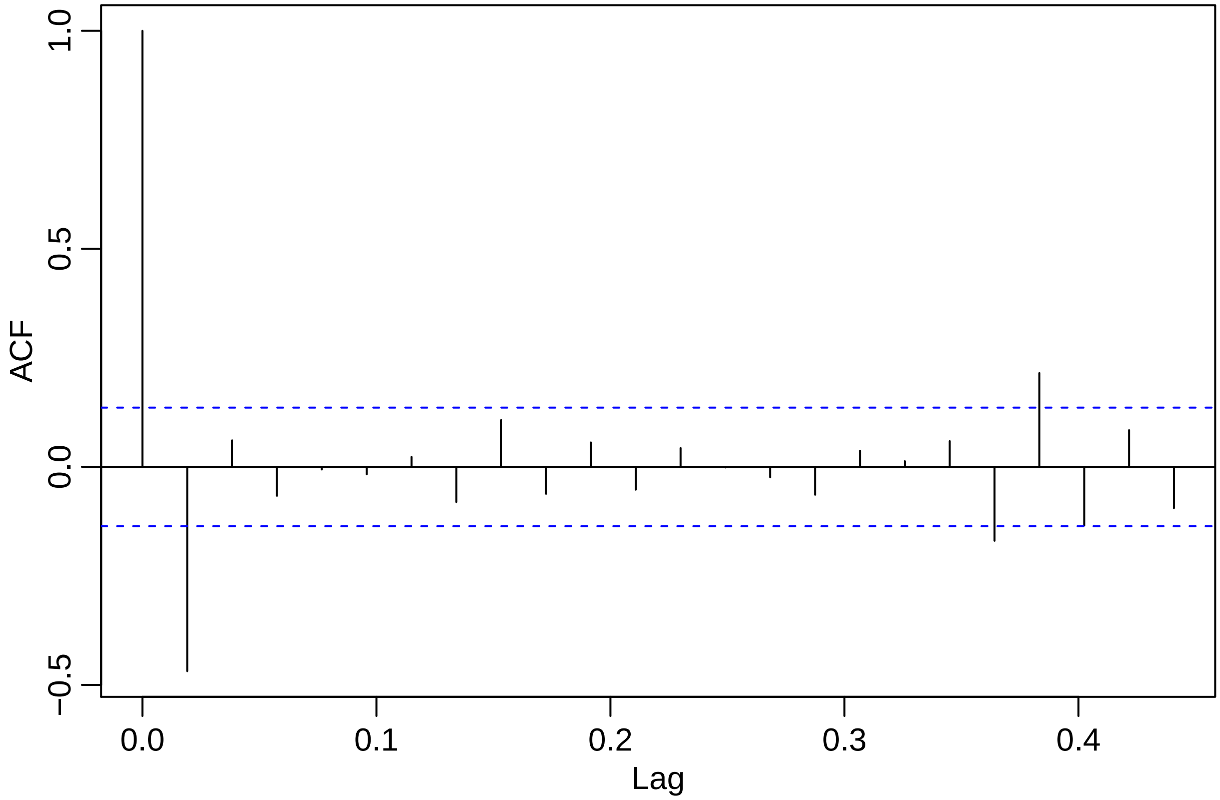
```
library(astsa)

ts <-read.csv("salesbyweek.csv")
ts.sales <-ts(ts[,2],start=decimal_date(ymd("2017-10-31")),freq=365.25/7)

tsplot(ts.sales,type="l")
```



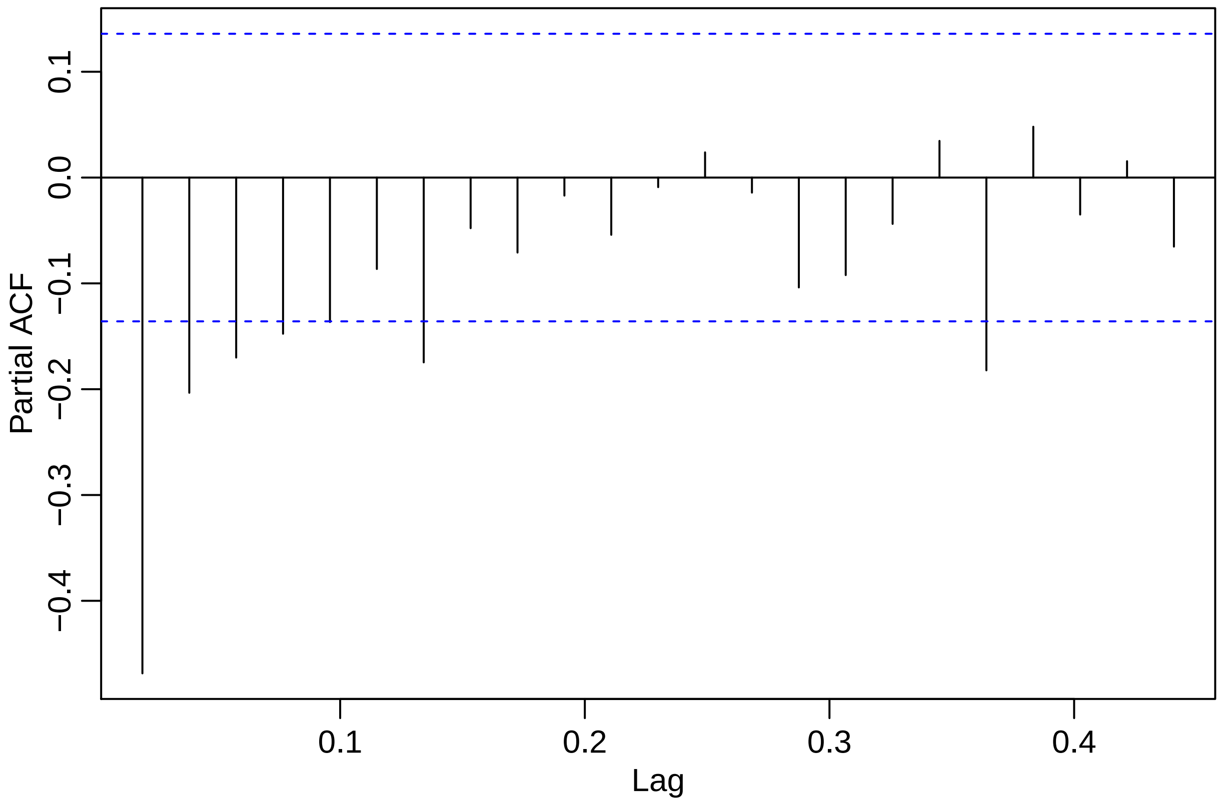

```r
diff.log <- diff(log(ts.sales))
tsplot(diff.log, xlab = "", ylab = "Log(Sales)", main="")
```




```
acf(diff.log)
```



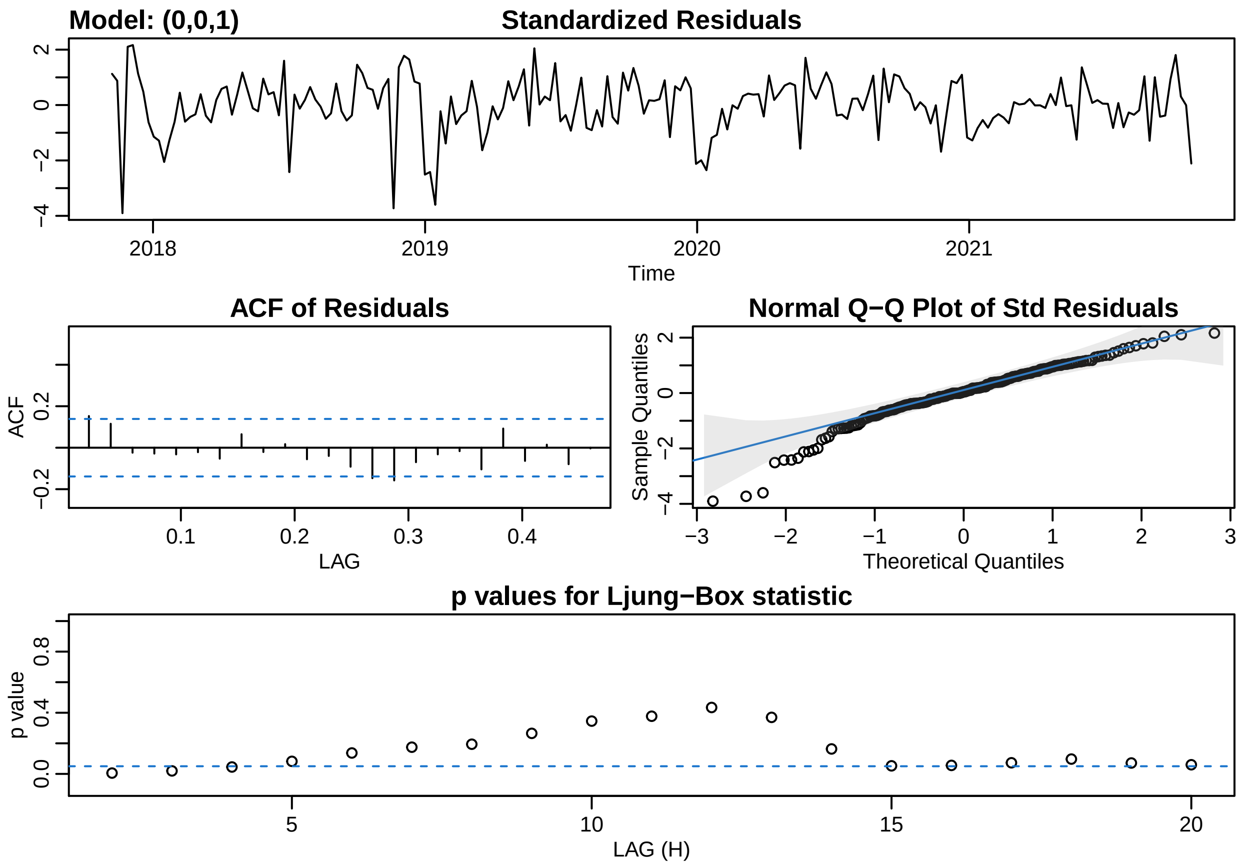

```
pacf(diff.log)
```




```
#Interpretation 1: Maybe the ACF is cutting off at lag 1 and the PACF is tailing off. This would sugges
#diff.log, which is equivalent to an ARIMA(0,1,1) for log.
#Interpretation 2: Maybe the ACF is tailing off and the PACF is cutting off at lag 7. This would sugges
# which is equivalent to an ARIMA(7,1,0) for log.p.
fit.ma1 <- sarima(diff.log,0,0,1)
```

```
## initial  value -1.727602
## iter   2 value -1.885715
## iter   3 value -1.937388
## iter   4 value -1.939513
## iter   5 value -1.940382
## iter   6 value -1.940943
## iter   7 value -1.940945
## iter   8 value -1.941149
## iter   9 value -1.941154
## iter   9 value -1.941154
## iter   9 value -1.941154
## final  value -1.941154
## converged
## initial  value -1.941013
## iter   2 value -1.941118
## iter   3 value -1.941127
## iter   4 value -1.941127
## iter   4 value -1.941127
## final  value -1.941127
## converged
```

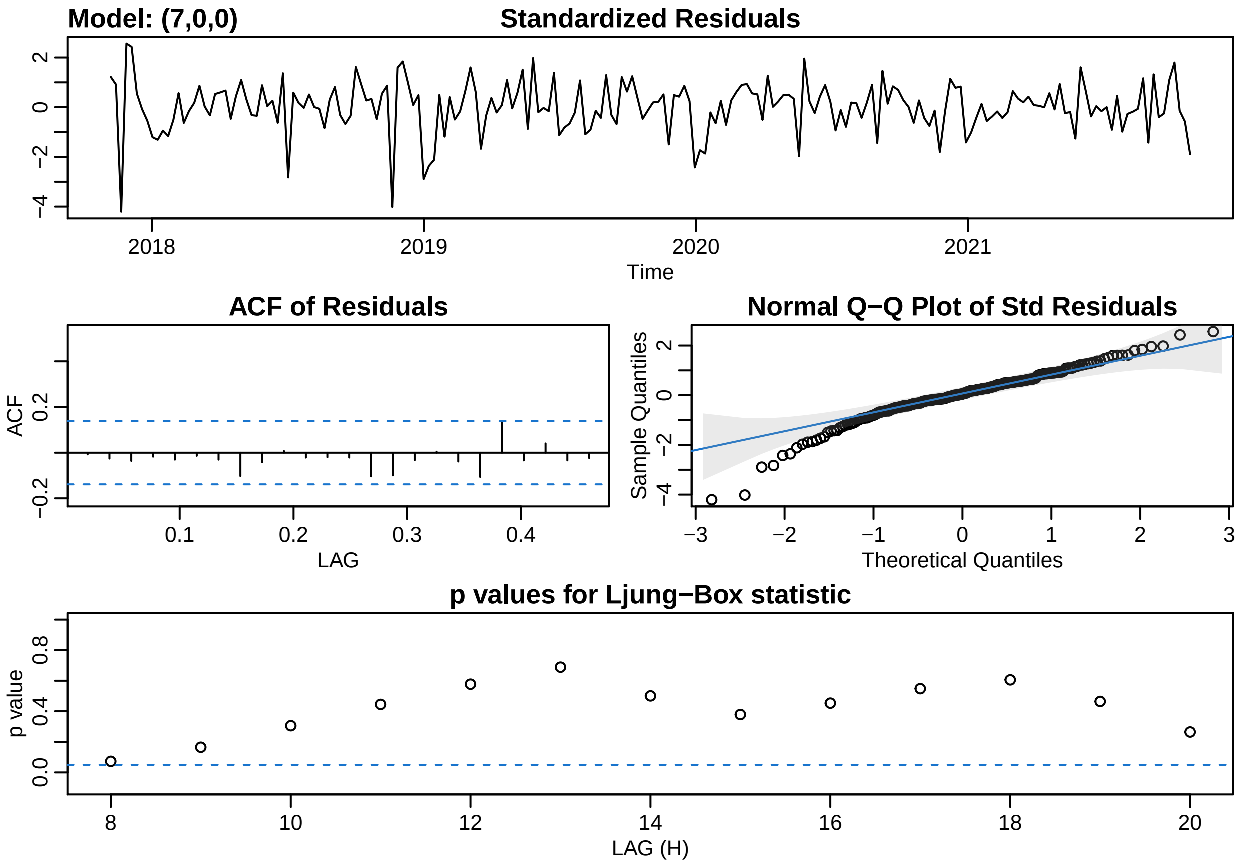
```
## Warning in sqrt(diag(fitit$var.coef)): NaNs produced

## Warning in sqrt(diag(fitit$var.coef)): NaNs produced
```




```r
fit.ar7 <- sarima(diff.log,7,0,0)
```

```
## initial  value -1.810866
## iter   2 value -1.912088
## iter   3 value -1.961913
## iter   4 value -1.987677
## iter   5 value -2.009984
## iter   6 value -2.014632
## iter   7 value -2.018646
## iter   8 value -2.022368
## iter   9 value -2.022612
## iter  10 value -2.022629
## iter  11 value -2.022630
## iter  11 value -2.022630
## iter  11 value -2.022630
## final  value -2.022630
## converged
## initial  value -1.942988
## iter   2 value -1.944924
## iter   3 value -1.945774
## iter   4 value -1.946771
## iter   5 value -1.947080
## iter   6 value -1.947093
```

```
## iter   7 value -1.947094
## iter   7 value -1.947094
## iter   7 value -1.947094
## final  value -1.947094
## converged
```




```
fit.ma1$AIC
```

```
## [1] -1.015531
```

```
fit.ma1$BIC
```

```
## [1] -0.9673938
```

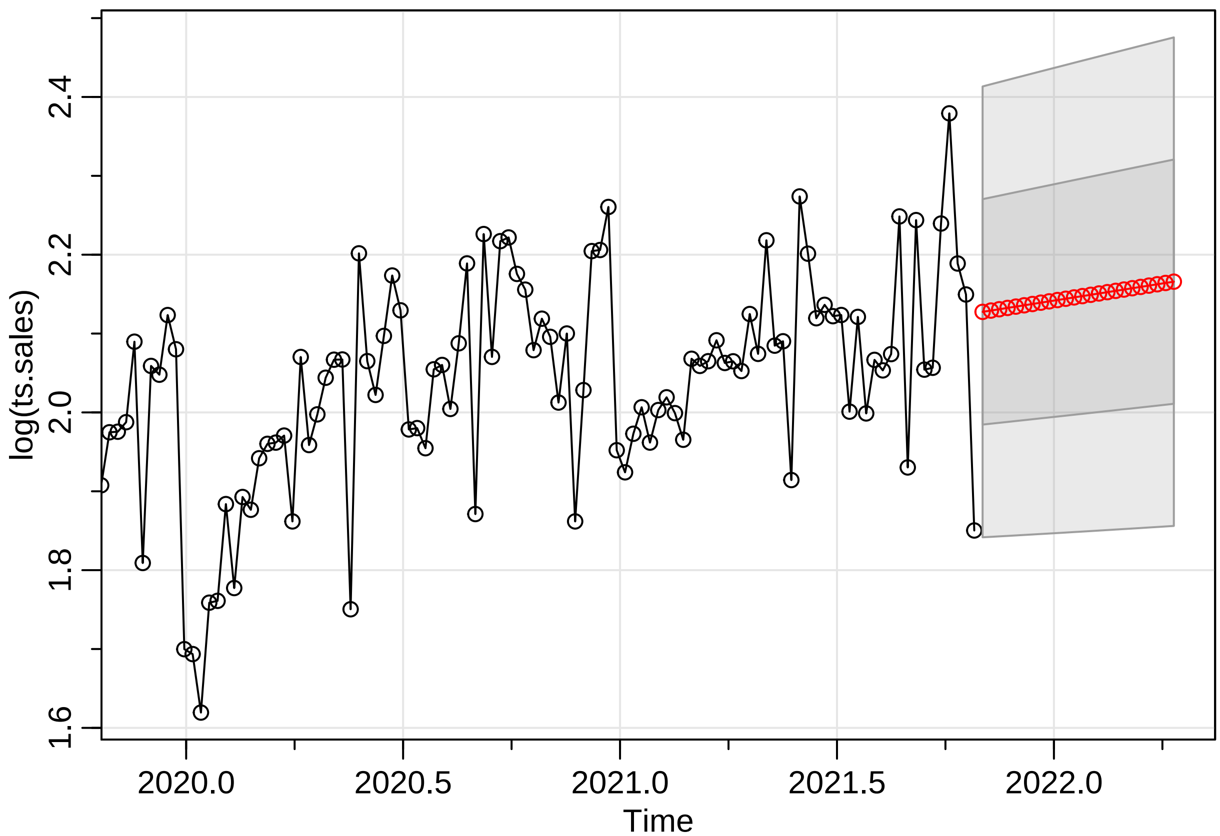
```
fit.ar7$AIC
```

```
## [1] -0.9697724
```

```
fit.ar7$BIC
```

```
## [1] -0.8253597
```

```
# Choose MA(1)

# Final model without considering the seasonality
ip.pred <- sarima.for(log(ts.sales),24,0,1,1)
```




```
exp(ip.pred$pred)
```

```
## Time Series:
## Start = 2021.83561268788
## End = 2022.2764066646
## Frequency = 52.1785714285714
##  [1] 8.393915 8.407905 8.421918 8.435955 8.450015 8.464098 8.478206 8.492336
##  [9] 8.506490 8.520668 8.534869 8.549094 8.563343 8.577615 8.591912 8.606232
## [17] 8.620576 8.634943 8.649335 8.663751 8.678191 8.692655 8.707143 8.721655
```

```
exp(ip.pred$pred-1.96*ip.pred$se)
```

```
## Time Series:
## Start = 2021.83561268788
## End = 2022.2764066646
## Frequency = 52.1785714285714
##  [1] 6.343145 6.346995 6.350874 6.354780 6.358713 6.362672 6.366659 6.370671
##  [9] 6.374710 6.378775 6.382866 6.386983 6.391124 6.395291 6.399483 6.403700
## [17] 6.407942 6.412208 6.416498 6.420813 6.425151 6.429514 6.433900 6.438309
```

```
exp(ip.pred$pred+1.96*ip.pred$se)
```

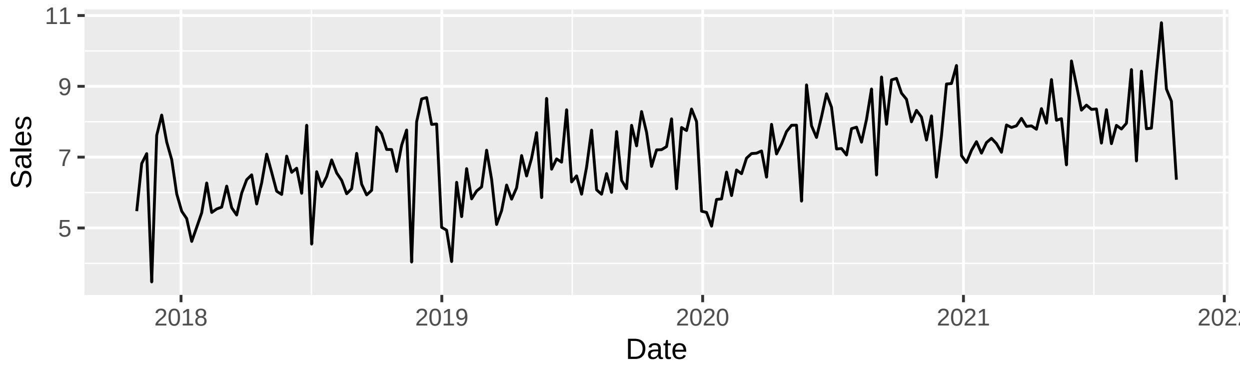
```
## Time Series:
## Start = 2021.83561268788
## End = 2022.2764066646
## Frequency = 52.1785714285714
##  [1] 11.10771 11.13800 11.16834 11.19871 11.22912 11.25957 11.29006 11.32059
##  [9] 11.35116 11.38177 11.41243 11.44312 11.47386 11.50463 11.53545 11.56632
```

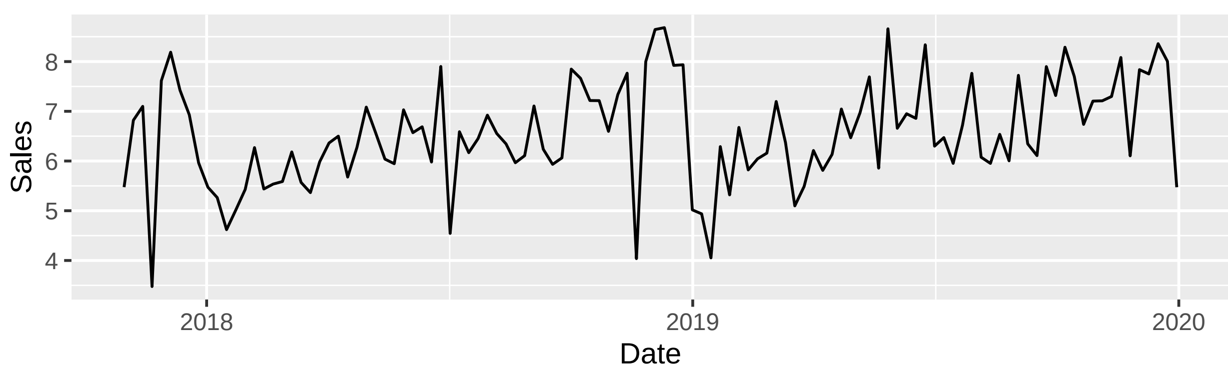
```
## [17] 11.59722 11.62817 11.65916 11.69020 11.72128 11.75240 11.78357 11.81479
```

```r
library(lubridate)
library(astsa)
library(ggplot2)
library(forecast)
```

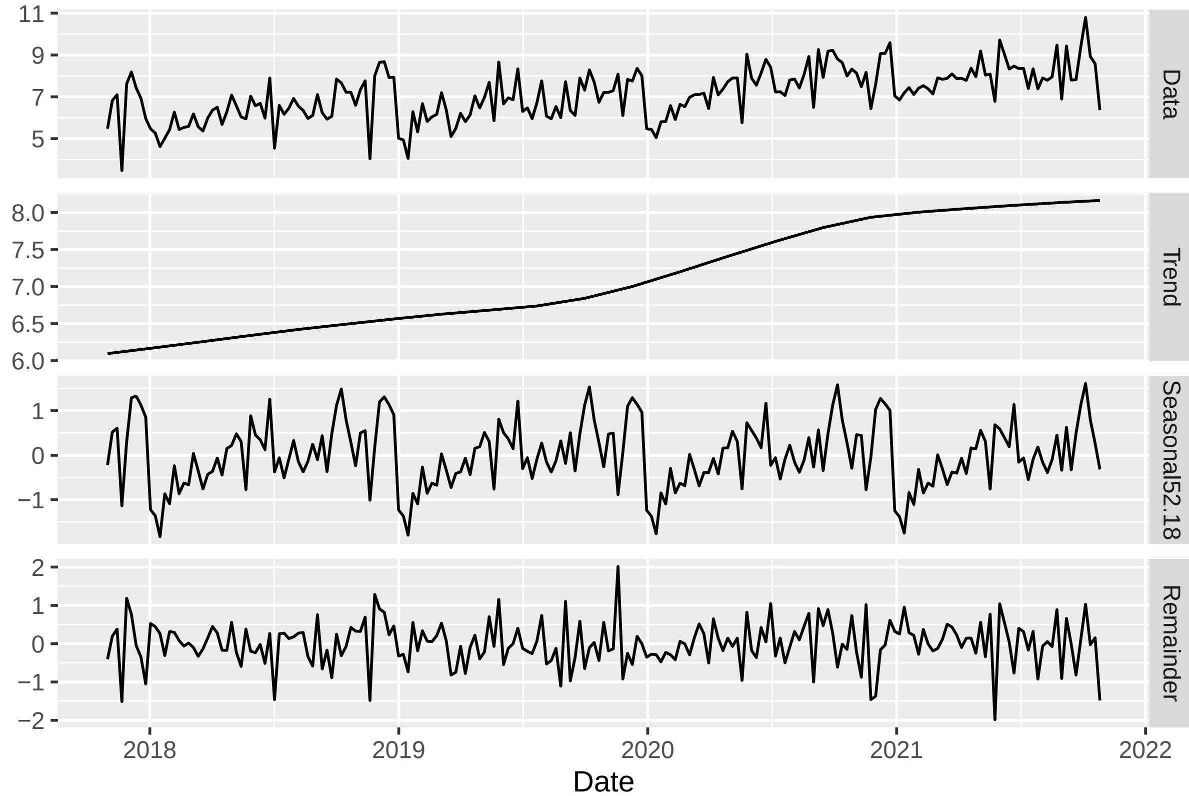
```
## Warning: package 'forecast' was built under R version 4.0.5

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

##
## Attaching package: 'forecast'

## The following object is masked from 'package:astsa':
##
##     gas
```

```r
ts <-read.csv("salesbyweek.csv")
ts.sales <-ts(ts[,2],start=decimal_date(ymd("2017-10-31")),freq=365.25/7)


library(ggplot2)
p1 <- autoplot(ts.sales) +
  ylab("Sales") + xlab("Date")
p2 <- autoplot(window(ts.sales, end=2020)) +
  ylab("Sales") + xlab("Date")
gridExtra::grid.arrange(p1,p2)
```





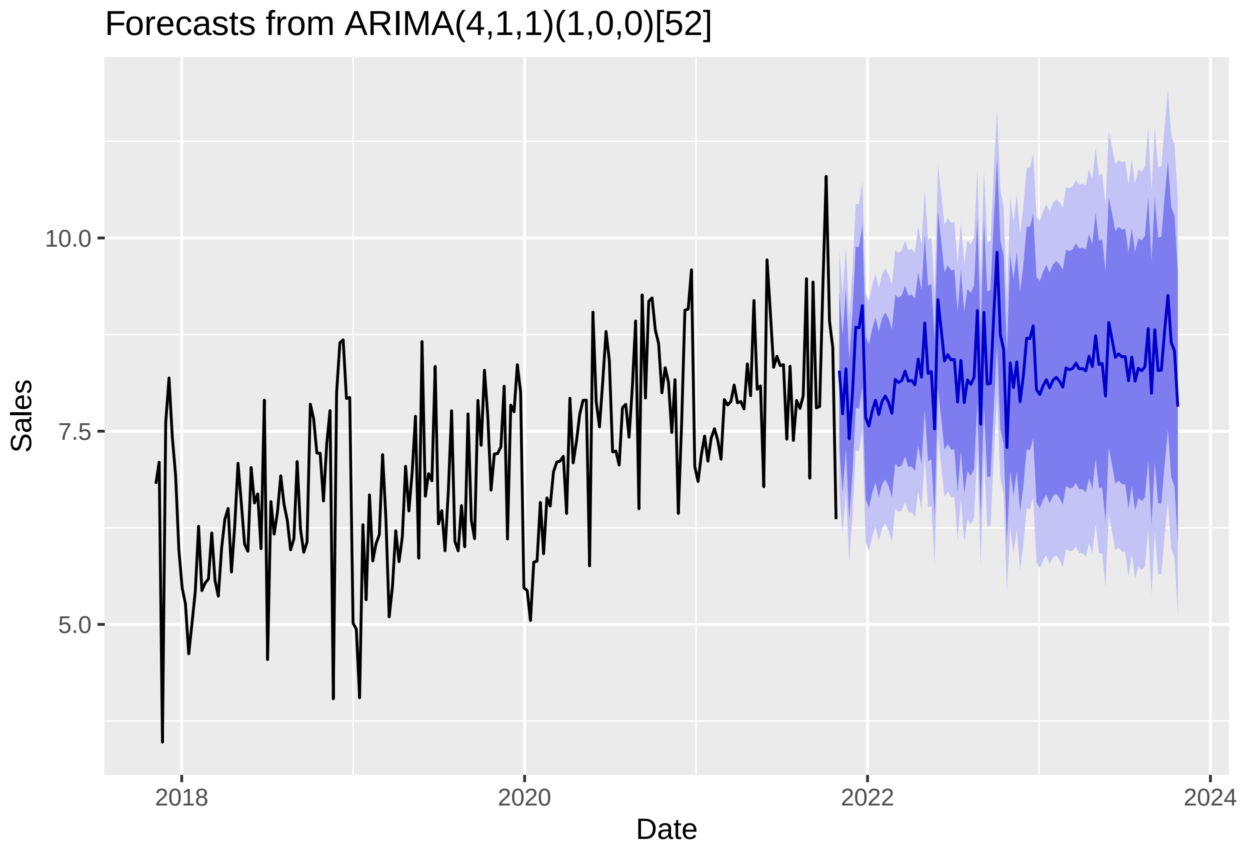

```
ts.sales %>% mstl() %>%
  autoplot() + xlab("Date")
```



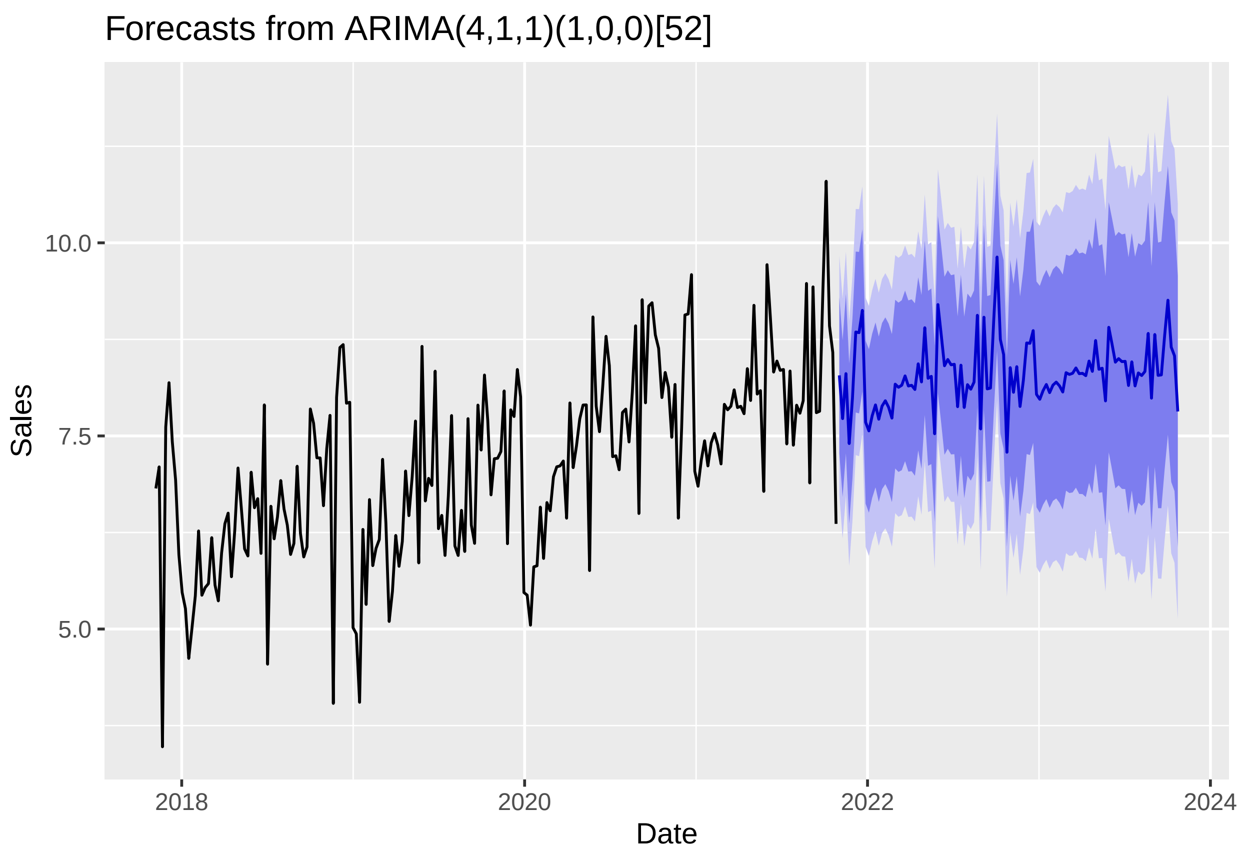

```
#Seasonality
library(forecast)

# Automated forecasting using an ARIMA model
fit <- auto.arima(ts.sales)
fit
```

```
## Series: ts.sales
## ARIMA(4,1,1)(1,0,0)[52]
##
## Coefficients:
##           ar1     ar2      ar3      ar4      ma1    sar1
##       -0.0224  0.0921  -0.0009  -0.0436  -0.9086  0.5699
## s.e.   0.0872  0.0836   0.0737   0.0790   0.0459  0.0616
##
## sigma^2 estimated as 0.6235:  log likelihood=-254.03
## AIC=522.07   AICc=522.63   BIC=545.43
```

```
#Complex seasonality
fit %>%
  forecast() %>%
  autoplot(include=208) +
    ylab("Sales") + xlab("Date")
```




```
#Complex seasonality
fit %>%
  forecast() %>%
  autoplot(include=208) +
    ylab("Sales") + xlab("Date")
```




```
forecast(fit)
```

```
##          Point Forecast    Lo 80     Hi 80    Lo 95     Hi 95
## 2021.836       8.283811 7.271851  9.295772 6.736152  9.831471
## 2021.855       7.723821 6.709451  8.738191 6.172476  9.275166
## 2021.874       8.306597 7.275650  9.337544 6.729899  9.883294
## 2021.893       7.401187 6.365975  8.436399 5.817967  8.984407
## 2021.912       7.993056 6.955920  9.030192 6.406893  9.579218
## 2021.931       8.845136 7.803519  9.886754 7.252120 10.438152
## 2021.951       8.836805 7.791473  9.882136 7.238108 10.435501
## 2021.970       9.125539 8.075875 10.175203 7.520216 10.730862
## 2021.989       7.676894 6.622882  8.730907 6.064921  9.288867
## 2022.008       7.565085 6.506814  8.623357 5.946599  9.183572
## 2022.027       7.761533 6.698982  8.824084 6.136502  9.386564
## 2022.046       7.901558 6.834781  8.968335 6.270063  9.533053
## 2022.066       7.716028 6.645038  8.787018 6.078091  9.353966
## 2022.085       7.886638 6.811452  8.961823 6.242283  9.530992
## 2022.104       7.955509 6.876145  9.034873 6.304765  9.606254
## 2022.123       7.870162 6.786634  8.953689 6.213050  9.527274
## 2022.142       7.730513 6.642838  8.818188 6.067058  9.393969
## 2022.161       8.170390 7.078584  9.262197 6.500616  9.840165
## 2022.181       8.129714 7.033791  9.225637 6.453645  9.805784
## 2022.200       8.156420 7.056396  9.256444 6.474079  9.838761
## 2022.219       8.277205 7.173096  9.381315 6.588615  9.965795
## 2022.238       8.146423 7.038243  9.254603 6.451608  9.841238
## 2022.257       8.155835 7.043599  9.268071 6.454817  9.856853
```

```
## 2022.276       8.100861 6.984584  9.217137 6.393663  9.808059
## 2022.296       8.433581 7.313278  9.553884 6.720226 10.146937
## 2022.315       8.198562 7.074247  9.322877 6.479071  9.918053
## 2022.334       8.900892 7.772579 10.029205 7.175287 10.626497
## 2022.353       8.245900 7.113604  9.378196 6.514203  9.977598
## 2022.372       8.271486 7.135220  9.407751 6.533717 10.009254
## 2022.391       7.527950 6.387729  8.668172 5.784132  9.271769
## 2022.411       9.200629 8.056466 10.344793 7.450782 10.950476
## 2022.430       8.812781 7.664689  9.960873 7.056926 10.568636
## 2022.449       8.408194 7.256186  9.560201 6.646351 10.170037
## 2022.468       8.489525 7.333615  9.645434 6.721714 10.257335
## 2022.487       8.420557 7.260759  9.580355 6.646799 10.194315
## 2022.506       8.427520 7.263846  9.591194 6.647835 10.207205
## 2022.526       7.878159 6.710622  9.045695 6.092566  9.663752
## 2022.545       8.416111 7.244725  9.587498 6.624630 10.207593
## 2022.564       7.868647 6.693422  9.043871 6.071296  9.665997
## 2022.583       8.164118 6.985069  9.343168 6.360918  9.967319
## 2022.602       8.104548 6.921686  9.287410 6.295517  9.913579
## 2022.621       8.197198 7.010536  9.383861 6.382355 10.012042
## 2022.641       9.062018 7.871567 10.252469 7.241380 10.882655
## 2022.660       7.590194 6.395967  8.784421 5.763781  9.416607
## 2022.679       9.036865 7.838873 10.234856 7.204695 10.869035
## 2022.698       8.108678 6.906934  9.310422 6.270769  9.946587
## 2022.717       8.119254 6.913769  9.324739 6.275624  9.962884
## 2022.736       9.013494 7.804280 10.222709 7.164160 10.862828
## 2022.756       9.815925 8.602993 11.028857 7.960905 11.670945
## 2022.775       8.748978 7.532340  9.965617 6.888290 10.609667
## 2022.794       8.553195 7.332861  9.773529 6.686856 10.419535
## 2022.813       7.288539 6.064521  8.512557 5.416565  9.160513
## 2022.832       8.383818 6.987672  9.779964 6.248597 10.519039
## 2022.851       8.064704 6.662073  9.467334 5.919565 10.209842
## 2022.871       8.396802 6.980009  9.813596 6.230004 10.563601
## 2022.890       7.880849 6.456306  9.305392 5.702198 10.059499
## 2022.909       8.218129 6.787587  9.648671 6.030303 10.405955
## 2022.928       8.703693 7.265318 10.142067 6.503889 10.903496
## 2022.947       8.698945 7.253274 10.144616 6.487981 10.909908
## 2022.966       8.863482 7.410133 10.316831 6.640777 11.086187
## 2022.986       8.037962 6.576955  9.498969 5.803545 10.272380
## 2023.005       7.974247 6.505667  9.442827 5.728248 10.220246
## 2023.024       8.086194 6.610056  9.562332 5.828636 10.343752
## 2023.043       8.165988 6.682354  9.649623 5.896966 10.435011
## 2023.062       8.060263 6.569168  9.551358 5.779830 10.340696
## 2023.081       8.157486 6.658968  9.656004 5.865700 10.449272
## 2023.100       8.196733 6.690829  9.702637 5.893651 10.499814
## 2023.120       8.148097 6.634842  9.661352 5.833774 10.462421
## 2023.139       8.068517 6.547947  9.589087 5.743006 10.394028
## 2023.158       8.319184 6.791334  9.847034 5.982539 10.655829
## 2023.177       8.296005 6.760909  9.831100 5.948278 10.643731
## 2023.196       8.311223 6.768916  9.853530 5.952468 10.669979
## 2023.215       8.380053 6.830568  9.929539 6.010320 10.749787
## 2023.235       8.305526 6.748896  9.862157 5.924866 10.686187
## 2023.254       8.310890 6.747147  9.874632 5.919352 10.702428
## 2023.273       8.279562 6.708740  9.850385 5.877196 10.681928
## 2023.292       8.469165 6.891294 10.047036 6.056020 10.882311
```

```
## 2023.311       8.335238 6.750350  9.920126 5.911361 10.759115
## 2023.330       8.735465 7.143591 10.327339 6.300904 11.170026
## 2023.350       8.362214 6.763385  9.961043 5.917015 10.807413
## 2023.369       8.376794 6.771039  9.982549 5.921004 10.832584
## 2023.388       7.953086 6.340435  9.565736 5.486750 10.419422
## 2023.407       8.906273 7.286756 10.525789 6.429436 11.383110
## 2023.426       8.685255 7.058901 10.311609 6.197961 11.172549
## 2023.445       8.454698 6.821535 10.087861 5.956991 10.952405
## 2023.465       8.501045 6.861102 10.140988 5.992969 11.009121
## 2023.484       8.461743 6.815048 10.108439 5.943340 10.980147
## 2023.503       8.465711 6.812291 10.119132 5.937023 10.994399
## 2023.522       8.152654 6.492536  9.812772 5.613723 10.691586
## 2023.541       8.459210 6.792421 10.125999 5.910076 11.008343
## 2023.560       8.147234 6.473801  9.820667 5.587939 10.706528
## 2023.580       8.315610 6.635559  9.995661 5.746194 10.885026
## 2023.599       8.281664 6.595021  9.968306 5.702166 10.861161
## 2023.618       8.334461 6.641252 10.027670 5.744921 10.924001
## 2023.637       8.827284 7.127534 10.527034 6.227741 11.426827
## 2023.656       7.988556 6.282290  9.694821 5.379047 10.598064
## 2023.675       8.812950 7.100194 10.525707 6.193515 11.432386
## 2023.695       8.284017 6.564794 10.003240 5.654692 10.913342
## 2023.714       8.290044 6.564378 10.015709 5.650866 10.929221
## 2023.733       8.799632 7.067549 10.531716 6.150639 11.448626
## 2023.752       9.256903 7.518424 10.995381 6.598130 11.915676
## 2023.771       8.648896 6.904047 10.393746 5.980379 11.317414
## 2023.790       8.537328 6.786130 10.288526 5.859102 11.215554
## 2023.810       7.816655 6.059133  9.574178 5.128756 10.504555
```

```r
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 4.0.5
```

```r
library(kableExtra)
```

```
## Warning: package 'kableExtra' was built under R version 4.0.5

##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```r
variablelist = c("General Popularity", "COVID Popularity")
methodname = c("GLM", "Decision Tree", "SVM", "KNN", "Lasso")

aclist = as.numeric(c(cm11$overall[1], cm12$overall[1], cm13$overall[1], cm14$overall[1], cm15$overall[
aclist2 = as.numeric(c(NA, cm22$overall[1], cm23$overall[1], cm24$overall[1], cm25$overall[1]))

aclistb = as.numeric(c(cm11$byClass[11], cm12$byClass[11], cm13$byClass[11], cm14$byClass[11], cm15$byCl
aclistb2 = as.numeric(c(NA, cm22$byClass[11], cm23$byClass[11], cm24$byClass[11], cm25$byClass[11]))

msgnf = c()

overalldata1 = data.frame(aclist,aclistb)
overalldata2 = data.frame(aclist2, aclistb2)
row.names(overalldata1) = methodname
```

Table 1: Classification for General Popularity

	Accuracy Rate	Balanced Accuracy Rate
GLM	0.8785461	0.7996125
Decision Tree	0.8705674	0.8345125
SVM	0.8812057	0.8635850
KNN	0.8608156	0.8440858
Lasso	0.8803191	0.7840465

Table 2: Classification for COVID Popularity

	Accuracy Rate	Balanced Accuracy Rate
GLM	NA	NA
Decision Tree	0.5789007	0.5624592
SVM	0.5691489	0.5492424
KNN	0.5629433	0.5499028
Lasso	0.4991135	0.5372197

```
colnames(overalldata1) = c("Accuracy Rate","Balanced Accuracy Rate")
row.names(overalldata2) = methodname
colnames(overalldata2) = c("Accuracy Rate","Balanced Accuracy Rate")


overalldata1 %>%
  kbl(caption = "Classification for General Popularity") %>%
  kable_paper("hover", full_width = F)
```

```
overalldata2 %>%
  kbl(caption = "Classification for COVID Popularity") %>%
  kable_paper("hover", full_width = F)
```

```
glm1%>%
  kbl(caption = "GLM for General Popularity") %>%
  kable_paper("hover", full_width = F)
```

```
glm2%>%
  kbl(caption = "GLM for COVID Popularity") %>%
  kable_paper("hover", full_width = F)
```

Table 3: GLM for General Popularity

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-6.1835348	0.4052910	-15.257024	0.0e+00
Store.Number	0.0096811	0.0005506	17.583406	0.0e+00
Pack	0.1045808	0.0123224	8.487011	0.0e+00
Bottle.Volume..ml.	0.0016921	0.0002127	7.955618	0.0e+00
State.Bottle.Retail	-0.0441104	0.0093000	-4.743043	2.1e-06

Table 4: GLM for COVID Popularity

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.0392380	0.0810043	0.4843936	0.6281066
State.Bottle.Retail	0.0127042	0.0035138	3.6155238	0.0002997