Data 624 Homework 3

Warner Alexis

2025-03-02

## Data Preprocessing

**Exercise 3.1** The UC Irvine Machine Learning Repository6 contains a data set related to glass identification. The data consist of 214 glass samples labeled as one of seven class categories. There are nine predictors, including the refractive index and percentages of eight elements: Na, Mg, Al, Si, K, Ca, Ba, and Fe.

1. Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors.

library(mlbench)  
data("Glass")  
str(Glass)

## 'data.frame': 214 obs. of 10 variables:  
## $ RI : num 1.52 1.52 1.52 1.52 1.52 ...  
## $ Na : num 13.6 13.9 13.5 13.2 13.3 ...  
## $ Mg : num 4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...  
## $ Al : num 1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...  
## $ Si : num 71.8 72.7 73 72.6 73.1 ...  
## $ K : num 0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...  
## $ Ca : num 8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...  
## $ Ba : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ Fe : num 0 0 0 0 0 0.26 0 0 0 0.11 ...  
## $ Type: Factor w/ 6 levels "1","2","3","5",..: 1 1 1 1 1 1 1 1 1 1 ...

library(corrplot)

## corrplot 0.94 loaded

library(reshape2)  
library(GGally)

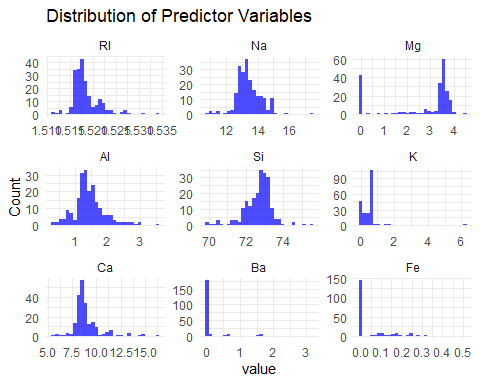
## Loading required package: ggplot2

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

library(caret)

## Loading required package: lattice

# use only numerical values   
# Convert data to long format, ensuring all numeric values  
glass\_long <- melt(Glass, id.vars = "Type")   
  
  
# histogram   
ggplot(glass\_long, aes(x= value)) +  
 geom\_histogram(bins=30, fill = 'blue', alpha=0.7) +  
 facet\_wrap(~ variable, scales = 'free') +  
 theme\_minimal() +  
 labs(title = 'Distribution of Predictor Variables', x = "value", y = "Count" )

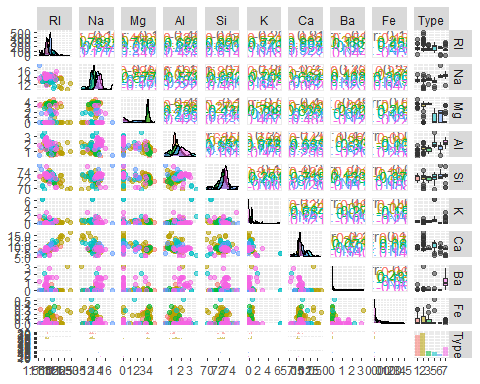


# view Scatter plot of for relationship   
ggpairs(Glass, aes(color=Type, alpha=0.5))

## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero  
  
## Warning in cor(x, y): the standard deviation is zero

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

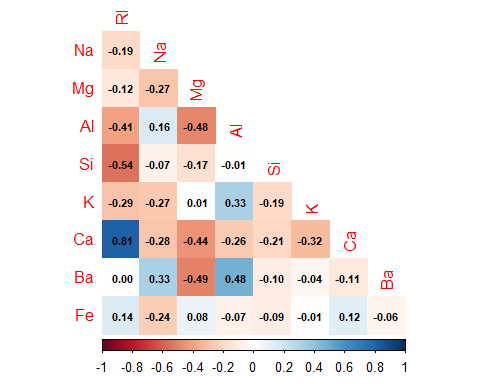
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# use corrplot library  
library(dplyr)

##   
## Attaching package: 'dplyr'  
##   
## The following objects are masked from 'package:stats':  
##   
## filter, lag  
##   
## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

Glass %>% select( -Type) %>% cor() %>% corrplot(,   
 method="color",  
 diag=FALSE,  
 type="lower",  
 addCoef.col = "black",  
 number.cex=0.70)



The pair plot provides a comprehensive visual analysis of the relationships between numerical variables in the **Glass** dataset. The diagonal elements display histograms and density plots, showing the distribution of each variable. Some variables, such as **RI, Si, and Ca**, exhibit skewed distributions, indicating potential transformations may be necessary. The lower triangle contains scatterplots illustrating relationships between variable pairs, with color-coded points representing different glass types. Notable patterns in these scatterplots suggest possible correlations.

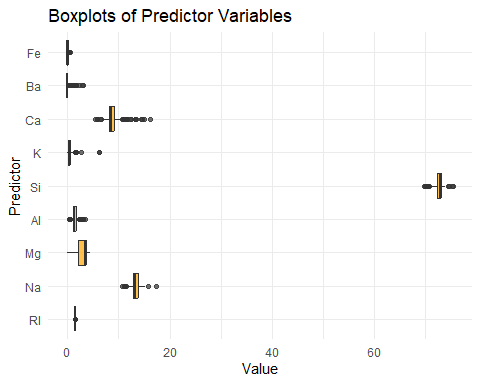
The upper triangle presents correlation values, highlighting significant relationships between variables. For example, **Si and RI, as well as Ca and Ba,** show strong positive correlations, whereas **Na and RI, along with Mg and Ca,** have moderate associations. The significance of these correlations, denoted by asterisks, indicates which variables may be most relevant for classification. Additionally, the rightmost column features boxplots that reveal variations in each variable across different glass types, emphasizing the importance of certain features such as **Si, Na, and Ca** in distinguishing between categories.

Several variables, particularly **K, Ba, and Fe**, exhibit outliers, which may impact modeling and require further preprocessing. Given these observations, transformations such as **logarithmic scaling or normalization** could help address skewness and improve classification accuracy. The insights from this visualization suggest that selecting relevant predictors and handling data irregularities appropriately will be crucial for developing an effective classification model for glass type identification.

1. Do there appear to be any outliers in the data? Are any predictors skewed?

To check skewness: A skewness value > 1 or < -1 indicates a highly skewed distribution.

# Boxplots for each predictor  
ggplot(glass\_long, aes(x = variable, y = value)) +  
 geom\_boxplot(fill = "orange", alpha = 0.7) +  
 theme\_minimal() +  
 coord\_flip() +  
 labs(title = "Boxplots of Predictor Variables", x = "Predictor", y = "Value")



# Load library for skewness  
library(e1071)  
  
# Calculate skewness for each numeric variable  
skewness\_values <- sapply(Glass[, -10], skewness)  
  
# Display skewness values  
print(skewness\_values)

## RI Na Mg Al Si K Ca   
## 1.6027151 0.4478343 -1.1364523 0.8946104 -0.7202392 6.4600889 2.0184463   
## Ba Fe   
## 3.3686800 1.7298107

Na appears to be approximately normally distributed with a slight right skew, while Al, RI, and Ca also exhibit right-skewed distributions. In contrast, Fe, Ba, and K are highly right-skewed. Si shows a left-skewed distribution, whereas Mg appears bimodal and also leans towards a left skew.

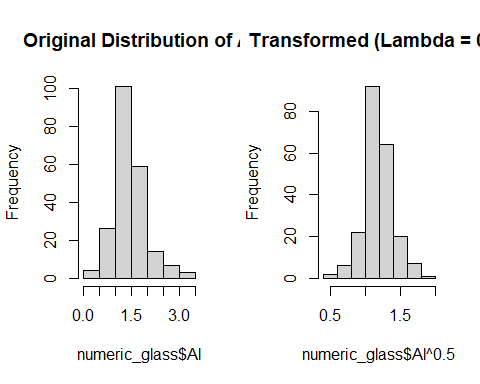
The boxplots reveal the presence of multiple outliers across most variables, with the exception of Mg, which appears to have fewer or no significant outliers.

1. Are there any relevant transformations of one or more predictors that might improve the classification model?

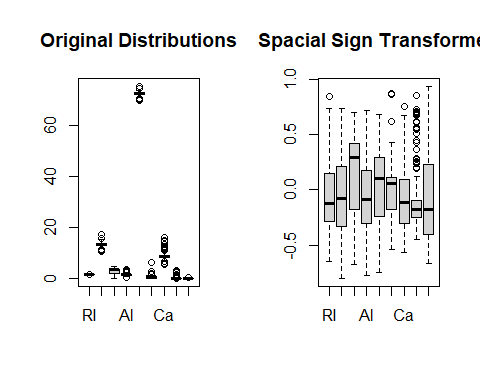
numeric\_glass <- Glass %>% select( -Type)  
par(mfrow=c(1,2))  
BoxCoxTrans(numeric\_glass$Al)

## Box-Cox Transformation  
##   
## 214 data points used to estimate Lambda  
##   
## Input data summary:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.290 1.190 1.360 1.445 1.630 3.500   
##   
## Largest/Smallest: 12.1   
## Sample Skewness: 0.895   
##   
## Estimated Lambda: 0.5

hist(numeric\_glass$Al, main='Original Distribution of Al')  
hist(numeric\_glass$Al\*\*.5, main='Transformed (Lambda = 0.5)')



# Spacial sign transformation of predictors  
boxplot(numeric\_glass, main='Original Distributions')  
boxplot(caret::spatialSign(scale(numeric\_glass)), main='Spacial Sign Transformed')



Yes, applying transformations to certain predictors in the **Glass** dataset can improve the classification model. Right-skewed variables such as **Fe, Ba, K, Al, RI, and Ca** would benefit from **Box-Cox, log, or square root transformations** to reduce skewness and stabilize variance. Left-skewed variables like **Si and Mg** may require **reflection and log transformation** to normalize their distributions. Additionally, **Mg**, which appears bimodal, could be handled through **clustering or smoothing techniques**. Outlier-prone variables like **Ba, Fe, and K** may require **Winsorization or robust scaling** to limit their impact on the model. These transformations help improve normality, feature scaling, and overall model performance, making classification more effective.

**Exercise 3.2** The soybean data can also be found at the UC Irvine Machine Learning Repository. Data were collected to predict disease in 683 soybeans. The 35 predictors are mostly categorical and include information on the environmental conditions (e.g., temperature, precipitation) and plant conditions (e.g., left spots, mold growth). The outcome labels consist of 19 distinct classes.

1. Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?

Degenerate distributions are those that have only a single possible value. In the dataset, **mycelium** and **sclerotia** appear to be degenerate, showing no variation. Additionally, **leaf.mild** and **leaf.malf** are nearly one-sided, with most observations falling into a single category when excluding missing values.

# Load necessary library  
library(mlbench)  
  
# Load the Soybean dataset  
data(Soybean)  
  
# Calculate missing percentage for each column  
missing\_percent <- colSums(is.na(Soybean)) / nrow(Soybean) \* 100  
  
# Convert to a data frame for better visualization  
missing\_df <- data.frame(Variable = names(missing\_percent), Missing\_Percentage = missing\_percent)  
  
# Print missing percentages sorted in descending order  
missing\_df <- missing\_df[order(-missing\_df$Missing\_Percentage), ]  
print(missing\_df)

## Variable Missing\_Percentage  
## hail hail 17.7159590  
## sever sever 17.7159590  
## seed.tmt seed.tmt 17.7159590  
## lodging lodging 17.7159590  
## germ germ 16.3982430  
## leaf.mild leaf.mild 15.8125915  
## fruiting.bodies fruiting.bodies 15.5197657  
## fruit.spots fruit.spots 15.5197657  
## seed.discolor seed.discolor 15.5197657  
## shriveling shriveling 15.5197657  
## leaf.shread leaf.shread 14.6412884  
## seed seed 13.4699854  
## mold.growth mold.growth 13.4699854  
## seed.size seed.size 13.4699854  
## leaf.halo leaf.halo 12.2986823  
## leaf.marg leaf.marg 12.2986823  
## leaf.size leaf.size 12.2986823  
## leaf.malf leaf.malf 12.2986823  
## fruit.pods fruit.pods 12.2986823  
## precip precip 5.5636896  
## stem.cankers stem.cankers 5.5636896  
## canker.lesion canker.lesion 5.5636896  
## ext.decay ext.decay 5.5636896  
## mycelium mycelium 5.5636896  
## int.discolor int.discolor 5.5636896  
## sclerotia sclerotia 5.5636896  
## plant.stand plant.stand 5.2708638  
## roots roots 4.5387994  
## temp temp 4.3923865  
## crop.hist crop.hist 2.3426061  
## plant.growth plant.growth 2.3426061  
## stem stem 2.3426061  
## date date 0.1464129  
## area.dam area.dam 0.1464129  
## Class Class 0.0000000  
## leaves leaves 0.0000000

head(Soybean)

## Class date plant.stand precip temp hail crop.hist area.dam  
## 1 diaporthe-stem-canker 6 0 2 1 0 1 1  
## 2 diaporthe-stem-canker 4 0 2 1 0 2 0  
## 3 diaporthe-stem-canker 3 0 2 1 0 1 0  
## 4 diaporthe-stem-canker 3 0 2 1 0 1 0  
## 5 diaporthe-stem-canker 6 0 2 1 0 2 0  
## 6 diaporthe-stem-canker 5 0 2 1 0 3 0  
## sever seed.tmt germ plant.growth leaves leaf.halo leaf.marg leaf.size  
## 1 1 0 0 1 1 0 2 2  
## 2 2 1 1 1 1 0 2 2  
## 3 2 1 2 1 1 0 2 2  
## 4 2 0 1 1 1 0 2 2  
## 5 1 0 2 1 1 0 2 2  
## 6 1 0 1 1 1 0 2 2  
## leaf.shread leaf.malf leaf.mild stem lodging stem.cankers canker.lesion  
## 1 0 0 0 1 1 3 1  
## 2 0 0 0 1 0 3 1  
## 3 0 0 0 1 0 3 0  
## 4 0 0 0 1 0 3 0  
## 5 0 0 0 1 0 3 1  
## 6 0 0 0 1 0 3 0  
## fruiting.bodies ext.decay mycelium int.discolor sclerotia fruit.pods  
## 1 1 1 0 0 0 0  
## 2 1 1 0 0 0 0  
## 3 1 1 0 0 0 0  
## 4 1 1 0 0 0 0  
## 5 1 1 0 0 0 0  
## 6 1 1 0 0 0 0  
## fruit.spots seed mold.growth seed.discolor seed.size shriveling roots  
## 1 4 0 0 0 0 0 0  
## 2 4 0 0 0 0 0 0  
## 3 4 0 0 0 0 0 0  
## 4 4 0 0 0 0 0 0  
## 5 4 0 0 0 0 0 0  
## 6 4 0 0 0 0 0 0

library(dplyr)  
library(tidyr)

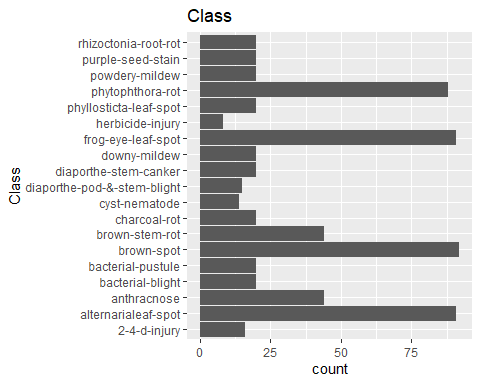
##   
## Attaching package: 'tidyr'

## The following object is masked from 'package:reshape2':  
##   
## smiths

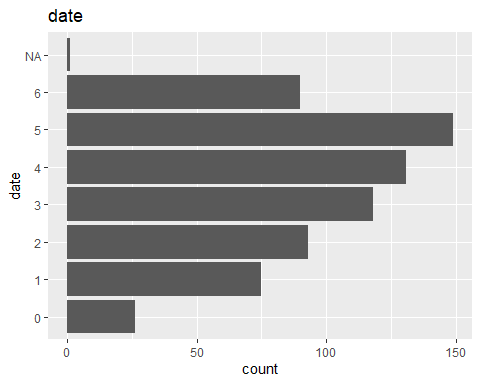
columns <- colnames(Soybean)  
  
lapply(columns,  
 function(col) {  
 ggplot(Soybean,   
 aes\_string(col)) + geom\_bar() + coord\_flip() + ggtitle(col)})

## Warning: `aes\_string()` was deprecated in ggplot2 3.0.0.  
## ℹ Please use tidy evaluation idioms with `aes()`.  
## ℹ See also `vignette("ggplot2-in-packages")` for more information.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

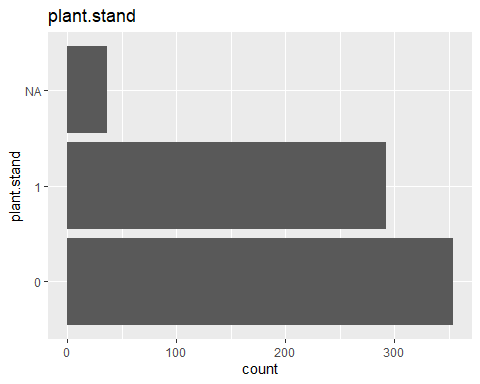
## [[1]]



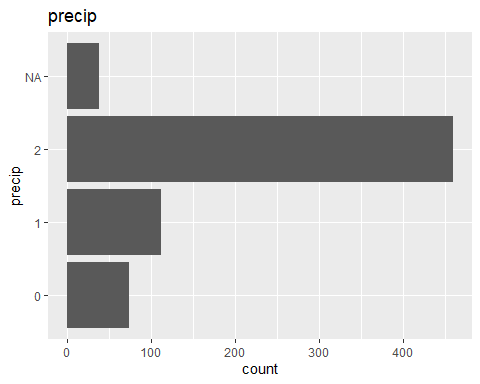
##   
## [[2]]



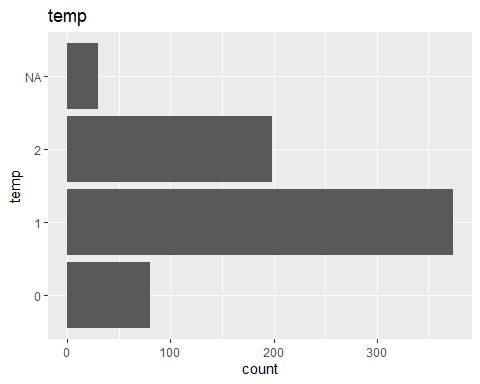
##   
## [[3]]



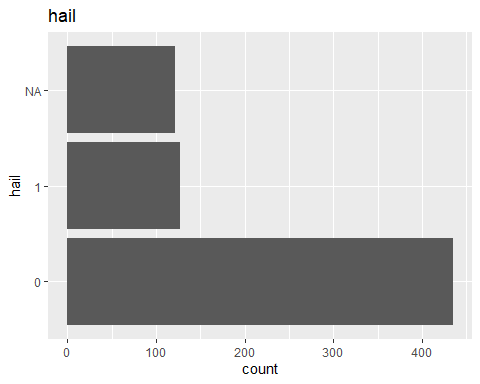
##   
## [[4]]



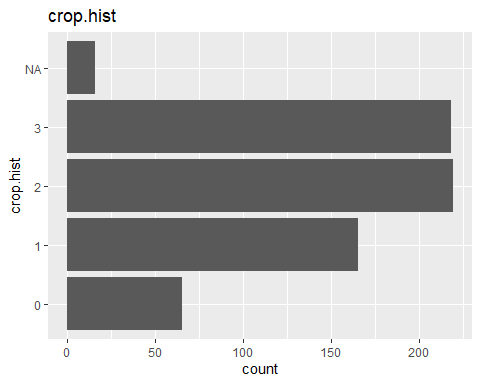
##   
## [[5]]



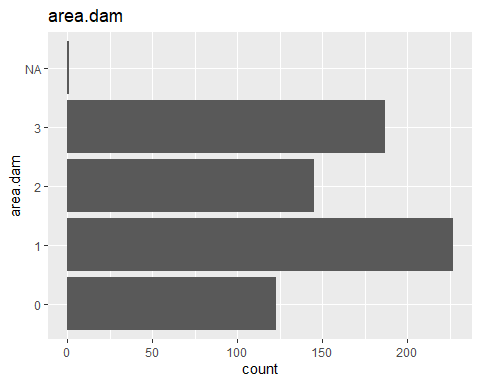
##   
## [[6]]



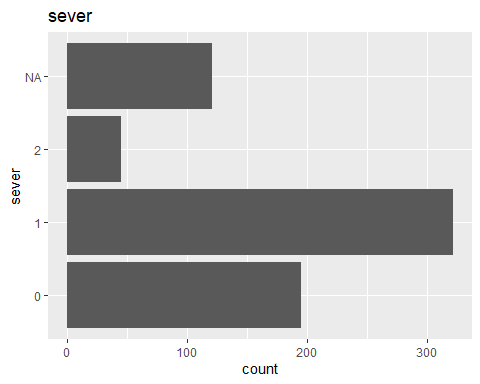
##   
## [[7]]



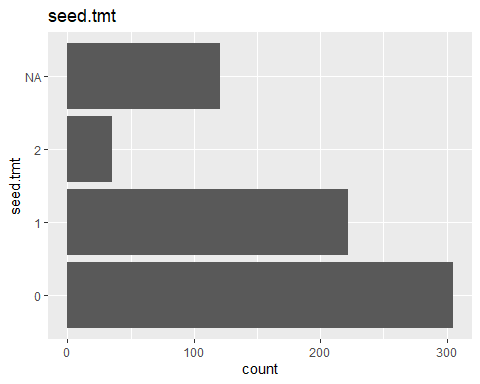
##   
## [[8]]



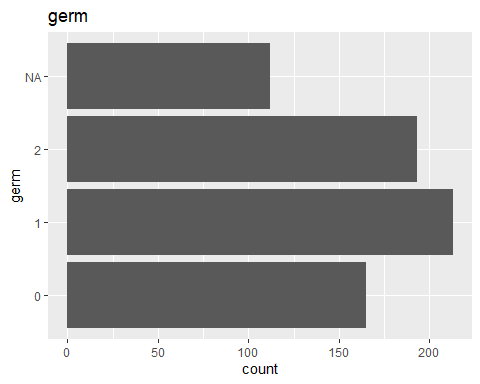
##   
## [[9]]



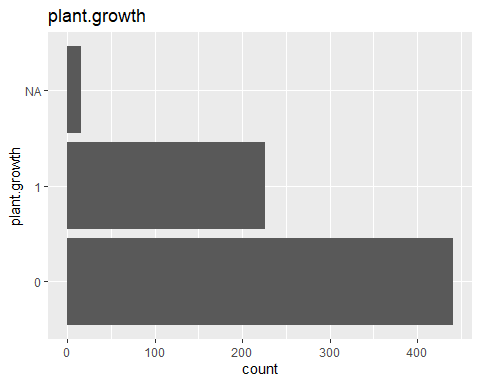
##   
## [[10]]



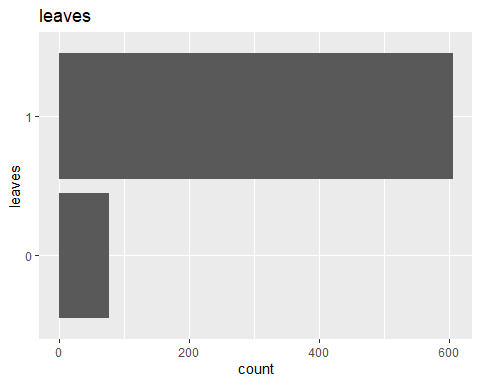
##   
## [[11]]



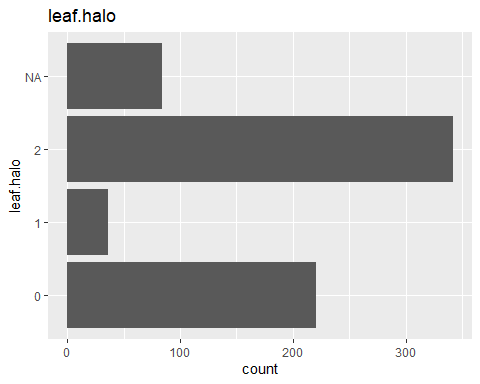
##   
## [[12]]



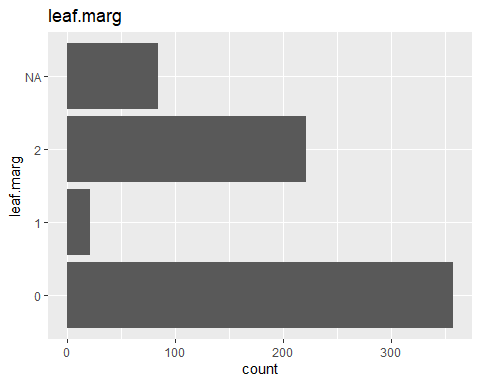
##   
## [[13]]



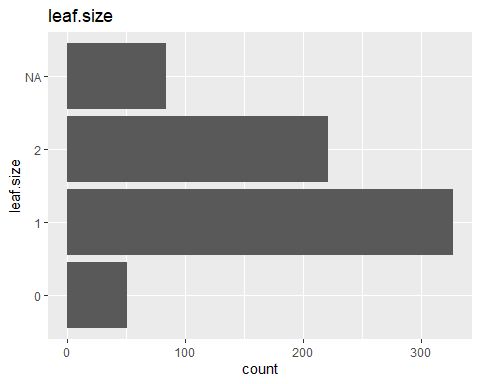
##   
## [[14]]



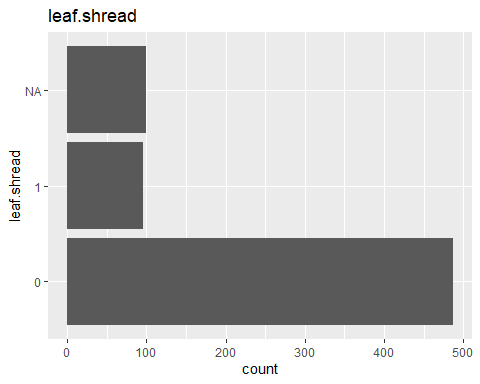
##   
## [[15]]



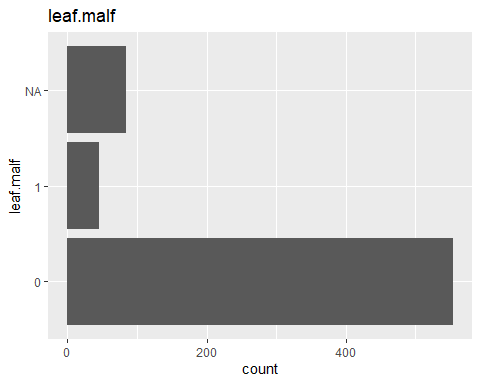
##   
## [[16]]



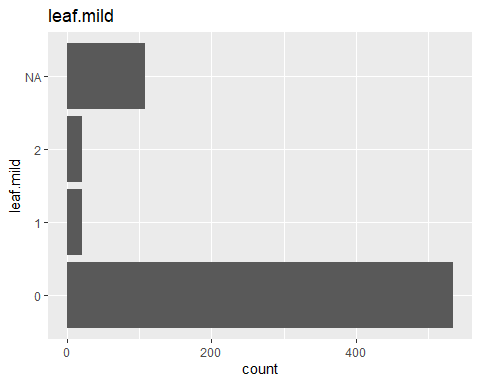
##   
## [[17]]



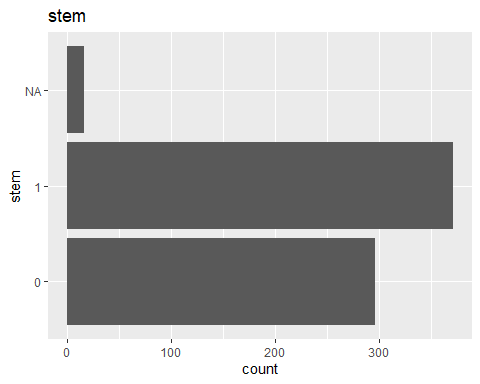
##   
## [[18]]



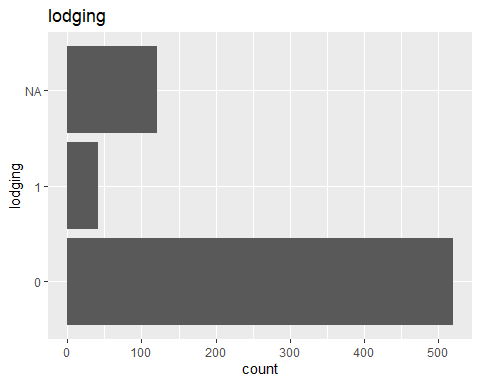
##   
## [[19]]



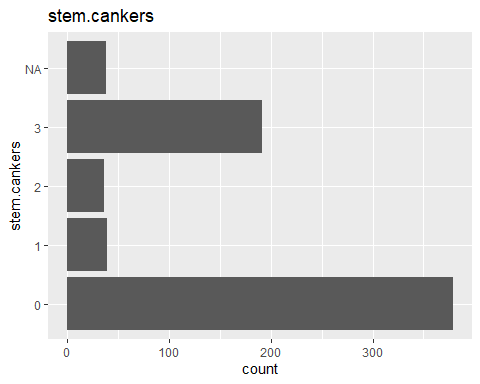
##   
## [[20]]



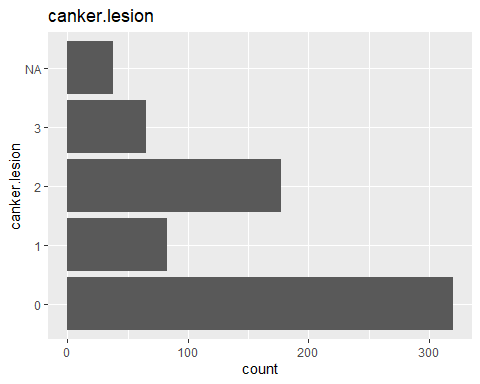
##   
## [[21]]



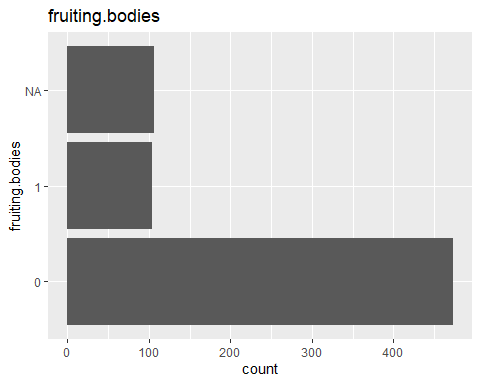
##   
## [[22]]



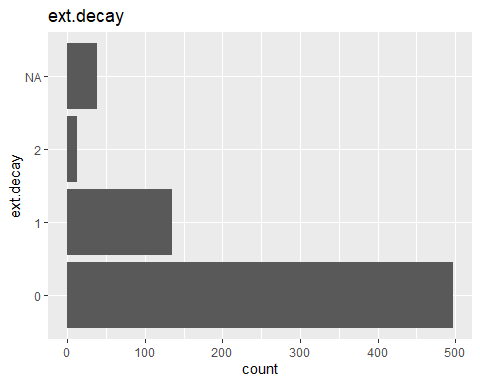
##   
## [[23]]



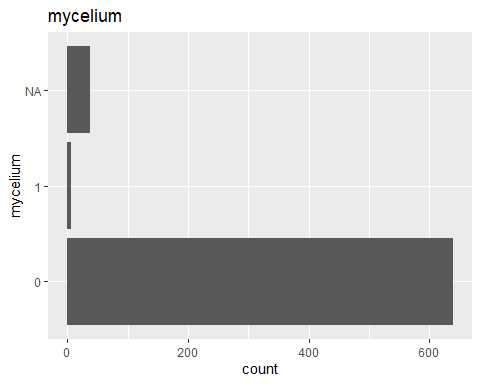
##   
## [[24]]



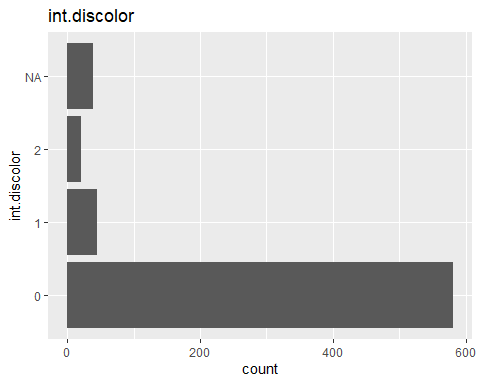
##   
## [[25]]



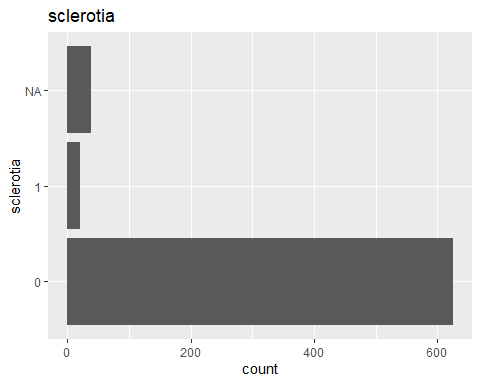
##   
## [[26]]



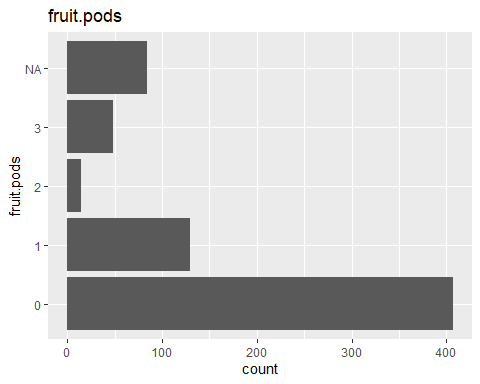
##   
## [[27]]



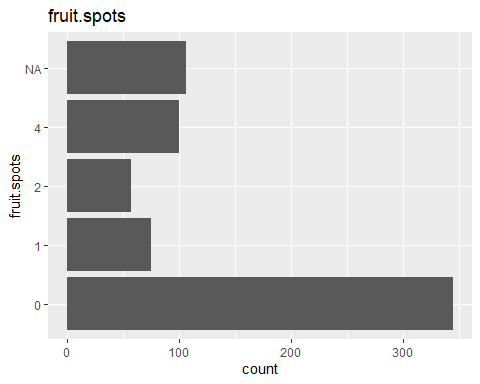
##   
## [[28]]



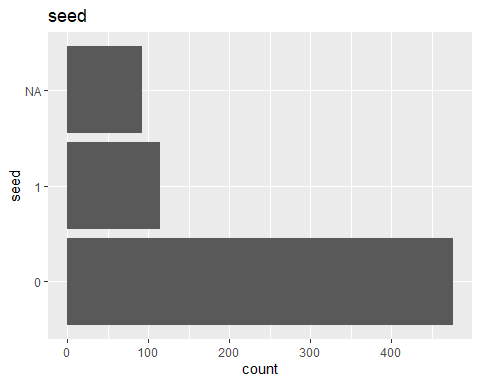
##   
## [[29]]



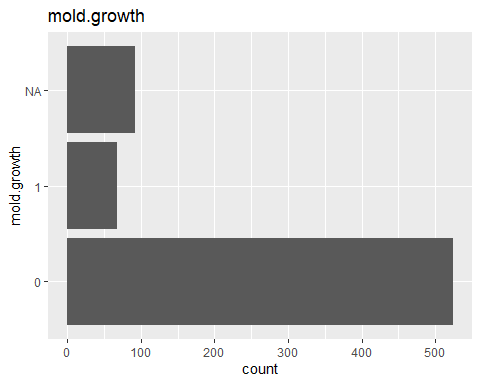
##   
## [[30]]



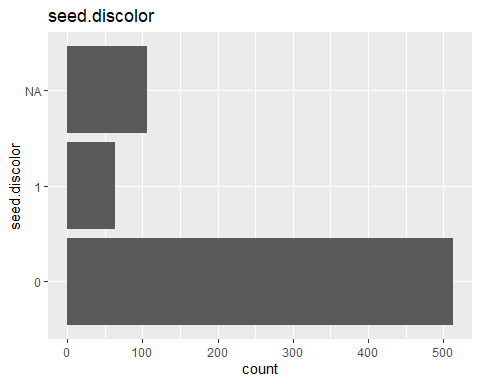
##   
## [[31]]



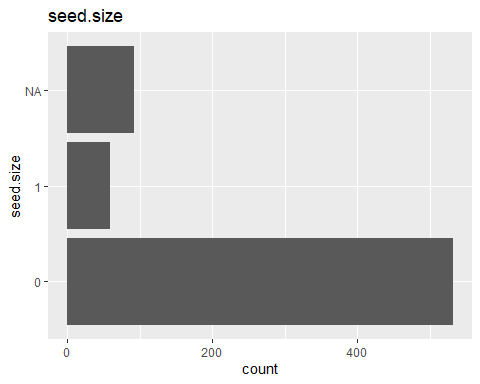
##   
## [[32]]



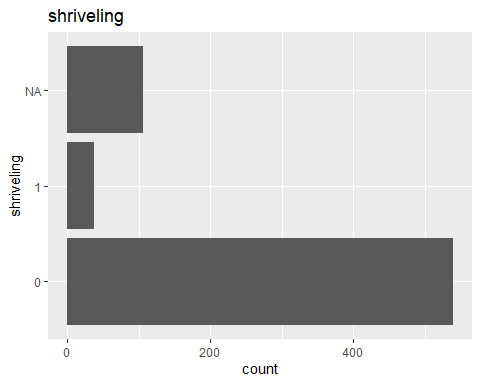
##   
## [[33]]



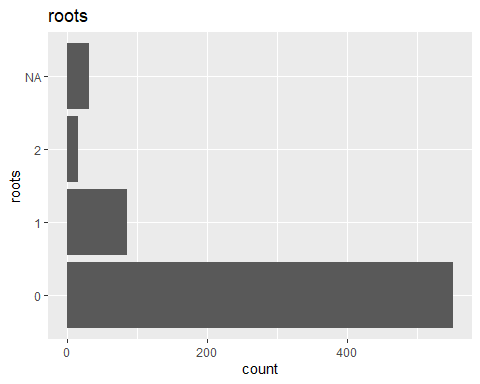
##   
## [[34]]



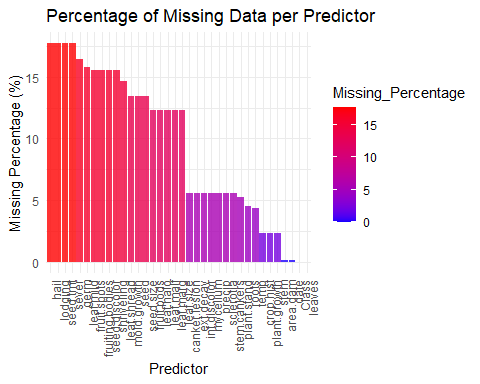
##   
## [[35]]



##   
## [[36]]



# Plot missing data percentages  
ggplot(missing\_df, aes(x = reorder(Variable, -Missing\_Percentage), y = Missing\_Percentage, fill = Missing\_Percentage)) +  
 geom\_bar(stat = "identity", alpha = 0.8) +  
 theme\_minimal() +  
 labs(title = "Percentage of Missing Data per Predictor",  
 x = "Predictor",   
 y = "Missing Percentage (%)") +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +  
 scale\_fill\_gradient(low = "blue", high = "red")



1. Roughly 18% of the data are missing. Are there particular predictors that are more likely to be missing? Is the pattern of missing data related to the classes?

The expected outcome of the bar plot is a clear visualization of the **percentage of missing values** for each variable in the **Soybean dataset**. The variables with the highest proportion of missing data will be displayed first, allowing for easy identification of the most affected predictors. Additionally, the bars will be **color-coded from blue to red**, helping to visually emphasize the severity of missingness. This plot provides a quick and effective way to assess data quality and determine whether imputation or variable removal may be necessary. If needed, threshold indicators can be added to highlight variables with excessive missing values that may require special handling.

1. Develop a strategy for handling missing data, either by eliminating predictors or imputation.

To manage missing data, different approaches can be considered based on the extent and pattern of missingness. One option is removing predictors with excessive missing values (e.g., more than 30%) or those that are degenerate. Alternatively, imputation methods such as mode imputation (for categorical predictors), k-NN imputation (estimating missing values based on similar observations), or Multiple Imputation (MICE) can be applied for more complex missing patterns. Finally, after handling missing data, it is important to evaluate the impact on the classification model to ensure that imputation or removal does not introduce bias or degrade performance.