# DS6306: Cast Study 01

Ben Goodwin/ Justin Ehly

10/6/2020

```
# Introduction #########
#
#
# The following is an analysis of one-hundred styles of beer brewed in the United States for the
executive team,
# CEO and CFO at Budweiser. Budweiser is interested in exploring the how many breweries are in t
he United States,
# how each beer is reported in terms of its International Bitterness Unit and Alcohol By Content
and basic summary
# statistics and conclusions we are able to uncover with the beer data provided. Statistics will
include handling
# missing data and explaining why it was possibly not included in the initial dataset, as well a
s uncover median
# and maximum (IBU and ABV) ratings by state. Conclusions will include basic summary statics on
 the ABV variable,
# any relationship between the IBU and ABV variables (such as dependencies, e.g. does a higher I
BU result in a
# higher ABV) and finally we will look to see if we can determine general beer styles (Ales and
 IPAs) based on
# ABV and IBU values. Additionally, we will report on any findings that are discovered during th
e analysis.
#
######################
#                    #
#     Libraries      #
#                    #
######################
######################
library(usmap)
library(ggplot2)
library(magrittr)
library(ggplot2)
library(GGally)
```

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

```
library(readr)
library(tibble)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
## v tidyr   1.1.2     v stringr 1.4.0
## v purrr   0.3.4     v forcats 0.5.0
## v dplyr   1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::extract()   masks magrittr::extract()
## x dplyr::filter()    masks stats::filter()
## x dplyr::lag()       masks stats::lag()
## x purrr::set_names() masks magrittr::set_names()
```

```
library(robustbase)
library(class)
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(e1071)
library(dplyr)
library(codebook)
library(future)
```

```
##
## Attaching package: 'future'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
library(fmsb)
```

```
## Registered S3 methods overwritten by 'fmsb':
##   method    from
##   print.roc pROC
##   plot.roc  pROC
```

```
library(ggraph)
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:future':
##
##     %->%, %<-%
```

```
## The following object is masked from 'package:class':
##
##     knn
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
```

```
## The following object is masked from 'package:tidyr':
##
##     crossing
```

```
## The following object is masked from 'package:tibble':
##
##     as_data_frame
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

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

```r
library(RColorBrewer)
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:igraph':
##
##     groups
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

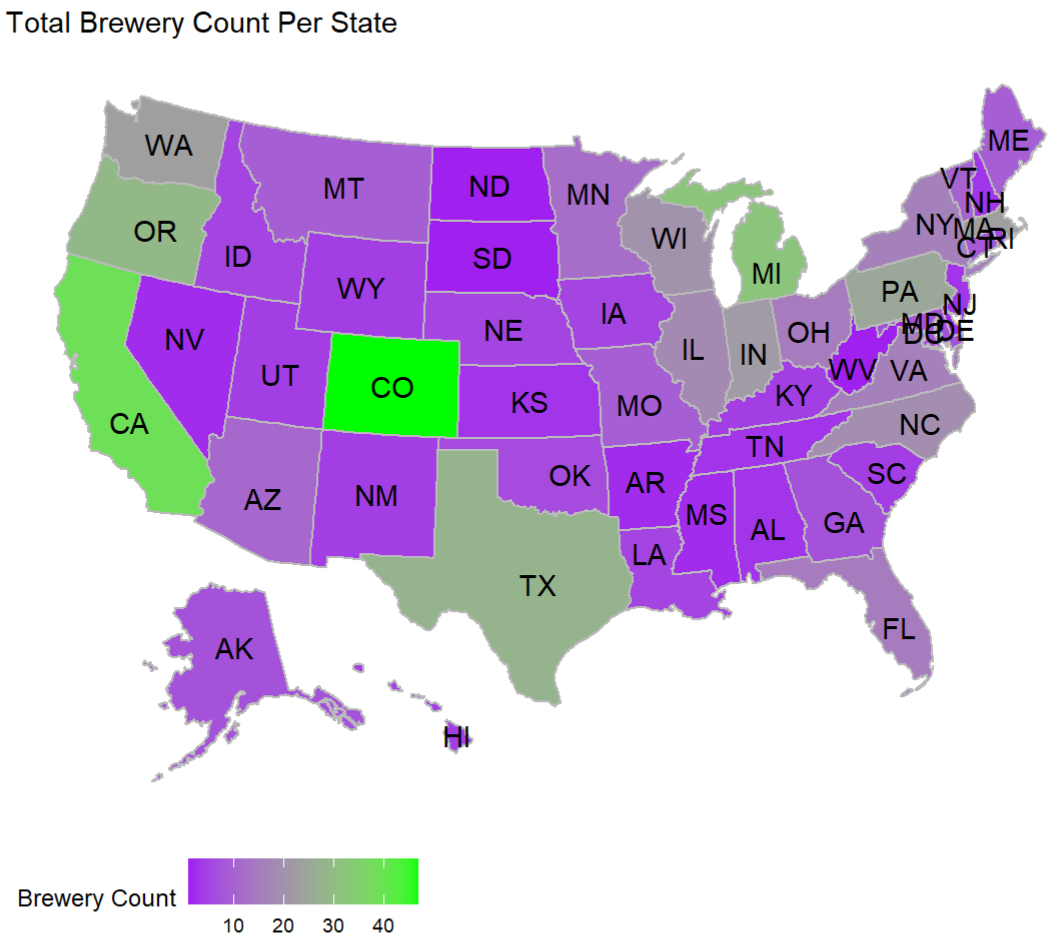
```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

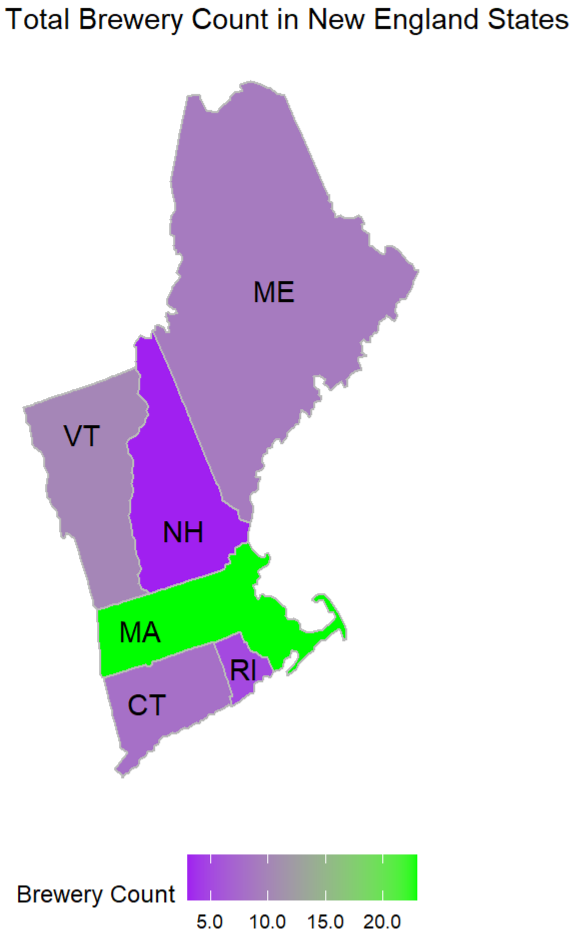
```
######################
######################
#                    #
#        Data        #
#                    #
######################
########################################################
#read in brewery data
setwd("C:/Users/justi/Documents/GitHub/MSDS6306/CaseStudy1/project_files/")
breweryDat <- read.csv("breweries.csv")
breweryDat$State <- trimws(breweryDat$State)

#datafile to organize states into census regions
regionData <- read.csv("state-geocodes-v2017.csv")
regionData <- dplyr::rename(regionData, "RegionNum" = "Region", "DivisionNum" = "Division", "FIP
S"="State..FIPS.", "Region" = "Region.1", "Division" = "Division.1")
regionData$State <- trimws(regionData$State)



#Ensure structure of data is compliant
#head(breweryDat)
#read in beer data
beerDat <- read.csv("beers.csv")
#Loop to fix leading decimal places on ABV
i <- 1
count <- length(beerDat$Name)
for (i in 1:count) {
 if(is.na(beerDat[i,3])){
  beerDat[i,3]=0
   }
   if(beerDat[i,3]<1){
    beerDat[i,3] <- beerDat[i,3]*100
  }
}

#Ensure structure of data is compliant
#head(beerDat)
########################################################
```

```
# Question 1 - How many breweries are in each state?
#
# During this analysis, we explored how many breweries are in each state and grouped the states
# by US Census Divisions. The data is visually displayed using maps of each USC Division below
# and summarized in a simple chart at the end.
#
######################
#                    #
#     Question 1     #
#                    #
######################
#
#########################################################
#Use Dplyr to group breweries by state
brewByState <- breweryDat %>%
  group_by(State) %>%
  dplyr::count()
#########################################################
#########################################################
#Add breweries by state to state information dataframe
statepop$brewByState <- brewByState$n
#########################################################
#########################################################
#Fix mismatched state brewery count to state info df
statepop[1,5] <- 3
statepop[2,5] <- 7
statepop[3,5] <- 11
statepop[4,5] <- 2
statepop[8,5] <- 2
statepop[9,5] <- 1
statepop[14,5] <- 18
statepop[15,5] <- 22
statepop[16,5] <- 5
statepop[20,5] <- 9
statepop[22,5] <- 23
statepop[25,5] <- 2
statepop[26,5] <- 9
statepop[28,5] <- 5
statepop[29,5] <- 2
statepop[30,5] <- 3
statepop[32,5] <- 4
statepop[34,5] <- 19
statepop[33,5] <- 16
statepop[35,5] <- 1
statepop[45,5] <- 4
statepop[46,5] <- 10
statepop[47,5] <- 16
statepop[49,5] <- 1
statepop[50,5] <- 20
#Check data
#View(statepop)
#View(brewByState)
#########################################################
```

```
############################################################
#Call plot functions to plot state brewery count on USmap
nationBrewPlot <- plot_usmap(data = statepop, values = "brewByState",labels=TRUE, color = "grey7
3") + scale_fill_continuous(low = "purple", high = "green", name = "Brewery Count", label = scal
es::comma) + theme(legend.position = "bottom")+labs(title = "Total Brewery Count Per State")
#display plot
nationBrewPlot
```

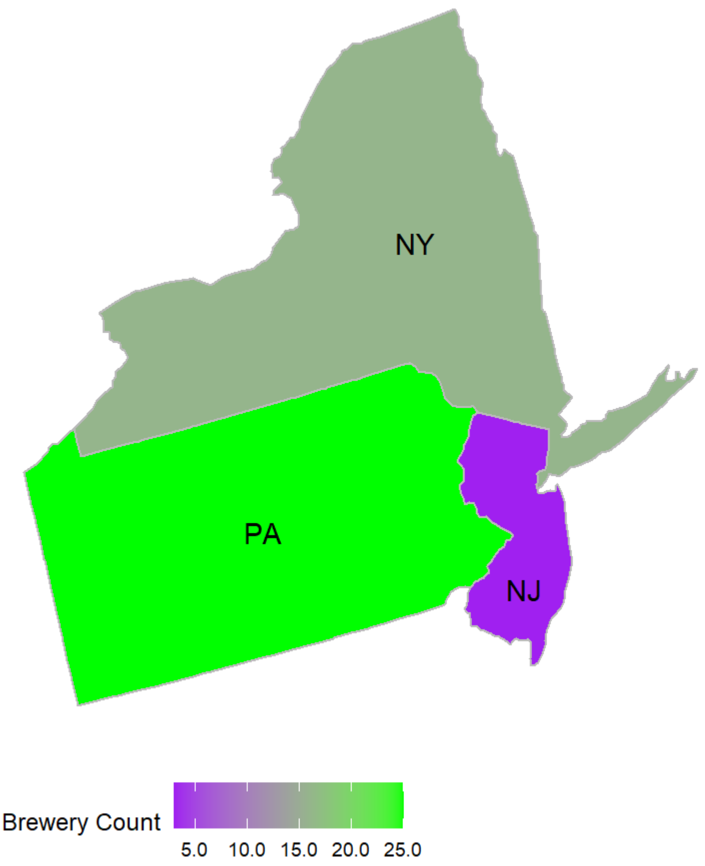


```
############################################################
############################################################
#Break down by region, NE first
NEplot <- plot_usmap(data=statepop, values = "brewByState",labels = TRUE,include = .new_england,
color = "grey73") + scale_fill_continuous(low = "purple", high = "green", name = "Brewery Count"
, label = scales::comma)+ theme(legend.position = "bottom")+labs(title = "Total Brewery Count in
New England States")
NEplot
```

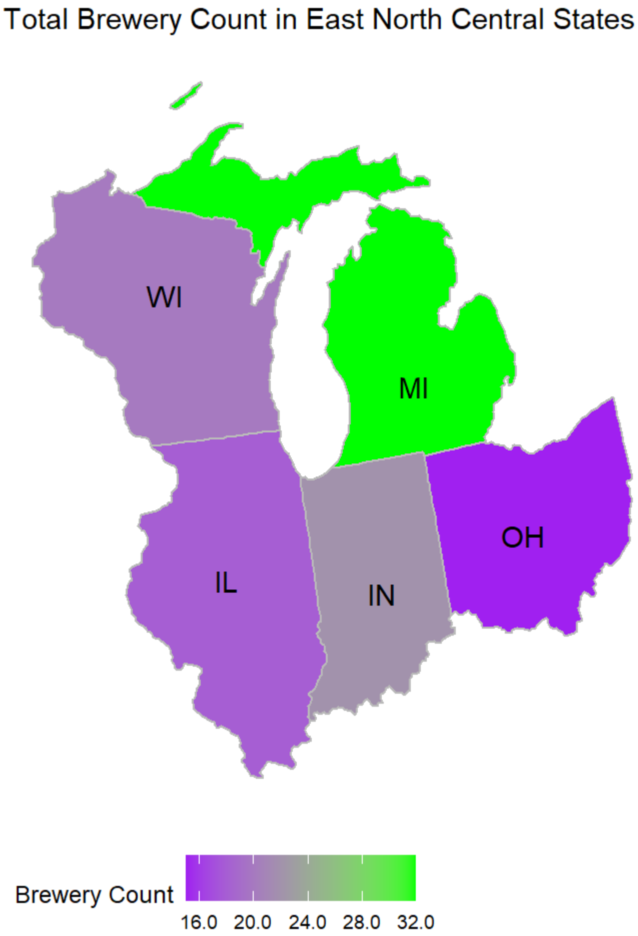


```
############################################################
############################################################
#Break down by region, Mid Atlantic second
MAplot <- plot_usmap(data=statepop, values = "brewByState",labels = TRUE,include = .mid_atlanti
c,color = "grey73") + scale_fill_continuous(low = "purple", high = "green", name = "Brewery Coun
t", label = scales::comma)+ theme(legend.position = "bottom")+labs(title = "Total Brewery Count
 in Middle Atlantic States")
MAplot
```

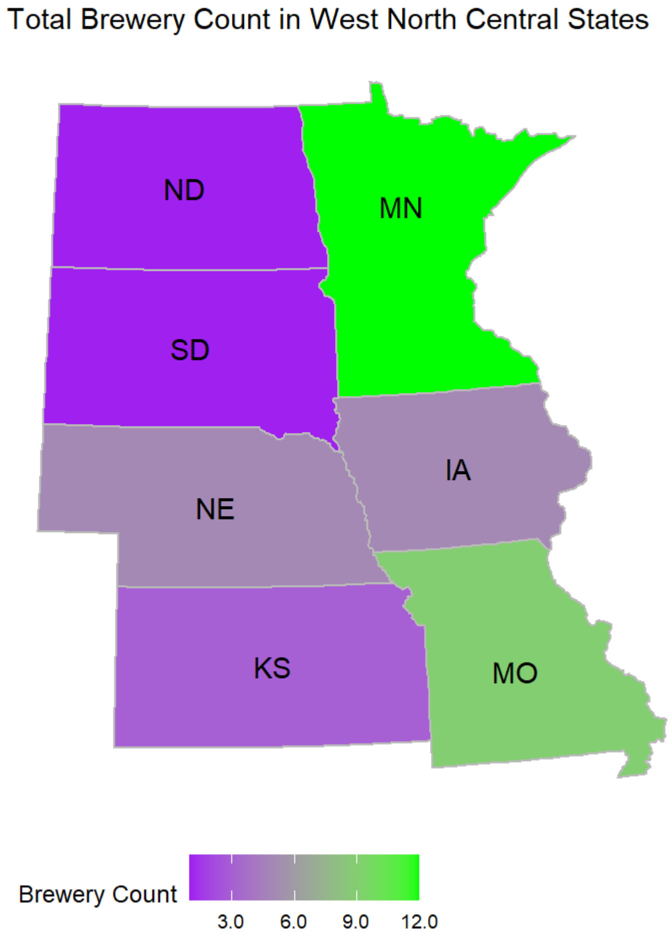
Total Brewery Count in Middle Atlantic States



```
###########################################################
###########################################################
#Break down by region, East North Central third
ENCplot <- plot_usmap(data=statepop, values = "brewByState",labels = TRUE,include = .east_north_
central,color = "grey73") + scale_fill_continuous(low = "purple", high = "green", name = "Brewer
y Count", label = scales::comma)+ theme(legend.position = "bottom")+labs(title = "Total Brewery
 Count in East North Central States")
ENCplot
```

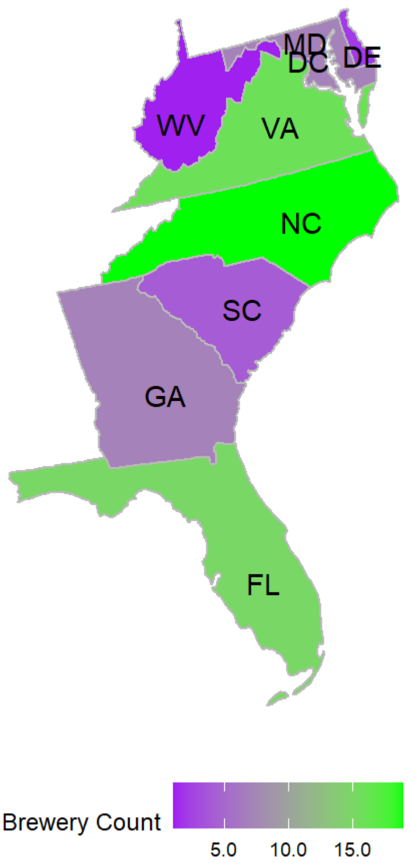


```
##########################################################
##########################################################
#Break down by region, West North Central fourth
WNCplot <- plot_usmap(data=statepop, values = "brewByState",labels = TRUE,include = .west_north_
central,color = "grey73") + scale_fill_continuous(low = "purple", high = "green", name = "Brewer
y Count", label = scales::comma)+ theme(legend.position = "bottom")+labs(title = "Total Brewery
 Count in West North Central States")
WNCplot
```

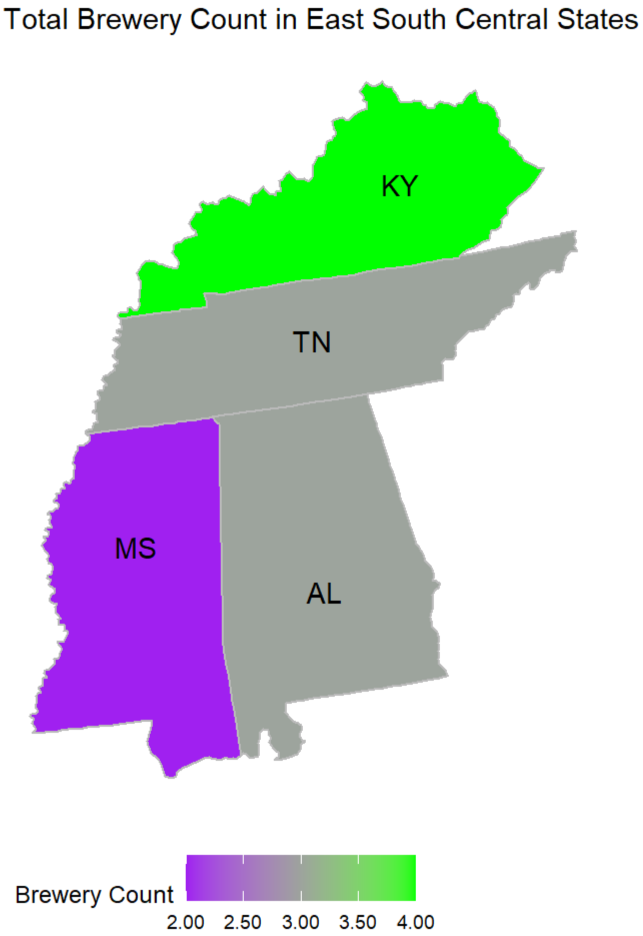


```
##########################################################
##########################################################
#Break down by region, South Atlantic fifth
SAplot <- plot_usmap(data=statepop, values = "brewByState",labels = TRUE,include = .south_atlant
ic,color = "grey73") + scale_fill_continuous(low = "purple", high = "green", name = "Brewery Cou
nt", label = scales::comma)+ theme(legend.position = "bottom")+labs(title = "Total Brewery Count
in South Atlantic States")
SAplot
```

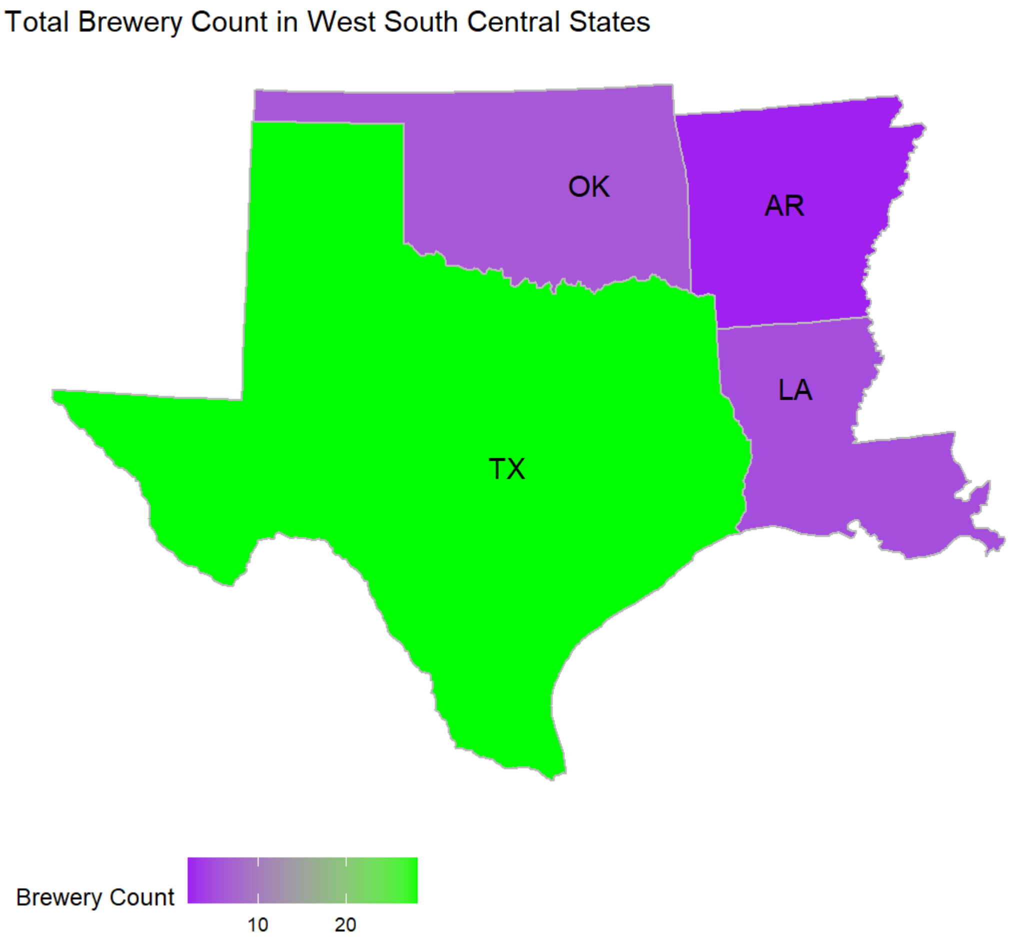
Total Brewery Count in South Atlantic States



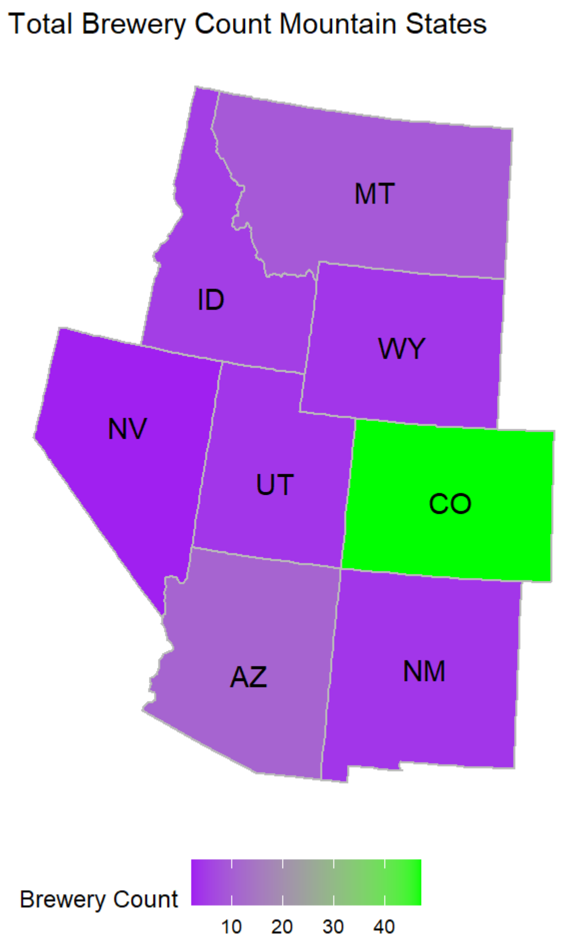
```
##############################################################
##############################################################
#Break down by region, East South Central sixth
ESCplot <- plot_usmap(data=statepop, values = "brewByState",labels = TRUE,include = .east_south_central,color = "grey73") + scale_fill_continuous(low = "purple", high = "green", name = "Brewery Count", label = scales::comma)+ theme(legend.position = "bottom")+labs(title = "Total Brewery Count in East South Central States")
ESCplot
```



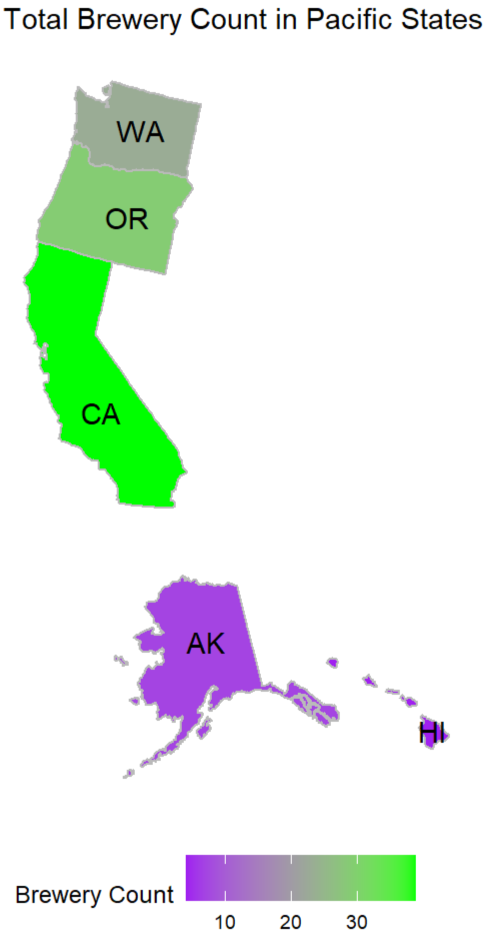
```
###########################################################
###########################################################
#Break down by region, West South Central seventh
WSCplot <- plot_usmap(data=statepop, values = "brewByState",labels = TRUE,include = .west_south_
central,color = "grey73") + scale_fill_continuous(low = "purple", high = "green", name = "Brewer
y Count", label = scales::comma)+ theme(legend.position = "bottom")+labs(title = "Total Brewery
 Count in West South Central States")
WSCplot
```



```
############################################################
############################################################
#Break down by region, Mountain eighth
Mplot <- plot_usmap(data=statepop, values = "brewByState",labels = TRUE,include = .mountain,color = "grey73") + scale_fill_continuous(low = "purple", high = "green", name = "Brewery Count", label = scales::comma)+ theme(legend.position = "bottom")+labs(title = "Total Brewery Count Mountain States")
Mplot
```


Total Brewery Count Mountain States

```
############################################################
############################################################
#Break down by region, Pacific ninth
Pplot <- plot_usmap(data=statepop, values = "brewByState",labels = TRUE,include = .pacific,color = "grey73") + scale_fill_continuous(low = "purple", high = "green", name = "Brewery Count", label = scales::comma)+ theme(legend.position = "bottom")+labs(title = "Total Brewery Count in Pacific States")
Pplot
```



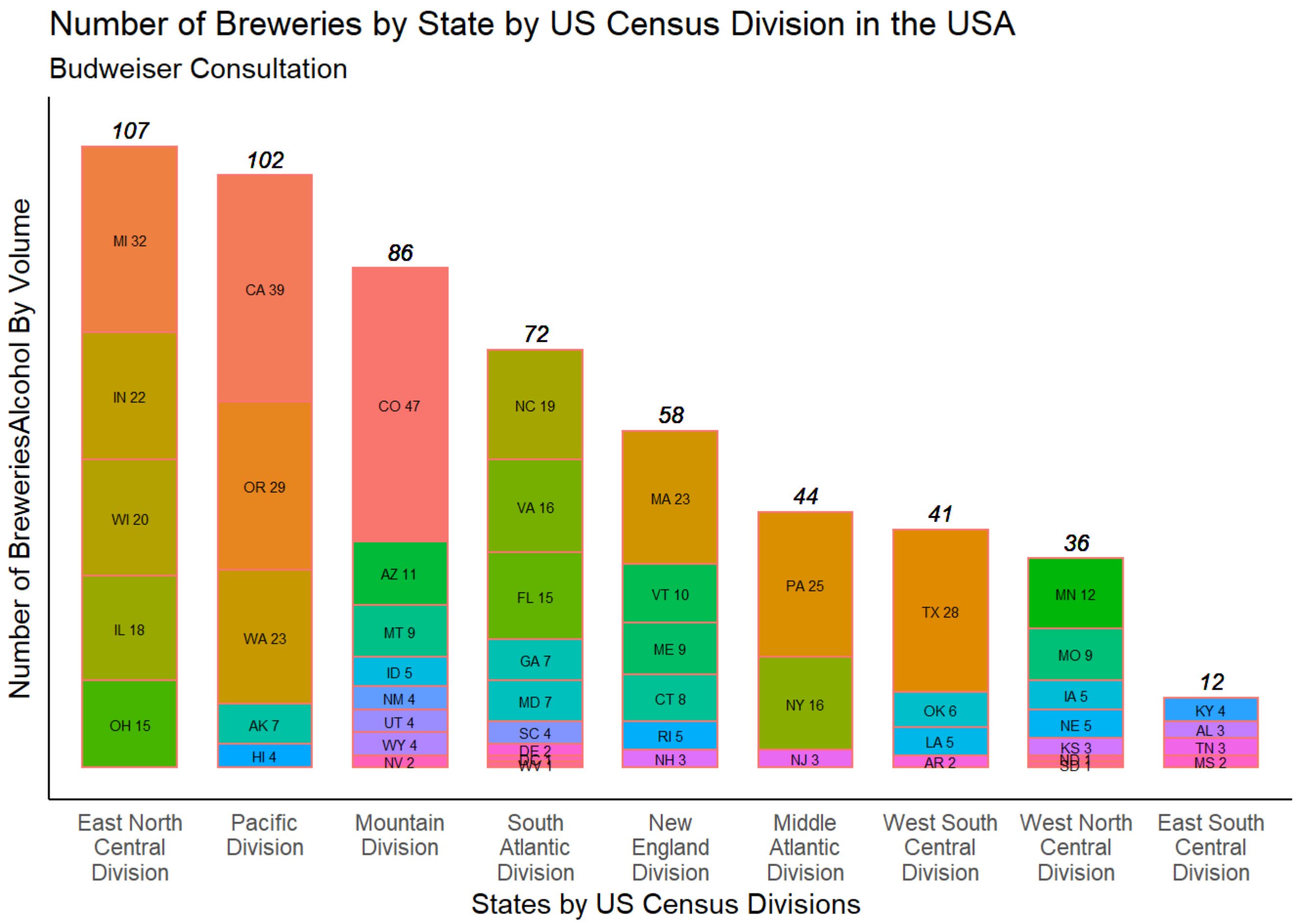
```
#########################################################
#########################################################

# Add label column to brewByState
brewByState$Label <- paste(brewByState$State, brewByState$n)
#Add Regional Data to brewByState
brewByState <- merge(brewByState, regionData, by = "State")
brewByState <- brewByState[,-c(4,5)]

# Sum up state brewery count by division for labeling
brewSum <- brewByState %>%
  group_by(Division) %>%
  dplyr::summarise(SumBreweries = sum(n))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
brewByState <- merge(brewByState, brewSum, by = "Division")

labelDF <- str_wrap(brewByState$Division, width = 10)
labelDF
```

```
##  [1] "East North\nCentral\nDivision" "East North\nCentral\nDivision"
##  [3] "East North\nCentral\nDivision" "East North\nCentral\nDivision"
##  [5] "East North\nCentral\nDivision" "East South\nCentral\nDivision"
##  [7] "East South\nCentral\nDivision" "East South\nCentral\nDivision"
##  [9] "East South\nCentral\nDivision" "Middle\nAtlantic\nDivision"
## [11] "Middle\nAtlantic\nDivision"    "Middle\nAtlantic\nDivision"
## [13] "Mountain\nDivision"            "Mountain\nDivision"
## [15] "Mountain\nDivision"            "Mountain\nDivision"
## [17] "Mountain\nDivision"            "Mountain\nDivision"
## [19] "Mountain\nDivision"            "Mountain\nDivision"
## [21] "New\nEngland\nDivision"        "New\nEngland\nDivision"
## [23] "New\nEngland\nDivision"        "New\nEngland\nDivision"
## [25] "New\nEngland\nDivision"        "New\nEngland\nDivision"
## [27] "Pacific\nDivision"             "Pacific\nDivision"
## [29] "Pacific\nDivision"             "Pacific\nDivision"
## [31] "Pacific\nDivision"             "South\nAtlantic\nDivision"
## [33] "South\nAtlantic\nDivision"     "South\nAtlantic\nDivision"
## [35] "South\nAtlantic\nDivision"     "South\nAtlantic\nDivision"
## [37] "South\nAtlantic\nDivision"     "South\nAtlantic\nDivision"
## [39] "South\nAtlantic\nDivision"     "South\nAtlantic\nDivision"
## [41] "West North\nCentral\nDivision" "West North\nCentral\nDivision"
## [43] "West North\nCentral\nDivision" "West North\nCentral\nDivision"
## [45] "West North\nCentral\nDivision" "West North\nCentral\nDivision"
## [47] "West North\nCentral\nDivision" "West South\nCentral\nDivision"
## [49] "West South\nCentral\nDivision" "West South\nCentral\nDivision"
## [51] "West South\nCentral\nDivision"
```

```
#### Bar Plot ####

#Plot overall breweries by state in bar chart

brewByState %>%
  ggplot(aes(x=reorder(Division,-SumBreweries), y=n,fill= reorder(State,-n))) +
  # Create stacked by chart organized by Division with States stacked in each bar
  geom_bar(aes(color = "#c8102e"),stat="identity", width= 0.7,
           position = position_stack(), show.legend = FALSE) +
  # Add state and ABV value to each state's chart position
  geom_text(aes(label = Label), size = 2, position = position_stack(vjust = .5), ) +
  # Add Division ABV Values to top of each chart stack
  geom_text(aes(Division, SumBreweries, label = SumBreweries), size = 3,
            nudge_y = 3, fontface = "italic") +
  # Label the chart objects
  scale_x_discrete(labels = function(x) str_wrap(x, width = 10)) +
  labs(title="Number of Breweries by State by US Census Division in the USA",
       subtitle="Budweiser Consultation",
       y = "Number of BreweriesAlcohol By Volume",
       x = "States by US Census Divisions ") +
  theme_classic() +
  # Adjust the X-axis labels, remove y-labels since this is a stacked chart
  theme(axis.text.y = element_blank(), axis.ticks = element_blank())
```



```
######################
#                    #
#     Question 2     #
#                    #
######################
################################################################################################
#############
# Question 2 - Merge the individual data sets
#
# We merged the breweries.csv dataset with the beers.csv dataset, additionally when we imported
# the individual datasets, we also imported a dataset that allows us to associate each beer with
# its brewery's US Census Division.
#
################################################################################################
#############
#Use Dplyr package to merge the two tables together
buzzbrews <- merge(breweryDat, beerDat, by.x = "Brew_ID", by.y = "Brewery_id", all = TRUE )
#Use Dplyr package to rename "Name.x" to "Brewery" and "Name.y" to "Beer"
buzzbrews <- dplyr::rename(buzzbrews, "Brewery" = "Name.x","Beer"="Name.y")
bzbwTestDf <- buzzbrews
#Check the results
#View(buzzbrews)
```

```
######################
#                    #
#     Question 3     #
#                    #
######################
###########################################
# Question 3 - Address the missing values in each column.
#
# During the initial exploratory process we discovered NA's in both the IBU and ABV columns.
# Upon further investigation we determined that some styles of beer, mixed or barrel aged beers
# do not have an ABV available at the time the brewery submits packaging labels to TTB, or Alcoh
ol
# and Tobacco Tax and Trade Bureau. The TTB is the federal agency that determines what can and c
annot
# be put on a beer label including the art, type size, verbiage, where elements are placed and e
tc.
# So beers without an AVB available either do not inlcude it, or add it to the bottom of the can
s
# or packaging at a later date.
#
# In terms of the missing IBU values, we determined that even though the IBU alludes to the bitt
erness
# of a beer's taste, it is somewhat misleading because it is derived from a test that measures d
ifferent
# chemical compounds that are known to cause bitter flavoring. For instance, a beer may have a h
igh IBU
# value, but due to other ingredients, such as added lactose or sucrose may actually have a swee
ter taste
# than would be expected from a high IBU. The other comfounding variable is if the brewery can a
fford the
# equipment used to generate an IBU value, smaller breweries simply cannot afford it while the l
arger
# breweries typically just use IBU as a quality control measure.
#
# Finally, we concluded that imputing data or filling in the missing gaps was a good idea for th
is
# analysis and that was done by taking an average of from similiar styles of beer and assigning
 that to
# beers in the same sytle classification that did not have values. Upon random testing of differ
ent imputed
# values, by googling beers that had missing values in the dataset and comparing that to the cre
ated averages,
# it was determined that the imputed values were very close to the actual values in the marketpl
ace.
#
###########################################
#Loop to fix numbering for Column 1 "brew ID"
iterations <- length(buzzbrews$Brew_ID)
for (i in 1:iterations) {
  buzzbrews[i,1]=i
}
#Fix no style beers to none
levels(buzzbrews$Style) <- c(levels(buzzbrews$Style), "none")
```

```
for (i in 1:iterations) {
  if(is.na(buzzbrews[i,9])){
  }
}
for (i in 1:iterations) {
  if((buzzbrews[i,9])==''){
    #print(buzzbrews[i,9])
    buzzbrews[i,9]="none"
  }
}
#Prep new df to contain style and averages
buzzbrews$Style <- as.factor(buzzbrews$Style)
#Create a data frame with each style and a variable for average IBU
styleCount <- as.data.frame(levels(buzzbrews$Style))
styleCount$`levels(buzzbrews$Style)` <- as.character(styleCount$`levels(buzzbrews$Style)`)
#View(styleCount)
#Initialize mean ibu to zero (to avoid problems with N/As)
styleCount$meanIbu <- 0
#Make beer count to keep track of total in each style
styleCount$beerCount <- 0
#Make column for total ibus
styleCount$totalIBU <- 0
styleCount$meanABV <- 0
styleCount$ABVbeerCount <- 0
styleCount$totalABV <- 0
#Checking
#View(styleCount)
#styleCount <- styleCount[-c(1), ]
#View(styleCount)
#Calculate mean IBU for each category and store it in IBU df
#Calculate average IBU for each style and add it to df
#outer loop for all the beers
ibuSum <- 0
beerCount <- 0
i <- 1
for (i in 1:iterations) {
  if(is.na(buzzbrews[i,8])) {
    buzzbrews[i,8]=0
  }

  #inner for each style
  for (j in 1:100) {

    if(buzzbrews[i,9]==styleCount[j,1]){
     #Compute IBU sum
     styleCount[j,4] <- styleCount[j,4]+buzzbrews[i,8]



     #Total of each beer count
     styleCount[j,3] <- styleCount[j,3]+1

     if(buzzbrews[i,8]==0){
       styleCount[j,3] <- styleCount[j,3]-1
     }



    }
    #Mean IBU for each style
    styleCount[j,2] <- styleCount[j,4]/styleCount[j,3]
    }}
#Add average column from style count to buzzbrews df
for (i in 1:iterations) {
  if(buzzbrews[i,8]==0){
    for(j in 1:100){
      if(buzzbrews[i,9]==styleCount[j,1]){
        buzzbrews[i,8]=styleCount[j,2]
      }
    }
  }
}
# View(styleCount)
# View(buzzbrews)
# Now do it all again for ABV
# Calculate average ABV for each style and add it to df
# outer loop for all the beers
AlcSum <- 0
AlcVeerCount <- 0
i <- 1
for (i in 1:iterations) {
  if(is.na(buzzbrews[i,7])) {
   buzzbrews[i,7]=0
  }


  #inner for each style
  for (j in 1:100) {

    if(buzzbrews[i,9]==styleCount[j,1]){

     #Compute ALC sum
     styleCount[j,7] <- styleCount[j,7]+buzzbrews[i,7]*100



     #Total of each beer count
     styleCount[j,6] <- styleCount[j,6]+1

     if(buzzbrews[i,7]==0){
       styleCount[j,6] <- styleCount[j,6]-1
     }



    }
    #Mean ABV for each style
    styleCount[j,5] <- (styleCount[j,7]/styleCount[j,6])/100
    }
}
#Add average column from style count to buzzbrews df
for (i in 1:iterations) {
  if(buzzbrews[i,7]==0){
    for(j in 1:100){
      if(buzzbrews[i,9]==styleCount[j,1]){
        buzzbrews[i,7]=styleCount[j,5]

      }
      }
  }
}
#kill NaN's for other alcohol types with no hops
i <- 1
for(i in 1:iterations){
  if(is.na(buzzbrews[i,8])){
    buzzbrews[i,8] <- 0
  }
}
#Check out end results - look for any leftover NAs in DF
sapply(buzzbrews, function(x) sum(is.na(x)))
```

```
## Brew_ID Brewery    City   State    Beer Beer_ID     ABV     IBU   Style  Ounces 
##       0       0       0       0       0       0       0       0       0       0
```

```
# Add in the regional data from the census bureau
buzzbrews <- merge(buzzbrews, regionData, by = "State")
# View new dataframe
view(buzzbrews)
```

```
######################
#                    #
#     Question 4     #
#                    #
######################
################################################################################################
#################
# Question 4 - Compute the median alcohol content and international bitterness unit for
# each state. Plot a bar chart to compare.
#
# We computed the MedStateABV and IBU for each state and created a visualisation that allowed
# us to further explore what those medians tell us. We found there appears to be a relationship
# between IBU and ABV where we can use IBU to estimate ABV of a given beer.
#
# We explored this further by developing a model to make predictions based on historical IBU
# and ABV data and were able to predict that a beer with 32 IBU could have an ABV of 5.72% and
# we were 97.5% confident that beer would at least fall between 3.24% and 8.21%.
#
################################################################################################
#################
buzzbrews$State <- trimws(buzzbrews$State)

# Group by state and compute
combineddf <- buzzbrews %>%
  group_by(State) %>%
  dplyr::summarise(MedStateIBU = median(IBU), MedStateABV = median(ABV))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

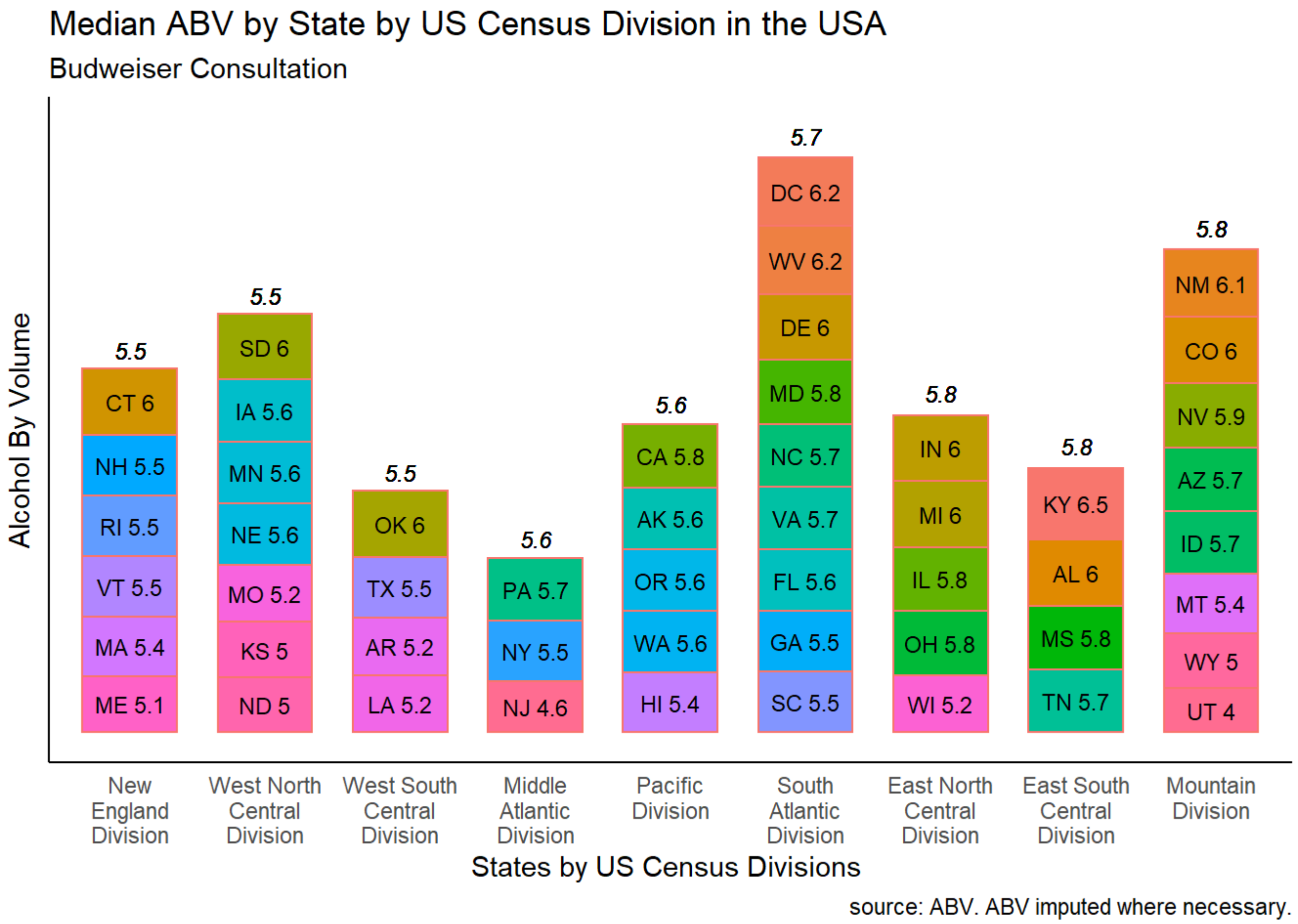
```
combineddf <- as.data.frame(combineddf)
combineddf$MedStateIBU <- as.numeric(combineddf$MedStateIBU)
combineddf$MedStateABV <- as.numeric(combineddf$MedStateABV)

# Divisional measurements
divisiondf <- buzzbrews %>%
  group_by(Division) %>%
  dplyr::summarise(MedDivIBU = median(IBU), MedDivABV = median(ABV))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# round values to xx.x ###
divisiondf$MedDivIBU <- round(divisiondf$MedDivIBU, digits = 1)
divisiondf$MedDivABV <- round(divisiondf$MedDivABV, digits = 1)
combineddf$MedStateIBU <- round(combineddf$MedStateIBU, digits = 1)
combineddf$MedStateABV <- round(combineddf$MedStateABV, digits = 1)

# Add regions to combinddf
combineddf <- merge(combineddf,regionData,by="State")

# Add in divisional values
combineddf <- merge(combineddf, divisiondf, by = "Division")

####### Create chart labels for stacked charts #####
combineddf$ABVlabel <- paste(combineddf$State, combineddf$MedStateABV)
combineddf$IBUlabel <- paste(combineddf$State, combineddf$MedStateIBU)

view(combineddf)

# Create sums of medians for labeling charts #
StateSums <- combineddf %>%
  group_by(Division) %>%
  dplyr::summarise(SumStateABV = sum(MedStateABV), SumStateIBU = sum(MedStateIBU))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

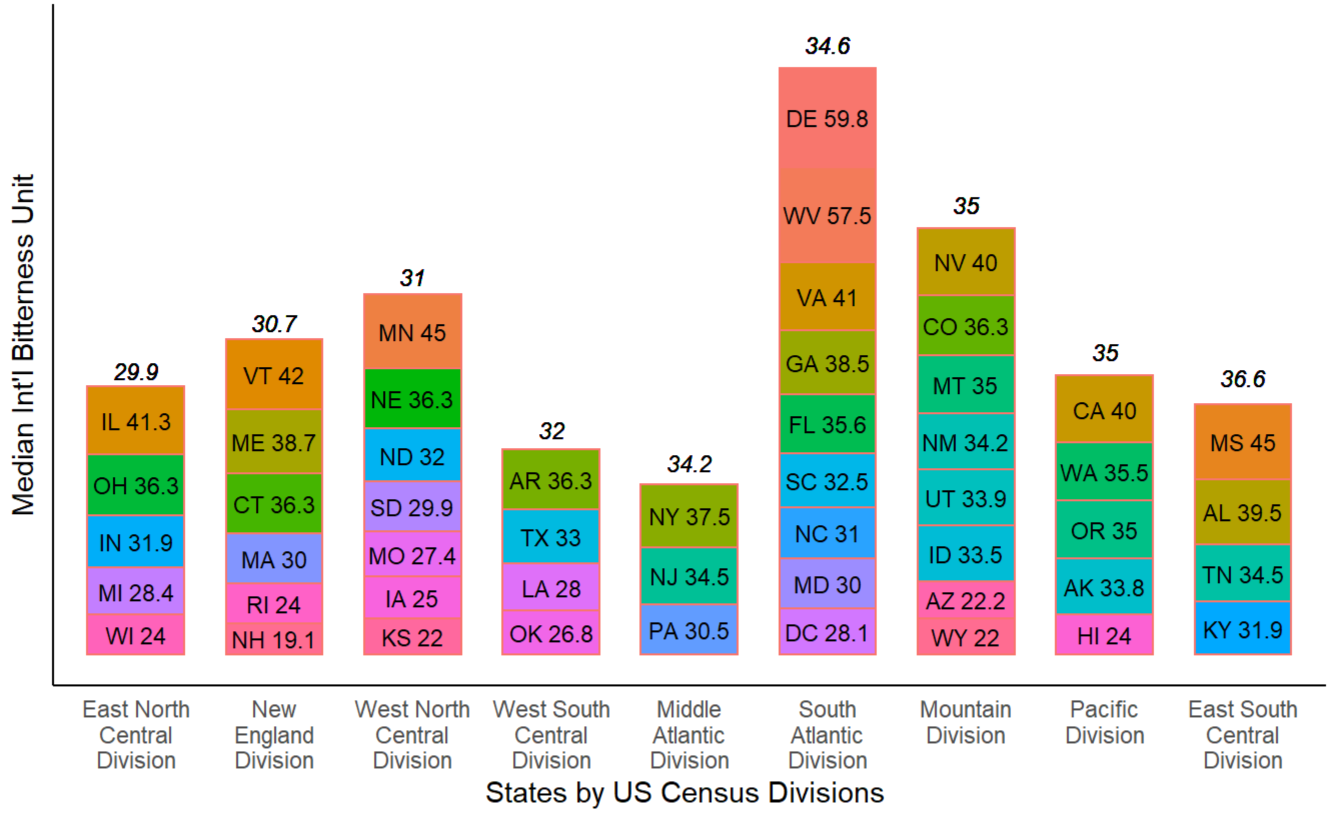
```
combineddf <- merge(combineddf, StateSums, by = "Division")

#
############################################################
#########                                   ################
######### Draw Bar Charts                   ################
#########                                   ################
############################################################

##### Create bar plot for ABV #####

combineddf %>%
  ggplot(aes(x=reorder(Division,MedDivABV ), y=MedStateABV,fill= reorder(State,-MedStateABV))) +
  # Create stacked by chart organized by Division with States stacked in each bar
  geom_bar(aes(color = "#c8102e"),stat="identity", width= 0.7, position = position_stack(), sho
w.legend = FALSE) +
  # Add state and ABV value to each state's chart position
  geom_text(aes(label = ABVlabel), size = 3, position = position_stack(vjust = 0.5)) +
  # Add Division ABV Values to top of each chart stack
  geom_text(aes(Division, MedDivABV + SumStateABV -3, label = MedDivABV), size = 3, vjust = 1, f
ontface = "italic") +
  # Label the chart objects
  labs(title="Median ABV by State by US Census Division in the USA",
       subtitle="Budweiser Consultation",
       caption="source: ABV. ABV imputed where necessary.",
       y = "Alcohol By Volume",
       x = "States by US Census Divisions ") +
  theme_classic() +
  # Remove y-labels and ticks since this is a stacked chart
  theme(axis.text.y = element_blank(), axis.ticks = element_blank()) +
  # Wrap X-axis labels
  scale_x_discrete(labels = function(x) str_wrap(x, width = 10))
```



```
#
##### Create bar plot for IBU #####
#
combineddf %>%
  ggplot(aes(x=reorder(Division, MedDivIBU), y=MedStateIBU,fill= reorder(State,-MedStateIBU))) +
  # Create stacked by chart organized by Division with States stacked in each bar
  geom_bar(aes(color = "#c8102e"),stat="identity", width= 0.7, position = position_stack(), sho
w.legend = FALSE) +
  # Add state and IBU value to each state's chart position
  geom_text(aes(label = IBUlabel), size = 3, position = position_stack(vjust = 0.5)) +
  # Add Division IBU Values to top of each chart stack
  geom_text(aes(Division, MedDivIBU + SumStateIBU - 15, label = MedDivIBU), size = 3, vjust = 1,
fontface = "italic") +
  # Label the chart objects
  labs(title="Median IBU by State by US Census Division in the USA",
       subtitle="Budweiser Consultation",
       caption="source: IBU. IBU imputed where necessary.",
       y = "Median Int'l Bitterness Unit",
       x = "States by US Census Divisions ") +
  theme_classic() +
  # Remove y-labels and ticks since this is a stacked chart
  theme(axis.text.y = element_blank(), axis.ticks = element_blank()) +
  # Wrap X-axis labels
  scale_x_discrete(labels = function(x) str_wrap(x, width = 10))
```

Median IBU by State by US Census Division in the USA

Budweiser Consultation



source: IBU. IBU imputed where necessary.

```
######################
#                    #
#     Question 5     #
#                    #
######################
################################################################################################
#############
# Question 5 - Which state has the maximum alcoholic (ABV) beer? Which state has the most bitter
(IBU) beer?
#
# We determined that the maximum observed IBU was 138 in Oregon for Bitter Bitch Imperial IPA th
at
# is an American Double/ Imperial IPA from the Astoria Brewing Company in Austoria, OR.
#
# We also determined that maximum observed ABV was 12.8% in Colorado for Lee Hill Series Vol. 5
 -
# Belgian Style Quadrupel Ale from Upslope Brewing Company in Boulder, CO.
#
################################################################################################
#############
#Figure out which has highest ABV
MaxStateABV <- arrange(buzzbrews, desc(ABV))
print(MaxStateABV[1,4])
```

```
## [1] "Boulder"
```

```
#Figure out which has highest IBU
maxIBU <- arrange(buzzbrews,desc(IBU))
print(maxIBU[1,4])
```

```
## [1] "Astoria"
```

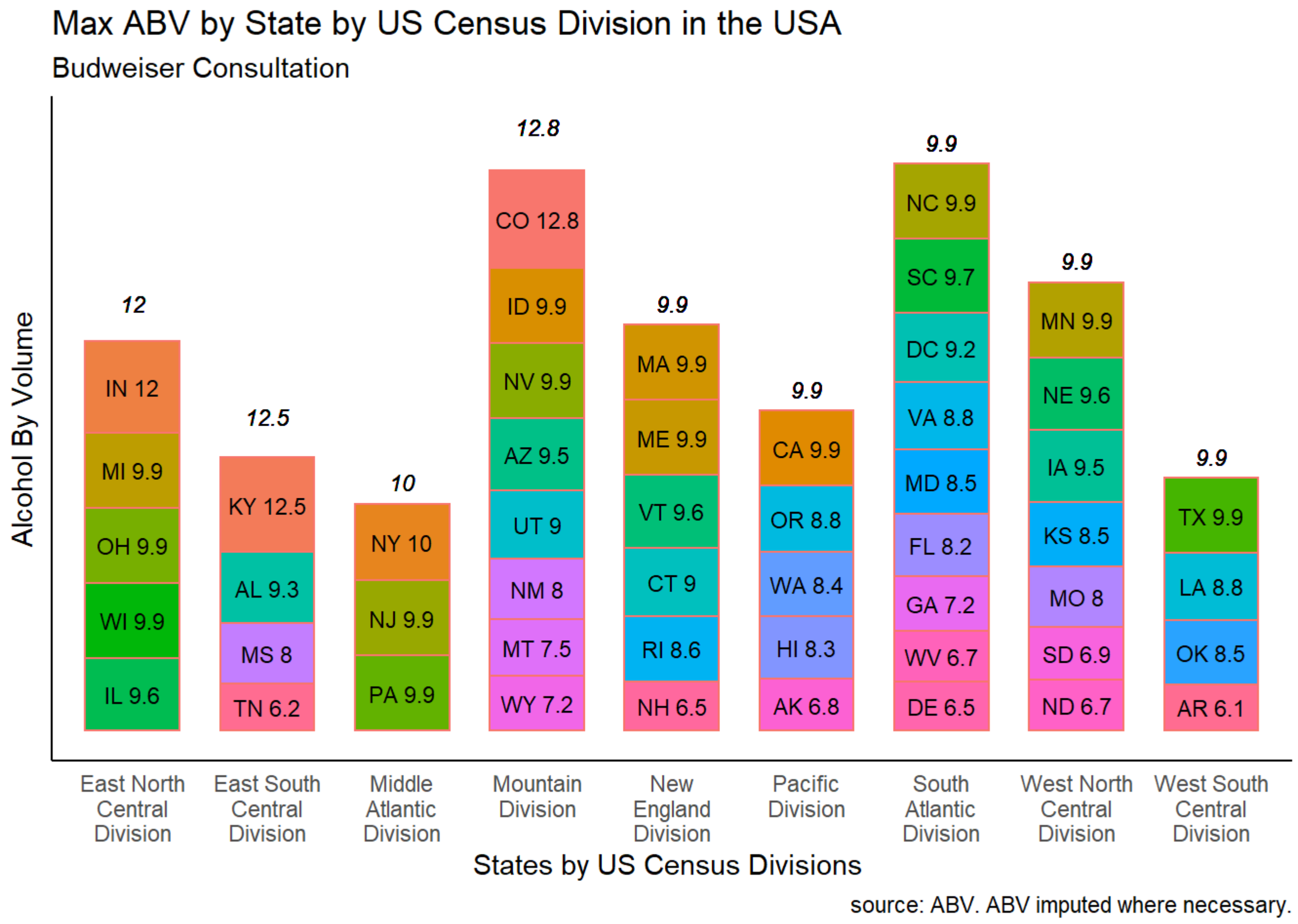
```
##### Question 5 Answer #####
## Colorado has the highest ABV = 12.8, Oregon has the highest IBU = 138.
###################################################
###### Create DF for just the max ABV & IBU values ######
# State measurements
maxStateValues <- buzzbrews %>%
  group_by(State) %>%
  dplyr::count(MaxStateABV = max(ABV), MaxStateIBU = max(IBU))
maxStateValues <- maxStateValues[,-4]
maxStateValues <- as.data.frame(maxStateValues)
maxStateValues$State <- trimws(maxStateValues$State)
str(maxStateValues)
```

```
## 'data.frame':    51 obs. of  3 variables:
##  $ State      : chr  "AK" "AL" "AR" "AZ" ...
##  $ MaxStateABV: num  6.8 9.3 6.1 9.5 9.9 ...
##  $ MaxStateIBU: num  71 103 45.7 99 115 ...
```

```
view(maxStateValues)

# Divisional measurements
divMaxValdf <- buzzbrews %>%
  group_by(Division) %>%
  dplyr::count(MaxDivABV = max(ABV), MaxDivIBU = max(IBU))
divMaxValdf <- divMaxValdf[,-4]
divMaxValdf <- as.data.frame(divMaxValdf)



# round values to xx.x ###
maxStateValues$MaxStateABV <- round(maxStateValues$MaxStateABV, digits = 1)
maxStateValues$MaxStateIBU <- round(maxStateValues$MaxStateIBU, digits = 1)

# Add regions to maxStateValues
maxStateValues <- merge(maxStateValues,regionData,by="State")

# Add in divisional values
maxStateValues <- merge(maxStateValues, divMaxValdf, by = "Division")

####### Create chart labels for stacked charts #####
maxStateValues$ABVmaxLabel <- paste(maxStateValues$State, maxStateValues$MaxStateABV)
maxStateValues$IBUmaxLabel <- paste(maxStateValues$State, maxStateValues$MaxStateIBU)

view(maxStateValues)

# Create sums of max values for labeling charts #
StateMaxSums <- maxStateValues %>%
  group_by(Division) %>%
  dplyr::summarise(SumStateABV = sum(MaxStateABV), SumStateIBU = sum(MaxStateIBU))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

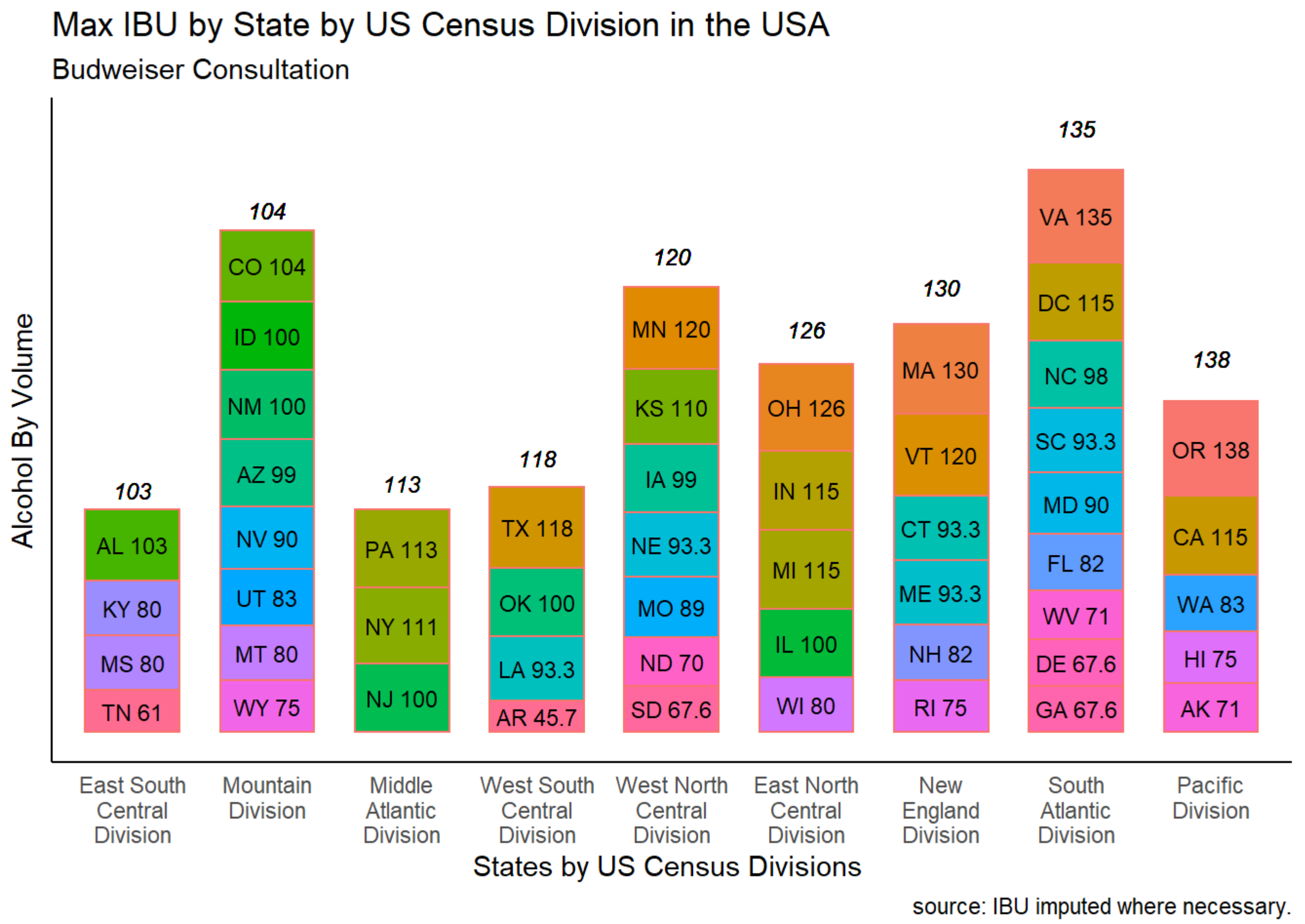
```
maxStateValues <- merge(maxStateValues, StateMaxSums, by = "Division")
###################################################
###### Plot for Max ABV ###########################

maxStateValues %>%
  ggplot(aes(x=Division, y=MaxStateABV,fill= reorder(State,-MaxStateABV))) +
  # Create stacked by chart organized by Division with States stacked in each bar
  geom_bar(aes(color = "#c8102e"),stat="identity", width= 0.7,
           position = position_stack(), show.legend = FALSE) +
  # Add state and ABV value to each state's chart position
  geom_text(aes(label = ABVmaxLabel), size = 3, position = position_stack(vjust = 0.5)) +
  # Add Division ABV Values to top of each chart stack
  geom_text(aes(Division, MaxDivABV + SumStateABV, label = MaxDivABV), size = 3,
            nudge_y = -7, fontface = "italic") +
  # Label the chart objects
  labs(title="Max ABV by State by US Census Division in the USA",
       subtitle="Budweiser Consultation",
       caption="source: ABV. ABV imputed where necessary.",
       y = "Alcohol By Volume",
       x = "States by US Census Divisions ") +
  theme_classic() +
  # Remove y-labels and ticks since this is a stacked chart
  theme(axis.text.y = element_blank(), axis.ticks = element_blank()) +
  # Wrap X-axis labels
  scale_x_discrete(labels = function(x) str_wrap(x, width = 10))
```


Max ABV by State by US Census Division in the USA

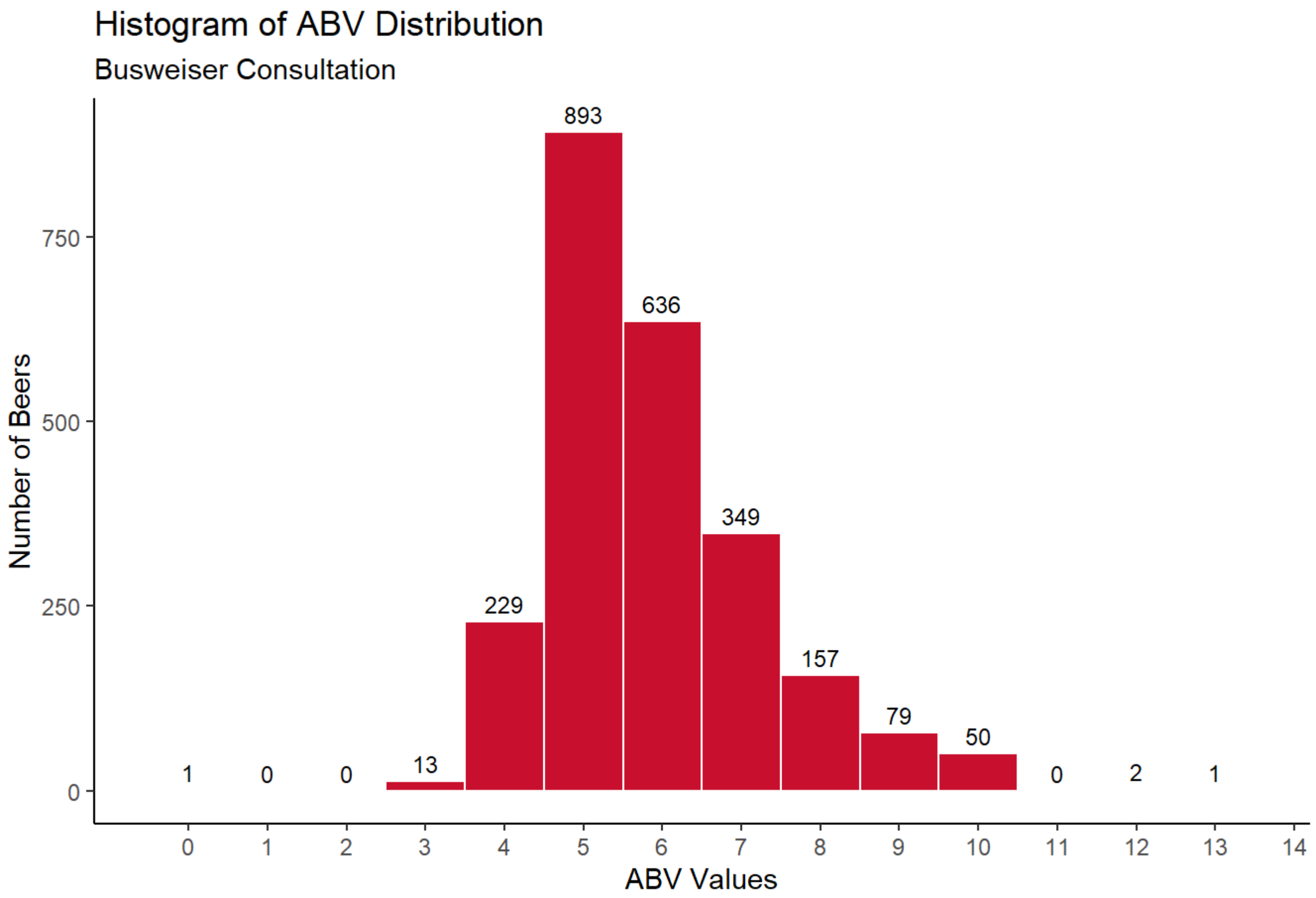
Budweiser Consultation

source: ABV. ABV imputed where necessary.

```
##################################################
############### Chart Max IBU #####################

maxStateValues %>%
  ggplot(aes(x=reorder(Division,MaxDivIBU), y=MaxStateIBU,fill= reorder(State,-MaxStateIBU))) +
  # Create stacked by chart organized by Division with States stacked in each bar
  geom_bar(aes(color = "#c8102e"),stat="identity", width= 0.7,
           position = position_stack(), show.legend = FALSE) +
  # Add state and ABV value to each state's chart position
  geom_text(aes(label = IBUmaxLabel), size = 3, position = position_stack(vjust = 0.5)) +
  # Add Division ABV Values to top of each chart stack
  geom_text(aes(Division, MaxDivIBU + SumStateIBU, label = MaxDivIBU),
            size = 3, nudge_y = -75, fontface = "italic") +
  # Label the chart objects
  labs(title="Max IBU by State by US Census Division in the USA",
       subtitle="Budweiser Consultation",
       caption="source: IBU imputed where necessary.",
       y = "Alcohol By Volume",
       x = "States by US Census Divisions ") +
  theme_classic() +
  # Remove y-labels and ticks since this is a stacked chart
  theme(axis.text.y = element_blank(), axis.ticks = element_blank()) +
  # Wrap X-axis labels
  scale_x_discrete(labels = function(x) str_wrap(x, width = 10))
```



```
######################
#                    #
#     Question 6     #
#                    #
######################
################################################################################################
#############
# Question 6 - Comment on the summary statistics and distribution of the ABV variable.
#
# We observed summary statistics from the ABV data showing that once we filled in the missing
# values as best as we could, there was a range of 0.10% to 12.80% with a median of 5.65% (media
n
# is simply the middle value if we were to arrange all the ABVs in either descending or ascendin
g order).
#
# We also found a very common range within the overall range that went from 5.0% ABV to 6.70% AB
V and
# upon further review noticed this is where many commonly mass produced beers fall, for example:
Bud
# Ice (5.5%), Bud Light Platinum (6%), Natural Ice (5.9%), Bud Ice (5.5%), Budweiser (5%), Blue
 Moon
# (5%), Stella Artois (5%), Heinekin (5%), Pabst Blue Ribbon (4.74%) and Miller Genuinine Draft
 (4.6%).
#
################################################################################################
#############
# Check on the distribution of ABV

# Add histogram of ABV distribution
buzzbrews %>%
  ggplot(aes(x = ABV)) +
  geom_histogram(binwidth = 1, color = "#ffffff", fill = "#c8102e") +
  # Adjust the scale to be able to mark-up the chart later in PowerPoint to match the
  # summary statistics - 1Q, middle 50%, 4Q
  scale_x_continuous(breaks = seq(0, 14, by = 1)) +
  stat_bin(binwidth = 1, aes(label = ..count..), vjust = -0.5, geom = "text", size = 3) +
  # Adjust titles
  labs(title = "Histogram of ABV Distribution",
     subtitle = "Busweiser Consultation",
     x = "ABV Values",
     y = "Number of Beers",
     caption = "source: ABV imputed where necessary.") +
  theme_classic()
```


Histogram of ABV Distribution

Busweiser Consultation

source: ABV imputed where necessary.

```
ABVsummary <- summary(buzzbrews$ABV)
ABVsummary
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.100   5.000   5.650   5.975   6.700  12.800
```

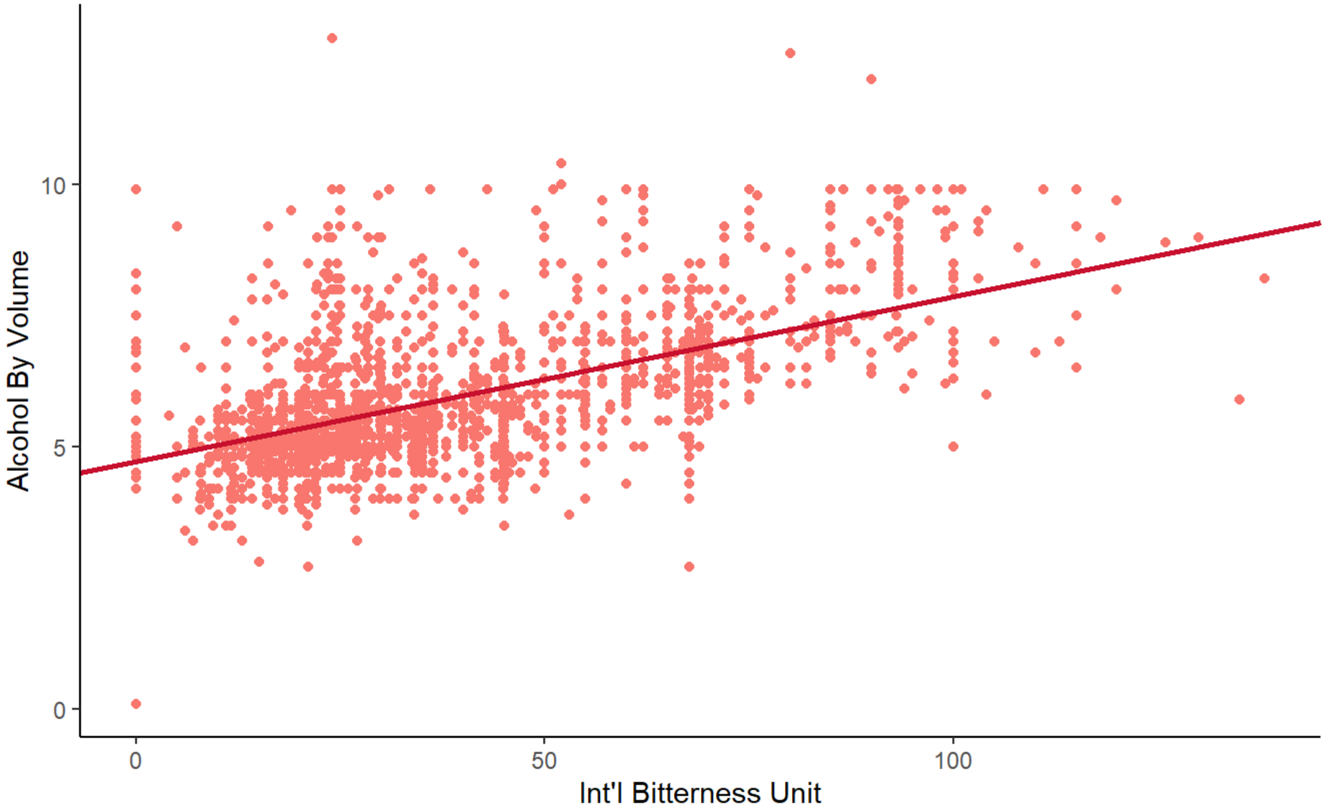
```
######################
#                    #
#     Question 7     #
#                    #
######################
################################################################################################
#############
# Question 7 - Is there an apparent relationship between the bitterness of the beer and its
# alcoholic content? Draw a scatter plot.  Make your best judgment of a relationship and
# EXPLAIN your answer.
#
# We used a scatter plot to viusally explore if there is any sort of relationship between
# IBU and ABV, in other words can IBU determine ABV or can ABV be used to determine IBU.
# There was evidence of a positive relationship, but and we will discuss this further shortly,
# it appears one can potentially predict the other.
#
################################################################################################
#############
# Label Ales, IPAs and neither - this is also setting up for the next question,
# but we want to be able to utilize it here
buzzbrews$IPAAle = case_when(grepl("\\bIPA\\b", buzzbrews$Beer, ignore.case = TRUE) ~ "IPA",
                             grepl("\\bindia pale ale\\b", buzzbrews$Beer, ignore.case = TRUE) ~
"IPA",
                             grepl("\\bale\\b", buzzbrews$Beer, ignore.case = TRUE ) ~ "Ale",
                             TRUE ~ "Neither")
view(buzzbrews)


## Calculate slope and intercept of line of best fit ##
comparisonCoef <- coef(lm(ABV ~ IBU, buzzbrews))
comparisonCoef
```

```
## (Intercept)         IBU
##  4.71799073  0.03142639
```

```
#  (Intercept)    MaxIBU
#   4.71799073  0.03142639

# Plot graph to show the correlation between ABV and IBU with an AB line
buzzbrews %>%
  ggplot(aes(x = IBU, y = ABV, color = "#c8102e")) +
  geom_point(show.legend = FALSE, na.rm = TRUE) +
  geom_abline(intercept =  comparisonCoef[1] , slope = comparisonCoef[2], color = "#c8102E", siz
e = 1) +
  theme_classic() +
  labs(title = "IBU vs ABV",
       subtitle = "Budweiser Consultation",
       y = "Alcohol By Volume",
       x = "Int'l Bitterness Unit",
       caption="ABV and IBU values imputed where necessary.")
```

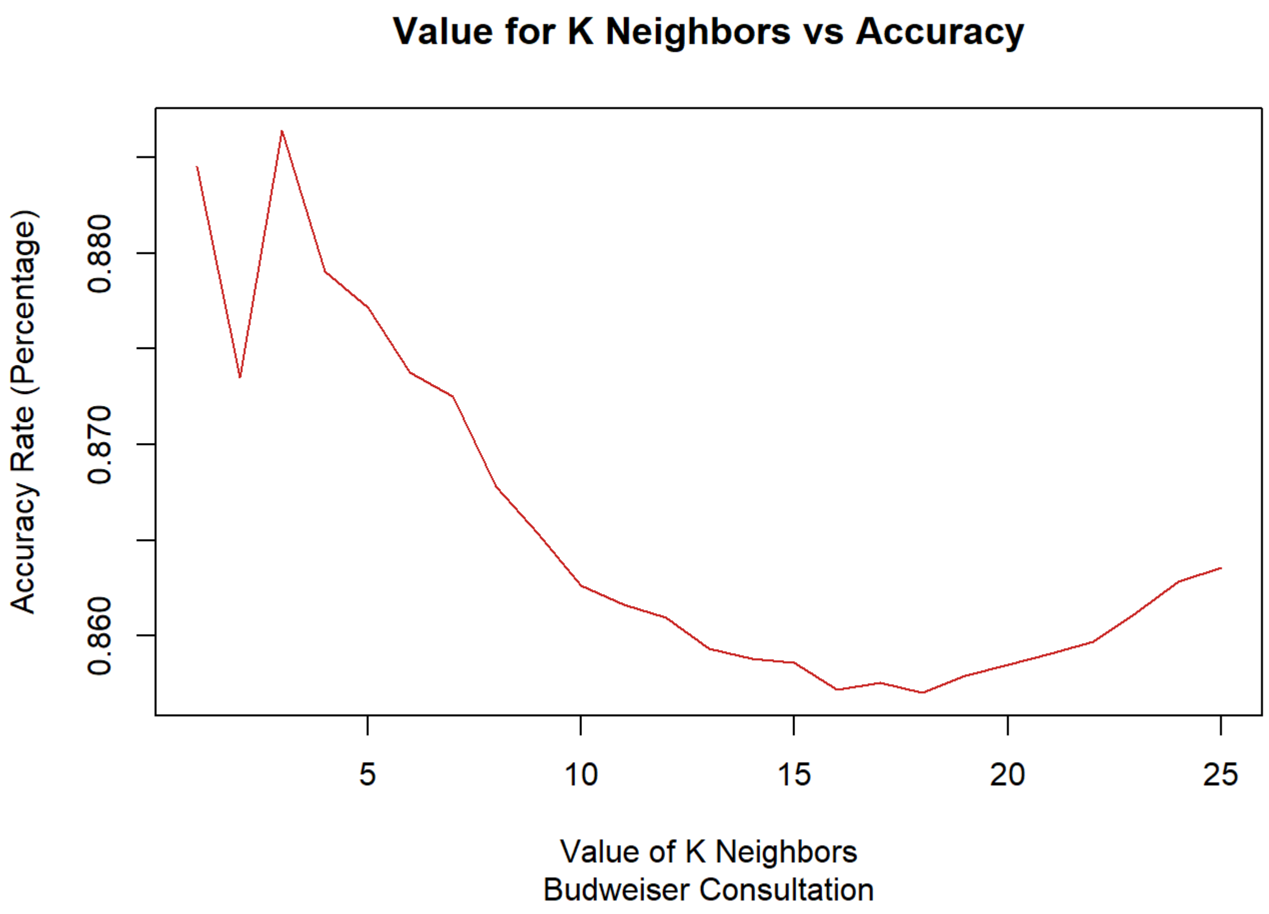
IBU vs ABV

Budweiser Consultation



ABV and IBU values imputed where necessary.

```
######################
#                    #
#     Question 8     #
#                    #
######################
################################################################################################
#############
# Question 8 - .  Budweiser would also like to investigate the difference with respect to IBU an
d ABV
# between IPAs (India Pale Ales) and other types of Ale (any beer with “Ale” in its name other t
han IPA).
# You decide to use KNN classification to investigate this relationship.  Provide statistical ev
idence one
# way or the other. You can of course assume your audience is comfortable with percentages … KNN
is very easy
# to understand conceptually.
# In addition, while you have decided to use KNN to investigate this relationship (KNN is requir
ed) you may
# also feel free to supplement your response to this question with any other methods or techniqu
es you have
# learned.  Creativity and alternative solutions are always encouraged.
#
# ## ## Response: ## ## #
# Knowing the client is very interested in seeing a kNN classifier to see if there is a
# difference between IPA and Ale beer (identified by name), we set up 2 tests, one looking
# at the difference between IPA and Ale and one that ran between IPA, Ale and Neither.
#
# First kNN Test - IPA vs Ale.
#
# In the first KNN test to classify IPA and Ales based on IBU and AVB values we saw an
# 87% accuracy overall with 90% to correctly classify Ale's and 83% accuracy to classify IPAs.
#
# Second kNN Test - IPA vs Ale vs Neither
#
# The second kNN test we ran was against all three classifications of IPa, Ale and Niether,
# again we started out exploring what the appropriate number of "neighbors" was to compare
# to since there is so many observations so close together (think New York city and all the
# noise generated). We found that generally 8-9 neighbors were the best estimation (we
# randomly parsed the data 100 times to find the best neighbors value).
# Our classifier was accurate in determining if a beer was an Ale, IPA or Neither between 59% an
d 67% of the # time (maybe 63%-64% to be more precise) when we used 8-9 nearest neighbors.
# Next we created some random pairings of IBU and ABV to see how the classifier handled the
# data and we saw a mean probabilities for IPAs 80% of the time, Neither at 59% of the time
# and Ale's 58% of the time.
# We also look a look at the ranges for IBU and ABV for each of the 3 broad types of beers IPA,
# Ale or "Neither" and found that Ales and Neither have very similar ABV Maxes and Minimums
# Ale: 3.5 - 12.8 and Neither: 0.1 - 12.5 (so it makes sense that it should be difficult to tell
# them apart just on the ABV). Similarly the IBU values for Ales are 7-120 and Neither are 0-13
0,
# with Neither encompasing Ales. Buy contrast, IPAs have ABU between 4 and 9.9 and IBU's between
# 19 and 138. While these numbers don't sound like a lot contrast, when we look at a scatter plo
t,
# we can visually confirm a lot of similiaritie between Ales and Neither versus IPAs that have a
# clustering with little overlap between the other two.
#
# NB Test to compare results
#
# We also utilized a Naive Bayes Classifier and in summary the classifier had 87% accuracy and
# correctly selects IPA about 91% of the time and Ales round # 83% of the time when just compari
ng
# the two.
#
# NB Test 2: IPA vs Ale vs Neither
#
# We also ran test a test using NB to see if we can better classify Ales, IPAs and Neither and
# found NB has a  also has a 63% accuracy rate. NB correctly selects IPA's 74% of the time,
# Neither (Ale or IPA) around 84% of the time and Ale's 0% of the time. These are actually
# quite different from the kNN classifier values, suggesting that using the nearest neighbor
# might be more accurate with data that is so overlapped in terms of ABV and IBU values.
#
#######################################################
###                                                 ###
###   For only  IPA $ Ale                           ###
###   Find the best value of K and train the model  ###
###                                                 ###
#######################################################

# Create new DF so not to confuse with buzzbrews
# Filter Ales and IPAs only
buzzKNN <- dplyr::filter(buzzbrews, IPAAle == "IPA" | IPAAle == "Ale")

iterations = 100
numks = 25
splitPerc = .70
set.seed(33)

masterAcc = matrix(nrow = iterations, ncol = numks)

for(j in 1:iterations)
{
accs = data.frame(accuracy = numeric(30), k = numeric(30))
trainIndices = sample(1:dim(buzzKNN)[1],round(splitPerc * dim(buzzKNN)[1]))
train = buzzKNN[trainIndices,]
test = buzzKNN[-trainIndices,]
for(i in 1:numks)
  {
  classifications = class::knn(train[,c(7,8)],test[,c(7,8)],train$IPAAle, prob = TRUE, k = i)
  table(classifications,test$IPAAle)
  CM = confusionMatrix(table(classifications,test$IPAAle))
  masterAcc[j,i] = CM$overall[1]
  }
}

MeanAcc = colMeans(masterAcc)
# Visually find the best value of k by using it's location in the dataframe based on the highest
Mean value
plot(seq(1,numks,1),MeanAcc, type = "l",
     col = "#c8201e",
     main = "Value for K Neighbors vs Accuracy",
     sub = "Budweiser Consultation",
     xlab = "Value of K Neighbors",
     ylab = "Accuracy Rate (Percentage)")
```



```
# Locate the value of k based on the best MeanAcc in the dataframe
kvalue = match(max(MeanAcc), MeanAcc)
max(MeanAcc)
```

```
## [1] 0.8863958
```

```
kvalue
```

```
## [1] 3
```

```
####### Best value of k = 3 around 87%% Accuracy #####################
####### Train the model using k = 3 #####################

classifications = class::knn(train[,c(7,8)],test[,c(7,8)],train$IPAAle, prob = TRUE, k = kvalue)
table(classifications,test$IPAAle)
```

```
##
## classifications Ale IPA
##             Ale 152  20
##             IPA  16  95
```

```
CM = confusionMatrix(table(classifications,test$IPAAle))

### Get details of test using CM ####
  CM
```

```
## Confusion Matrix and Statistics
##
##
## classifications Ale IPA
##             Ale 152  20
##             IPA  16  95
##
##                Accuracy : 0.8728
##                  95% CI : (0.8283, 0.9093)
##     No Information Rate : 0.5936
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7349
##
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.9048
##             Specificity : 0.8261
##          Pos Pred Value : 0.8837
##          Neg Pred Value : 0.8559
##              Prevalence : 0.5936
##          Detection Rate : 0.5371
##    Detection Prevalence : 0.6078
##       Balanced Accuracy : 0.8654
##
##        'Positive' Class : Ale
##
```

```
####### Test the Classifier with some random data ###
classifyMyBeers <- data.frame(ABV = c(6,6,5,4,5, 12, 7),
       IBU = c(78, 65, 55, 38, 100, 148, 98))
classifications = class::knn(train[,c(7,8)],classifyMyBeers,train$IPAAle, prob = TRUE, k = kvalu
e)

classifications
```

```
## [1] IPA IPA Ale Ale IPA IPA IPA
## attr(,"prob")
## [1] 1.000 1.000 0.875 0.750 1.000 1.000 1.000
## Levels: Ale IPA
```

```
############# Summary data by classification #############
IPAAleSummary <- buzzKNN %>%
  group_by(IPAAle) %>%
  dplyr::summarise(ABV.min = min(ABV),
                   ABV.med = median(ABV),
                   ABV.max = max(ABV),
                   IBU.min = min(IBU),
                   IBU.med = median(IBU),
                   IBU.max = max(IBU))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
view(IPAAleSummary)
IPAAleSummary
```

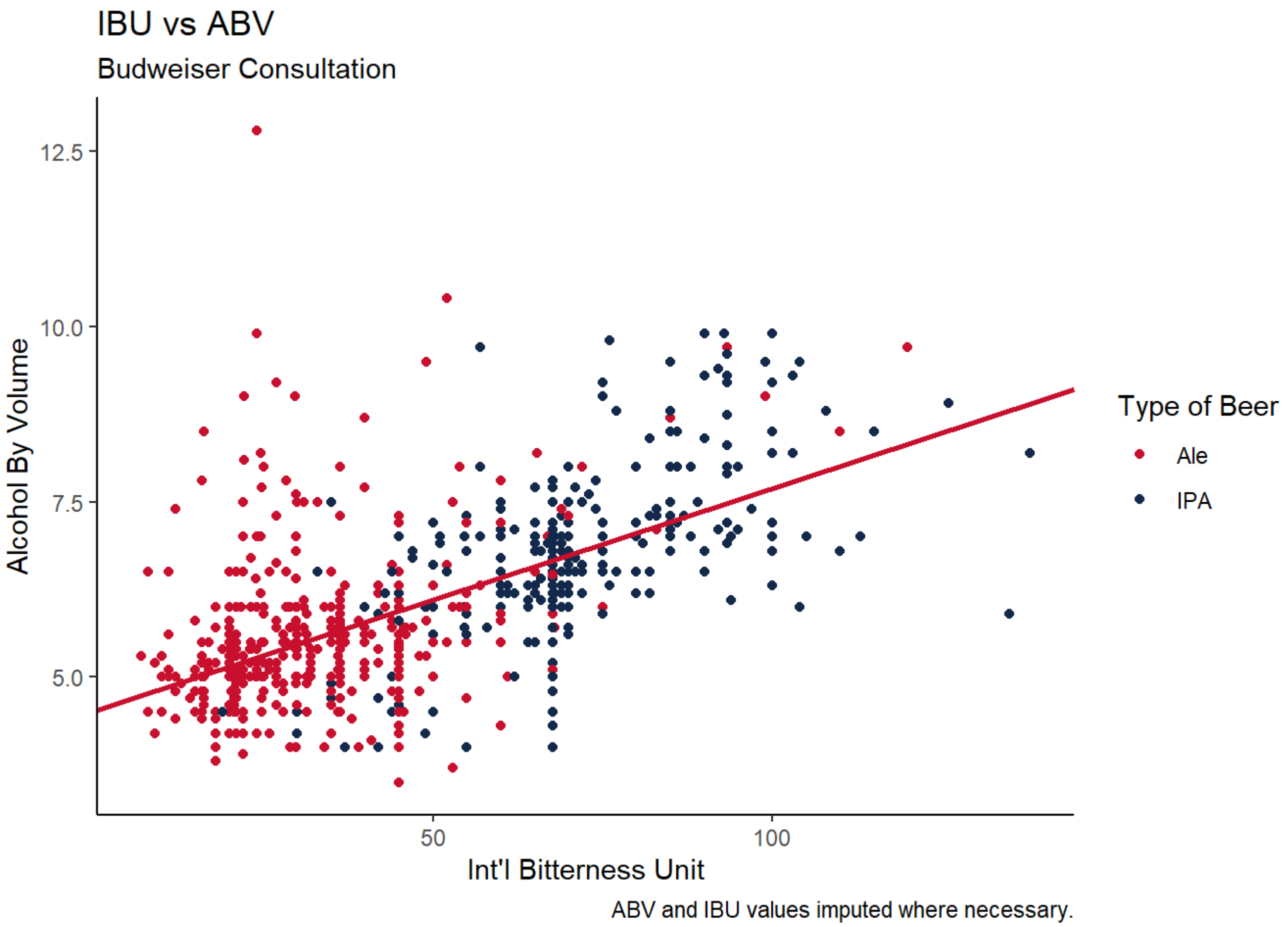
| IPAAle | ABV.min | ABV.med | ABV.max | IBU.min | IBU.med | IBU.max |
|---|---:|---:|---:|---:|---:|---:|
| <chr> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> |
| Ale | 3.5 | 5.4 | 12.8 | 7 | 31.00000 | 120 |
| IPA | 4.0 | 6.7 | 9.9 | 19 | 67.63455 | 138 |

2 rows

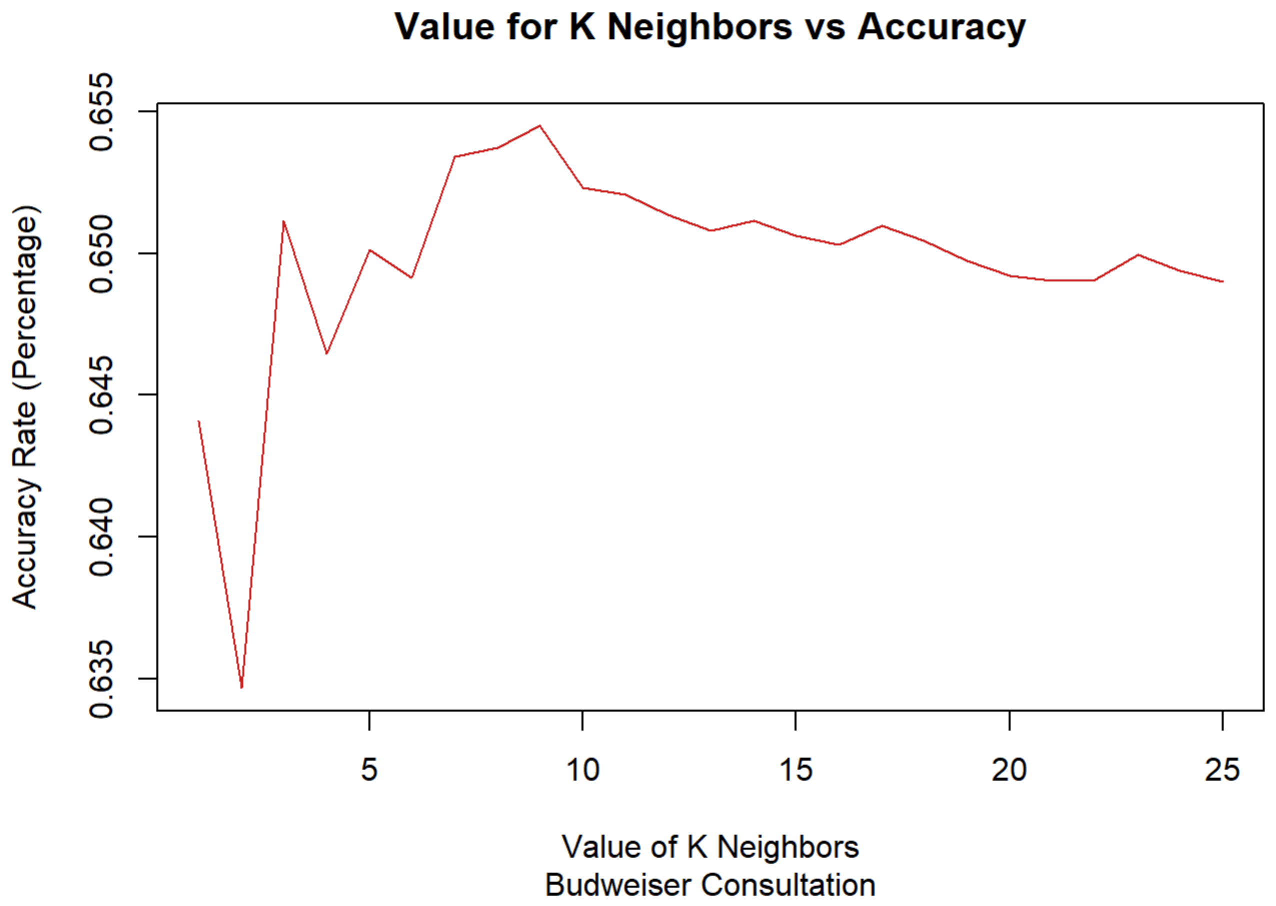
```
#########################################################
##### Replot and color by beer style ###################
comparisonCoef <- coef(lm(ABV ~ IBU, buzzKNN))
comparisonCoef
```

```
## (Intercept)         IBU
##  4.50992764  0.03172051
```

```
buzzKNN %>%
  ggplot(aes(x = IBU, y = ABV, color = IPAAle)) +
  geom_point(show.legend = TRUE, na.rm = TRUE) +
  geom_abline(intercept =  comparisonCoef[1] , slope = comparisonCoef[2], color = "#c8102E", siz
e = 1) +
  theme_classic() +
  labs(title = "IBU vs ABV",
       subtitle = "Budweiser Consultation",
       y = "Alcohol By Volume",
       x = "Int'l Bitterness Unit",
       caption="ABV and IBU values imputed where necessary.") +
  scale_color_manual(values = c("#c8102e","#13294b","#b1b3b3"),
                     name = "Type of Beer",
                     breaks = c("Ale", "IPA", "Neither"),
                     labels = c("Ale", "IPA", "Neither"))
```


IBU vs ABV
Budweiser Consultation
12.5
10.0
7.5
5.0
Alcohol By Volume
50
100
Int'l Bitterness Unit
Type of Beer
Ale
IPA
ABV and IBU values imputed where necessary.

```
#################################
#                               #
#           2nd kNN test        #
#       IPA - Ale = Niether     #
#################################

buzzKNN <- buzzbrews

iterations = 100
numks = 25
splitPerc = .70
set.seed(33)

masterAcc = matrix(nrow = iterations, ncol = numks)

for(j in 1:iterations)
{
accs = data.frame(accuracy = numeric(30), k = numeric(30))
trainIndices = sample(1:dim(buzzKNN)[1],round(splitPerc * dim(buzzKNN)[1]))
train = buzzKNN[trainIndices,]
test = buzzKNN[-trainIndices,]
for(i in 1:numks)
  {
  classifications = class::knn(train[,c(7,8)],test[,c(7,8)],train$IPAAle, prob = TRUE, k = i)
  table(classifications,test$IPAAle)
  CM = confusionMatrix(table(classifications,test$IPAAle))
  masterAcc[j,i] = CM$overall[1]
  }
}

MeanAcc = colMeans(masterAcc)
# Visually find the best value of k by using it's location in the dataframe based on the highest
Mean value
plot(seq(1,numks,1),MeanAcc, type = "l",
     col = "#c8201e",
     main = "Value for K Neighbors vs Accuracy",
     sub = "Budweiser Consultation",
     xlab = "Value of K Neighbors",
     ylab = "Accuracy Rate (Percentage)")
```



```
# Locate the value of k based on the best MeanAcc in the dataframe
kvalue = match(max(MeanAcc), MeanAcc)
max(MeanAcc)
```

```
## [1] 0.6544813
```

```
####### Best value of k = 8 between 59% - 67% Accuracy #####################
kvalue
```

```
## [1] 9
```

```
###### test the model using kvalue #####
classifications = class::knn(train[,c(7,8)],test[,c(7,8)],train$IPAAle, prob = TRUE, k = kvalue)
table(classifications,test$IPAAle)
```

```
##
## classifications Ale IPA Neither
##         Ale      67  12      49
##         IPA       9  64      55
##         Neither 112  30     325
```

```
CM = confusionMatrix(table(classifications,test$IPAAle))

### Get details of test using CM ####
  CM
```

```
## Confusion Matrix and Statistics
## 
## 
## classifications Ale IPA Neither
##         Ale      67  12      49
##         IPA       9  64      55
##         Neither 112  30     325
## 
## Overall Statistics
## 
##                Accuracy : 0.6307          
##                  95% CI : (0.5944, 0.666)
##     No Information Rate : 0.5934          
##     P-Value [Acc > NIR] : 0.02199         
## 
##                   Kappa : 0.3221          
## 
##  Mcnemar's Test P-Value : 4.24e-07        
## 
## Statistics by Class:
## 
##                      Class: Ale Class: IPA Class: Neither
## Sensitivity             0.35638    0.60377         0.7576
## Specificity             0.88598    0.89627         0.5170
## Pos Pred Value          0.52344    0.50000         0.6959
## Neg Pred Value          0.79664    0.92941         0.5938
## Prevalence              0.26003    0.14661         0.5934
## Detection Rate          0.09267    0.08852         0.4495
## Detection Prevalence    0.17704    0.17704         0.6459
## Balanced Accuracy       0.62118    0.75002         0.6373
```

```
####### Test the Classifier with some random data at the kvalue###
classifyMyBeers <- data.frame(ABV = c(6,6,5,4,5, 12, 7), 
       IBU = c(78, 65, 55, 38, 100, 148, 98))
classifications = class::knn(train[,c(7,8)],classifyMyBeers,train$IPAAle, prob = TRUE, k = kvalu
e)

classifications
```

```
## [1] Neither Ale     Ale     Neither IPA     Neither IPA    
## attr(,"prob")
## [1] 0.6666667 0.6666667 0.5000000 0.4444444 0.7777778 0.6666667 0.8181818
## Levels: Ale IPA Neither
```

```
############# Summary data by classification #############
IPAAleSummary <- buzzKNN %>%
  group_by(IPAAle) %>%
  dplyr::summarise(ABV.min = min(ABV),
                   ABV.med = median(ABV),
                   ABV.max = max(ABV),
                   IBU.min = min(IBU),
                   IBU.med = median(IBU),
                   IBU.max = max(IBU))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
view(IPAAleSummary)
IPAAleSummary
```

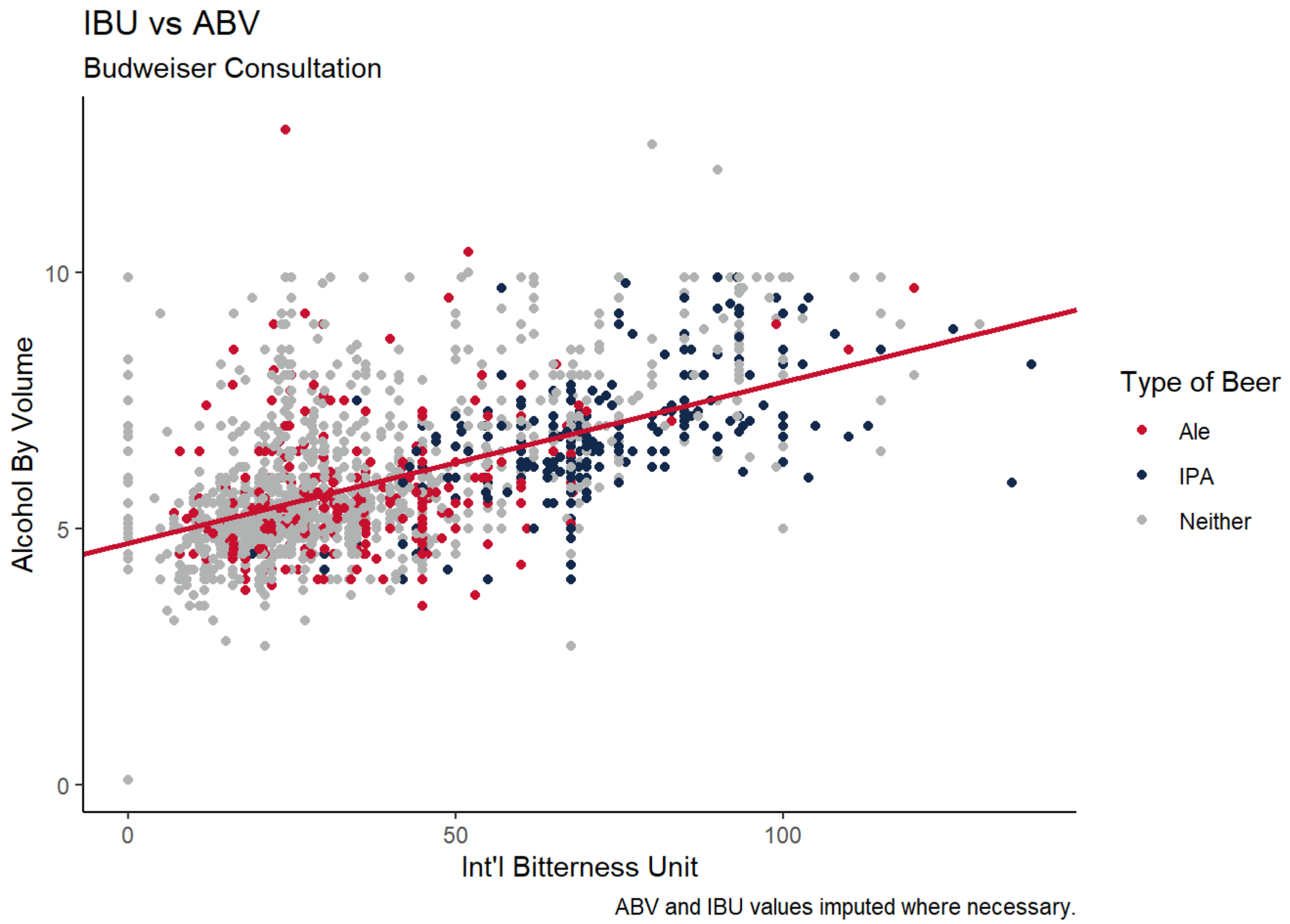
| IPAAle<br><chr> | ABV.min<br><dbl> | ABV.med<br><dbl> | ABV.max<br><dbl> | IBU.min<br><dbl> | IBU.med<br><dbl> | IBU.max<br><dbl> |
|---|---:|---:|---:|---:|---:|---:|
| Ale | 3.5 | 5.4 | 12.8 | 7 | 31.00000 | 120 |
| IPA | 4.0 | 6.7 | 9.9 | 19 | 67.63455 | 138 |
| Neither | 0.1 | 5.5 | 12.5 | 0 | 28.00000 | 130 |

3 rows

```
#
########################################################
#     Replot and color by beer style for all 3 classes #
########################################################
#
# Coefficient of the ABLine for the plot
comparisonCoef <- coef(lm(ABV ~ IBU, buzzKNN))
comparisonCoef
```

```
## (Intercept)         IBU
##  4.71799073  0.03142639
```

```
# plotgraph with ABLine overlay
buzzKNN %>%
  ggplot(aes(x = IBU, y = ABV, color = IPAAle)) +
  geom_point(show.legend = TRUE, na.rm = TRUE) +
  geom_abline(intercept =  comparisonCoef[1] , slope = comparisonCoef[2], color = "#c8102E", siz
e = 1) +
  theme_classic() +
  labs(title = "IBU vs ABV",
       subtitle = "Budweiser Consultation",
       y = "Alcohol By Volume",
       x = "Int'l Bitterness Unit",
       caption="ABV and IBU values imputed where necessary.") +
  scale_color_manual(values = c("#c8102e","#13294b","#b1b3b3"),
                     name = "Type of Beer",
                     breaks = c("Ale", "IPA", "Neither"),
                     labels = c("Ale", "IPA", "Neither"))
```



```
################################################################################################
#############################################
#Alternative technique for classifcation
#Hypothesis: Naive Bayes is a stronger ML technique for categorical data
#We implemented a Naive Bayes classifer based on IBU and ABV as a predictor of categotization of
style of beer (IPA,Ale, or Neither)
################################################################################################
#############################################
#Create new DF for Naive Bayes classifier (Don't want to interfere with original buzzbrews DF)
bayesDat <- dplyr::filter(buzzbrews, IPAAle == "IPA" | IPAAle == "Ale")
#Make the classifier happy and convert outcome to factor
bayesDat$IPAAle <- as.factor(bayesDat$IPAAle)
#Run this loop to run classifier 100 times to determine mean accuracy
iterations = 100
masterAcc = matrix(nrow = iterations,ncol=3)
#Begin the loop
for(j in 1:iterations)
{
#change seed each iteration
set.seed(j)

#Determine training and testing indicies
trainIndices = sample(seq(1:length(bayesDat$Beer)),round(.8*length(bayesDat$Beer)))
trainBeer = bayesDat[trainIndices,]
testBeer = bayesDat[-trainIndices,]

#Generate model, table, and confusion matrix
model = naiveBayes(trainBeer[,c(7,8)],trainBeer$IPAAle)
table(predict(model,testBeer[,c(7,8)]),testBeer$IPAAle)
CM = confusionMatrix(table(predict(model,testBeer[,c(7,8)]),testBeer$IPAAle))

#Insert current accuracies
masterAcc[j,1] = CM$overall[1]
masterAcc[j,2] = CM$byClass[1]
masterAcc[j,3] = CM$byClass[2]
}
#Mean accuracy
MeanAcc = colMeans(masterAcc)

MeanAcc
```

```
## [1] 0.8710053 0.8941920 0.8351311
```

```
#Confusion matrix
CM
```

```
## Confusion Matrix and Statistics
## 
## 
##      Ale IPA
##   Ale 102  13
##   IPA  11  63
## 
##                Accuracy : 0.873
##                  95% CI : (0.817, 0.9169)
##     No Information Rate : 0.5979
##     P-Value [Acc > NIR] : <2e-16
## 
##                   Kappa : 0.7348
## 
##  Mcnemar's Test P-Value : 0.8383
## 
##             Sensitivity : 0.9027
##             Specificity : 0.8289
##          Pos Pred Value : 0.8870
##          Neg Pred Value : 0.8514
##              Prevalence : 0.5979
##          Detection Rate : 0.5397
##    Detection Prevalence : 0.6085
##       Balanced Accuracy : 0.8658
## 
##        'Positive' Class : Ale
## 
```

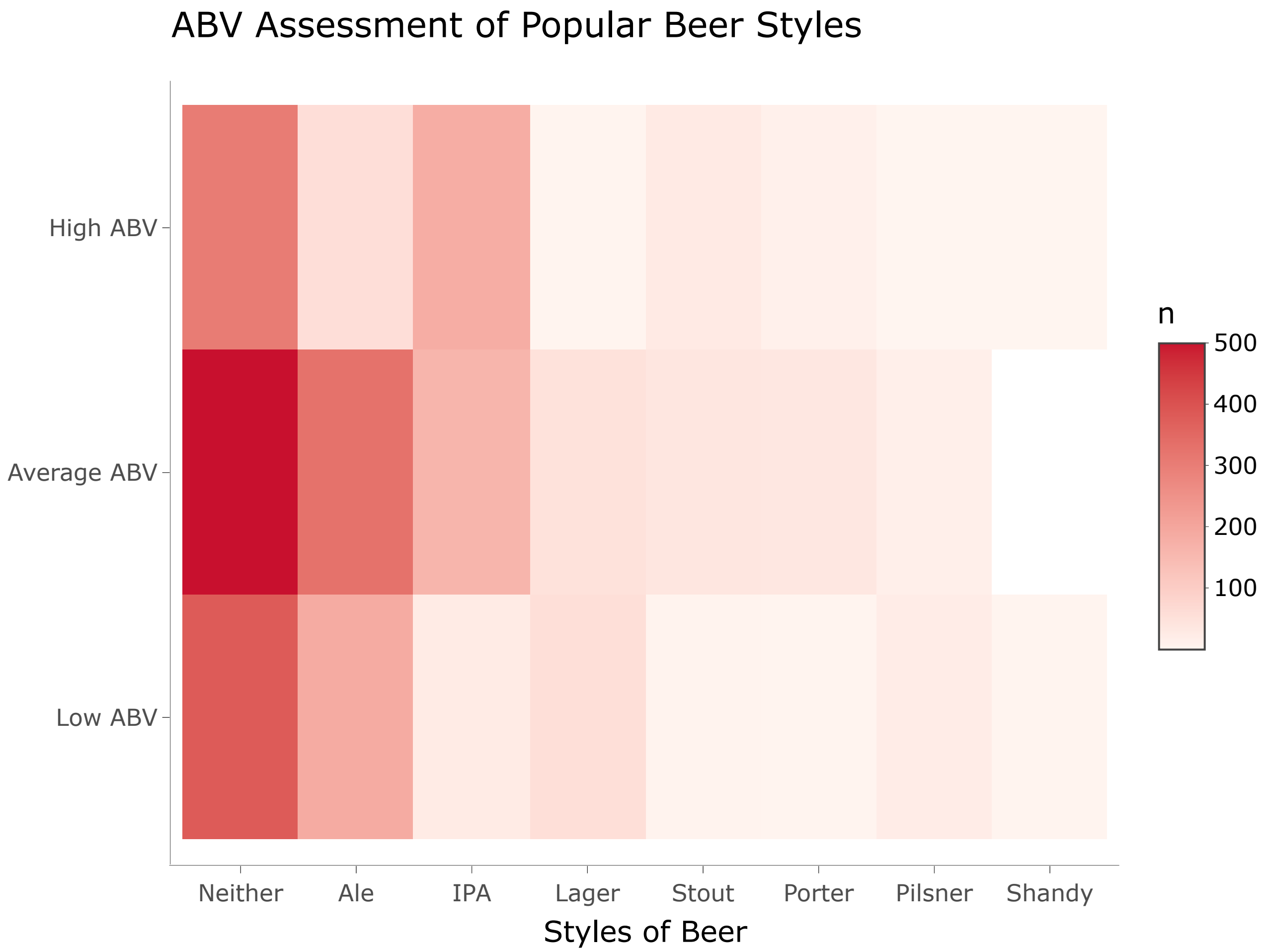
```
##################################
# 2nd Test of NB using all IPA vs Ale vs Neither
#
##################################
#Create new DF for Naive Bayes classifier (Don't want to interfere with original buzzbrews DF)
bayesDat <- buzzbrews
#Make the classifier happy and convert outcome to factor
bayesDat$IPAAle <- as.factor(bayesDat$IPAAle)
#Run this loop to run classifier 100 times to determine mean accuracy
iterations = 100
masterAcc = matrix(nrow = iterations,ncol=3)
#Begin the loop
for(j in 1:iterations)
{
#change seed each iteration
set.seed(j)

#Determine training and testing indicies
trainIndices = sample(seq(1:length(bayesDat$Beer)),round(.7*length(bayesDat$Beer)))
trainBeer = bayesDat[trainIndices,]
testBeer = bayesDat[-trainIndices,]

#Generate model, table, and confusion matrix
model = naiveBayes(trainBeer[,c(7,8)],trainBeer$IPAAle)
table(predict(model,testBeer[,c(7,8)]),testBeer$IPAAle)
CM = confusionMatrix(table(predict(model,testBeer[,c(7,8)]),testBeer$IPAAle))

#Insert current accuracies
masterAcc[j,1] = CM$overall[1]
masterAcc[j,2] = CM$byClass[1]
masterAcc[j,3] = CM$byClass[2]
}
#Mean accuracy
MeanAcc = colMeans(masterAcc)

MeanAcc
```

```
## [1] 0.6301521 0.0000000 0.6806239
```

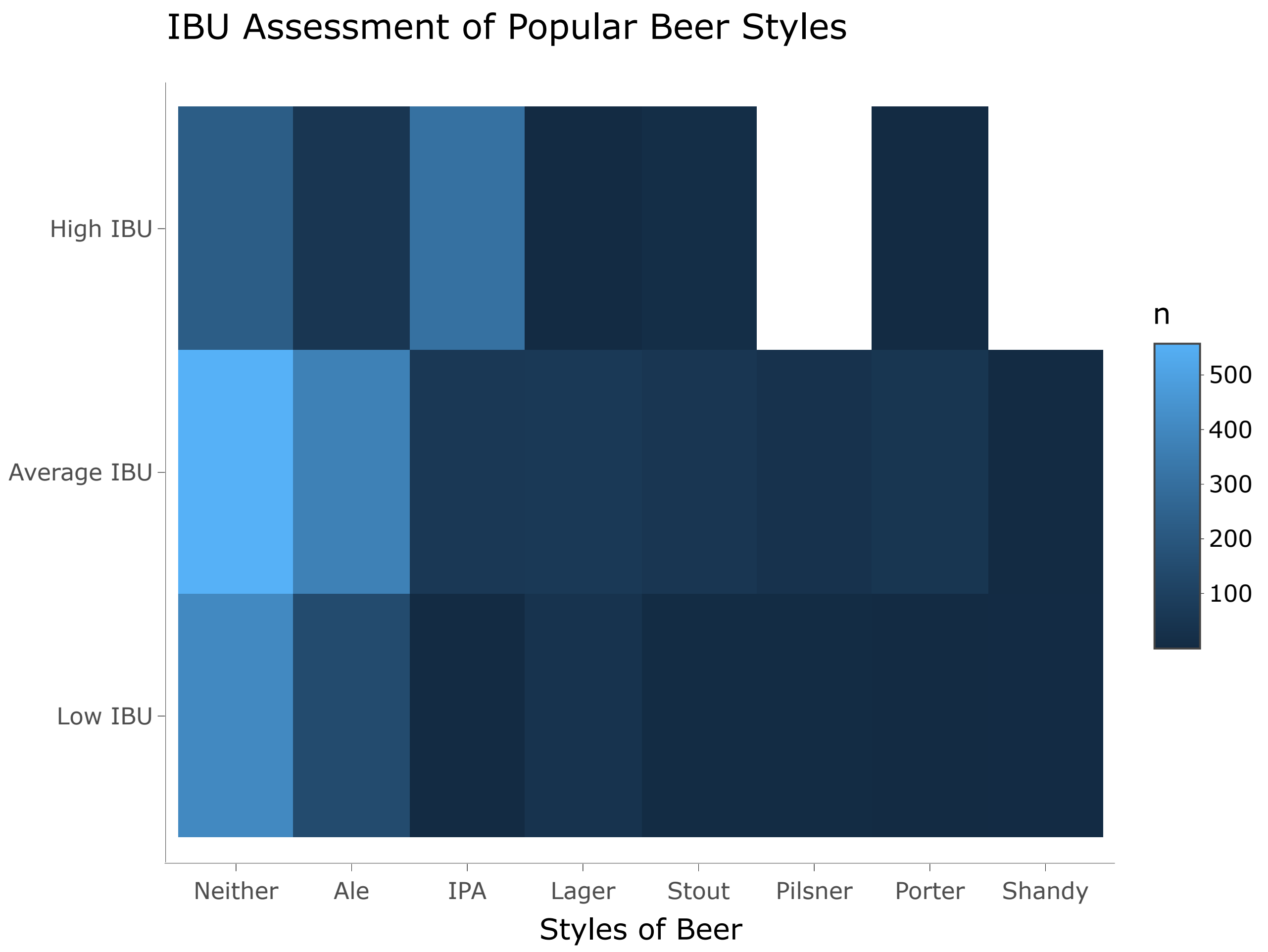
```
#Confusion matrix
CM
```

```
## Confusion Matrix and Statistics
## 
##            
##           Ale IPA Neither
##   Ale       0   0       0
##   IPA       9  78      65
##   Neither 157  40     374
## 
## Overall Statistics
##                                           
##                Accuracy : 0.6252          
##                  95% CI : (0.5887, 0.6606)
##     No Information Rate : 0.6072          
##     P-Value [Acc > NIR] : 0.1706          
##                                           
##                   Kappa : 0.229           
##                                           
##  Mcnemar's Test P-Value : <2e-16          
## 
## Statistics by Class:
## 
##                      Class: Ale Class: IPA Class: Neither
## Sensitivity              0.0000     0.6610         0.8519
## Specificity              1.0000     0.8777         0.3063
## Pos Pred Value              NaN     0.5132         0.6550
## Neg Pred Value           0.7704     0.9299         0.5724
## Prevalence               0.2296     0.1632         0.6072
## Detection Rate           0.0000     0.1079         0.5173
## Detection Prevalence     0.0000     0.2102         0.7898
## Balanced Accuracy        0.5000     0.7694         0.5791
```

```
######################
#                    #
#     Question 9     #
#                    #
######################
################################################################################################
#############
#Knock their socks off!  Find one other useful inference from the data that you feel Budweiser m
ay be able to find value in.  You must convince them why it is important and back up your convic
tion with appropriate statistical evidence.
################################################################################################
#############
##################################################################
#
##### Response #####
#
# There appears to be opportunity to create a new product where one does not
# exist in the Shandy category. Shandy is similar to a lemon seltzer, Budweiser
# has several products on the market already, but not a Budweiser labeled product
# that I could find. There are four Shandy's in the data set, three that have an AVB
# higher than 5.0 and one with an AVB higher than 6.7, leave opportunity at all levels
# of AVB, but particularly an AVB greater than 5.0.
# There also appears to be an opportunity to create an Ale with an AVB greater than 6.7
# as there are only 58 Ales out of 573 about 10% that have an AVB greater than 6.7.
# Finally, there appears to be an opportunity to create a Lager with an AVB greater
# than 6.7 as there are only 3 out of 109 Lagers, less than 2.7% in the data set that
# have an AVB greater than 6.7.
#####################################################################################
####################################################################################


# add lager, stout, pilsner, zymurgy, shandy, porter to the broader style list
buzzbrews$IPAAle = case_when(grepl("\\bIPA\\b", buzzbrews$Beer, ignore.case = TRUE) ~ "IPA",
                             grepl("\\bindia pale ale\\b", buzzbrews$Beer, ignore.case = TRUE) ~
"IPA",
                             grepl("\\bale\\b", buzzbrews$Beer, ignore.case = TRUE ) ~ "Ale",
                             grepl("\\blager\\b", buzzbrews$Beer, ignore.case = TRUE) ~ "Lager",
                             grepl("\\bstout\\b", buzzbrews$Beer, ignore.case = TRUE) ~ "Stout",
                             grepl("\\bpilsner\\b", buzzbrews$Beer, ignore.case = TRUE) ~ "Pilsn
er",
                             grepl("\\bzymurgy\\b", buzzbrews$Beer, ignore.case = TRUE) ~ "Zymur
gy",
                             grepl("\\bshandy\\b", buzzbrews$Beer, ignore.case = TRUE) ~ "Shand
y",
                             grepl("\\bporter\\b", buzzbrews$Beer, ignore.case = TRUE) ~ "Porte
r",
                             TRUE ~ "Neither")
buzzbrews$IPAAle <- as.factor(buzzbrews$IPAAle)

# plot style by ABV
beermeABV <- buzzbrews %>%
  mutate(ABVControlFact = cut(ABV, breaks = c(0,5,6.7,12.8),
                              labels = c("Low ABV","Average ABV","High ABV"))) %>%
  count(IPAAle, ABVControlFact) %>%
  ggplot(aes(x=reorder(IPAAle, -n), ABVControlFact)) +
  geom_tile(mapping = aes(fill = n)) +
  scale_fill_gradient(low = "#fff5f0", high = "#c8102e") +
  labs(title = "ABV Assessment of Popular Beer Styles",
       subtitle = "Budweiser Consultation",
       y = NULL,
       x = "Styles of Beer",
       caption="ABV values imputed where necessary.") +
  theme_classic()
ggplotly(beermeABV)
```


ABV Assessment of Popular Beer Styles

```
# plot style by IBU
beermeIBU <- buzzbrews %>%
  mutate(IBUControlFact = cut(IBU, breaks = c(-10,21,57, 138),
                              labels = c("Low IBU","Average IBU","High IBU"))) %>%
  count(IPAAle, IBUControlFact) %>%
  ggplot(aes(x=reorder(IPAAle, -n), IBUControlFact)) +
  geom_tile(mapping = aes(fill = n)) +
  labs(title = "IBU Assessment of Popular Beer Styles",
       subtitle = "Budweiser Consultation",
       y = NULL,
       x = "Styles of Beer",
       caption="IBU values imputed where necessary.") +
  theme_classic()
ggplotly(beermeIBU)
```



```
view(summary(buzzbrews$IPAAle))
```

```
##############################################################
#                                                            #
######        Budweiser Case Study Conclusion           ######
#                                                            #
##############################################################
#
# In conclusion to this case study for Budweiser we observed which states have the most brewerie
s, the importance or lack of importance, usefulness and ability to discover possible business op
portunities using AVB and IBU values. We discovered that beyond just the numerical values associ
ated with IBUs, it can be used to estimate bitterness as well as to establish a system of qualit
y control for breweries. We discovered that most beers in the USA have an AVB between 5.0 and 6.
7 and there are extreme outliers to these values, such as Lee Hill Series Vol 5 made in Colorado
with a 12.8% ABv. We also discovered that IBU and AVB values are independent from one another, b
ut share a positive correlated relationship that can help in identifying some beer styles from o
thers, IPA's from Ale's for example. Overall, we find that AVB and IBU values are established me
tric for the beer industry to determine quality, estimate bitterness, describe alcohol by volume
of beer and make for great marketing tools.
```