# BAN Course Project

## Wilkinson,Hunter

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.4.0 ✔ purrr 1.0.1   
## ✔ tibble 3.1.8 ✔ dplyr 1.0.10  
## ✔ tidyr 1.3.0 ✔ stringr 1.5.0   
## ✔ readr 2.1.3 ✔ forcats 1.0.0

## Warning: package 'ggplot2' was built under R version 4.2.2

## Warning: package 'tidyr' was built under R version 4.2.2

## Warning: package 'purrr' was built under R version 4.2.2

## Warning: package 'dplyr' was built under R version 4.2.2

## Warning: package 'stringr' was built under R version 4.2.2

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(tidymodels)

## Warning: package 'tidymodels' was built under R version 4.2.2

## ── Attaching packages ────────────────────────────────────── tidymodels 1.0.0 ──  
## ✔ broom 1.0.3 ✔ rsample 1.1.1  
## ✔ dials 1.1.0 ✔ tune 1.0.1  
## ✔ infer 1.0.4 ✔ workflows 1.1.2  
## ✔ modeldata 1.1.0 ✔ workflowsets 1.0.0  
## ✔ parsnip 1.0.3 ✔ yardstick 1.1.0  
## ✔ recipes 1.0.4

## Warning: package 'broom' was built under R version 4.2.2

## Warning: package 'dials' was built under R version 4.2.2

## Warning: package 'scales' was built under R version 4.2.2

## Warning: package 'infer' was built under R version 4.2.2

## Warning: package 'modeldata' was built under R version 4.2.2

## Warning: package 'parsnip' was built under R version 4.2.2

## Warning: package 'recipes' was built under R version 4.2.2

## Warning: package 'rsample' was built under R version 4.2.2

## Warning: package 'tune' was built under R version 4.2.2

## Warning: package 'workflows' was built under R version 4.2.2

## Warning: package 'workflowsets' was built under R version 4.2.2

## Warning: package 'yardstick' was built under R version 4.2.2

## ── Conflicts ───────────────────────────────────────── tidymodels\_conflicts() ──  
## ✖ scales::discard() masks purrr::discard()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ recipes::fixed() masks stringr::fixed()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ yardstick::spec() masks readr::spec()  
## ✖ recipes::step() masks stats::step()  
## • Dig deeper into tidy modeling with R at https://www.tmwr.org

library(GGally)

## Warning: package 'GGally' was built under R version 4.2.2

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

library(ggplot2)  
library(mice) #package for imputation

## Warning: package 'mice' was built under R version 4.2.2

##   
## Attaching package: 'mice'  
##   
## The following object is masked from 'package:stats':  
##   
## filter  
##   
## The following objects are masked from 'package:base':  
##   
## cbind, rbind

library(VIM) #visualizing missingness

## Warning: package 'VIM' was built under R version 4.2.2

## Loading required package: colorspace  
## Loading required package: grid  
## VIM is ready to use.  
##   
## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues  
##   
## Attaching package: 'VIM'  
##   
## The following object is masked from 'package:recipes':  
##   
## prepare  
##   
## The following object is masked from 'package:datasets':  
##   
## sleep

library(skimr) #alternative way to view dataset summaries

## Warning: package 'skimr' was built under R version 4.2.2

#install.packages("naniar")  
library(naniar)

## Warning: package 'naniar' was built under R version 4.2.2

##   
## Attaching package: 'naniar'  
##   
## The following object is masked from 'package:skimr':  
##   
## n\_complete

library(gridExtra)

## Warning: package 'gridExtra' was built under R version 4.2.2

##   
## Attaching package: 'gridExtra'  
##   
## The following object is masked from 'package:dplyr':  
##   
## combine

library(dplyr)  
library(corrplot)

## Warning: package 'corrplot' was built under R version 4.2.2

## corrplot 0.92 loaded

library(caret)

## Warning: package 'caret' was built under R version 4.2.2

## Loading required package: lattice  
##   
## Attaching package: 'caret'  
##   
## The following objects are masked from 'package:yardstick':  
##   
## precision, recall, sensitivity, specificity  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

library(rpart) #for classification trees

##   
## Attaching package: 'rpart'  
##   
## The following object is masked from 'package:dials':  
##   
## prune

library(rpart.plot) #for plotting trees

## Warning: package 'rpart.plot' was built under R version 4.2.2

library(rattle)

## Warning: package 'rattle' was built under R version 4.2.2

## Loading required package: bitops  
## Rattle: A free graphical interface for data science with R.  
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.  
##   
## Attaching package: 'rattle'  
##   
## The following object is masked from 'package:VIM':  
##   
## wine

library(RColorBrewer)  
library(ranger) #for random forests

## Warning: package 'ranger' was built under R version 4.2.2

##   
## Attaching package: 'ranger'  
##   
## The following object is masked from 'package:rattle':  
##   
## importance

#install.packages("randomForest")  
library(randomForest) #also for random forests

## Warning: package 'randomForest' was built under R version 4.2.2

## randomForest 4.7-1.1  
## Type rfNews() to see new features/changes/bug fixes.  
##   
## Attaching package: 'randomForest'  
##   
## The following object is masked from 'package:ranger':  
##   
## importance  
##   
## The following object is masked from 'package:rattle':  
##   
## importance  
##   
## The following object is masked from 'package:gridExtra':  
##   
## combine  
##   
## The following object is masked from 'package:dplyr':  
##   
## combine  
##   
## The following object is masked from 'package:ggplot2':  
##   
## margin

library(skimr)  
library(GGally)  
library(gridExtra)  
library(vip) #variable importance

## Warning: package 'vip' was built under R version 4.2.2

##   
## Attaching package: 'vip'  
##   
## The following object is masked from 'package:utils':  
##   
## vi

###Read in Data

library(readr)  
ames\_student\_1 <- read\_csv("ames\_student-1.csv")

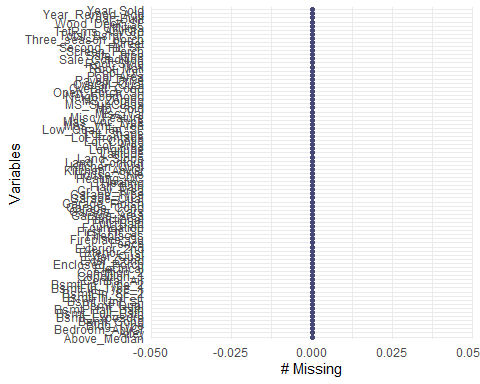
## Rows: 2053 Columns: 81  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (47): MS\_SubClass, MS\_Zoning, Street, Alley, Lot\_Shape, Land\_Contour, Ut...  
## dbl (34): Lot\_Frontage, Lot\_Area, Year\_Built, Year\_Remod\_Add, Mas\_Vnr\_Area, ...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

###Variable Conversions

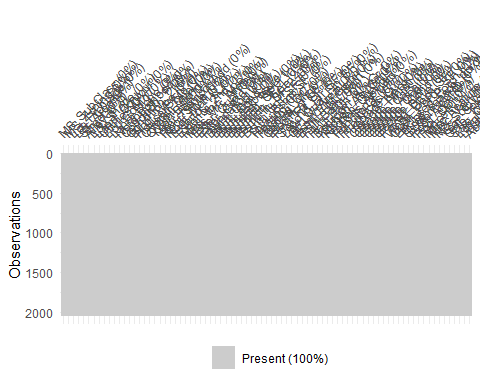
ames= ames\_student\_1 %>% mutate\_if(is.character, as\_factor) %>% mutate(Year\_Built = as\_factor(Year\_Built)) %>%   
 mutate(Year\_Remod\_Add = as\_factor(Year\_Remod\_Add)) %>% mutate(BsmtFin\_SF\_1 = as\_factor(BsmtFin\_SF\_1)) %>%   
 mutate(Bsmt\_Full\_Bath = as\_factor(Bsmt\_Full\_Bath)) %>% mutate(Bsmt\_Half\_Bath = as\_factor(Bsmt\_Half\_Bath)) %>%  
 mutate(Full\_Bath = as\_factor(Full\_Bath)) %>% mutate(Half\_Bath = as\_factor(Half\_Bath)) %>%   
 mutate(Bedroom\_AbvGr = as\_factor(Bedroom\_AbvGr)) %>% mutate(Kitchen\_AbvGr = as\_factor(Kitchen\_AbvGr)) %>%   
 mutate(Fireplaces = as\_factor(Fireplaces)) %>% mutate(Garage\_Cars = as\_factor(Garage\_Cars)) %>%   
 mutate(Mo\_Sold = as\_factor(Mo\_Sold)) %>% mutate(Year\_Sold = as\_factor(Year\_Sold)) %>%   
 mutate(Pool\_Area = as\_factor(Pool\_Area)) %>% mutate(TotRms\_AbvGrd = as\_factor(TotRms\_AbvGrd))

###Missing Data (No missing data)

gg\_miss\_var(ames)



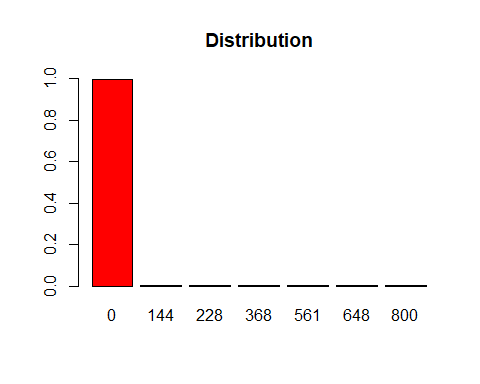
vis\_miss(ames)



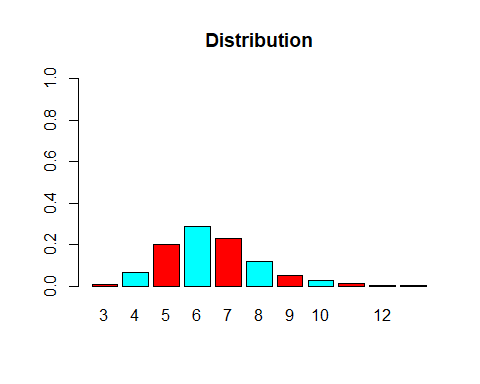
###Check for Imbalance among catergorical variables

####Remove(Pool\_Area)

barplot(prop.table(table(ames$Pool\_Area)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

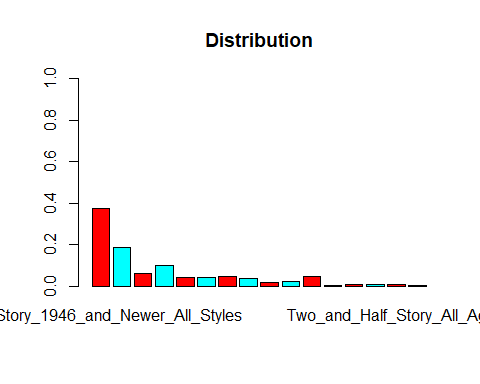


barplot(prop.table(table(ames$TotRms\_AbvGrd)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

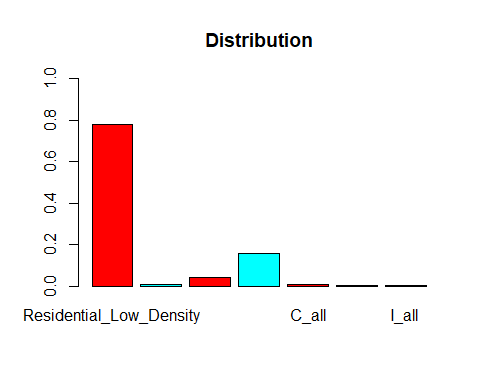


####Remove(MS\_Zoning,Street,Alley,Land\_Contour,Utilities,Lot\_Config)

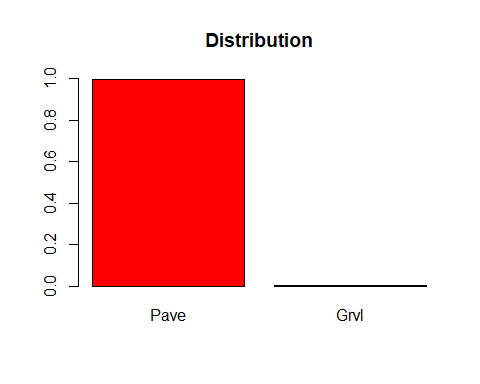
barplot(prop.table(table(ames$MS\_SubClass)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



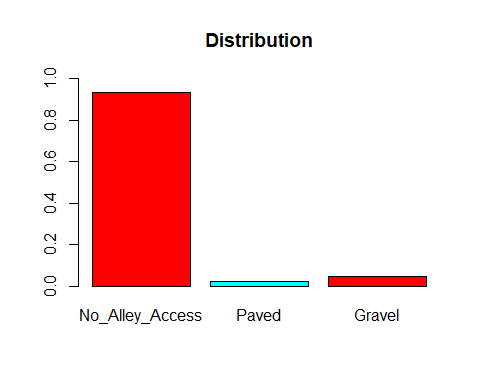
barplot(prop.table(table(ames$MS\_Zoning)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



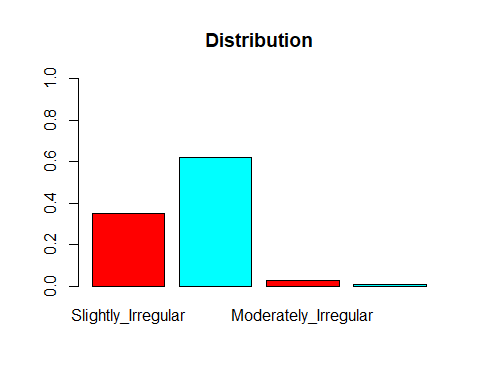
barplot(prop.table(table(ames$Street)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



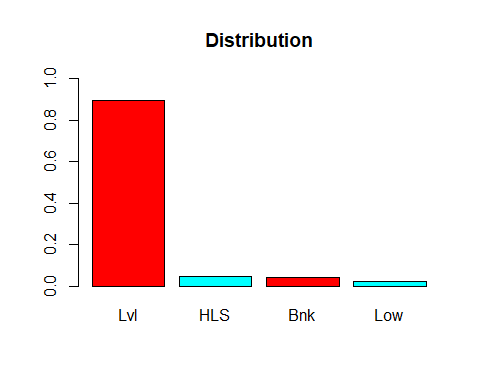
barplot(prop.table(table(ames$Alley)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



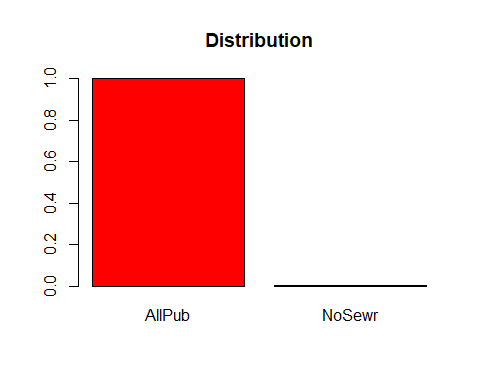
barplot(prop.table(table(ames$Lot\_Shape)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



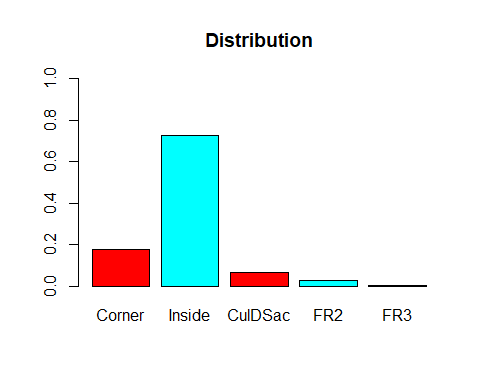
barplot(prop.table(table(ames$Land\_Contour)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



barplot(prop.table(table(ames$Utilities)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

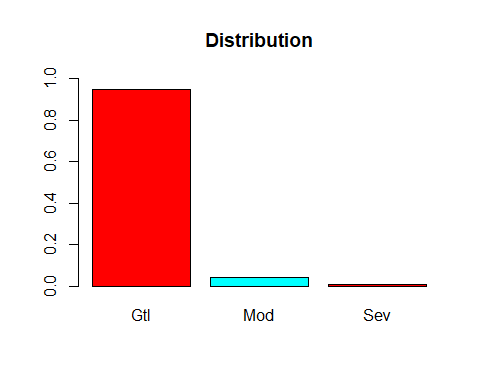


barplot(prop.table(table(ames$Lot\_Config)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

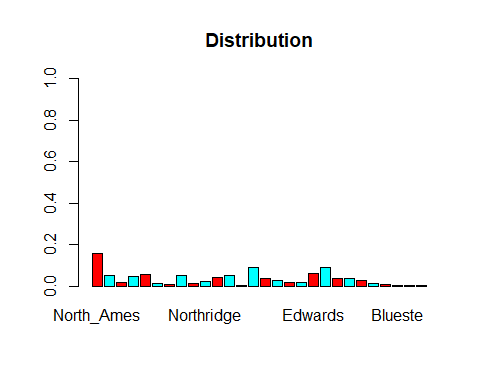


####Remove(Land\_Slope,Condition\_1,Condition\_2,Bldg\_Type)

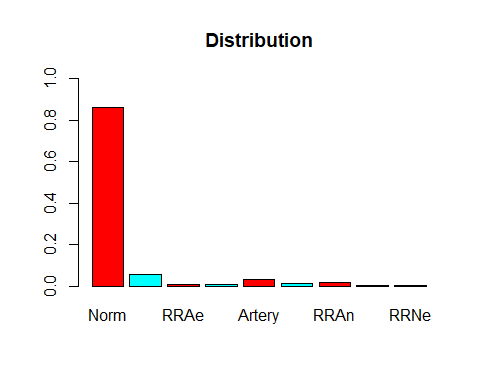
barplot(prop.table(table(ames$Land\_Slope)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



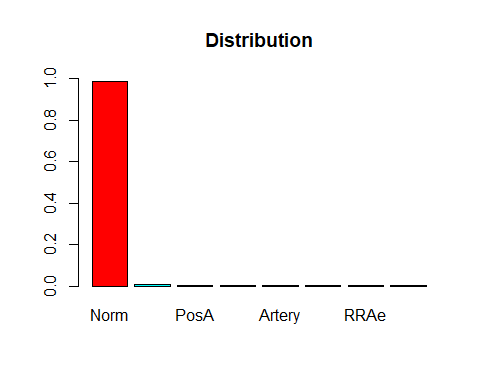
barplot(prop.table(table(ames$Neighborhood)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



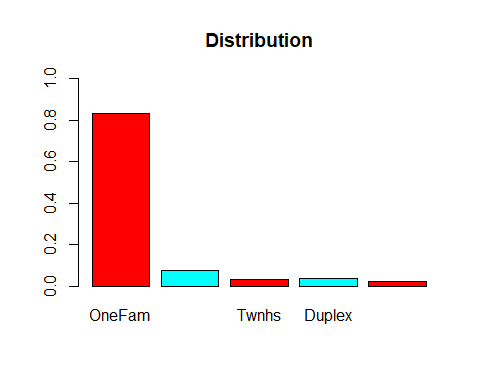
barplot(prop.table(table(ames$Condition\_1)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



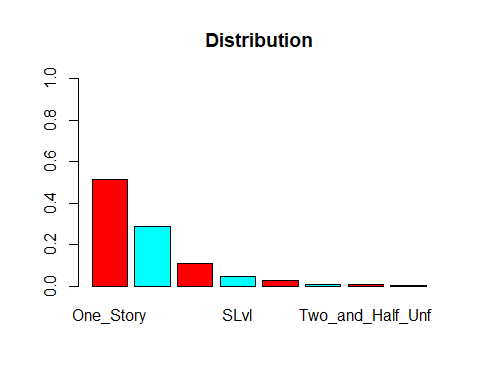
barplot(prop.table(table(ames$Condition\_2)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



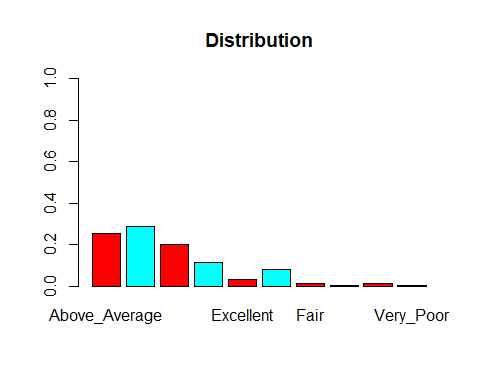
barplot(prop.table(table(ames$Bldg\_Type)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



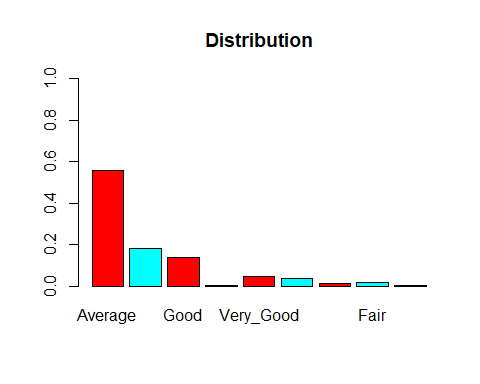
barplot(prop.table(table(ames$House\_Style)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



barplot(prop.table(table(ames$Overall\_Qual)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

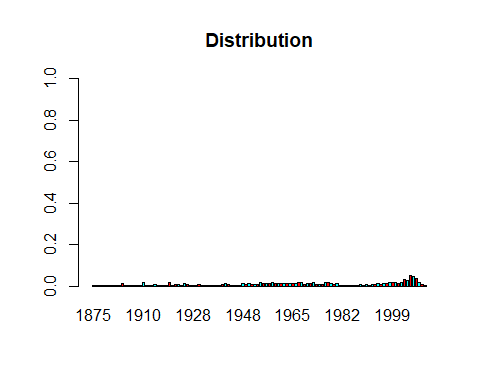


barplot(prop.table(table(ames$Overall\_Cond)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

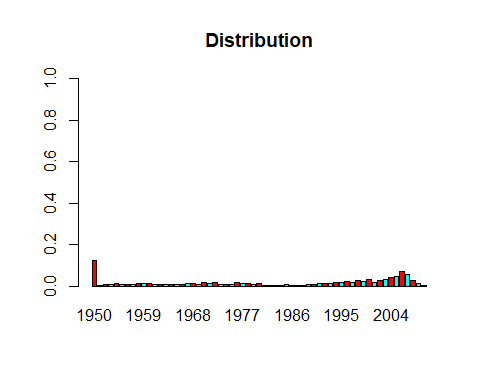


####Remove(Roof\_Matl,Roof\_Style)

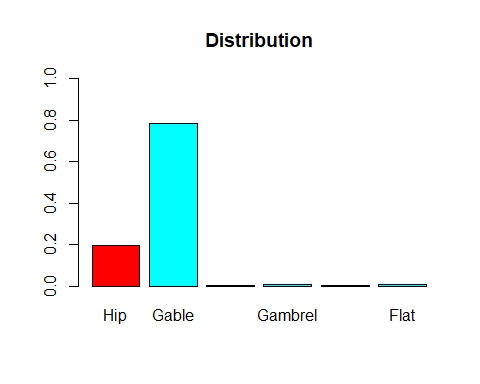
barplot(prop.table(table(ames$Year\_Built)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



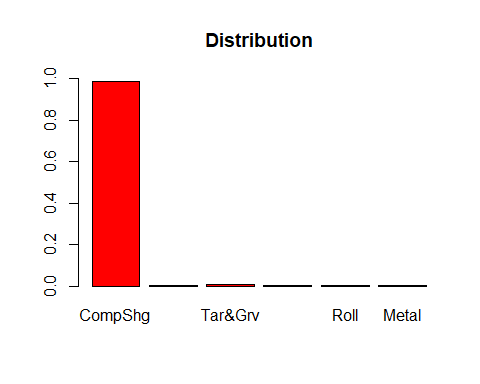
barplot(prop.table(table(ames$Year\_Remod\_Add)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



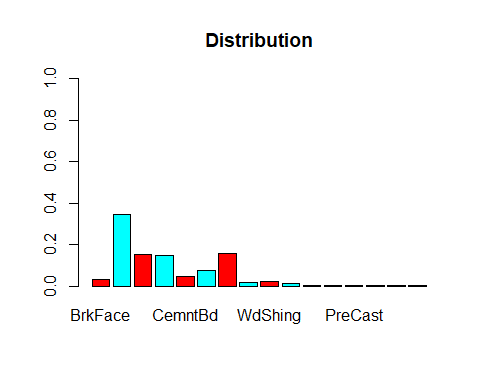
barplot(prop.table(table(ames$Roof\_Style)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



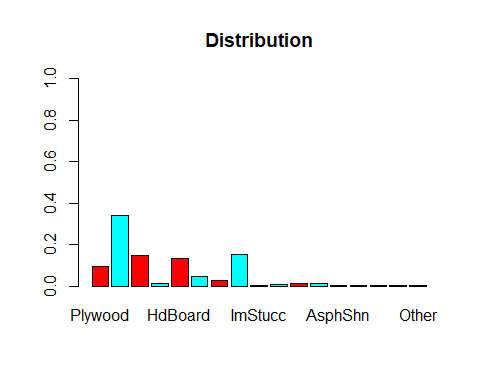
barplot(prop.table(table(ames$Roof\_Matl)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



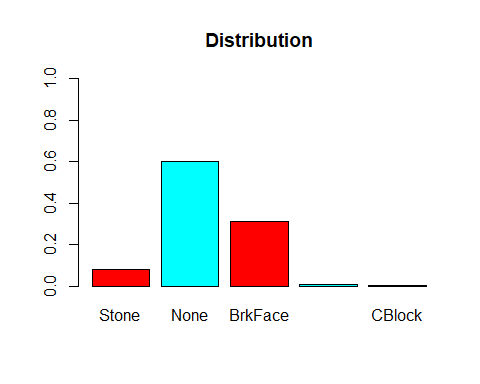
barplot(prop.table(table(ames$Exterior\_1st)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



barplot(prop.table(table(ames$Exterior\_2nd)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

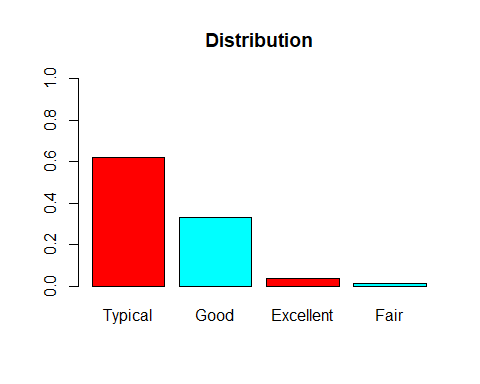


barplot(prop.table(table(ames$Mas\_Vnr\_Type)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

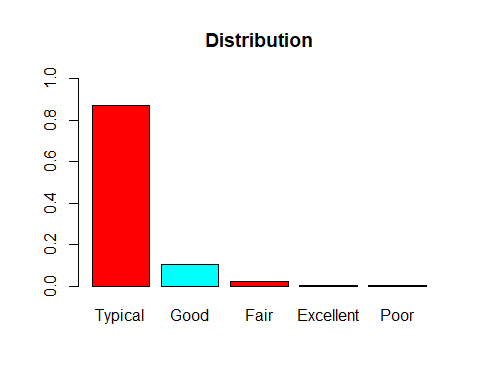


####Remove(Exter\_Cond,Bsmt\_Cond,Bsmt\_Exposure Heating)

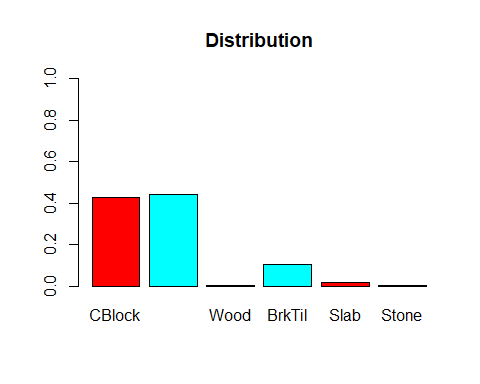
barplot(prop.table(table(ames$Exter\_Qual)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



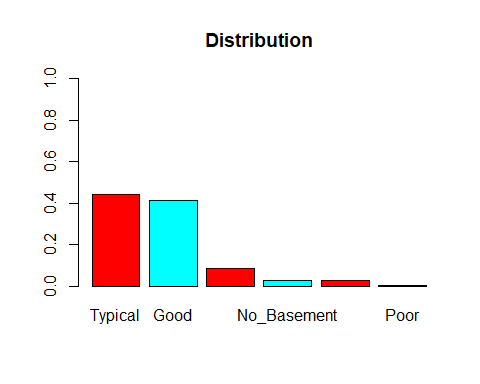
barplot(prop.table(table(ames$Exter\_Cond)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



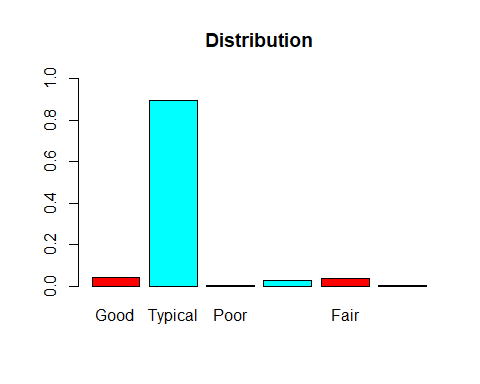
barplot(prop.table(table(ames$Foundation)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



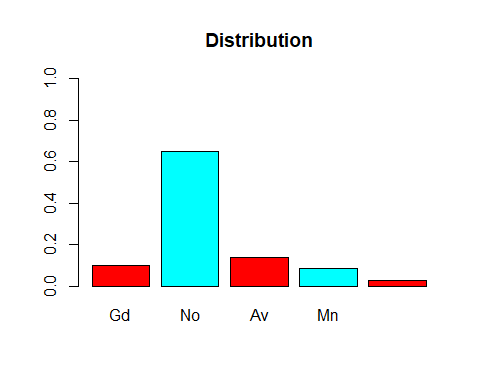
barplot(prop.table(table(ames$Bsmt\_Qual)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



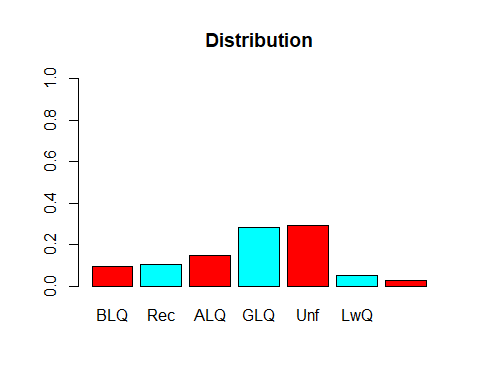
barplot(prop.table(table(ames$Bsmt\_Cond)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



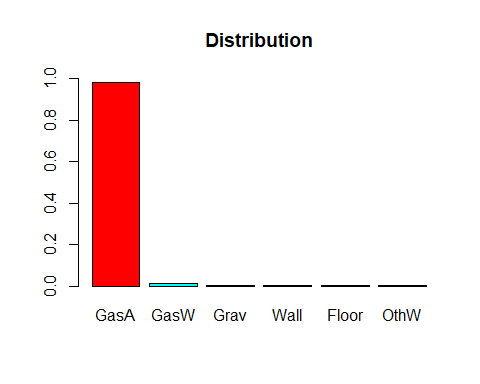
barplot(prop.table(table(ames$Bsmt\_Exposure)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



barplot(prop.table(table(ames$BsmtFin\_Type\_1)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

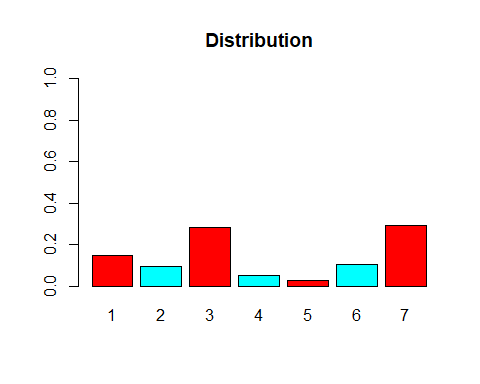


barplot(prop.table(table(ames$Heating)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

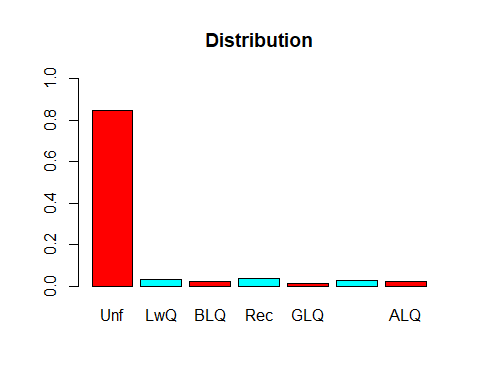


####Remove(BsmtFin\_Type\_2)

barplot(prop.table(table(ames$BsmtFin\_SF\_1)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

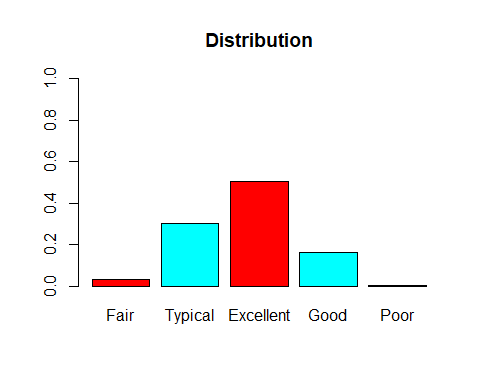


barplot(prop.table(table(ames$BsmtFin\_Type\_2)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

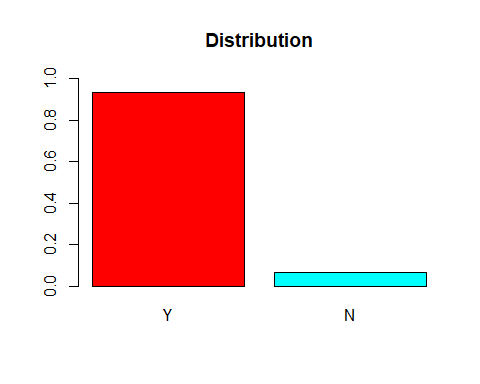


####Remove(Central\_Air,Electrical,Bsmt\_Half\_Bath)

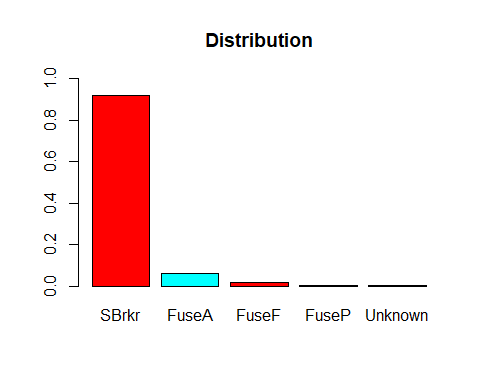
barplot(prop.table(table(ames$Heating\_QC)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



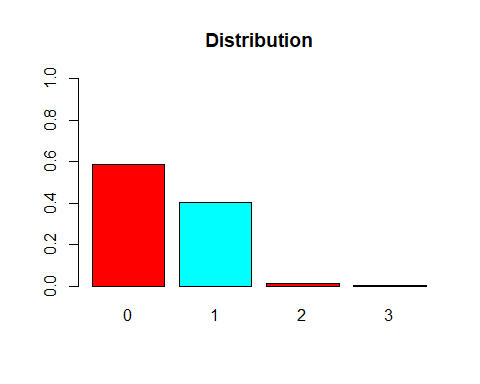
barplot(prop.table(table(ames$Central\_Air)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



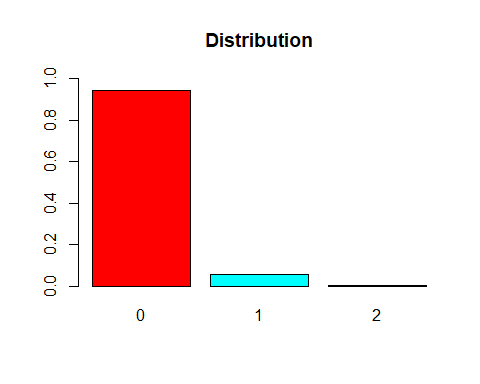
barplot(prop.table(table(ames$Electrical)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



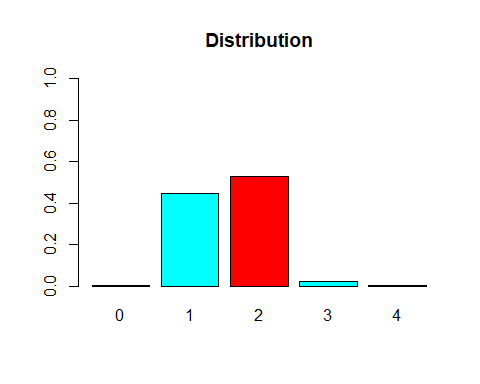
barplot(prop.table(table(ames$Bsmt\_Full\_Bath)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



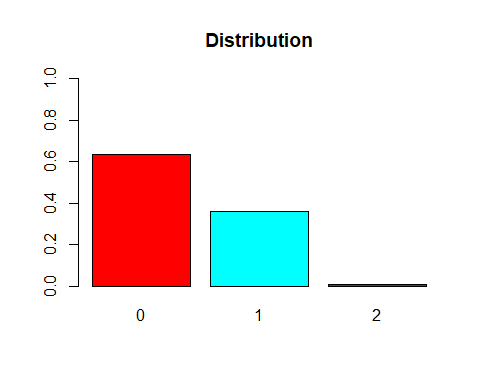
barplot(prop.table(table(ames$Bsmt\_Half\_Bath)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



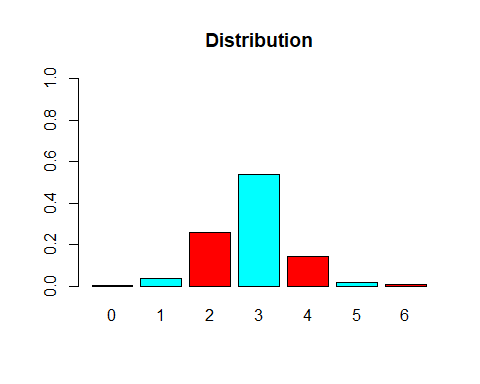
barplot(prop.table(table(ames$Full\_Bath)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



barplot(prop.table(table(ames$Half\_Bath)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

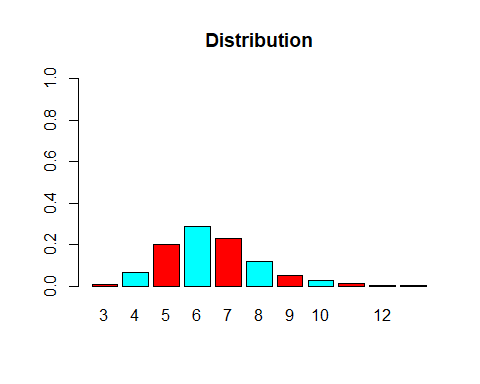


barplot(prop.table(table(ames$Bedroom\_AbvGr)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

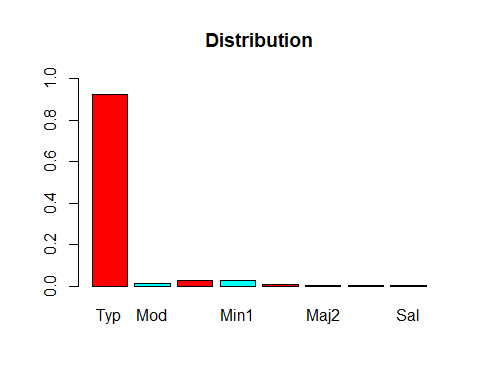


####Remove(Functional)

barplot(prop.table(table(ames$TotRms\_AbvGrd)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

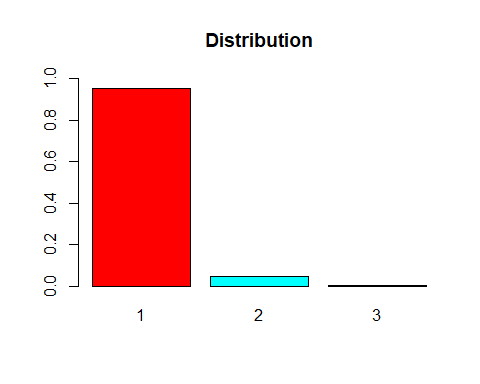


barplot(prop.table(table(ames$Functional)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

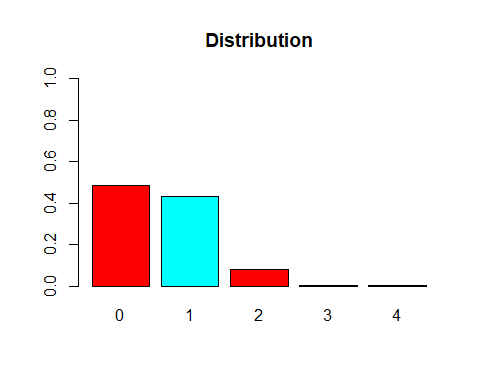


####Remove(Kitchen\_AbvGr,Garage\_Qual,Garage\_Cond)

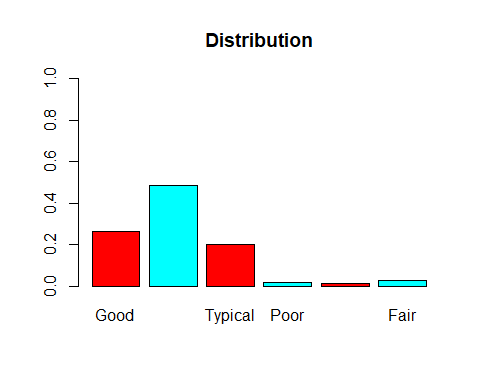
barplot(prop.table(table(ames$Kitchen\_AbvGr)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



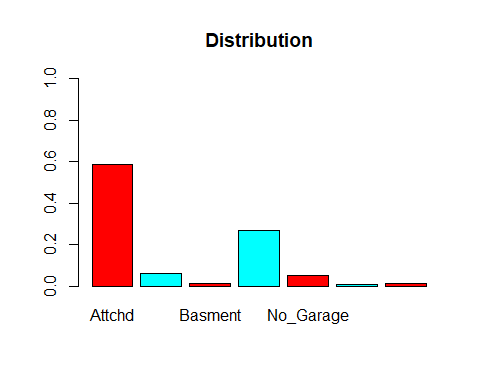
barplot(prop.table(table(ames$Fireplaces)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



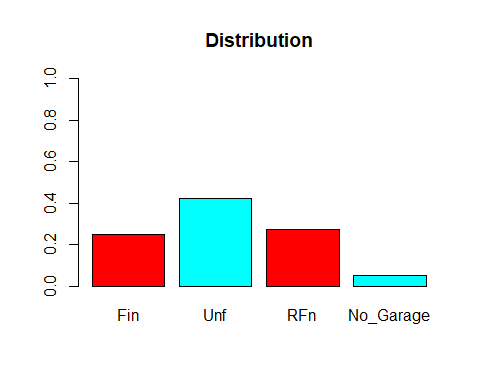
barplot(prop.table(table(ames$Fireplace\_Qu)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



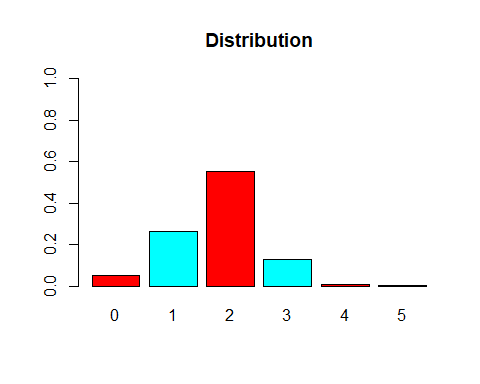
barplot(prop.table(table(ames$Garage\_Type)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



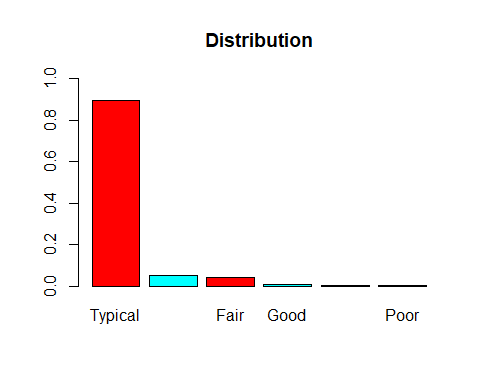
barplot(prop.table(table(ames$Garage\_Finish)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



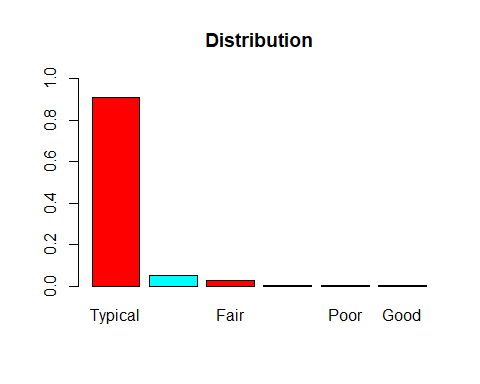
barplot(prop.table(table(ames$Garage\_Cars)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



barplot(prop.table(table(ames$Garage\_Qual)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

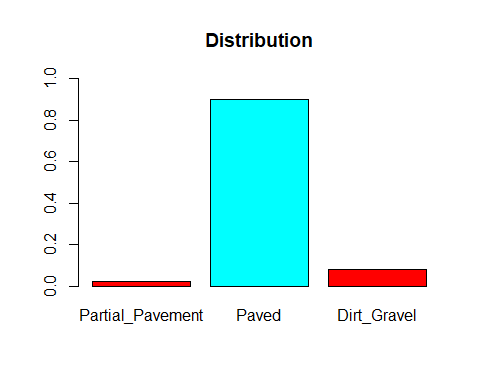


barplot(prop.table(table(ames$Garage\_Cond)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")

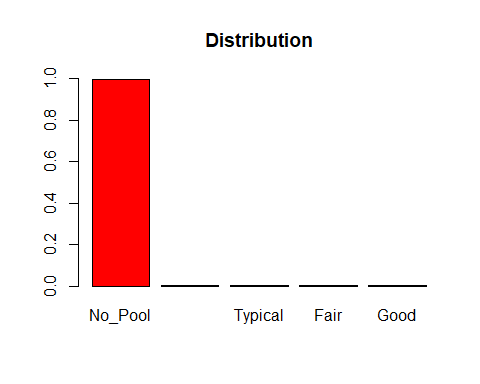


####Remove(Paved\_Drive,Pool\_QC,Fence, Misc\_Feature,Sale\_Type,Sale\_Condition)

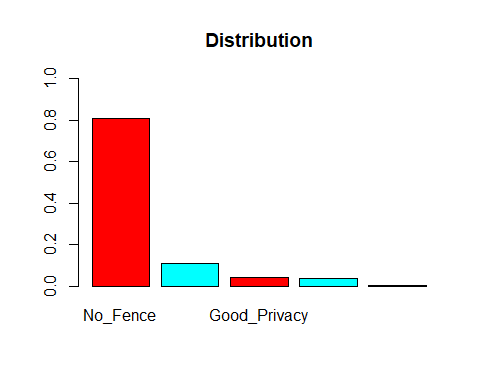
barplot(prop.table(table(ames$Paved\_Drive)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



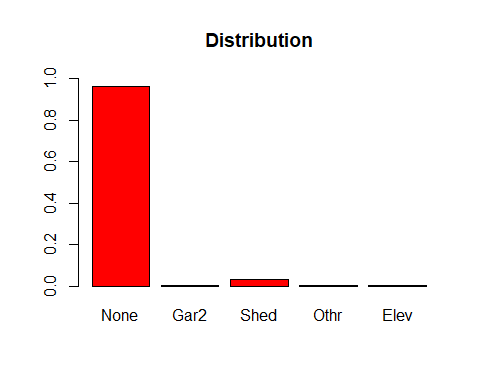
barplot(prop.table(table(ames$Pool\_QC)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



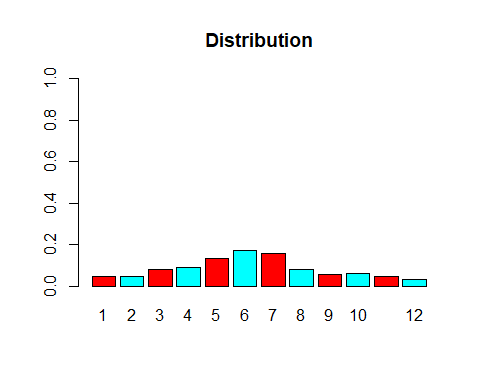
barplot(prop.table(table(ames$Fence)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



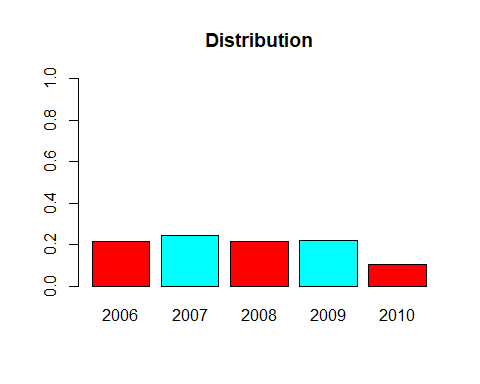
barplot(prop.table(table(ames$Misc\_Feature)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



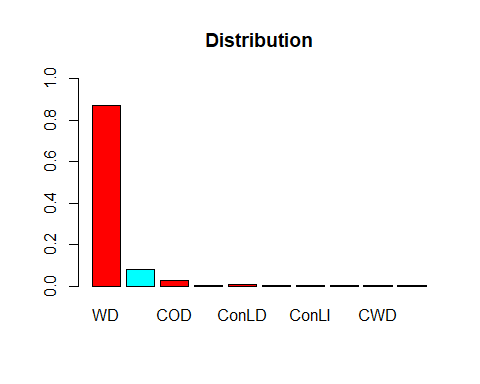
barplot(prop.table(table(ames$Mo\_Sold)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



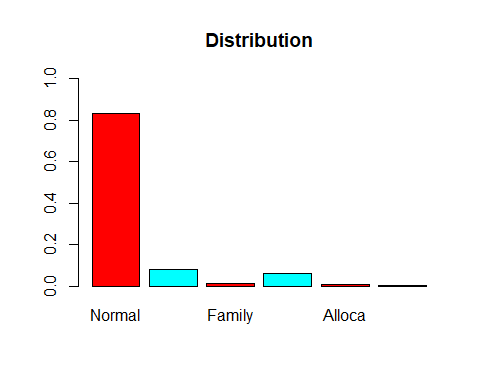
barplot(prop.table(table(ames$Year\_Sold)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



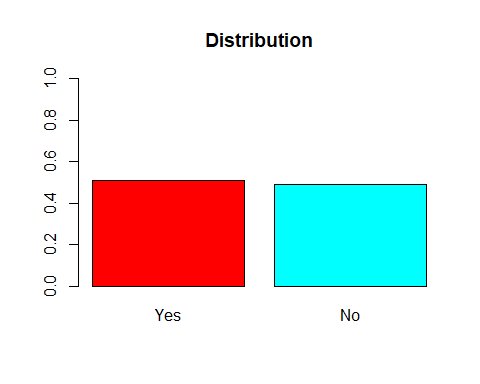
barplot(prop.table(table(ames$Sale\_Type)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



barplot(prop.table(table(ames$Sale\_Condition)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



barplot(prop.table(table(ames$Above\_Median)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Distribution")



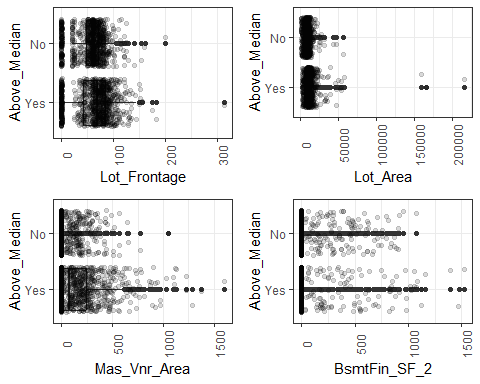
###Remove Imbalanced Catagorical Variables

ames2 <- select(ames, - c(MS\_Zoning,Street,Alley,Land\_Contour,Utilities,Land\_Slope,Condition\_1,Condition\_2,Bldg\_Type,Roof\_Matl,Exter\_Cond,Bsmt\_Cond, Heating,Central\_Air,Electrical,Bsmt\_Half\_Bath,Kitchen\_AbvGr,Garage\_Qual,Garage\_Cond,Paved\_Drive,Pool\_QC,Fence, Misc\_Feature,Sale\_Type,Sale\_Condition,Latitude,Longitude,Lot\_Config,Roof\_Style,Bsmt\_Exposure,Garage\_Type,Pool\_Area,BsmtFin\_Type\_2,Functional))

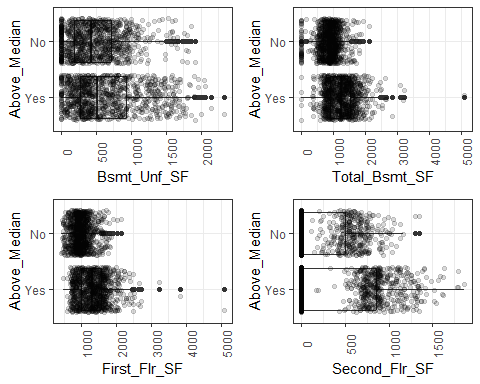
###Numerical Significance Box Plots

####(Remove(Lot\_Frontage,Lot\_Area,BsmtFin\_SF\_2))

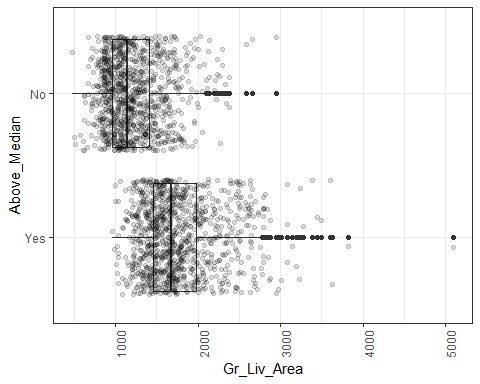
p1=ggplot(ames2,aes(x=Lot\_Frontage,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
p2=ggplot(ames2,aes(x=Lot\_Area,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
p3=ggplot(ames2,aes(x=Mas\_Vnr\_Area,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
p4=ggplot(ames2,aes(x=BsmtFin\_SF\_2,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
grid.arrange(p1,p2,p3,p4)

 ####(Remove(Bsmt\_Unf\_SF,Second\_Flr\_SF))

p1=ggplot(ames2,aes(x=Bsmt\_Unf\_SF,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
p2=ggplot(ames2,aes(x=Total\_Bsmt\_SF,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
p3=ggplot(ames2,aes(x=First\_Flr\_SF,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
p4=ggplot(ames2,aes(x=Second\_Flr\_SF,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
grid.arrange(p1,p2,p3,p4)

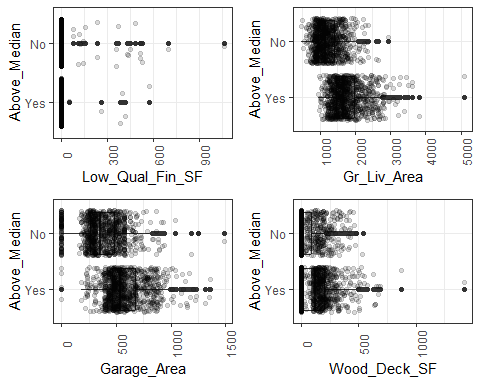


ggplot(ames2,aes(x=Gr\_Liv\_Area,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))



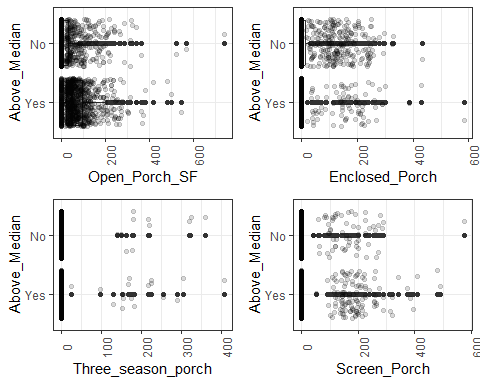
####(Remove(Low\_Qual\_Fin\_SF))

p1=ggplot(ames2,aes(x=Low\_Qual\_Fin\_SF,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
p2=ggplot(ames2,aes(x=Gr\_Liv\_Area,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
p3=ggplot(ames2,aes(x=Garage\_Area,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
p4=ggplot(ames2,aes(x=Wood\_Deck\_SF,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
grid.arrange(p1,p2,p3,p4)



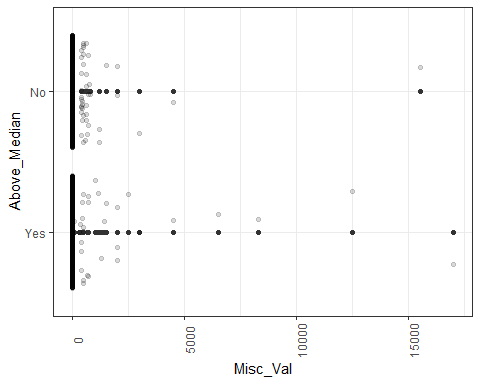
####(Remove(Enclosed\_Porch,Three\_season\_porch,Screen\_Porch))

p1=ggplot(ames2,aes(x=Open\_Porch\_SF,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
p2=ggplot(ames2,aes(x=Enclosed\_Porch,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
p3=ggplot(ames2,aes(x=Three\_season\_porch,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
p4=ggplot(ames2,aes(x=Screen\_Porch,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))  
grid.arrange(p1,p2,p3,p4)



####(Remove(Misc\_Val))

ggplot(ames2,aes(x=Misc\_Val,y=Above\_Median)) + geom\_boxplot() + geom\_jitter(alpha = 0.15) + theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90))

 ####Remove non-significant numeric values

ames2 <- select(ames2, - c(Lot\_Frontage,Lot\_Area,BsmtFin\_SF\_2,Bsmt\_Unf\_SF,Second\_Flr\_SF,Low\_Qual\_Fin\_SF,Enclosed\_Porch,Three\_season\_porch,Screen\_Porch,Misc\_Val))

#multicollinearity test

####Remove(Gr\_Liv\_Area,First\_Flr\_SF,Total\_Bsmt\_SF)

corrdata<- select(ames2, c (Gr\_Liv\_Area,Garage\_Area,Wood\_Deck\_SF,Open\_Porch\_SF,Mas\_Vnr\_Area,Total\_Bsmt\_SF,First\_Flr\_SF))  
head(corrdata)

## # A tibble: 6 × 7  
## Gr\_Liv\_Area Garage\_Area Wood\_Deck\_SF Open\_Porch\_SF Mas\_Vnr\_A…¹ Total…² First…³  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1656 528 210 62 112 1080 1656  
## 2 896 730 140 0 0 882 896  
## 3 1329 312 393 36 108 1329 1329  
## 4 2110 522 0 0 0 2110 2110  
## 5 1629 482 212 34 0 928 928  
## 6 1604 470 360 36 20 926 926  
## # … with abbreviated variable names ¹​Mas\_Vnr\_Area, ²​Total\_Bsmt\_SF,  
## # ³​First\_Flr\_SF

cor(corrdata)

## Gr\_Liv\_Area Garage\_Area Wood\_Deck\_SF Open\_Porch\_SF Mas\_Vnr\_Area  
## Gr\_Liv\_Area 1.0000000 0.4698324 0.24947385 0.32003486 0.4134423  
## Garage\_Area 0.4698324 1.0000000 0.23885523 0.20980706 0.3647154  
## Wood\_Deck\_SF 0.2494738 0.2388552 1.00000000 0.04207457 0.1864713  
## Open\_Porch\_SF 0.3200349 0.2098071 0.04207457 1.00000000 0.1211212  
## Mas\_Vnr\_Area 0.4134423 0.3647154 0.18647125 0.12112122 1.0000000  
## Total\_Bsmt\_SF 0.4177324 0.4752770 0.23916781 0.23566051 0.4035991  
## First\_Flr\_SF 0.5511429 0.4864755 0.23395662 0.22844243 0.4010746  
## Total\_Bsmt\_SF First\_Flr\_SF  
## Gr\_Liv\_Area 0.4177324 0.5511429  
## Garage\_Area 0.4752770 0.4864755  
## Wood\_Deck\_SF 0.2391678 0.2339566  
## Open\_Porch\_SF 0.2356605 0.2284424  
## Mas\_Vnr\_Area 0.4035991 0.4010746  
## Total\_Bsmt\_SF 1.0000000 0.7814968  
## First\_Flr\_SF 0.7814968 1.0000000

####Remove variables with multicollinearity

ames2 <- select(ames2, - c(Gr\_Liv\_Area,First\_Flr\_SF,Total\_Bsmt\_SF))

####Remove variables that lack significance in initial Log reg

ames2 <- select(ames2, - c(Year\_Sold,Mo\_Sold,Open\_Porch\_SF,Garage\_Cars,Kitchen\_Qual,Bedroom\_AbvGr,Full\_Bath,Bsmt\_Full\_Bath,Heating\_QC,BsmtFin\_Type\_1,Bsmt\_Qual,Mas\_Vnr\_Area,Mas\_Vnr\_Type,Exterior\_2nd,Exterior\_1st,Lot\_Shape,MS\_SubClass,House\_Style,Overall\_Cond,Year\_Built,Half\_Bath,Fireplaces,Exter\_Qual))

####rewriting only significant levels of varaibles

ames2<-ames2 %>%  
 mutate(PConc\_Foundation = case\_when(Foundation == "PConc" ~ 'Yes',  
 Foundation == "CBlock" | Foundation == "Wood" | Foundation == "BrkTil" |Foundation  
 == "Stone" |Foundation == "Slab" ~ "No"))  
ames2<-ames2 %>%  
 mutate(BsmtFin\_SF\_17 = case\_when(BsmtFin\_SF\_1 == "7" ~ '7',  
 BsmtFin\_SF\_1 == "2" | BsmtFin\_SF\_1 == "3" | BsmtFin\_SF\_1 == "4" | BsmtFin\_SF\_1 ==  
 "5" | BsmtFin\_SF\_1 == "6" | BsmtFin\_SF\_1 == "1" ~ "Other"))   
  
ames2<-ames2 %>%  
 mutate(Fireplace\_Qu\_1 = case\_when(Fireplace\_Qu == "No\_Fireplace" | Fireplace\_Qu == "Poor" | Fireplace\_Qu == "Fair" ~ 'Subpar', Fireplace\_Qu == "Typical" | Fireplace\_Qu == "Excellent" | Fireplace\_Qu == "Good" ~ "Good"))   
  
ames2<-ames2 %>%  
 mutate(Unfinished\_Garage = case\_when(Garage\_Finish == "Unf" ~ 'Unfinished',  
 Garage\_Finish == "Fin" | Garage\_Finish == "RFn" | Garage\_Finish == "No\_Garage" ~ "Other"))   
  
ames2<-ames2 %>%  
 mutate(Overall\_Qual2 = case\_when(Overall\_Qual == "Average" ~ 'Average', Overall\_Qual == "Good" ~ 'Good', Overall\_Qual == "Very\_Good" ~ 'Very Good', Overall\_Qual == "Below\_Average" ~ 'Below Average', Overall\_Qual == "Above Average" | Overall\_Qual == "Excellent" | Overall\_Qual == "Fair" | Overall\_Qual == "Poor" | Overall\_Qual == "Very\_Excellent" | Overall\_Qual == "Very\_Poor" | Overall\_Qual == "Above\_Average" ~ "Other"))   
  
ames2<-ames2 %>%  
 mutate(TotRms\_AbvGrd2 = case\_when(TotRms\_AbvGrd == "6" ~ '6', TotRms\_AbvGrd == "7" ~ '7', TotRms\_AbvGrd == "8" ~ '8', TotRms\_AbvGrd == "15" | TotRms\_AbvGrd == "12" | TotRms\_AbvGrd == "3" | TotRms\_AbvGrd == "4" | TotRms\_AbvGrd == "5" | TotRms\_AbvGrd == "11" | TotRms\_AbvGrd == "10" | TotRms\_AbvGrd == "9" ~ "Other"))

####Remove overlapping variables

ames2 <- select(ames2, - c(Foundation,BsmtFin\_SF\_1,Fireplace\_Qu,Garage\_Finish,Overall\_Qual,TotRms\_AbvGrd))

###Test and Train Set

set.seed(123)   
Median\_split = initial\_split(ames2, prob = 0.80, strata = Above\_Median)  
train = training(Median\_split)  
test = testing(Median\_split)

#Log Reg Model

Log\_model =   
 logistic\_reg(mode = "classification") %>% #note the use of logistic\_reg and mode = "classification"  
 set\_engine("glm") #standard logistic regression engine is glm  
  
Median\_recipe = recipe(Above\_Median ~., train) %>%  
 step\_other(Neighborhood,Year\_Remod\_Add, threshold = 0.1) %>%  
 step\_dummy(Neighborhood,Year\_Remod\_Add)  
  
logreg\_wf = workflow() %>%  
 add\_recipe(Median\_recipe) %>%   
 add\_model(Log\_model)  
  
Median\_fit = fit(logreg\_wf, train)

summary(Median\_fit$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.2066 -0.2815 -0.0161 0.3183 2.9226   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 4.5463066 0.5632068 8.072 6.91e-16 \*\*\*  
## Garage\_Area -0.0038180 0.0005546 -6.884 5.83e-12 \*\*\*  
## Wood\_Deck\_SF -0.0024316 0.0007559 -3.217 0.001296 \*\*   
## PConc\_FoundationYes -1.4786435 0.2122067 -6.968 3.22e-12 \*\*\*  
## BsmtFin\_SF\_17Other -0.7228966 0.2240433 -3.227 0.001253 \*\*   
## Fireplace\_Qu\_1Subpar 1.7901897 0.1911149 9.367 < 2e-16 \*\*\*  
## Unfinished\_GarageUnfinished 1.0261404 0.1949734 5.263 1.42e-07 \*\*\*  
## Overall\_Qual2Below Average 1.0606718 0.5073219 2.091 0.036552 \*   
## Overall\_Qual2Good -2.5051540 0.2868055 -8.735 < 2e-16 \*\*\*  
## Overall\_Qual2Other -1.0111061 0.2135585 -4.735 2.20e-06 \*\*\*  
## Overall\_Qual2Very Good -4.7680585 1.0648058 -4.478 7.54e-06 \*\*\*  
## TotRms\_AbvGrd27 -0.8674737 0.2510292 -3.456 0.000549 \*\*\*  
## TotRms\_AbvGrd28 -1.0853205 0.3381198 -3.210 0.001328 \*\*   
## TotRms\_AbvGrd2Other 0.5150167 0.2203582 2.337 0.019430 \*   
## Neighborhood\_other -0.6680064 0.2486905 -2.686 0.007229 \*\*   
## Year\_Remod\_Add\_other -1.3859853 0.3337738 -4.152 3.29e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2133.10 on 1538 degrees of freedom  
## Residual deviance: 808.08 on 1523 degrees of freedom  
## AIC: 840.08  
##   
## Number of Fisher Scoring iterations: 8

###Train Accuracy Test

Logpred = predict(Median\_fit, train, type = "class")  
head(Logpred)

## # A tibble: 6 × 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 No   
## 3 No   
## 4 No   
## 5 No   
## 6 No

confusionMatrix(Logpred$.pred\_class,train$Above\_Median,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 682 88  
## No 100 669  
##   
## Accuracy : 0.8778   
## 95% CI : (0.8604, 0.8938)  
## No Information Rate : 0.5081   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.7557   
##   
## Mcnemar's Test P-Value : 0.4224   
##   
## Sensitivity : 0.8721   
## Specificity : 0.8838   
## Pos Pred Value : 0.8857   
## Neg Pred Value : 0.8700   
## Prevalence : 0.5081   
## Detection Rate : 0.4431   
## Detection Prevalence : 0.5003   
## Balanced Accuracy : 0.8779   
##   
## 'Positive' Class : Yes   
##

###Test Accuracy Test

Logpred = predict(Median\_fit, test, type = "class")  
head(Logpred)

## # A tibble: 6 × 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 Yes   
## 3 Yes   
## 4 Yes   
## 5 Yes   
## 6 Yes

confusionMatrix(Logpred$.pred\_class,test$Above\_Median,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 236 27  
## No 25 226  
##   
## Accuracy : 0.8988   
## 95% CI : (0.8695, 0.9235)  
## No Information Rate : 0.5078   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.7976   
##   
## Mcnemar's Test P-Value : 0.8897   
##   
## Sensitivity : 0.9042   
## Specificity : 0.8933   
## Pos Pred Value : 0.8973   
## Neg Pred Value : 0.9004   
## Prevalence : 0.5078   
## Detection Rate : 0.4591   
## Detection Prevalence : 0.5117   
## Balanced Accuracy : 0.8987   
##   
## 'Positive' Class : Yes   
##

#Tree Model

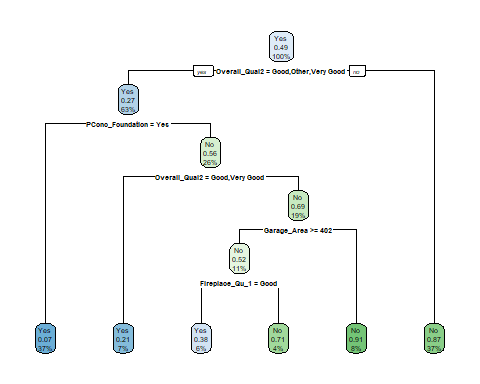
Median\_recipe = recipe(Above\_Median ~., train) %>%  
 step\_other(Neighborhood,Year\_Remod\_Add, threshold = 0.1) %>%  
 step\_dummy(Neighborhood,Year\_Remod\_Add)  
  
tree\_model = decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification")  
  
Median\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(Median\_recipe)  
  
Median\_fit = fit(Median\_wflow, train)

#look at the tree's fit  
Median\_fit %>%  
 pull\_workflow\_fit() %>%  
 pluck("fit")

## Warning: `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.  
## ℹ Please use `extract\_fit\_parsnip()` instead.

## n= 1539   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 1539 757 Yes (0.50812216 0.49187784)   
## 2) Overall\_Qual2=Good,Other,Very Good 974 266 Yes (0.72689938 0.27310062)   
## 4) PConc\_Foundation=Yes 576 43 Yes (0.92534722 0.07465278) \*  
## 5) PConc\_Foundation=No 398 175 No (0.43969849 0.56030151)   
## 10) Overall\_Qual2=Good,Very Good 106 22 Yes (0.79245283 0.20754717) \*  
## 11) Overall\_Qual2=Other 292 91 No (0.31164384 0.68835616)   
## 22) Garage\_Area>=402 166 80 No (0.48192771 0.51807229)   
## 44) Fireplace\_Qu\_1=Good 97 37 Yes (0.61855670 0.38144330) \*  
## 45) Fireplace\_Qu\_1=Subpar 69 20 No (0.28985507 0.71014493) \*  
## 23) Garage\_Area< 402 126 11 No (0.08730159 0.91269841) \*  
## 3) Overall\_Qual2=Average,Below Average 565 74 No (0.13097345 0.86902655) \*

#extract the tree's fit from the fit object  
tree = Median\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
#plot the tree  
rpart.plot(tree)



Median\_fit$fit$fit$fit$cptable

## CP nsplit rel error xerror xstd  
## 1 0.55085865 0 1.0000000 1.0369881 0.02590630  
## 2 0.07265522 1 0.4491413 0.4808454 0.02202188  
## 3 0.01519155 3 0.3038309 0.3513871 0.01959475  
## 4 0.01000000 5 0.2734478 0.3196830 0.01886525

set.seed(123)  
folds = vfold\_cv(train, v = 5)

tree\_model = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification")  
  
tree\_grid = grid\_regular(cost\_complexity(),  
 levels = 25) #try 25 sensible values for cp  
  
Median\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(Median\_recipe)  
  
tree\_res =   
 Median\_wflow %>%   
 tune\_grid(  
 resamples = folds,  
 grid = tree\_grid  
 )  
  
tree\_res

## # Tuning results  
## # 5-fold cross-validation   
## # A tibble: 5 × 4  
## splits id .metrics .notes   
## <list> <chr> <list> <list>   
## 1 <split [1231/308]> Fold1 <tibble [50 × 5]> <tibble [0 × 3]>  
## 2 <split [1231/308]> Fold2 <tibble [50 × 5]> <tibble [0 × 3]>  
## 3 <split [1231/308]> Fold3 <tibble [50 × 5]> <tibble [0 × 3]>  
## 4 <split [1231/308]> Fold4 <tibble [50 × 5]> <tibble [0 × 3]>  
## 5 <split [1232/307]> Fold5 <tibble [50 × 5]> <tibble [0 × 3]>

tree\_res %>%   
 collect\_metrics()

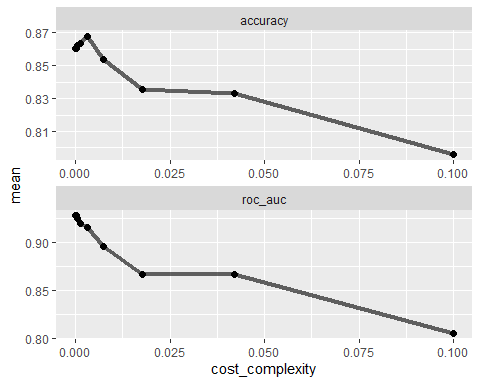
## # A tibble: 50 × 7  
## cost\_complexity .metric .estimator mean n std\_err .config   
## <dbl> <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 1 e-10 accuracy binary 0.860 5 0.00252 Preprocessor1\_Model01  
## 2 1 e-10 roc\_auc binary 0.928 5 0.00883 Preprocessor1\_Model01  
## 3 2.37e-10 accuracy binary 0.860 5 0.00252 Preprocessor1\_Model02  
## 4 2.37e-10 roc\_auc binary 0.928 5 0.00883 Preprocessor1\_Model02  
## 5 5.62e-10 accuracy binary 0.860 5 0.00252 Preprocessor1\_Model03  
## 6 5.62e-10 roc\_auc binary 0.928 5 0.00883 Preprocessor1\_Model03  
## 7 1.33e- 9 accuracy binary 0.860 5 0.00252 Preprocessor1\_Model04  
## 8 1.33e- 9 roc\_auc binary 0.928 5 0.00883 Preprocessor1\_Model04  
## 9 3.16e- 9 accuracy binary 0.860 5 0.00252 Preprocessor1\_Model05  
## 10 3.16e- 9 roc\_auc binary 0.928 5 0.00883 Preprocessor1\_Model05  
## # … with 40 more rows

cost\_complexity()

## Cost-Complexity Parameter (quantitative)  
## Transformer: log-10 [1e-100, Inf]  
## Range (transformed scale): [-10, -1]

tree\_res %>%  
 collect\_metrics() %>%  
 ggplot(aes(cost\_complexity, mean)) +  
 geom\_line(size = 1.5, alpha = 0.6) +  
 geom\_point(size = 2) +  
 facet\_wrap(~ .metric, scales = "free", nrow = 2)

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.

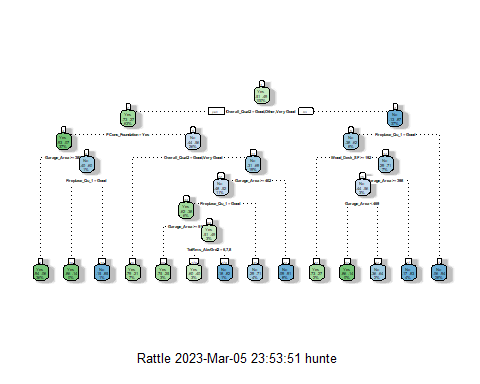


best\_tree = tree\_res %>%  
 select\_best("accuracy")  
  
best\_tree

## # A tibble: 1 × 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.00316 Preprocessor1\_Model21

final\_wf =   
 Median\_wflow %>%   
 finalize\_workflow(best\_tree)

final\_fit = fit(final\_wf, train)  
  
tree = final\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
fancyRpartPlot(tree, tweak = 1.5)



###Train Accuracy Test

treepred = predict(final\_fit, train, type = "class")  
head(treepred)

## # A tibble: 6 × 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 No   
## 3 No   
## 4 No   
## 5 No   
## 6 No

confusionMatrix(treepred$.pred\_class,train$Above\_Median,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 698 90  
## No 84 667  
##   
## Accuracy : 0.8869   
## 95% CI : (0.8701, 0.9023)  
## No Information Rate : 0.5081   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.7738   
##   
## Mcnemar's Test P-Value : 0.7047   
##   
## Sensitivity : 0.8926   
## Specificity : 0.8811   
## Pos Pred Value : 0.8858   
## Neg Pred Value : 0.8881   
## Prevalence : 0.5081   
## Detection Rate : 0.4535   
## Detection Prevalence : 0.5120   
## Balanced Accuracy : 0.8868   
##   
## 'Positive' Class : Yes   
##

###Test Accuracy Test

treepred = predict(final\_fit, test, type = "class")  
head(treepred)

## # A tibble: 6 × 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 Yes   
## 3 Yes   
## 4 Yes   
## 5 Yes   
## 6 Yes

confusionMatrix(treepred$.pred\_class,test$Above\_Median,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 240 32  
## No 21 221  
##   
## Accuracy : 0.8969   
## 95% CI : (0.8673, 0.9218)  
## No Information Rate : 0.5078   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.7936   
##   
## Mcnemar's Test P-Value : 0.1696   
##   
## Sensitivity : 0.9195   
## Specificity : 0.8735   
## Pos Pred Value : 0.8824   
## Neg Pred Value : 0.9132   
## Prevalence : 0.5078   
## Detection Rate : 0.4669   
## Detection Prevalence : 0.5292   
## Balanced Accuracy : 0.8965   
##   
## 'Positive' Class : Yes   
##

#Random Forest

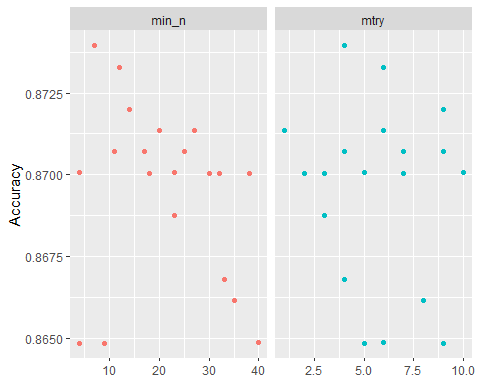
set.seed(123)  
rf\_folds = vfold\_cv(train, v = 5)

rf\_model = rand\_forest(mtry = tune(), min\_n = tune(), trees = 100) %>% #add tuning of mtry and min\_n parameters  
 #setting trees to 100 here should also speed things up a bit, but more trees might be better  
 set\_engine("ranger", importance = "permutation") %>% #added importance metric  
 set\_mode("classification")  
  
Median\_wflow =   
 workflow() %>%   
 add\_model(rf\_model) %>%   
 add\_recipe(Median\_recipe)  
  
set.seed(123)  
rf\_res = tune\_grid(  
 Median\_wflow,  
 resamples = rf\_folds,  
 grid = 20 #try 20 different combinations of the random forest tuning parameters  
)

## i Creating pre-processing data to finalize unknown parameter: mtry

###Look at parameter performance (borrowed from <https://juliasilge.com/blog/sf-trees-random-tuning/>)

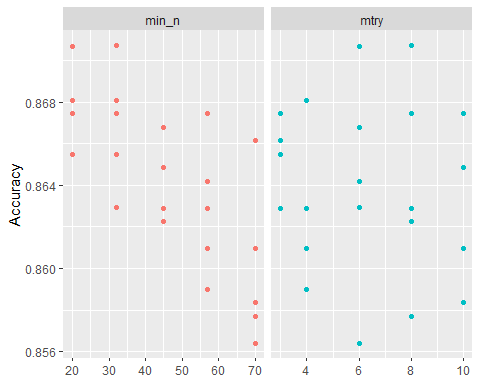
rf\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 select(mean, min\_n, mtry) %>%  
 pivot\_longer(min\_n:mtry,  
 values\_to = "value",  
 names\_to = "parameter"  
 ) %>%  
 ggplot(aes(value, mean, color = parameter)) +  
 geom\_point(show.legend = FALSE) +  
 facet\_wrap(~parameter, scales = "free\_x") +  
 labs(x = NULL, y = "Accuracy")



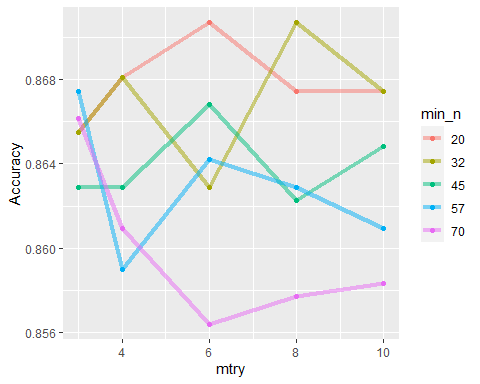
###Refining the parameters

rf\_model = rand\_forest(mtry = tune(), min\_n = tune(), trees = 100) %>% #add tuning of mtry and min\_n parameters  
 #setting trees to 100 here should also speed things up a bit, but more trees might be better  
 set\_engine("ranger", importance = "permutation") %>% #added importance metric  
 set\_mode("classification")  
  
Median\_wflow =   
 workflow() %>%   
 add\_model(rf\_model) %>%   
 add\_recipe(Median\_recipe)  
  
rf\_grid = grid\_regular(  
 mtry(range = c(3, 10)), #these values determined through significant trial and error  
 min\_n(range = c(20, 70)), #these values determined through significant trial and error  
 levels = 5  
)  
  
set.seed(123)  
rf\_res\_tuned = tune\_grid(  
 Median\_wflow,  
 resamples = rf\_folds,  
 grid = rf\_grid #use the tuning grid  
)

rf\_res\_tuned %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 select(mean, min\_n, mtry) %>%  
 pivot\_longer(min\_n:mtry,  
 values\_to = "value",  
 names\_to = "parameter"  
 ) %>%  
 ggplot(aes(value, mean, color = parameter)) +  
 geom\_point(show.legend = FALSE) +  
 facet\_wrap(~parameter, scales = "free\_x") +  
 labs(x = NULL, y = "Accuracy")

 An alternate view of the parameters

rf\_res\_tuned %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 mutate(min\_n = factor(min\_n)) %>%  
 ggplot(aes(mtry, mean, color = min\_n)) +  
 geom\_line(alpha = 0.5, size = 1.5) +  
 geom\_point() +  
 labs(y = "Accuracy")



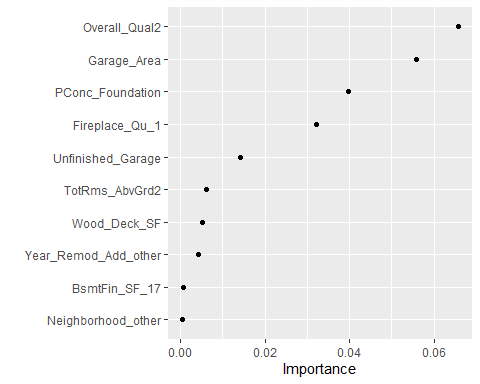
best\_rf = select\_best(rf\_res\_tuned, "accuracy")  
  
final\_rf = finalize\_workflow(  
 Median\_wflow,  
 best\_rf  
)  
  
final\_rf

## ══ Workflow ════════════════════════════════════════════════════════════════════  
## Preprocessor: Recipe  
## Model: rand\_forest()  
##   
## ── Preprocessor ────────────────────────────────────────────────────────────────  
## 2 Recipe Steps  
##   
## • step\_other()  
## • step\_dummy()  
##   
## ── Model ───────────────────────────────────────────────────────────────────────  
## Random Forest Model Specification (classification)  
##   
## Main Arguments:  
## mtry = 8  
## trees = 100  
## min\_n = 32  
##   
## Engine-Specific Arguments:  
## importance = permutation  
##   
## Computational engine: ranger

#fit the finalized workflow to our training data  
final\_rf\_fit = fit(final\_rf, train)

Check out variable importance

final\_rf\_fit %>% pull\_workflow\_fit() %>% vip(geom = "point")



Predictions

trainpredrf = predict(final\_rf\_fit, train)  
head(trainpredrf)

## # A tibble: 6 × 1  
## .pred\_class  
## <fct>   
## 1 Yes   
## 2 No   
## 3 No   
## 4 No   
## 5 No   
## 6 No

Confusion matrix

confusionMatrix(trainpredrf$.pred\_class, train$Above\_Median,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 716 68  
## No 66 689  
##   
## Accuracy : 0.9129   
## 95% CI : (0.8977, 0.9265)  
## No Information Rate : 0.5081   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.8258   
##   
## Mcnemar's Test P-Value : 0.9312   
##   
## Sensitivity : 0.9156   
## Specificity : 0.9102   
## Pos Pred Value : 0.9133   
## Neg Pred Value : 0.9126   
## Prevalence : 0.5081   
## Detection Rate : 0.4652   
## Detection Prevalence : 0.5094   
## Balanced Accuracy : 0.9129   
##   
## 'Positive' Class : Yes   
##

Predictions on test

testpredrf = predict(final\_rf\_fit, test)  
head(testpredrf)

## # A tibble: 6 × 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 Yes   
## 3 Yes   
## 4 Yes   
## 5 Yes   
## 6 Yes

confusionMatrix(testpredrf$.pred\_class, test$Above\_Median,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 240 27  
## No 21 226  
##   
## Accuracy : 0.9066   
## 95% CI : (0.8781, 0.9303)  
## No Information Rate : 0.5078   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.8131   
##   
## Mcnemar's Test P-Value : 0.4705   
##   
## Sensitivity : 0.9195   
## Specificity : 0.8933   
## Pos Pred Value : 0.8989   
## Neg Pred Value : 0.9150   
## Prevalence : 0.5078   
## Detection Rate : 0.4669   
## Detection Prevalence : 0.5195   
## Balanced Accuracy : 0.9064   
##   
## 'Positive' Class : Yes   
##