Data Pre-Processing and Exponential Smoothing

Week2 homework

Subhalaxmi Rout

06/14/2021

library(fpp2)  
library(mlbench)  
library(psych) # skew  
library(corrplot) # correlation  
library(PerformanceAnalytics) # correlation and histogram  
library(DataExplorer) # histogram  
library(VIM) # kNN impute  
library(caret) # near zero variance  
library(dplyr) # arrange  
library(tidyr) # gather

### Forecasting: Principles and Practices (Chapter 7 - Exponential smoothing)

#### 7.1

Consider the pigs series — the number of pigs slaughtered in Victoria each month.

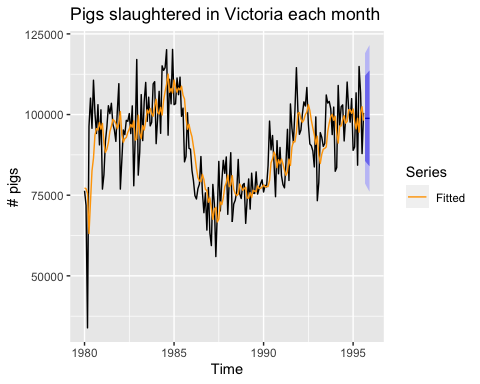
**Use the ses() function in R to find the optimal values of**  
 **and $\l\_0$, and generate forecasts for the next four months.**

model = ses(pigs)  
  
fc <- model %>%  
 forecast(h = 4)  
  
summary(fc)

##   
## Forecast method: Simple exponential smoothing  
##   
## Model Information:  
## Simple exponential smoothing   
##   
## Call:  
## ses(y = pigs)   
##   
## Smoothing parameters:  
## alpha = 0.2971   
##   
## Initial states:  
## l = 77260.0561   
##   
## sigma: 10308.58  
##   
## AIC AICc BIC   
## 4462.955 4463.086 4472.665   
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set 385.8721 10253.6 7961.383 -0.922652 9.274016 0.7966249 0.01282239  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## Sep 1995 98816.41 85605.43 112027.4 78611.97 119020.8  
## Oct 1995 98816.41 85034.52 112598.3 77738.83 119894.0  
## Nov 1995 98816.41 84486.34 113146.5 76900.46 120732.4  
## Dec 1995 98816.41 83958.37 113674.4 76092.99 121539.8

The optimal value of alpha = 0.2971 and Initial states = 77260.0561.

autoplot(fc, series = "Data") +  
 autolayer(fc$fitted, series="Fitted") +  
 labs(y="# pigs", title="Pigs slaughtered in Victoria each month ", color = "Serise") +  
 scale\_color\_manual(name="Series",   
 values = c("Data"="gray50",   
 "Fitted"="orange"))



**Compute a 95% prediction interval for the first forecast using**  
 **where s is the standard deviation of the residuals. Compare** **your interval with the interval produced by R.**

forecast <- 98816.41  
s <- sd(fc$residuals)  
y1 <- forecast - 1.96 \* s  
print(y1)

## [1] 78679.97

y2 <- forecast + 1.96 \* s  
print(y2)

## [1] 118952.8

95% prediction interval for the first forecast = (78679.97, 118952.8)

#### 7.2

**Write your own function to implement simple exponential smoothing. The function should take arguments y (the time series), alpha (the smoothing parameter**  
**α) and level (the initial level $\l\_0$). It should return the forecast of the next observation in the series. Does it give the same forecast as ses()?**

To answer this question follow below approach:

* First create own time series
* apply ses() to calculate alpha, and initial value i.e l
* create own simple exponential smoothing function
* Pass time series, alpha, and l as parameters
* compare forecast value by using ses() and own created my\_ses()

# time serise  
myts <- ts(c(10,12,13, 14, 15, 2, 8, 36, 28, 12, 16, 22, 26, 23, 34, 11, 18, 19, 24, 5), frequency = 1)  
  
# forecast for next observation  
fc <- ses(myts) %>% forecast::forecast(h = 1)  
  
# calculate alpha and l value  
alpha <- fc$model$par[1]  
print(alpha)

## alpha   
## 0.000100015

l <- fc$model$par[2]  
print(l)

## l   
## 17.39929

Created own ses function named my\_ses. Pass the parameters to see the forecast and compair the result.

my\_ses <- function(ts = myts, alpha = alpha, l = l){  
 forecast <- l  
 for(i in 1:length(myts)){  
 forecast <- alpha\*myts[i] + (1 - alpha)\*forecast   
 }  
 paste0("Forecast of next observation : ", forecast)  
}  
  
# user defined function  
my\_ses(ts = myts, alpha = alpha, l = l)

## [1] "Forecast of next observation : 17.3992915009543"

# Pre-defined function  
fc

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 21 17.39929 5.396422 29.40216 -0.957514 35.7561

Yes, my\_ses() function gives the same result as the ses().

#### 7.3

**Modify your function from the previous exercise to return the sum of squared errors rather than the forecast of the next observation. Then use the optim() function to find the optimal values of and $\l\_0$. Do you get the same values as the ses() function?**

#myts  
  
my\_ses\_err <- function(par = c(alpha, l), ts){  
 err <- 0  
 sse <- 0  
 alpha <- par[1]  
 l <- par[2]  
 forecast <- l  
   
 for(i in 1:length(ts)){  
 err <- ts[i] - forecast  
 sse <- sse + err \*\* 2  
   
 forecast <- alpha\*ts[i] + (1 - alpha)\*forecast   
 }  
   
 return(sse)  
  
}  
  
optimal\_value <- optim(par = c(0.5, myts[1]), ts = myts, fn = my\_ses\_err)  
  
paste0("Optimal value of alpha : ", optimal\_value$par[1])

## [1] "Optimal value of alpha : -0.607763263360704"

paste0("Optimal value of l : ", optimal\_value$par[2])

## [1] "Optimal value of l : 11.837619120558"

Optimal value of alpha and l using ses().

paste0("Optimal value of alpha using ses() : ", fc$model$par[1])

## [1] "Optimal value of alpha using ses() : 0.000100014964653795"

paste0("Optimal value of l using ses() : ", fc$model$par[2])

## [1] "Optimal value of l using ses() : 17.399287399534"

The alpha value lies between 0 and 1 so anything below zero considers as 0. using my\_ses\_err() gets negative alpha values i.e zero. From ses() alpha value almost zero. The different optimal value getting for l.

Get alpha value almost same but different l value.

### Applied predictive modeling (Chapter 3 - Data Pre-processing)

#### 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. The data can be accessed via: library(mlbench)

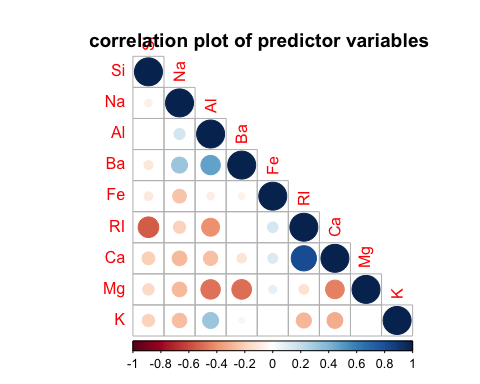
**Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors.**

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 ...

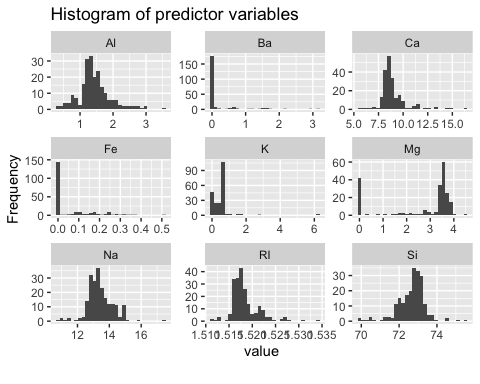
Remove the target variable Type from the Glass dataset named as data. Visualize the distribution and relationship between predictors by histogram and correlation plot.

# remove traget variable  
data <- Glass[, -10]  
  
# corelation of predictors  
corrplot(cor(data), type="lower", order="hclust", mar=c(0,0,1,0)) + title("correlation plot of predictor variables")

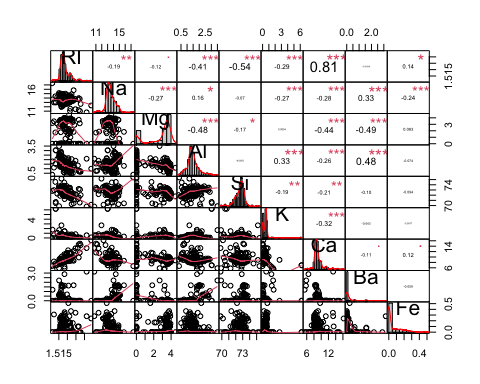


## numeric(0)

# histogram of predictors  
plot\_histogram(data,   
 geom\_histogram\_args = list(bins = 30L),   
 title = "Histogram of predictor variables",  
 nrow = 3L,  
 ncol = 3L)



# correlation and distribution of predictors  
chart.Correlation(data, histogram = TRUE, method = "pearson")



The correlation plot shows:

* very high positive relationship between Ca and RI
* RI and Si, and Mg and Al have negative relationship
* Al and Ba positively correlate with each other

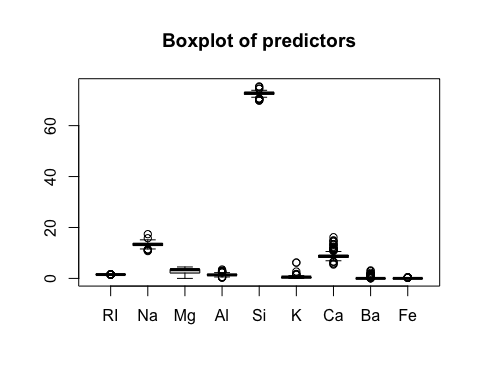
Histogram shows:

* Ba, Fe, K completely right tail skewed distribution
* RI, and Ca have slightly right tail skewed
* Mg has 2 peaks, mostly left tail skewed
* Na, Al, and Si looks symmetrical

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

Yes, outliers are present in the data. Below box-plor shoes the outliers of predictor.

boxplot(data, main = "Boxplot of predictors")

 Above histogram shows predictors are skewed. To verify lets have a look on skewness. If the predictor distribution is roughly symmetric, the skewness values will be close to zero. As the distribution becomes more right skewed, the skewness statistic becomes larger. Similarly, as the distribution becomes more left skewed, the value becomes negative.

skewValues <- apply(data, 2, skew)  
skewValues

## 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

Skewvalue shows the same result that we get from histogram.

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

Most of the predictors are skewed and having outliers, we can perform box-cox or Scaling and Centering transformation on each predictor variable might improve the model classification.

#### 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.**

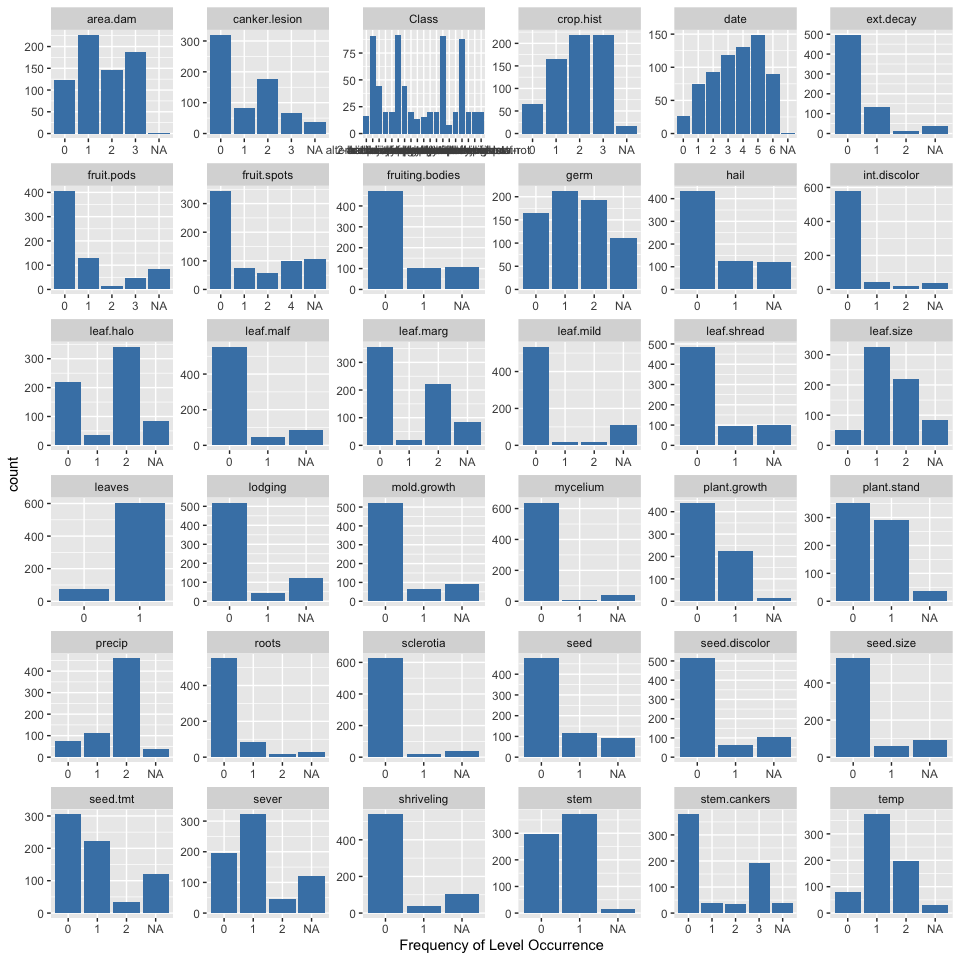
**Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?**

data("Soybean")  
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 ...  
## $ 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 ...

Soybean dataset have 35 predictor variables and 1 target variable. Below histogram shows the distribution of the categorical predictors.

Soybean %>%  
 gather(variable, level) %>%  
 ggplot(aes(x = level)) +  
 geom\_bar(fill = 'steelblue') +   
 xlab("Frequency of Level Occurrence") +  
 facet\_wrap(~variable, scales = 'free')



nearZeroVar function could be used to find the degenrate variables for Soybean dataset.

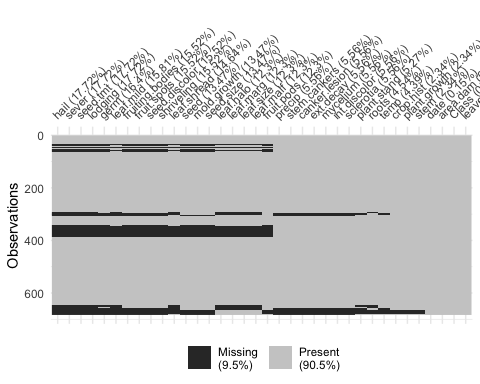
nzero <- nearZeroVar(Soybean)  
names(Soybean)[nzero]

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

There are 3 predictors “leaf.mild” “mycelium” “sclerotia” have non zero variance. It would be better too remove these variables from the model.

**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?**

visdat::vis\_miss(Soybean, sort\_miss = TRUE)



Total 9.5% of data having NAs. Top 5 variables have high missing values are : ‘hail’: 17.72%, ‘sever’:17.72%, ‘seed.tmt’: 17.72%, ‘lodging’:17.72% and ‘germ’:16.4%.

Soybean %>% filter\_all(any\_vars(is.na(.))) %>%  
 group\_by(Class) %>%  
 summarise(Count = n()) %>%  
 arrange(desc(Count))

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

phytophthora-rot class has high number of NA’s.

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

We would prefer to Eliminate predictors if the NA’S has more than 90%. In this dataset, most of the NA’S lies below 20%. We would go with the KNN imputation. The K value that we are decide to apply is 20 (~ close to square root of number of observationa). To perform this use VIM pacakge. We can see after apply kNN() there are some additional variables added to the data. Remove all additional variables.

Soybean\_impute <- kNN(data = Soybean, k = 26)

Soybean\_impute <- subset(Soybean\_impute, select = -c(Class\_imp:roots\_imp))  
head(Soybean\_impute)

## 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

colSums(is.na(Soybean\_impute))

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

Few colums still have NAs, most of the variable are factor. Impute mode for the remaining missing values. Missing value columns are, plant.stand , precip, temp, germ, and leaf.size.

mode <- function(df, ...) {  
 tb <- table(df)  
 which(tb == max(tb)) %>% sort %>% `[`(., 1) %>% names  
}  
  
impute <- function(df, fun) {  
 fval <- fun(df, na.rm = TRUE)  
 y <- ifelse(is.na(df), fval, df)  
 return(y)  
}  
  
Soybean\_impute\_2 <- Soybean\_impute %>%   
 mutate(  
 plant.stand = impute(plant.stand, mode),  
 precip = impute(precip, mode),  
 temp = impute(temp, mode),  
 germ = impute(germ, mode),  
 leaf.size = impute(leaf.size, mode)  
 ) %>%   
 mutate\_if(is.factor, as.numeric)  
  
visdat::vis\_miss(Soybean\_impute\_2)

