DATA 624: PREDICTIVE ANALYTICS HW4

Gabriel Campos

Last edited February 25, 2024

```r
library(mlbench)
library(dplyr)
library(ggplot2)
library(tsibble)
library(tidyr)
library(corrplot)
library(cowplot)
library(psych)
library(MASS)
library(gridExtra)
library(tidyr)
```

Instructions

Do problems 3.1 and 3.2 in the Kuhn and Johnson book Applied Predictive Modeling. Please submit your Rpubs link along with your .pdf for your run code.

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. The data can be accessed via:

```r
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 ...
```
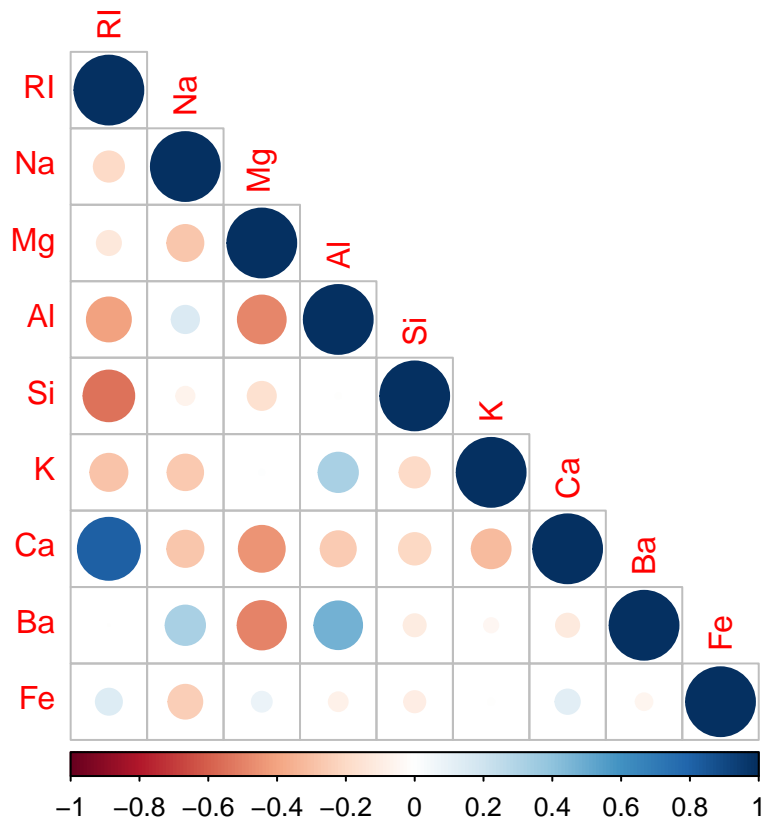
(a)

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

```
Glass %>%
  dplyr::select(-10)%>%
  gather() %>%
  ggplot(aes(x=value))+
  geom_histogram(fill="lightgreen")+
  facet_wrap(~key,scales = "free")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```
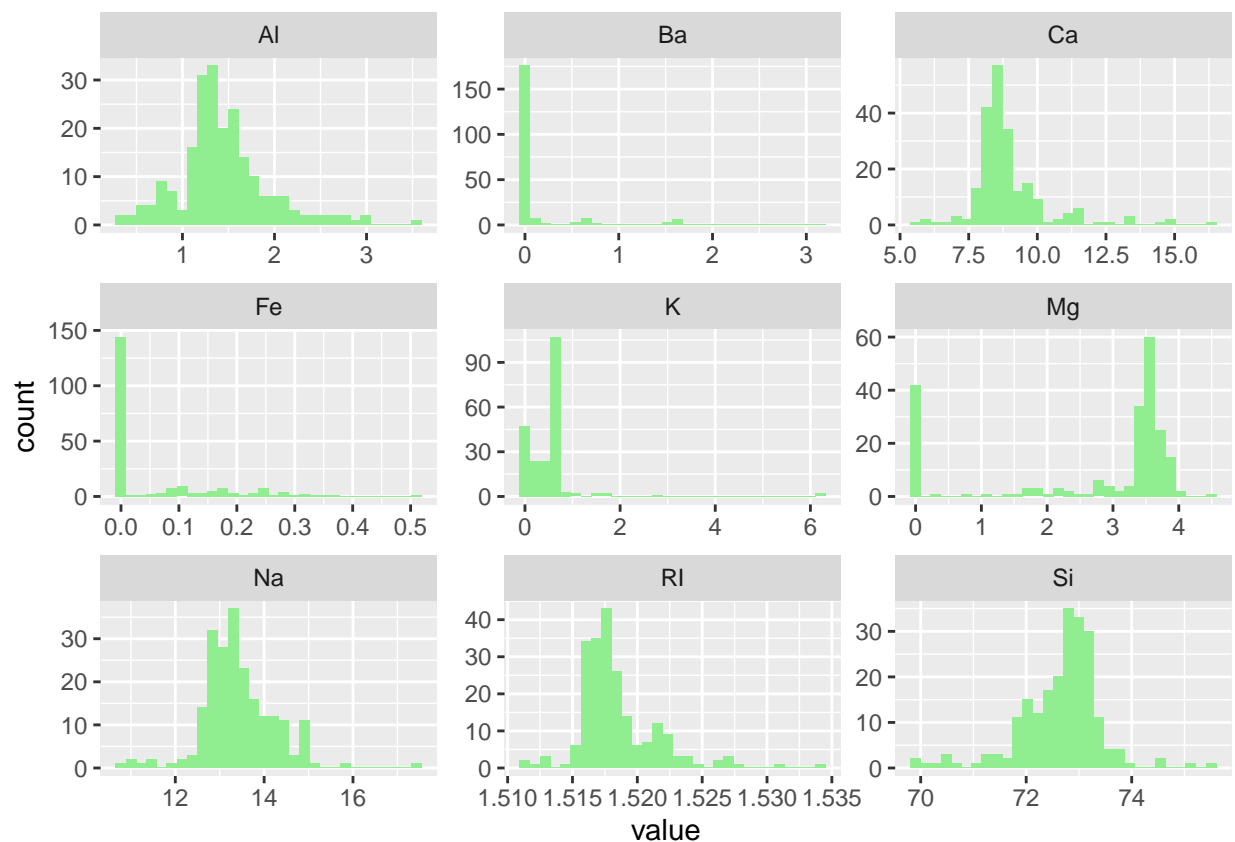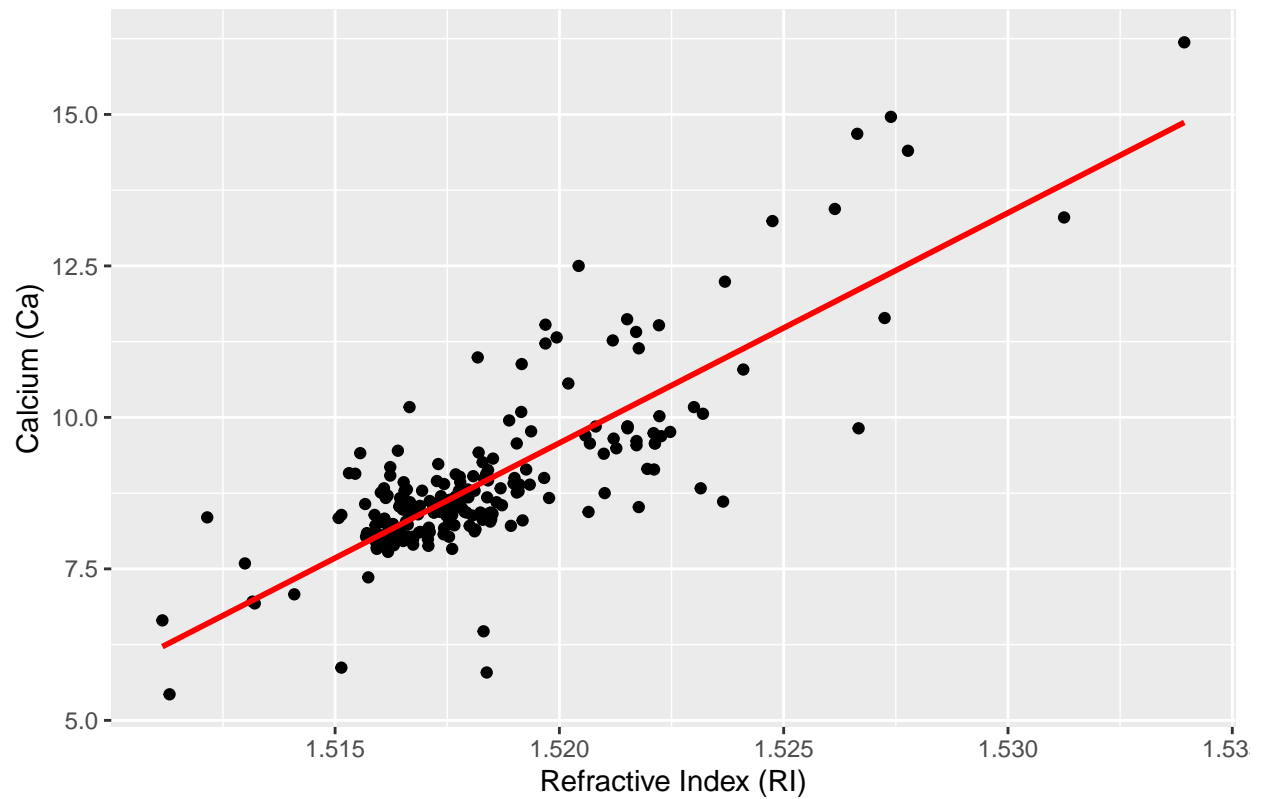


```
corrplot(cor(Glass%>%dplyr::select(-10)),type="lower")
```
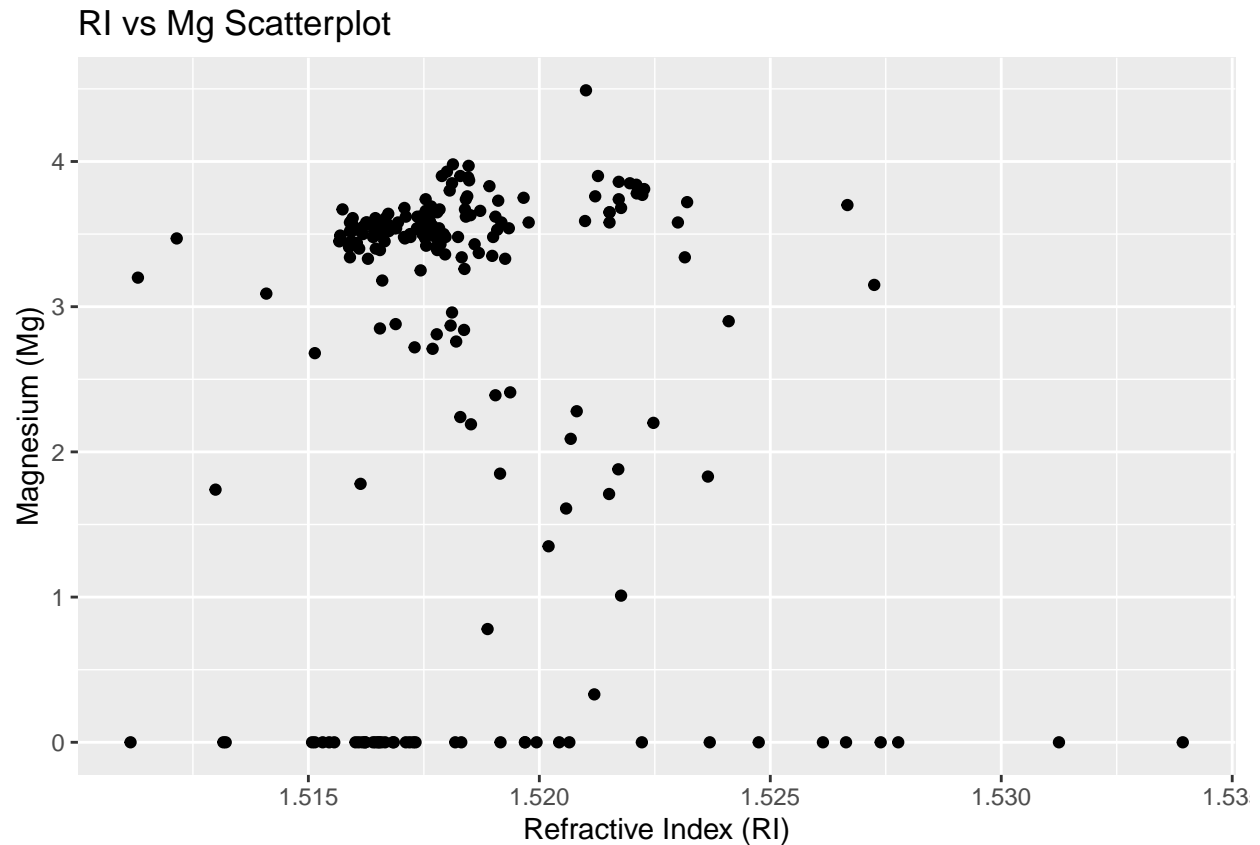
```
Glass%>%
  dplyr::select("RI","Ca")%>%
  ggplot(aes(x=RI,y=Ca))+
  geom_point()+
  geom_smooth(method="lm", se = FALSE,color="red")+
  labs(x = "Refractive Index (RI)", y = "Calcium (Ca)",
       title = "RI vs Ca Scatterplot" )
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

RI vs Ca Scatterplot


```
Glass%>%
  dplyr::select("RI","Mg")%>%
  ggplot(aes(x=RI,y=Mg))+
  geom_point()+
  labs(x = "Refractive Index (RI)", y = "Magnesium (Mg)",
       title = "RI vs Mg Scatterplot" )
```

RI vs Mg Scatterplot

In order to make the plots `Type` was excluded, as the values were not numeric and did not provide any real insight if it were to be plotted.

Noting the plots I can assess the following:

- All plots with exception of `Mg` and `Si` appear to be right skewed to some degree
- Of the right skewed properties `Ba` and `Fe` is centered around 0
- `Ai` and `Ba` is the most centered
- `RI` and `Ca` is shown to have the highest positive correlation while `RI` and `Mg` has the lowest.
- none of the degrees of relationship seems prominent

(b)

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

```
Glass %>%
  dplyr::select(-10)%>%
  gather() %>%
  ggplot(aes(value))+
  geom_boxplot()+
  facet_wrap(~key,scales = "free")
```
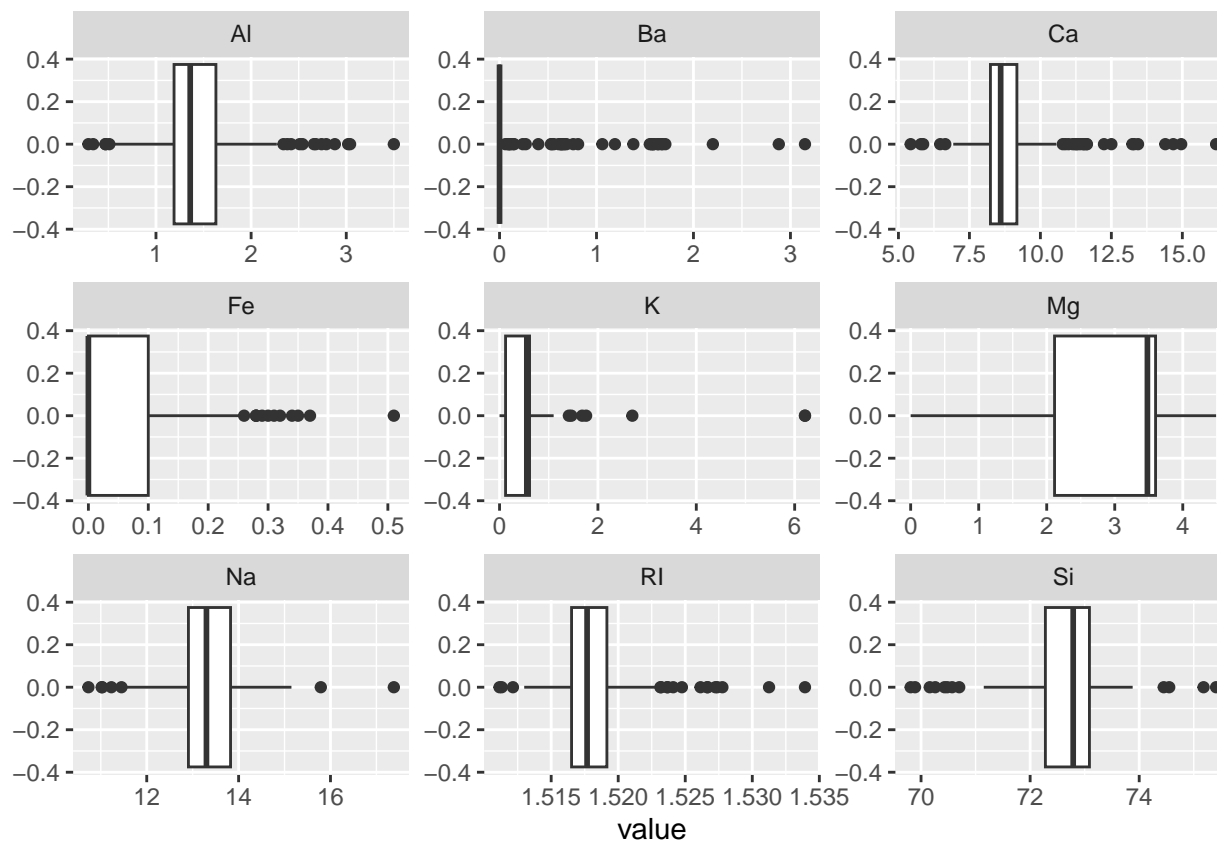
All of the attribute except for `Mg` have outliers in the data. Skewness is shown below. As noted all have a degree of skewness, right and left skewness is described in (a) with `Ba` having the most notable degree.

```r
data.frame(describe(Glass))%>%
                      dplyr::select(skew)
```

```
##               skew
## RI       1.6027151
## Na       0.4478343
## Mg      -1.1364523
## Al       0.8946104
## Si      -0.7202392
## K        6.4600889
## Ca       2.0184463
## Ba       3.3686800
## Fe       1.7298107
## Type*    1.0377535
```

(c)

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

```r
df_glass<-Glass
df_glass$log_Ba<-log(Glass$Ba)
df_glass$log_Fe<-log(Glass$Fe)
```

```
df_glass$log_K<-log(Glass$K)

plot_Ba <- ggplot(df_glass, aes(x = log_Ba)) +
  geom_histogram(bins = 20, fill = "darkblue",
                 color = "black") +
  labs(title = "Histogram of log-transformed Ba",
       x = "log(Be)", y = "Frequency")

plot_Fe<-ggplot(df_glass, aes(x = log_Fe)) +
  geom_histogram(bins = 20, fill = "blue",
                 color = "black") +
  labs(title = "Histogram of log-transformed Fe",
       x = "log(Be)", y = "Frequency")

plot_K<-ggplot(df_glass, aes(x = log_K)) +
  geom_histogram(bins = 20, fill = "lightblue",
                 color = "black") +
  labs(title = "Histogram of log-transformed K",
       x = "log(Be)", y = "Frequency")

plot_grid(plot_Ba,plot_Fe,plot_K, ncol = 3)
```
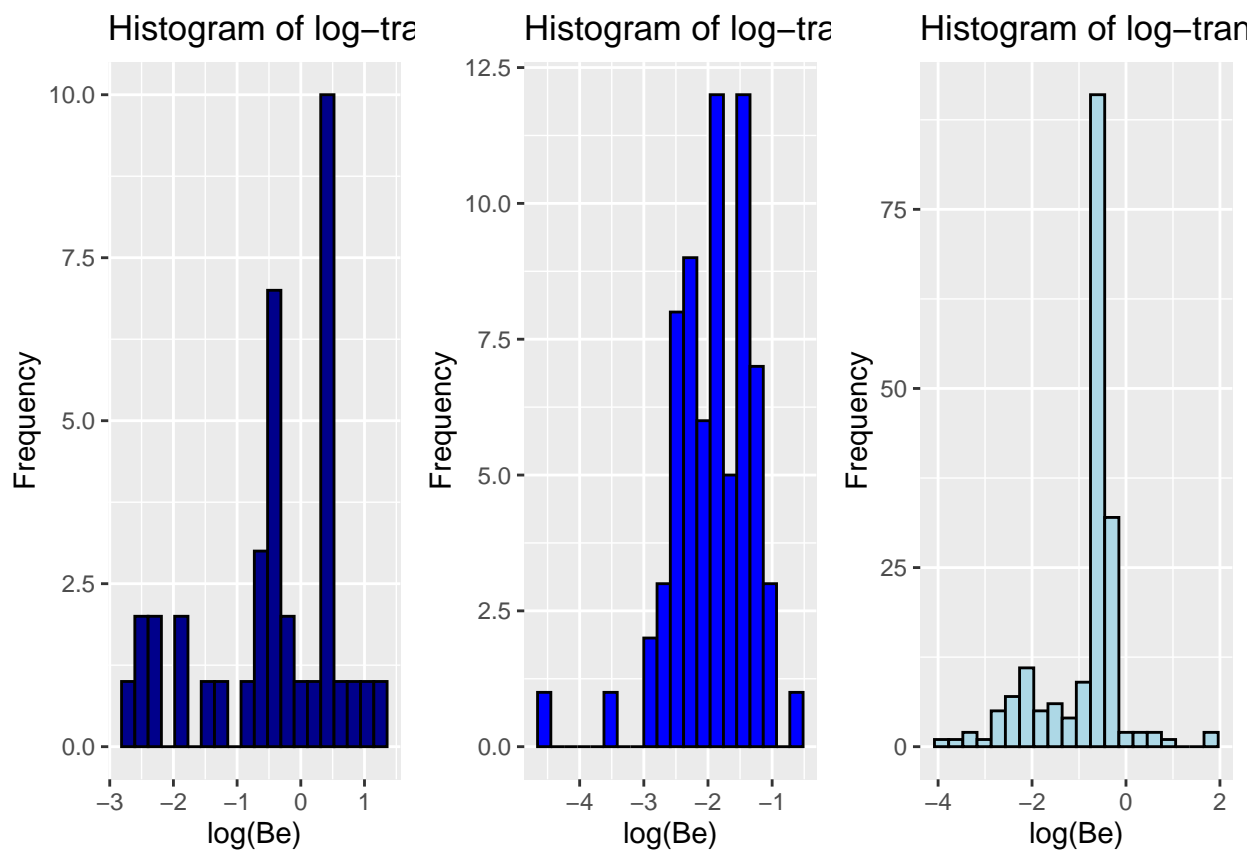


```
# Box-Cox transformation for specified columns
df_glass_transformed <- df_glass
```

```r
# Columns for Box-Cox transformation
columns <- c("RI", "Na", "Al", "Si", "Ca")

for (col in columns) {
  transformed_col <- boxcox(df_glass[[col]] ~ 1, plotit=FALSE)
  lambda <- transformed_col$x[which.max(transformed_col$y)]
  if (lambda == 0) {
    df_glass_transformed[[paste0("boxcox_", col)]] <- log(df_glass[[col]])
  } else {
    df_glass_transformed[[paste0("boxcox_", col)]] <- (df_glass[[col]]^lambda - 1) / lambda
  }
}

# Replace null values with 0
df_glass_transformed[is.na(df_glass_transformed)] <- 0

# Create ggplot visualizations for each transformed column
plots <- list()
for (col in paste0("boxcox_", columns)) {
  plots[[col]] <- ggplot(df_glass_transformed, aes(x = !!sym(col))) +
    geom_histogram(bins = 20, fill = "orange", color = "black") +
    labs(title = paste("Histogram of", col), x = col, y = "Frequency")
}

# Arrange plots in columns of 3
grid.arrange(grobs = plots, ncol = 3)
```
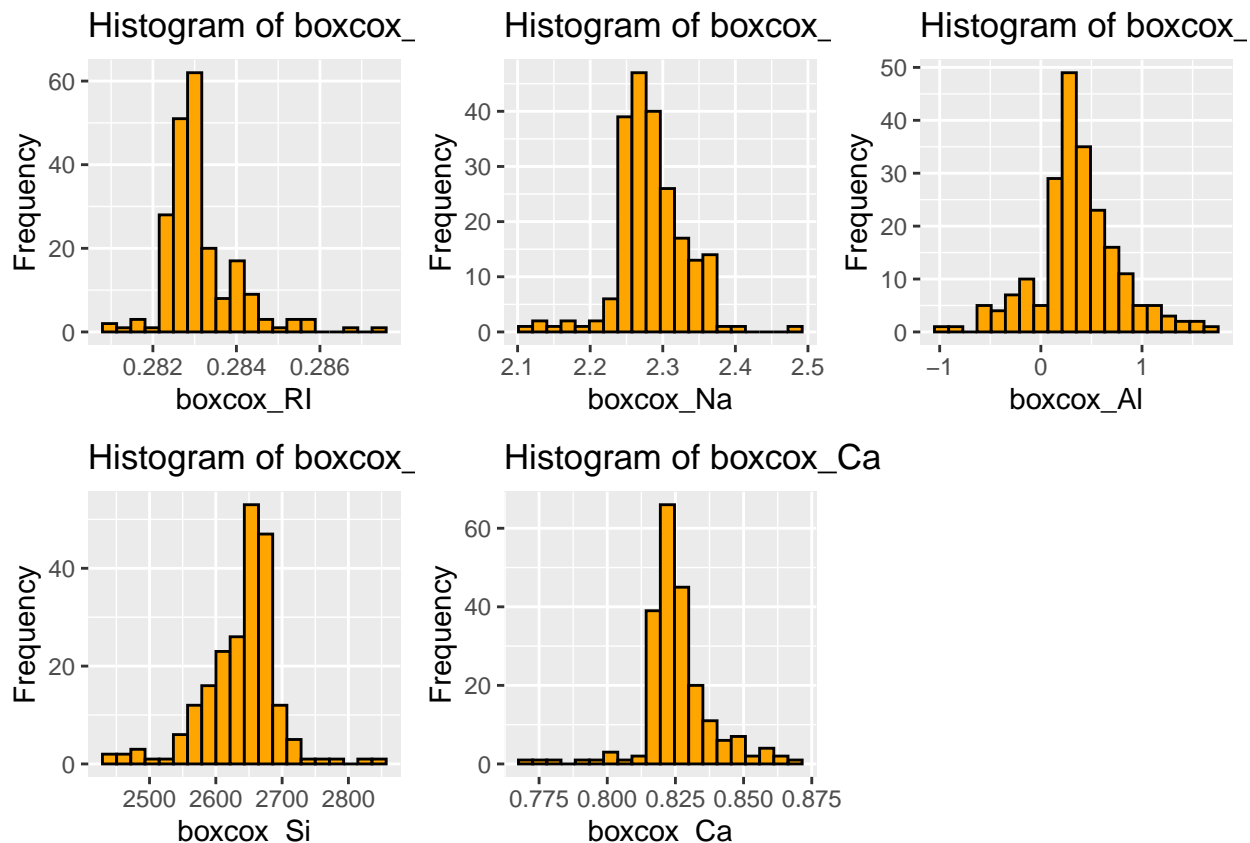
Histogram of boxcox_ (boxcox_RI), Histogram of boxcox_ (boxcox_Na), Histogram of boxcox_ (boxcox_Al), Histogram of boxcox_ (boxcox_Si), Histogram of boxcox_Ca

```r
rm(list = ls(pattern = "(lambda|plot|glass|^col|col$)"))
```

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 environmen tal conditions (e.g., temperature, precipitation) and plant conditions (e.g., left spots, mold growth). The outcome labels consist of 19 distinct classes.

The data can be loaded via:

```r
data(Soybean)
## See ?Soybean for details
```

(a)

Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?
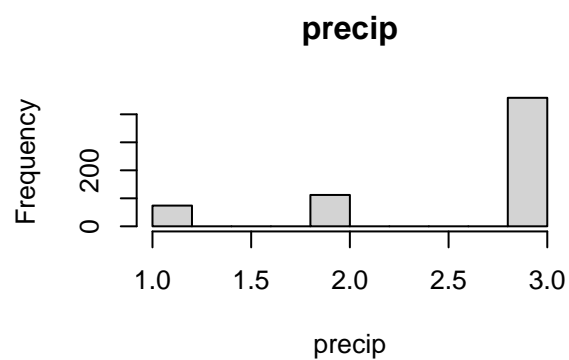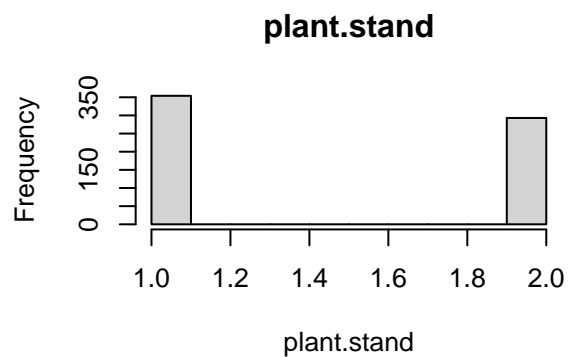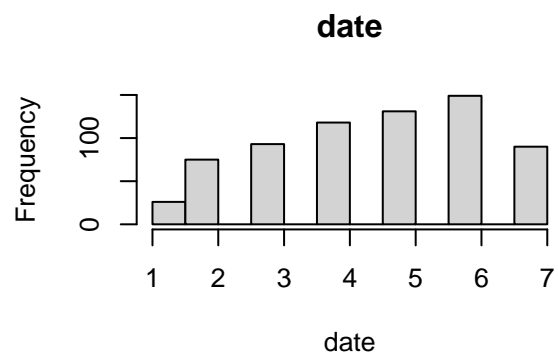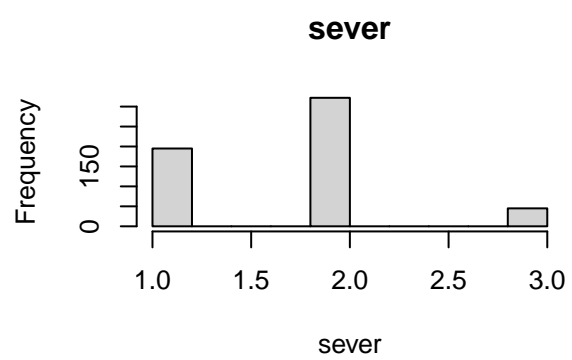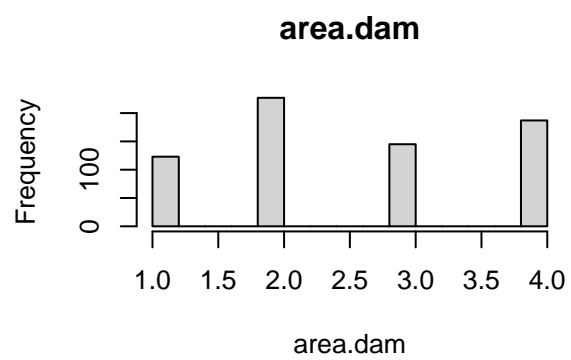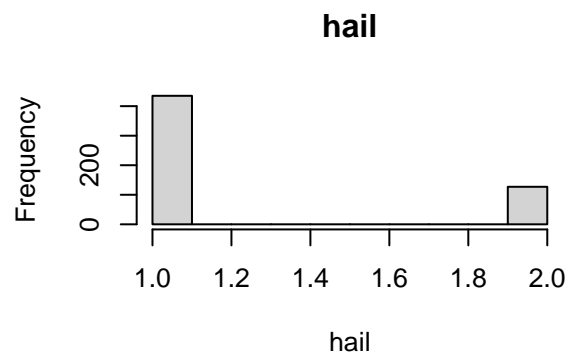
```r
str(Soybean)
```

```
## 'data.frame':    683 obs. of  36 variables:
##  $ Class          : Factor w/ 19 levels "2-4-d-injury",..: 11 11 11 11 11 11 11 11 11 11 ...
```
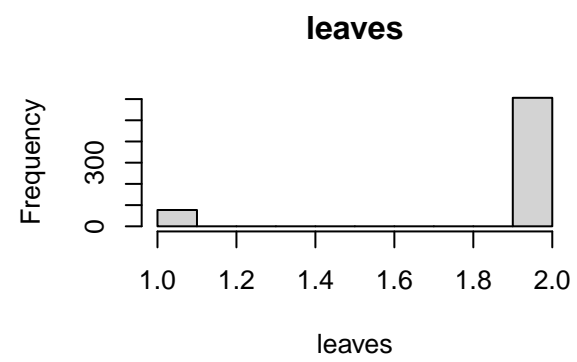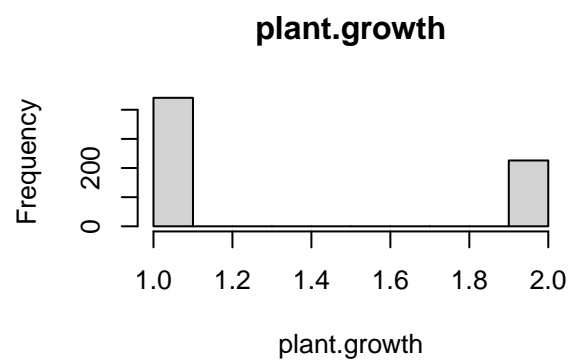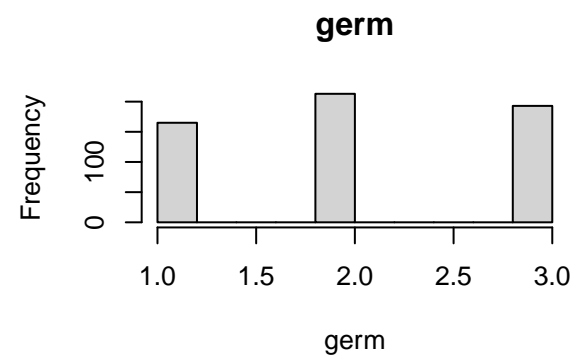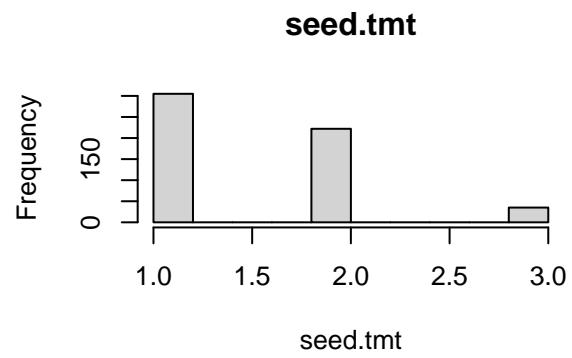
9

```
##  $ date          : Factor w/ 7 levels "0","1","2","3",..: 7 5 4 4 7 6 6 5 7 5 ...
##  $ plant.stand   : Ord.factor w/ 2 levels "0"<"1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ precip        : Ord.factor w/ 3 levels "0"<"1"<"2": 3 3 3 3 3 3 3 3 3 3 ...
##  $ temp          : Ord.factor w/ 3 levels "0"<"1"<"2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ hail          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
##  $ crop.hist     : Factor w/ 4 levels "0","1","2","3": 2 3 2 2 3 4 3 2 4 3 ...
##  $ area.dam      : Factor w/ 4 levels "0","1","2","3": 2 1 1 1 1 1 1 1 1 1 ...
##  $ sever         : Factor w/ 3 levels "0","1","2": 2 3 3 3 2 2 2 2 2 3 ...
##  $ seed.tmt      : Factor w/ 3 levels "0","1","2": 1 2 2 1 1 1 2 1 2 1 ...
##  $ germ          : Ord.factor w/ 3 levels "0"<"1"<"2": 1 2 3 2 3 2 1 3 2 3 ...
##  $ plant.growth  : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ leaves        : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ leaf.halo     : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ leaf.marg     : Factor w/ 3 levels "0","1","2": 3 3 3 3 3 3 3 3 3 3 ...
##  $ leaf.size     : Ord.factor w/ 3 levels "0"<"1"<"2": 3 3 3 3 3 3 3 3 3 3 ...
##  $ leaf.shread   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ leaf.malf     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ leaf.mild     : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ stem          : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ lodging       : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 2 1 1 1 ...
##  $ stem.cankers  : Factor w/ 4 levels "0","1","2","3": 4 4 4 4 4 4 4 4 4 4 ...
##  $ canker.lesion : Factor w/ 4 levels "0","1","2","3": 2 2 1 1 2 1 2 2 2 2 ...
##  $ fruiting.bodies: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ext.decay     : Factor w/ 3 levels "0","1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ mycelium      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ int.discolor  : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sclerotia     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ fruit.pods    : Factor w/ 4 levels "0","1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ fruit.spots   : Factor w/ 4 levels "0","1","2","4": 4 4 4 4 4 4 4 4 4 4 ...
##  $ seed          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ mold.growth   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ seed.discolor : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ seed.size     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ shriveling    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ roots         : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
```
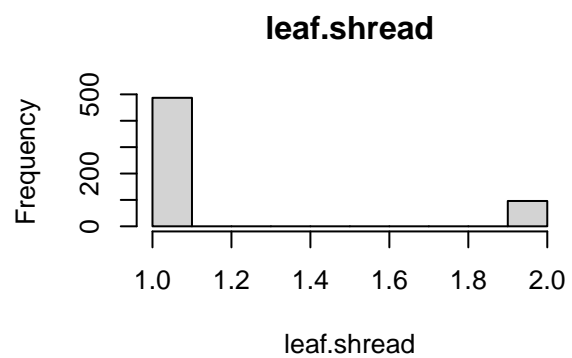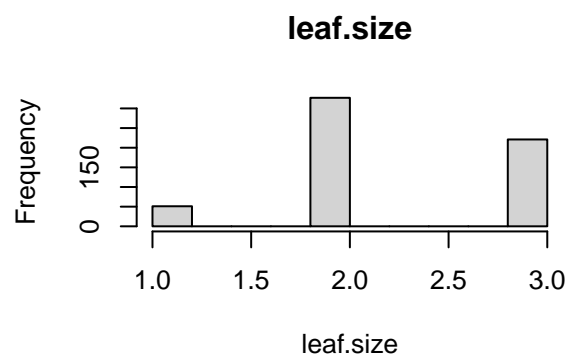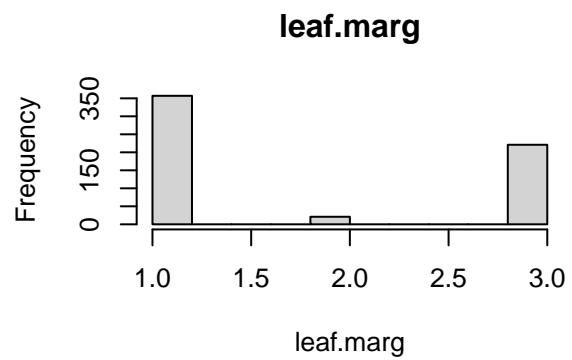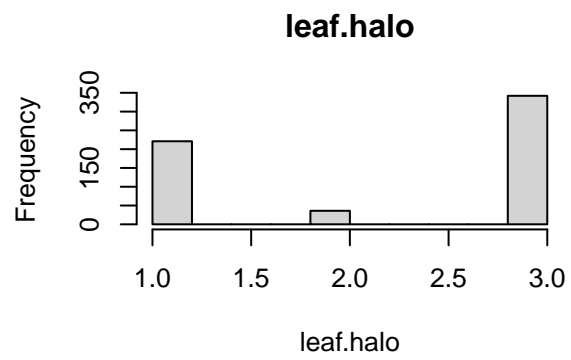
```r
df_soybean <- Soybean#%>%dplyr::select(-1)

par(mfrow=c(2,2))
for (col in 2:ncol(Soybean)) {
    hist( as.numeric(Soybean[,col]),main =   colnames(Soybean)[col], xlab = colnames(Soybean)[col])
}
```
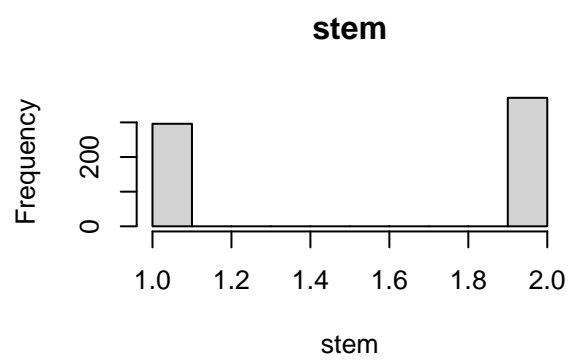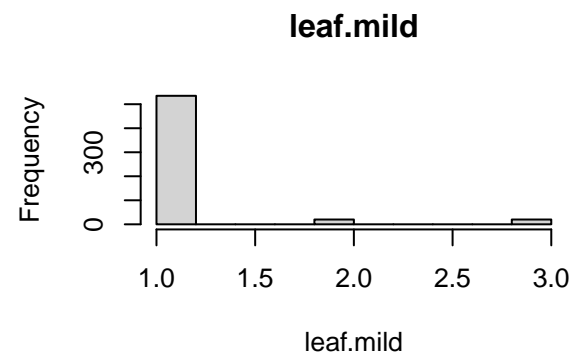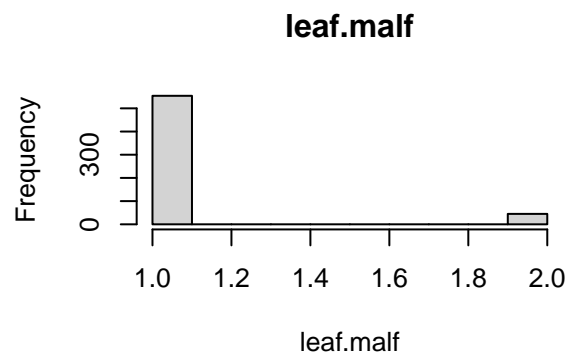
date

plant.stand

precip

temp

hail

crop.hist

area.dam

sever

seed.tmt

germ

plant.growth

leaves

leaf.halo

leaf.marg

leaf.size

leaf.shread

leaf.malf

leaf.mild

stem

lodging

stem.cankers

stem.cankers

canker.lesion

canker.lesion

fruiting.bodies

fruiting.bodies

ext.decay

ext.decay

mycelium

int.discolor

sclerotia

fruit.pods

fruit.spots

seed

mold.growth

seed.discolor

seed.size

shriveling

roots


```r
# Function to count distinct numeric values, including NAs, in a column
count_distinct_numeric <- function(column) {
  n_distinct(na.omit(column))
}

# Apply the function to each column of the data frame 'B'
distinct_counts <- sapply(Soybean, count_distinct_numeric)

# Reorder distinct counts in ascending order
distinct_counts <- distinct_counts[order(distinct_counts)]

# Remove columns with NA counts from the print output
distinct_counts <- distinct_counts[!is.na(distinct_counts)]

# Print the distinct counts for each column
print(distinct_counts)
```

```
##      plant.stand            hail      plant.growth           leaves      leaf.shread
##                2               2                 2                2                2
##        leaf.malf            stem           lodging   fruiting.bodies          mycelium
##                2               2                 2                2                2
##         sclerotia            seed       mold.growth      seed.discolor         seed.size
##                2               2                 2                2                2
##        shriveling          precip              temp            sever          seed.tmt
##                2               3                 3                3                3
```
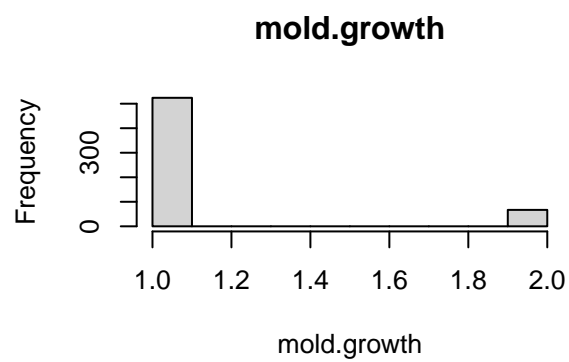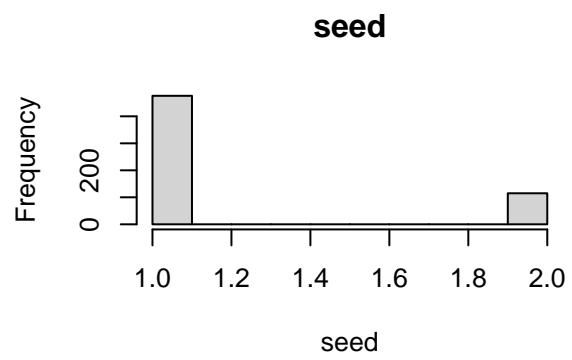
```
##          germ        leaf.halo       leaf.marg       leaf.size      leaf.mild
##             3               3               3               3              3
##     ext.decay    int.discolor           roots       crop.hist       area.dam
##             3               3               3               4              4
##  stem.cankers    canker.lesion      fruit.pods     fruit.spots           date
##             4               4               4               4              7
##         Class
##            19
```

The distribution of `mycelium` and `sclerotia` appears to be degenerate, considering the low frequency count and minimal distinct values. They are significantly less then the other attributes to where visually `mycelium` almost appeared to have just 1 value.

(b)

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?

```
df_soybean%>%
  summarise_all(list(~is.na(.))) %>%
  pivot_longer(everything(), names_to = "variables", values_to = "missing") %>%
  count(variables, missing) %>%
  ggplot(aes(y = reorder(variables, n), x = n, fill = missing)) +
  geom_col(position = "fill") +
  geom_text(aes(label = ifelse(missing, "NA", "Non-NA")),
            position = position_fill(vjust = 0.5),
            color = "white", size = 4) +  # Add data labels
  labs(title = "NA Proportion",
       x = "Proportion") +
  scale_fill_manual(values = c("grey", "darkgreen")) +
  theme_minimal()
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## i The deprecated feature was likely used in the dplyr package.
##   Please report the issue at <https://github.com/tidyverse/dplyr/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

NA Proportion
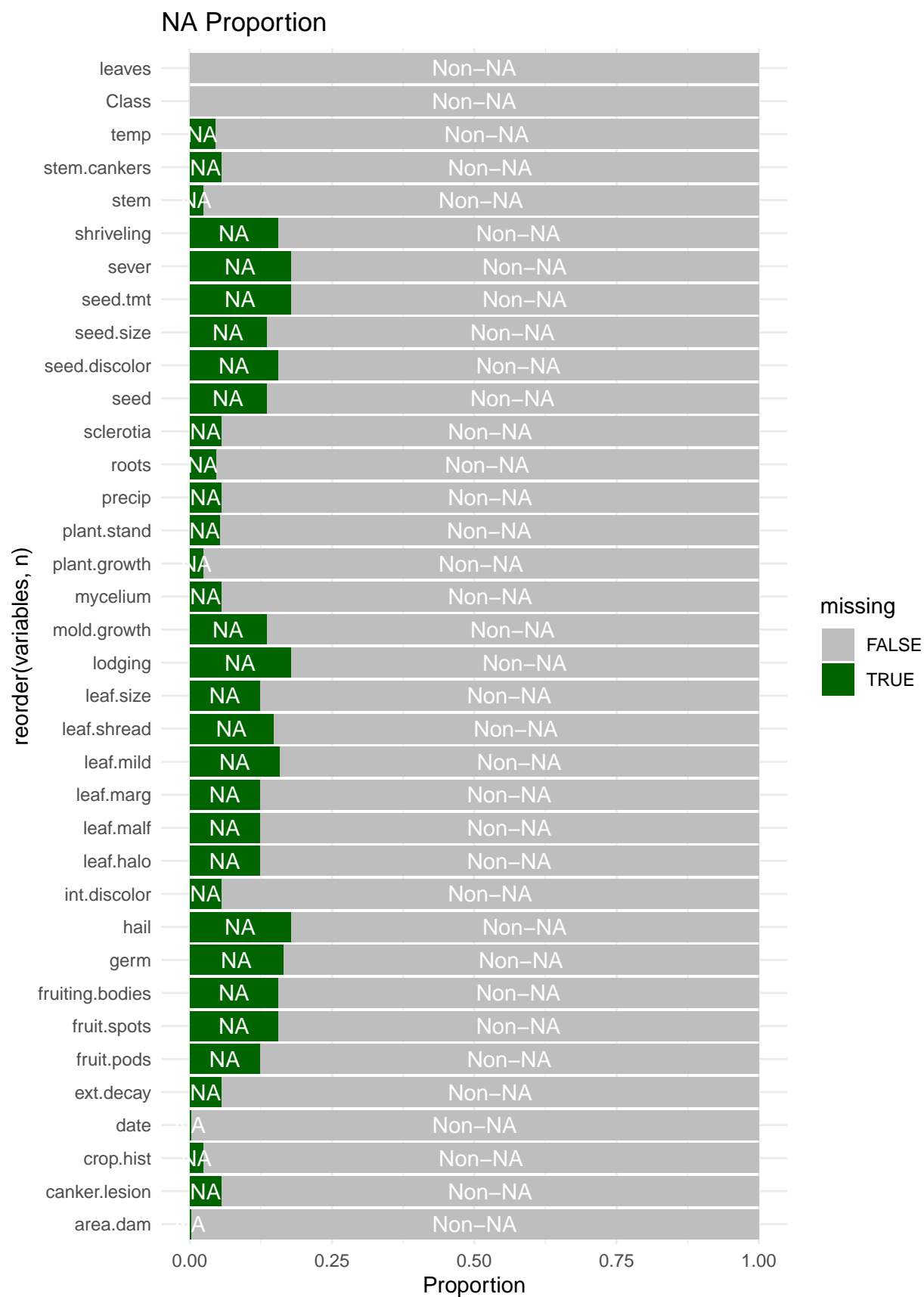
```
df_incomplete <- df_soybean[!complete.cases(df_soybean),]
(df_incomplete %>%
                group_by(Class) %>%
                  tally())
```

```
## # A tibble: 5 x 2
##   Class                         n
##   <fct>                     <int>
## 1 2-4-d-injury                 16
## 2 cyst-nematode                14
## 3 diaporthe-pod-&-stem-blight  15
## 4 herbicide-injury              8
## 5 phytophthora-rot             68
```

```
df_soybean%>%
  filter(!Class %in% c("2-4-d-injury","cyst-nematode",
                  "diaporthe-pod-&-stem-blight",
                  "herbicide-injury","phytophthora-rot"))%>%
  summarise_all(list(~is.na(.))) %>%
  pivot_longer(everything(),
              names_to = "variables",
              values_to = "missing") %>%
  count(variables, missing) %>%
  ggplot(aes(y = reorder(variables, n),
            x = n, fill = missing)) +
  geom_col(position = "fill") +
  geom_text(aes(label = ifelse(missing,
                              "NA", "Non-NA")),
          position = position_fill(vjust = 0.5),
          color = "white", size = 4) +  # Add data labels
  labs(title = "NA Proportion",
      x = "Proportion") +
  scale_fill_manual(values = c("grey", "darkgreen")) +
  theme_minimal()
```
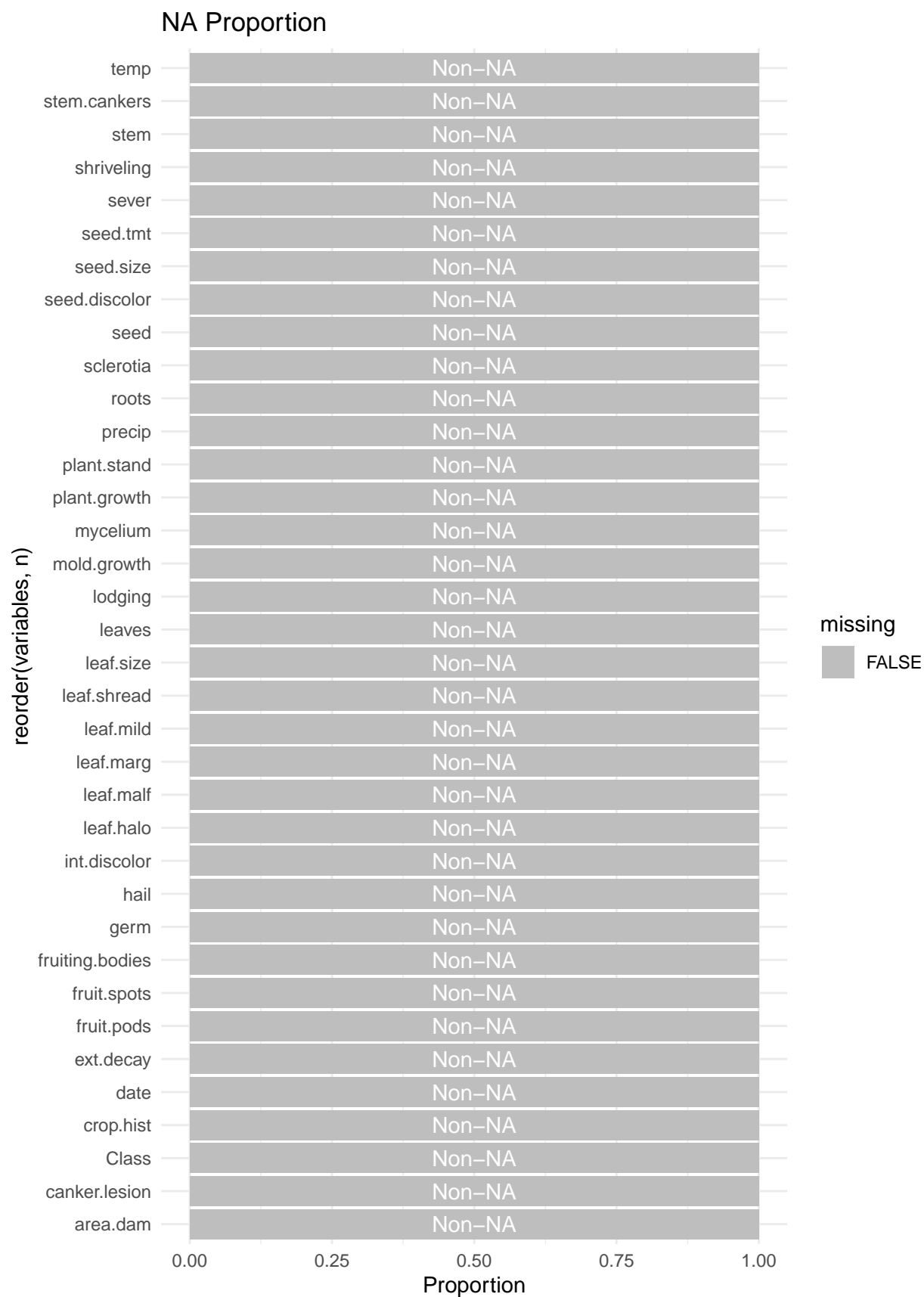
NA Proportion

(c)

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

I wouldnt just remove the classes although its seems much easier to just do that. I would say replace with zero where it makes sense, for binary values like `hail` or `lodging` since it is more logical to defualt to "no" unless reason to believe otherwise. I say this assuming `hail`=yes likelihood being significantly smaller if presuming. For values like `severe` I'd much rather not put a value, unless an "unknown" metric is put in, as it's frequency is likely to be very low. For the remaining, I would likely want to get the frequency and substitute the categorical metric with its mode for normalization with box-cox, and see if it better fits the model we want to predict with.