APM Chapter 3 HW: Data Pre-Processing

Alex Ptacek

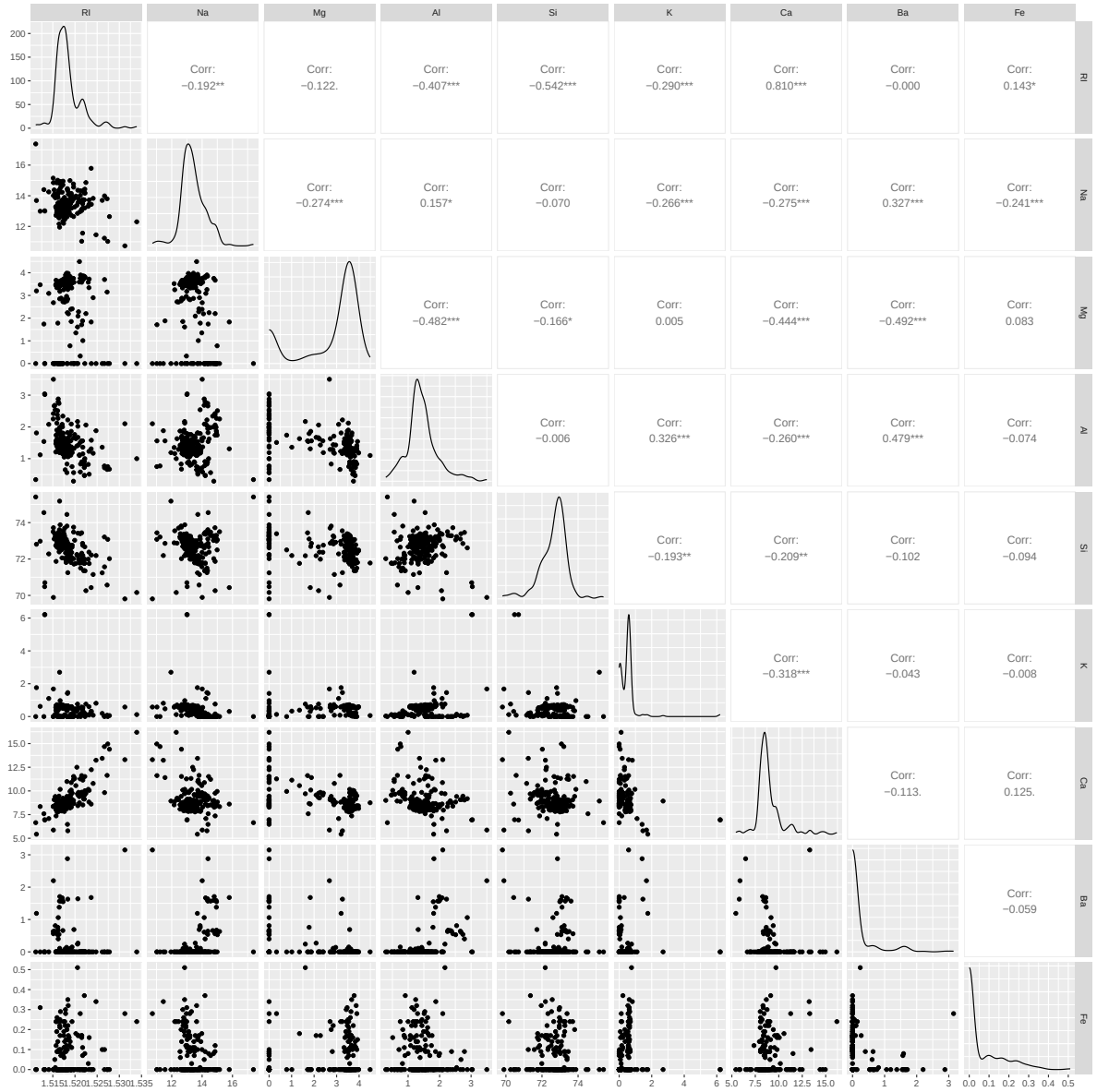
```
library(tidyverse)
library(AppliedPredictiveModeling)
library(caret)
library(corrplot)
library(e1071)
library(mlbench)
library(ggpubr)
library(GGally)
library(nnet)
```

Question 3.1

The UC Irvine Machine Learning Repository 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.

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

```
data(Glass)

Glass |>
  select(1:9) |>
  ggpairs()
```



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

Answer: Based on correlation and distribution matrix illustrated above, we can see that the following predictor matchups contain noticeable outliers: K against RI, Na, Al, Si, Ca, Ba, and Fe; Ba against RI, Na, K, and Fe. Clearly, K and Ba are the common denominator in all of these outlier observations, so they likely contain the outlier values. As further evidence, we can see that K and Ba are obviously right-skewed. Additionally, Fe is obviously right-skewed. Other candidates for skewness that are not as obvious are

RI, Mg, and Ca. Therefore, I decided to compute their skewness statistic below. Since none of the skewness statistics are close to 0, it is reasonable to assume that they are skewed.

```
skewed_predictors <- Glass |>
  select(RI, Mg, Ca)

apply(skewed_predictors, 2, skewness)
```

```
       RI        Mg        Ca
 1.602715 -1.136452  2.018446
```

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

Answer: Based on the analysis above, we can recommend several potential transformations. The 6 predictors with skewness can be transformed with box-cox transformations. This would likely increase accuracy and stability of a predictive model. Several variables also appears collinear (most notably, RI and Ca). Therefore, we could recommend Principle Component Analysis (PCA) as a means of feature extraction/data reduction. Lastly, to resolve the outliers in our predictors (specifically K and Ba), we could apply a spatial sign transformation, which would increase our interpretablilty of a model where the sample contains outliers. All transformations combined, however, would lead to a significant decrease in interpretability of our model, and may not be necessary if our selected model is insensitive to irregular data. Additionally, further domain knowledge/data understanding would be needed to assess the reliability and appropriateness of these transformations. For example, we could also resolve outliers by simply removing them, and PCA may be substituted with the Partial Least Sqaures (PLS) method or a manual predictor reduction based on a set correlation threshold - if the higher variance predictors are not accounting for most of the variance in our response variable. Nonetheless, let's assume these caveats do not apply and proceed with the transformations below. Based on our results in part iii. (below), the transformations didn't have much of an effect on the accuracy of our multinomial logistic regression accuracy. In fact, the residual deviation is a bit higher, so our new model with transformations is slightly worse. The AIC statistic is slightly better for our new model, but that's because it's a function of the number of predictors we use. Since we have less predictors, the AIC is lower (better), but the accuracy (based on residual deviation) is worse. There are many reasons that could have contributed to this decrease in accuracy, particularly the caveats mentioned earlier and the caveats mentioned in part ii. (below).

i. Let's start by fitting a simple multinomial logistic regression model and extracting our accuracy measures, Residual Deviance and Akaike Information Criterion (AIC). These values will be the benchmark to see if our transformations increase model performance.

```r
raw_model <- multinom(Type ~ RI + Na + Mg + Al + Si + K + Ca + Ba + Fe, Glass)
```

```
# weights:  66 (50 variable)
initial  value 383.436526
iter  10 value 257.359885
iter  20 value 181.634208
iter  30 value 161.554088
iter  40 value 157.912577
iter  50 value 154.889493
iter  60 value 153.706333
iter  70 value 153.334999
iter  80 value 152.219340
iter  90 value 149.994098
iter 100 value 149.743843
final  value 149.743843
stopped after 100 iterations
```

```r
raw_model$deviance
```

```
[1] 299.4877
```

```r
AIC(raw_model)
```

```
[1] 399.4877
```

ii. Here, we apply our specified transformations to the predictor variables. I would like to caveat that ideally we would apply appropriate transformations targeted to groups of predictors that were designated for transformation. However, this gave me a ton of trouble and I could not figure out how to do so. So, I simply applied the transformations I was interested in to all predictors.

```r
# Add 1 to predictor values to initialize for Box Cox transformation
predictors <- Glass |>
  select(1:9) |>
  mutate(across(everything(), \(x) x = x + 1))


# Apply transformations to predictors
trans_model <- preProcess(predictors,
                          method = c("BoxCox", "center", "scale", "spatialSign", "pca"))
```

```r
trans_predictors <- predict(trans_model, predictors)


# Select our Principle Components
trans_predictors <- trans_predictors |>
  select(10:15)
```

iii. Lastly, we bind our predictors back to the response variable `Type` and re-run our multinomial logistic regression with our principle components. Then we observe the changes in our accuracy statistics.

```r
adj_Glass <- cbind(trans_predictors, Glass |> select(Type))

adj_model <- multinom(Type ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6, adj_Glass)
```

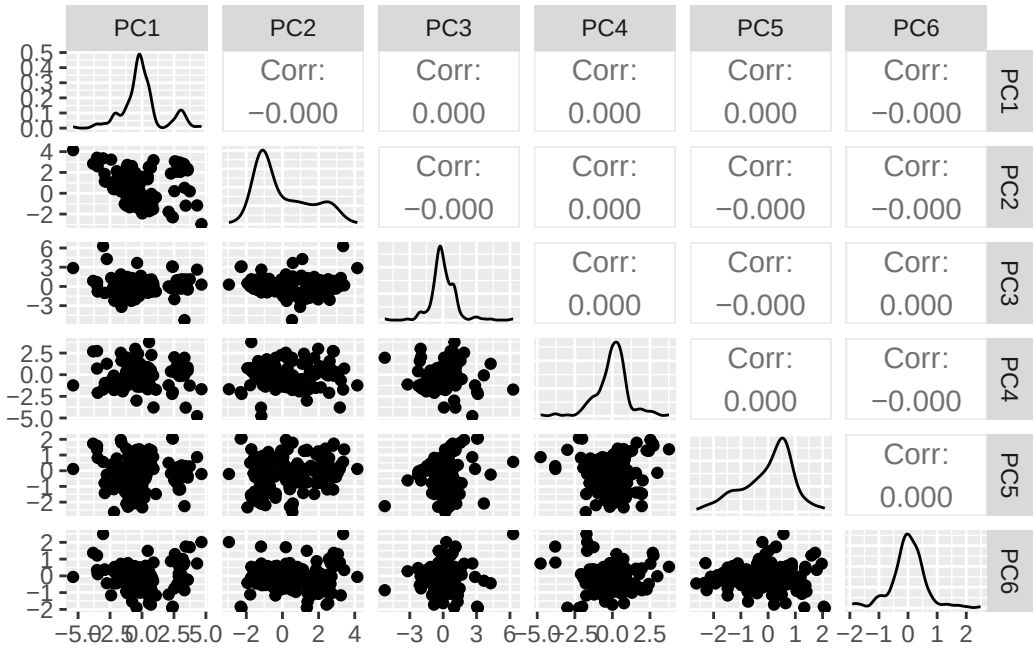
```
# weights:  48 (35 variable)
initial  value 383.436526
iter  10 value 170.460029
iter  20 value 162.183287
iter  30 value 161.110212
iter  40 value 160.200823
final  value 160.196480
converged
```

```r
# Create table to observe statistics of old and new model
tibble("Model (Old vs. New)" = c("old", "new"),
       "Deviance" = c(raw_model$deviance, adj_model$deviance),
       "AIC" = c(AIC(raw_model), AIC(adj_model)))
```

```
# A tibble: 2 x 3
  `Model (Old vs. New)` Deviance   AIC
  <chr>                    <dbl> <dbl>
1 old                       299.  399.
2 new                       320.  390.
```

iv. Bonus: Interesting to observe our principle components which have undergone transformation. The model computed by preProcess did a good job of outputting principle components with nearly normal distributions and no correlation.

```
adj_Glass |>
  select(1:6) |>
  ggpairs()
```



Question 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.

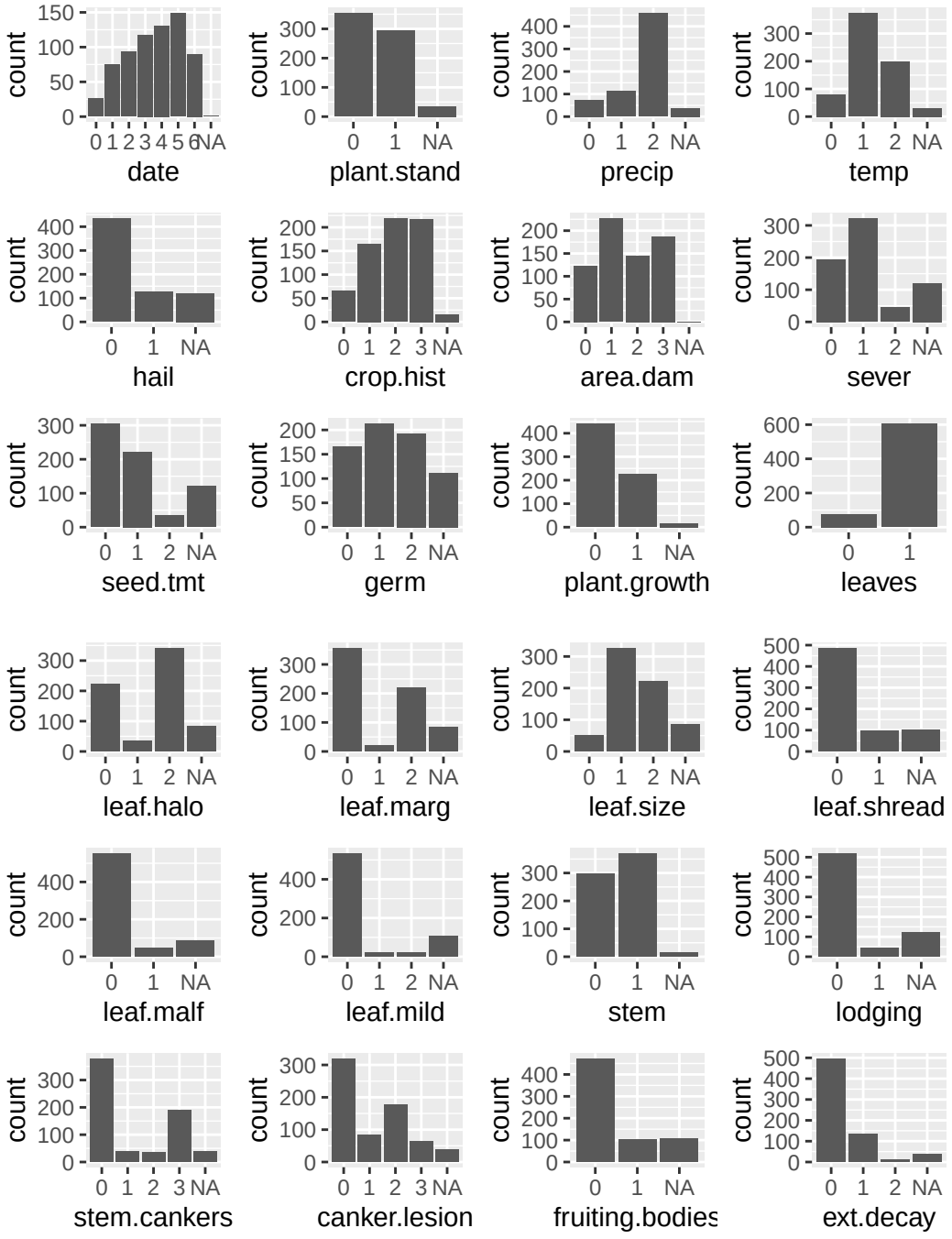
a. Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?

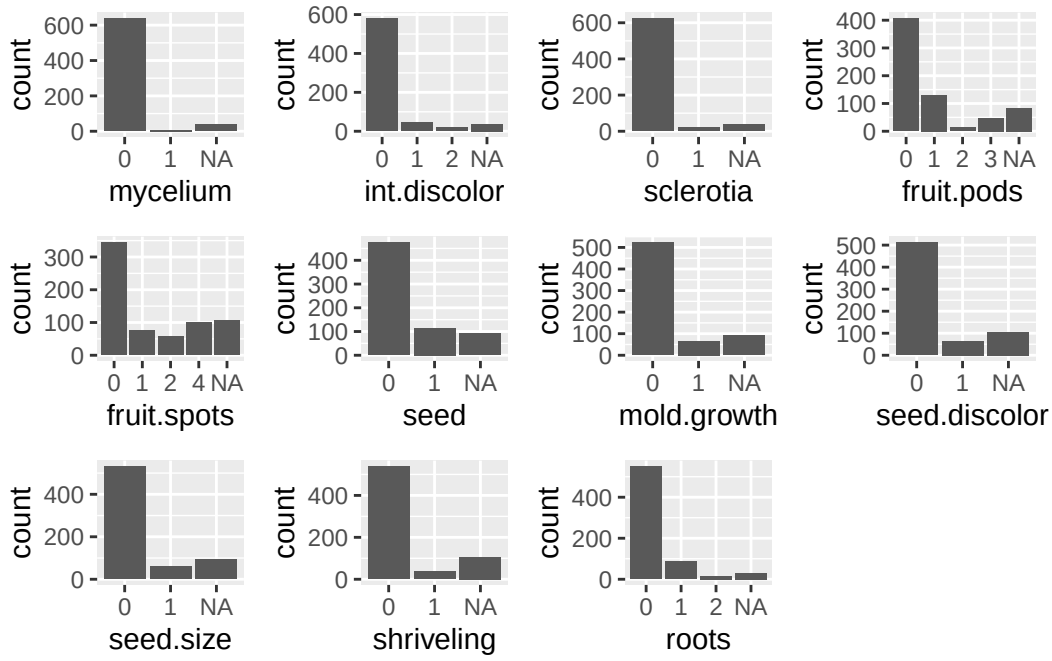
Answer: Looking at the distributions, we can see some severely uneven distributions. However, based on the rules of thumb in the chapter, none of the predictors qualify as having "degenerate" distributions. None of the predictors pass the first test, as we see even the highest ratios are only 1% or less. But, all of the predictors pass the second test. Although, `mycelium` and `sclerotia` are close to the 20x threshold. Interestingly, the

`nearZeroVar` function provided by the book says these variables do not pass the test and also adds `leaf.mild` to the list, which wasn't even on the radar with my calculations.

i. Visualize distributions of predictors.

```
data("Soybean")

# Splitting up predictors will help us load the distribution plots more cleanly
soybean_predictors1 <- Soybean |> select(2:13)
soybean_predictors2 <- Soybean |> select(14:25)
soybean_predictors3 <- Soybean |> select(26:36)


# Use purrr::map to visualize distributions for all predictors
predictor_distributions1 <- map(names(soybean_predictors1), ~ {
  ggplot(soybean_predictors1, aes(x = .data[[.x]])) +
    geom_bar()})

predictor_distributions2 <- map(names(soybean_predictors2), ~ {
  ggplot(soybean_predictors2, aes(x = .data[[.x]])) +
    geom_bar()})

predictor_distributions3 <- map(names(soybean_predictors3), ~ {
  ggplot(soybean_predictors3, aes(x = .data[[.x]])) +
    geom_bar()})
```


count
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
canker.lesion
fruiting.bodies
ext.decay



ii. Perform "degenerate" test from textbook chapter.

```
# First test for "degenerate" distributions is to get the ratio of unique counts
# to sample size. If unique_count/sample size is <= 10%, it doesn't pass this test.
Soybean |>
  summarise(across(2:36, \(x) n_distinct(x))) |>
  pivot_longer(cols = 1:35, names_to = "Predictors", values_to = "Unique_Count") |>
  mutate(counts_vs_sample_size = Unique_Count/683) |>
  arrange(desc(counts_vs_sample_size)) |>
  head(5)
```

```
# A tibble: 5 x 3
  Predictors    Unique_Count counts_vs_sample_size
  <chr>                <int>                 <dbl>
1 date                     8               0.0117
2 crop.hist                5               0.00732
3 area.dam                 5               0.00732
4 stem.cankers             5               0.00732
5 canker.lesion            5               0.00732
```

```
# Second test is to get the ratio of the highest to second highest counts for the
# distinct values in each predictor. If the highest count is at least 20x more
```

```
# than the second highest, it doesn't pass this test.
top2counts <- map_dfc(Soybean |> select(2:36), ~ {
  tibble(.x) |>
  count(.x, sort = TRUE) |>
  head(2) |>
  pull(n)
})

top2counts |>
  pivot_longer(cols = everything(), names_to = "Predictors", values_to = "Top2") |>
  arrange(Predictors) |>
  mutate(test = Top2/lead(Top2)) |>
  filter(Top2 == max(Top2), .by = Predictors) |>
  arrange(desc(test)) |>
  head(5)
```

```
# A tibble: 5 x 3
  Predictors    Top2  test
  <chr>        <int> <dbl>
1 mycelium       639 16.8
2 sclerotia      625 16.4
3 int.discolor   581 13.2
4 leaves         606  7.87
5 leaf.malf      554  6.60
```

```
# Try out function which performs test for us from book
nearZeroVar(Soybean |> select(2:36), names = TRUE, freqCut = 20)
```

```
[1] "leaf.mild" "mycelium"  "sclerotia"
```

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?

Answer: After my examination of the data, it is clear that there is systemic relationship between `Class` and likelihood of NA values in certain predictor values. The table in part i. was the first clue that this relationship was present, because there are groups of predictors that have equal NA counts. Next, I grouped these variables to see if they were indeed NA in the same rows and for the same Class, and I found that they were. Lastly, I calculated tables similar to those in step i. to examine the % of NA counts when the data was filtered to the Classes where these high NA counts were appearing. This gave me concusive evidence that Class has a clear and direct relationship with NA values for many of our predictors.

i. Create a table of our predictors ordered by the ratio of NA values to Total Count.

```
# Create dataset of NA counts
na_predictors <- Soybean |>
  select(2:36) |>
  summarise(across(everything(), \(x) sum(is.na(x)))) |>
  tibble() |>
  pivot_longer(cols = everything(), names_to = "Predictors", values_to = "NA_Counts")

# Create dataset of non-NA counts
not_na_predictors <- Soybean |>
  select(2:36) |>
  summarise(across(everything(), \(x) sum(!is.na(x)))) |>
  tibble() |>
  pivot_longer(cols = everything(), names_to = "Predictors", values_to = "Other_Counts")

# Combine datasets
predictor_na_vs_not_na <- na_predictors |>
  left_join(not_na_predictors, by = join_by(Predictors == Predictors))

# Compute percentage of NA counts to Total counts
predictor_na_vs_not_na |>
  mutate(na_ratio = NA_Counts/(Other_Counts+NA_Counts)) |>
  arrange(desc(na_ratio)) |>
  print(n = 14)
```

```
# A tibble: 35 x 4
   Predictors      NA_Counts Other_Counts na_ratio
   <chr>               <int>        <int>    <dbl>
 1 hail                  121          562    0.177
 2 sever                 121          562    0.177
 3 seed.tmt              121          562    0.177
 4 lodging               121          562    0.177
 5 germ                  112          571    0.164
 6 leaf.mild             108          575    0.158
 7 fruiting.bodies       106          577    0.155
 8 fruit.spots           106          577    0.155
 9 seed.discolor         106          577    0.155
10 shriveling            106          577    0.155
11 leaf.shread           100          583    0.146
12 seed                   92          591    0.135
13 mold.growth            92          591    0.135
```

```
14 seed.size              92          591    0.135
# i 21 more rows
```

ii. Examine predictors with high % of NAs grouped with predictors of equal NA_Counts

```
# Examine Predictors with high % of NAs grouped by equal NA_Counts
Soybean |>
  tibble() |>
  filter(is.na(hail)) |>
  count(Class, hail, sever, seed.tmt, lodging)
```

```
# A tibble: 5 x 6
  Class                       hail  sever seed.tmt lodging     n
  <fct>                       <fct> <fct> <fct>    <fct>   <int>
1 2-4-d-injury                <NA>  <NA>  <NA>     <NA>       16
2 cyst-nematode               <NA>  <NA>  <NA>     <NA>       14
3 diaporthe-pod-&-stem-blight <NA>  <NA>  <NA>     <NA>       15
4 herbicide-injury            <NA>  <NA>  <NA>     <NA>        8
5 phytophthora-rot            <NA>  <NA>  <NA>     <NA>       68
```

```
# Examine Predictors with high % of NAs grouped by equal NA_Counts
Soybean |>
  tibble() |>
  filter(is.na(fruiting.bodies)) |>
  count(Class,fruiting.bodies, fruit.spots, seed.discolor, shriveling)
```

```
# A tibble: 4 x 6
  Class            fruiting.bodies fruit.spots seed.discolor shriveling     n
  <fct>            <fct>           <fct>       <fct>         <fct>      <int>
1 2-4-d-injury     <NA>            <NA>        <NA>          <NA>          16
2 cyst-nematode    <NA>            <NA>        <NA>          <NA>          14
3 herbicide-injury <NA>            <NA>        <NA>          <NA>           8
4 phytophthora-rot <NA>            <NA>        <NA>          <NA>          68
```

```
# Examine Predictors with high % of NAs grouped by equal NA_Counts
Soybean |>
  tibble() |>
  filter(is.na(seed)) |>
  count(Class, seed, mold.growth, seed.size)
```

```
# A tibble: 3 x 5
  Class            seed  mold.growth seed.size     n
  <fct>            <fct> <fct>       <fct>     <int>
1 2-4-d-injury     <NA>  <NA>        <NA>         16
2 herbicide-injury <NA>  <NA>        <NA>          8
3 phytophthora-rot <NA>  <NA>        <NA>         68
```

iii. Re-examine high-NA-grouped predictors after filtering data to Class's with systematic NAs within certain predictors.

```
# Re-examine high-NA-grouped predictors after filtering Class
Soybean |>
  filter(grepl("injury|nematode|phytophthora|diaporthe-pod", Class)) |>
  select(hail, sever, seed.tmt, lodging) |>
  summarise(across(everything(), \(x) sum(is.na(x))/n())) |>
  tibble() |>
  pivot_longer(cols = everything(), names_to = "Predictors", values_to = "NA_%_of_Total")
```

```
# A tibble: 4 x 2
  Predictors `NA_%_of_Total`
  <chr>                <dbl>
1 hail                 0.858
2 sever                0.858
3 seed.tmt             0.858
4 lodging              0.858
```

```
# Re-examine high-NA-grouped predictors after filtering Class
Soybean |>
  filter(grepl("injury|nematode|phytophthora", Class)) |>
  select(fruiting.bodies, fruit.spots, seed.discolor, shriveling) |>
  summarise(across(everything(), \(x) sum(is.na(x))/n())) |>
  tibble() |>
  pivot_longer(cols = everything(), names_to = "Predictors", values_to = "NA_%_of_Total")
```

```
# A tibble: 4 x 2
  Predictors      `NA_%_of_Total`
  <chr>                    <dbl>
1 fruiting.bodies          0.841
2 fruit.spots              0.841
3 seed.discolor            0.841
4 shriveling               0.841
```

```r
# Re-examine high-NA-grouped predictors after filtering Class
Soybean |>
  filter(grepl("injury|phytophthora", Class)) |>
  select(seed, mold.growth, seed.size) |>
  summarise(across(everything(), \(x) sum(is.na(x))/n())) |>
  tibble() |>
  pivot_longer(cols = everything(), names_to = "Predictors", values_to = "NA_%_of_Total")
```

```
# A tibble: 3 x 2
  Predictors  `NA_%_of_Total`
  <chr>                 <dbl>
1 seed                  0.821
2 mold.growth           0.821
3 seed.size             0.821
```

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

Answer: Since there a strong relationship between % of NA values and Class, I would argue that NA values are informative, and including them in the model might increase accuracy. For example, using a Naive approach, an NA value in `seed` would predict a 14% chance of narrowing `Class` to 3 options, which is a pretty good start for a model with one predictor. Therefore, for predictors with >10% NA values, I would consider imputing a "dummy" value in place of the NAs, if it would help our selected model factor in NA values. Furthermore, I think it would be safe to remove all but 1 predictor of each group of predictors with eaual NA ratios, since we have proven that the missingness of these groups share Classes. However, we would also need further domain knowledge to determine the appropriateness of this. Next, for predictors with <10% NA values I would impute data using a predictive classification model. In summary, I would recommend removing predictors that were grouped by equal NA ratios (leaving 1 per group), imputing "dummy" values for predictors with >10% NA ratio, and imputing predicted values for predictors with <10% NA ratio.