# Data 624_Exercise 3.1_HW4

Jiaxin Zheng

2025-03-30

## Exercise 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)
```

```
## Warning: package 'mlbench' was built under R version 4.4.3
```

```
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.2

## Warning: package 'readr' was built under R version 4.4.3

## Warning: package 'dplyr' was built under R version 4.4.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggplot2)
library(reshape2)
```

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

```r
data(Glass)
str(Glass)
```

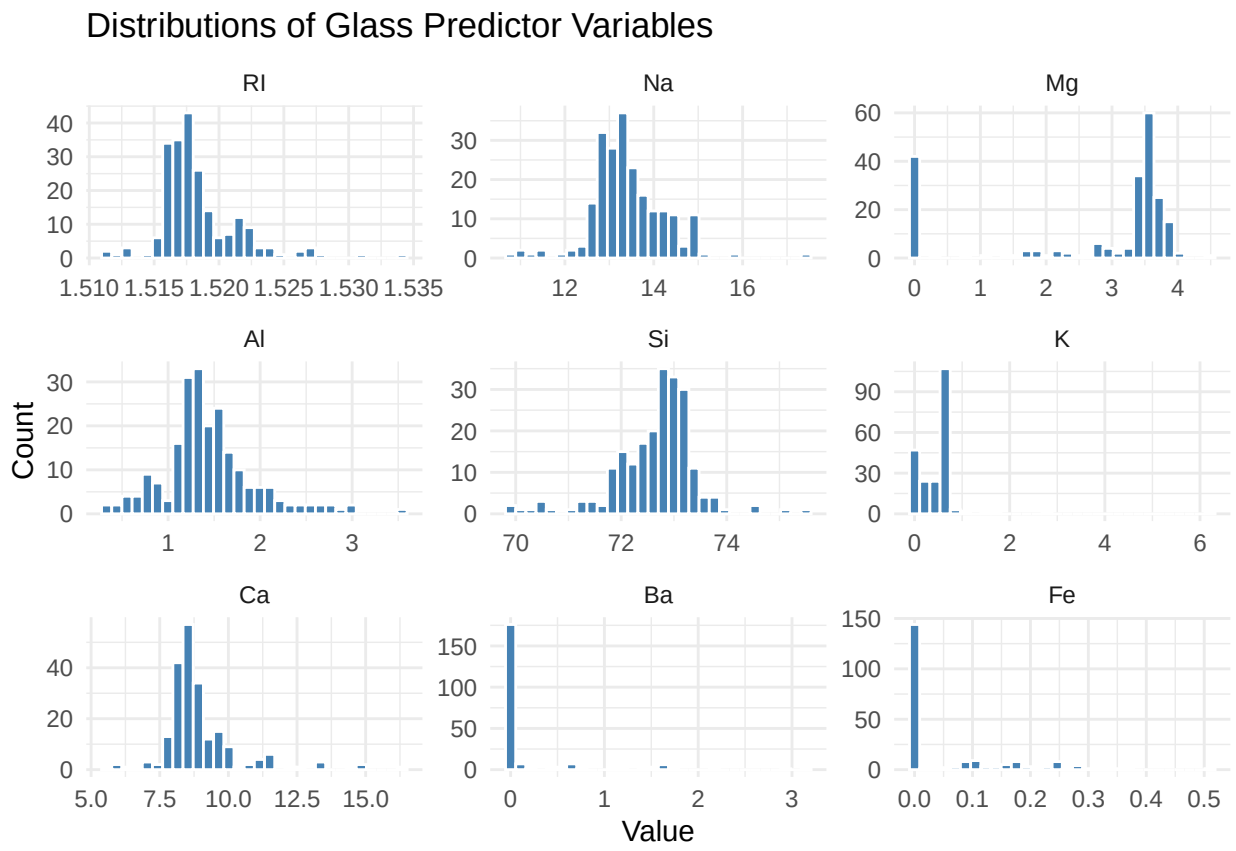
```
## 'data.frame':    214 obs. of  10 variables:
##  $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
##  $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
##  $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
##  $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
##  $ Si  : num  71.8 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.**

```r
# There is no missing value
sum(is.na(Glass))
```

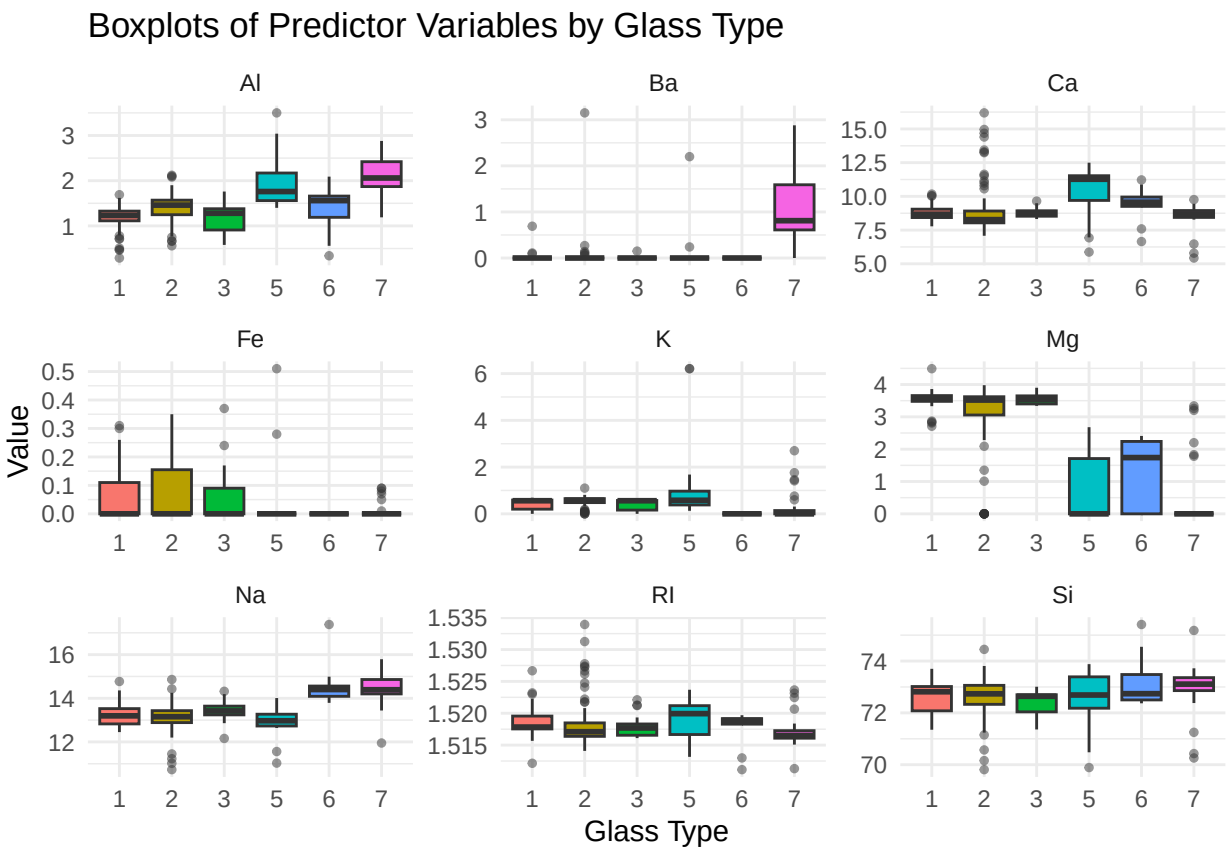
```
## [1] 0
```

```r
glass <- melt(Glass, id.vars = "Type")
ggplot(glass, aes(x = value)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "white") +
  facet_wrap(~variable, scales = "free") +
  theme_minimal() +
  labs(title = "Distributions of Glass Predictor Variables", x = "Value", y = "Count")
```




```
# Shape the data into long format
glass_long <- Glass %>%
  pivot_longer(cols = -Type, names_to = "Variable", values_to = "Value")

# Boxplot for each variable grouped by glass type
ggplot(glass_long, aes(x = Type, y = Value, fill = Type)) +
  geom_boxplot(outlier.size = 1, outlier.alpha = 0.5) +
  facet_wrap(~Variable, scales = "free", ncol = 3) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(title = "Boxplots of Predictor Variables by Glass Type",
       x = "Glass Type",
       y = "Value")
```

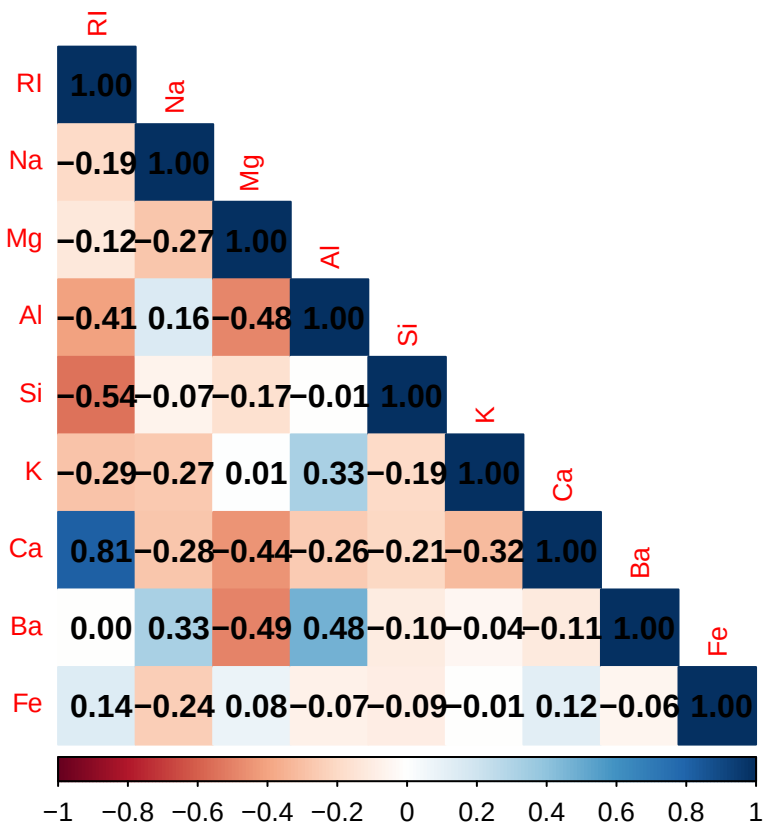



```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.4.2
```

```
## corrplot 0.95 loaded
```

```
cor_matrix <- cor(Glass[, 1:9])
corrplot(cor_matrix, method = "color", type = "lower", addCoef.col = "black", tl.cex = 0.8)
```




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

**From above's histogram plot:**

- RI: Right skewed with outliers, the tails is long.
- Na: Looks like normal distribution with some outliers.
- Mg: Left skewed, non-normal, and bi-modal.
- Al: Looks like right skewed, normal distributions.
- Si: Looks like a left skewed, but with long tails
- K: Non-normal with outliers.
- Ca: Right skewed with outliers.
- Ba: Is heavily right skewed and with outliers.
- Fe: Is heavily right skewed and with outliers.

**From the Boxplot:**

- We could see from the boxplots, there are numbers of outliers for all.

**From the correlation plot:**

- Correlations between RI and Si have a negative correlation of -0.54
- Correlations between RI and Ca have a positive correlation of 0.81.

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

- Box- Cox transformation maybe only works on positive values. Like K, Ba, Fe, I don't see much effect.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.2
```

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

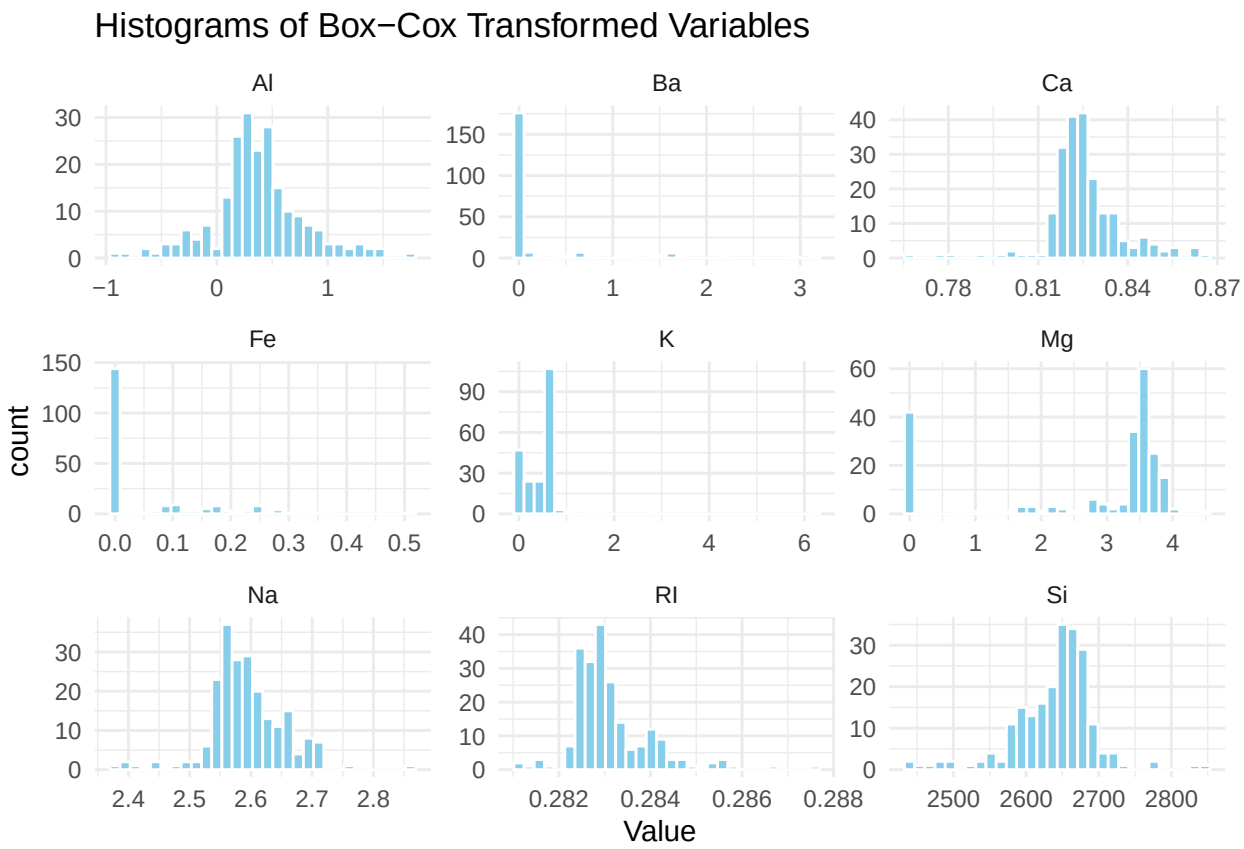
```
##
## Attaching package: 'caret'
```

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

```
# Preprocess using Box-Cox
glass_numeric <- Glass[, 1:9]
pp_boxcox <- preProcess(glass_numeric, method = "BoxCox")
glass_boxcox <- predict(pp_boxcox, glass_numeric)
```

```
glass_boxcox_long <- glass_boxcox %>%
  pivot_longer(cols = everything(), names_to = "Variable", values_to = "Value")

ggplot(glass_boxcox_long, aes(x = Value)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "white") +
  facet_wrap(~Variable, scales = "free") +
  theme_minimal() +
  labs(title = "Histograms of Box-Cox Transformed Variables")
```




## 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.

The data can be loaded via:

```
library(mlbench)
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 ...
```

```
sum(is.na(Soybean))
```

```
## [1] 2337
```

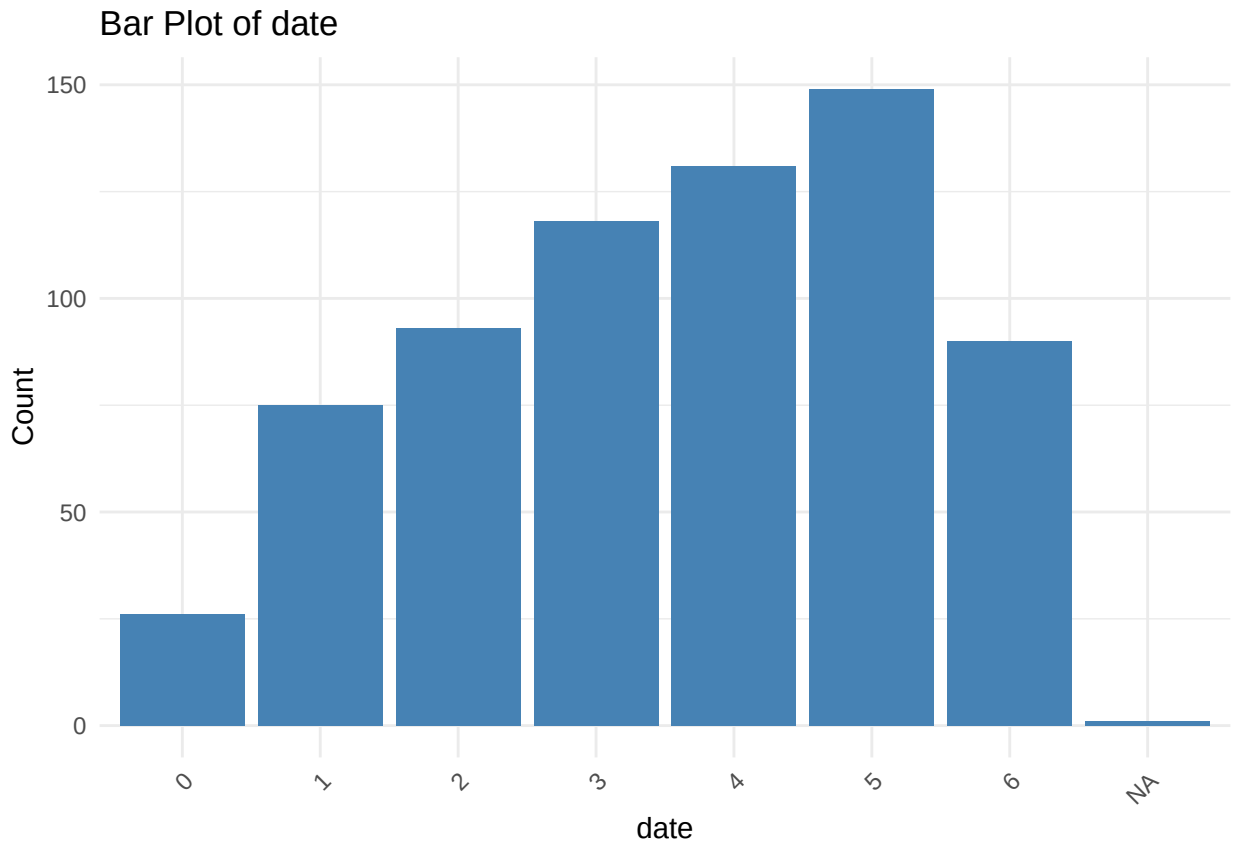
**a. Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?**

- There are many missing value, some of the predictors are also not very imbalanced.

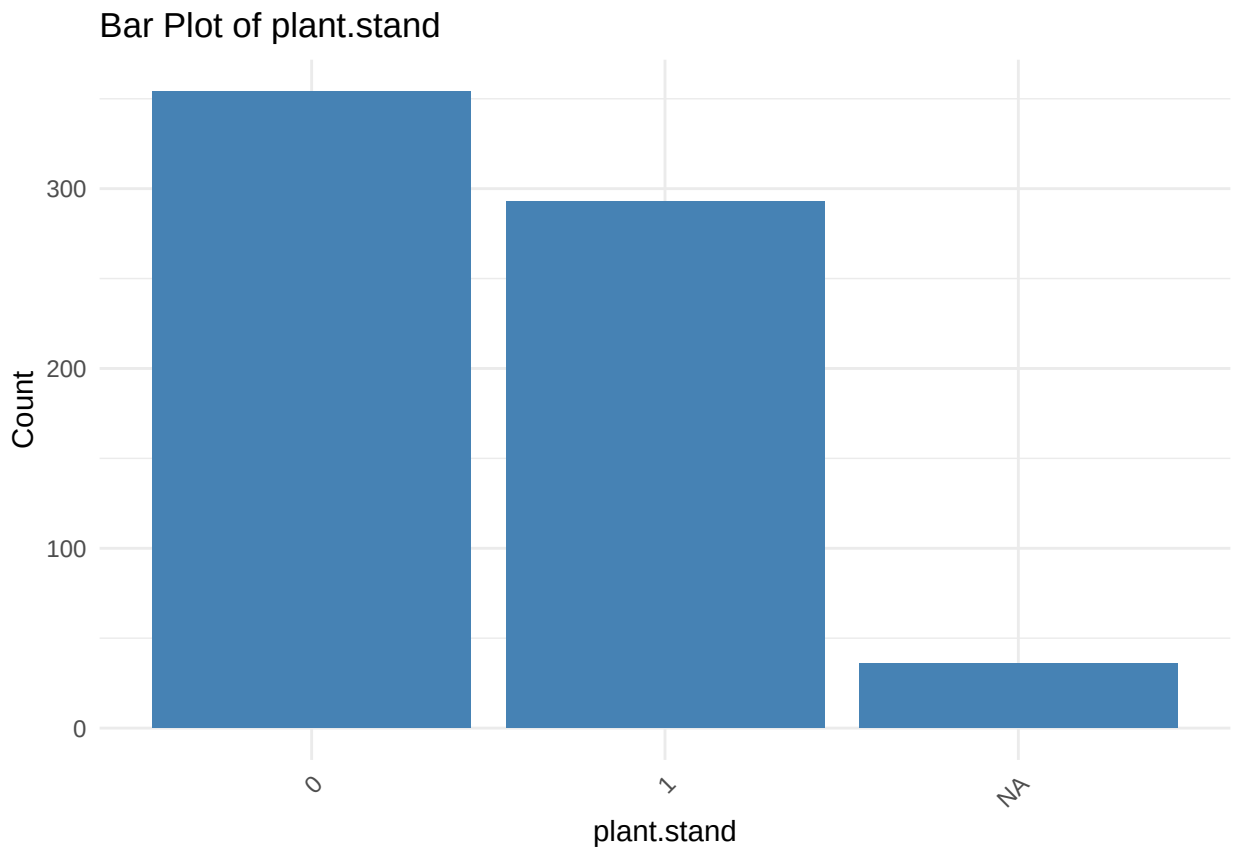
```
predictors <- Soybean %>%
  select(-Class)

for (predictor in names(predictors)) {
  print(
    ggplot(data = predictors, aes(x = as.factor(predictors[[predictor]]))) +
      geom_bar(fill = "steelblue") +
      labs(
        title = paste("Bar Plot of", predictor),
        x = predictor,
        y = "Count"
      ) +
      theme_minimal() +
      theme(axis.text.x = element_text(angle = 45, hjust = 1))
  )
}
```

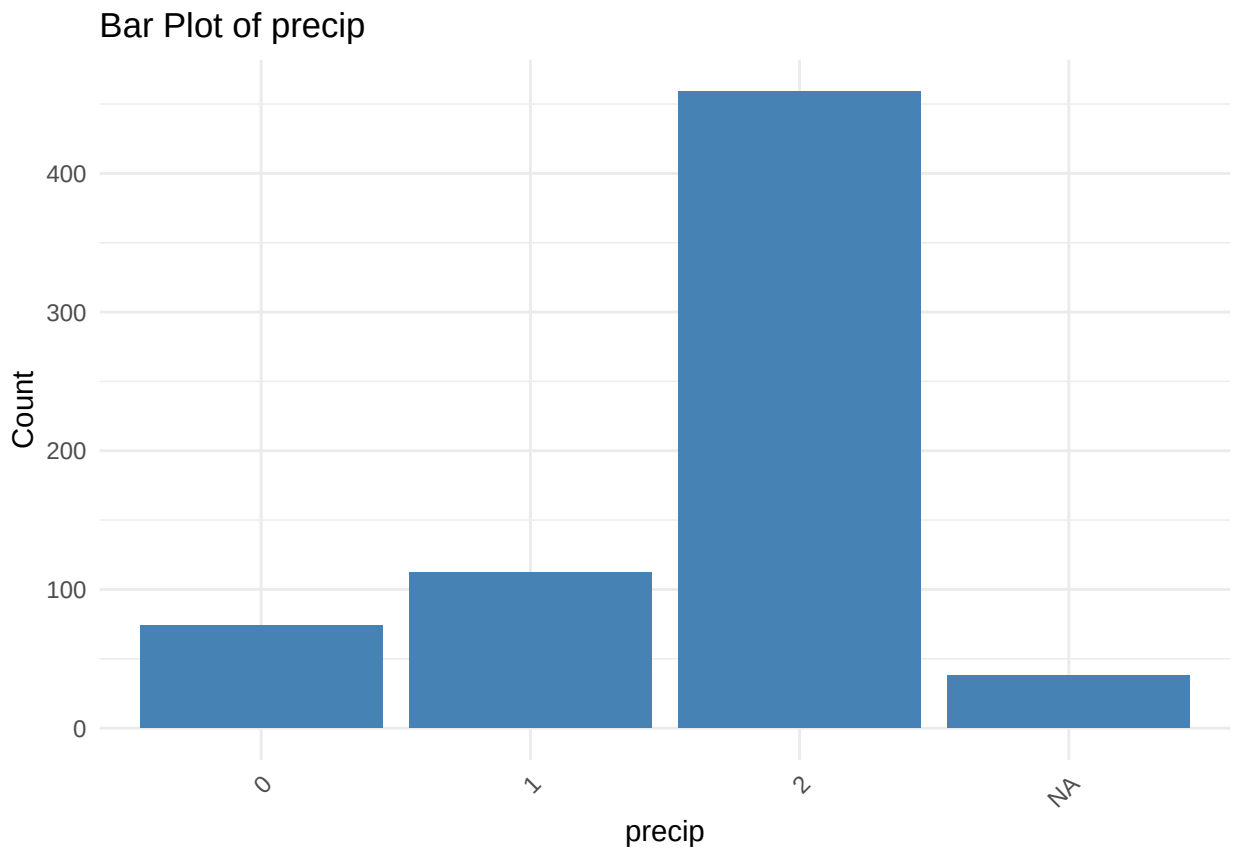
```
## Warning: Use of `predictors[[predictor]]` is discouraged.
## i Use `.data[[predictor]]` instead.
```



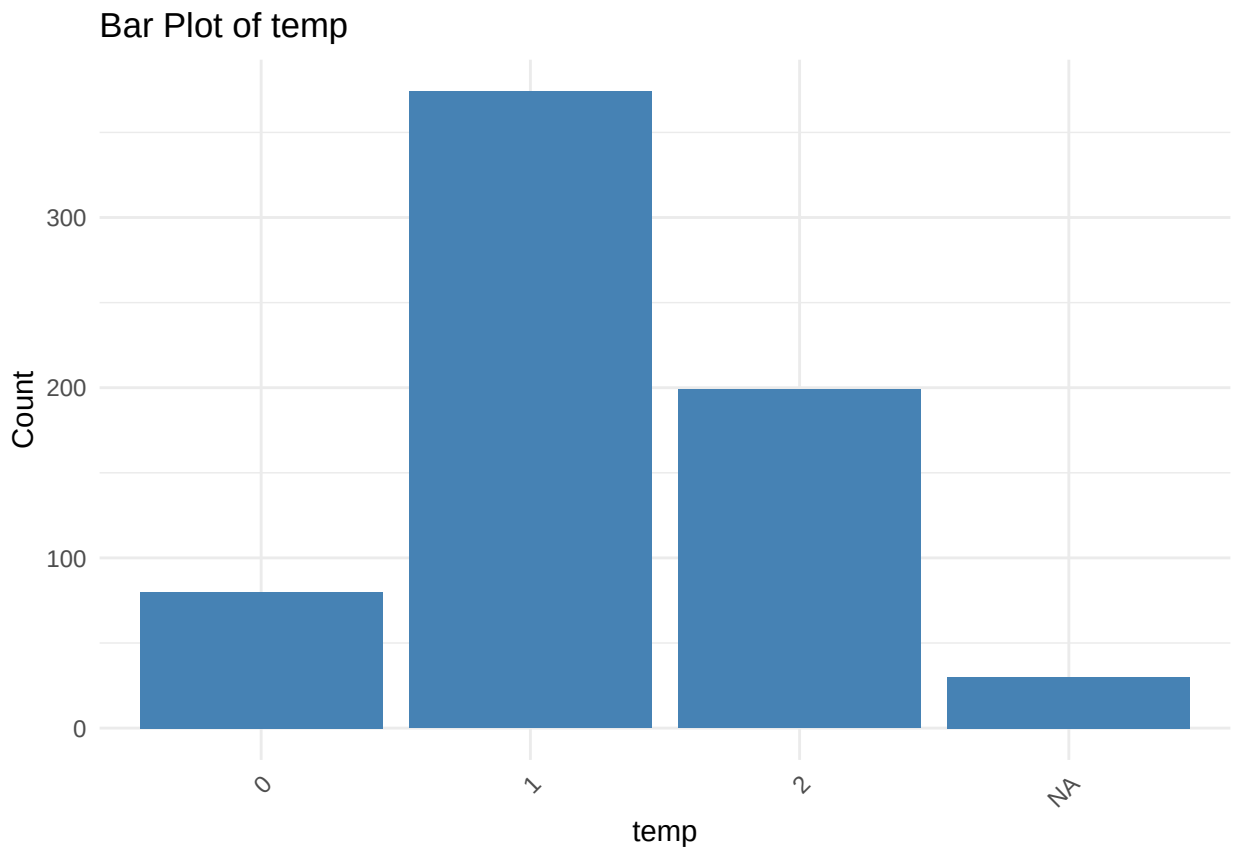

```
## Warning: Use of `predictors[[predictor]]` is discouraged.
## i Use `.data[[predictor]]` instead.
```



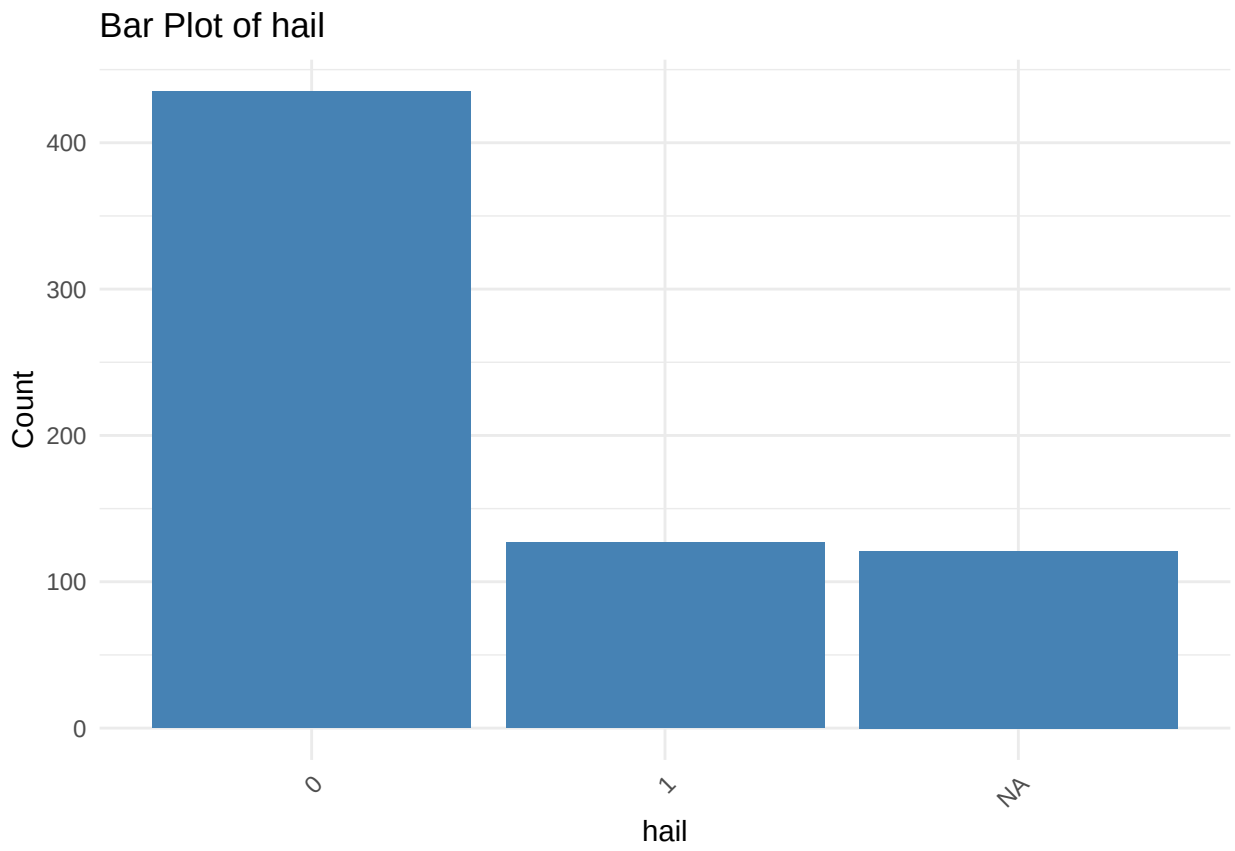

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



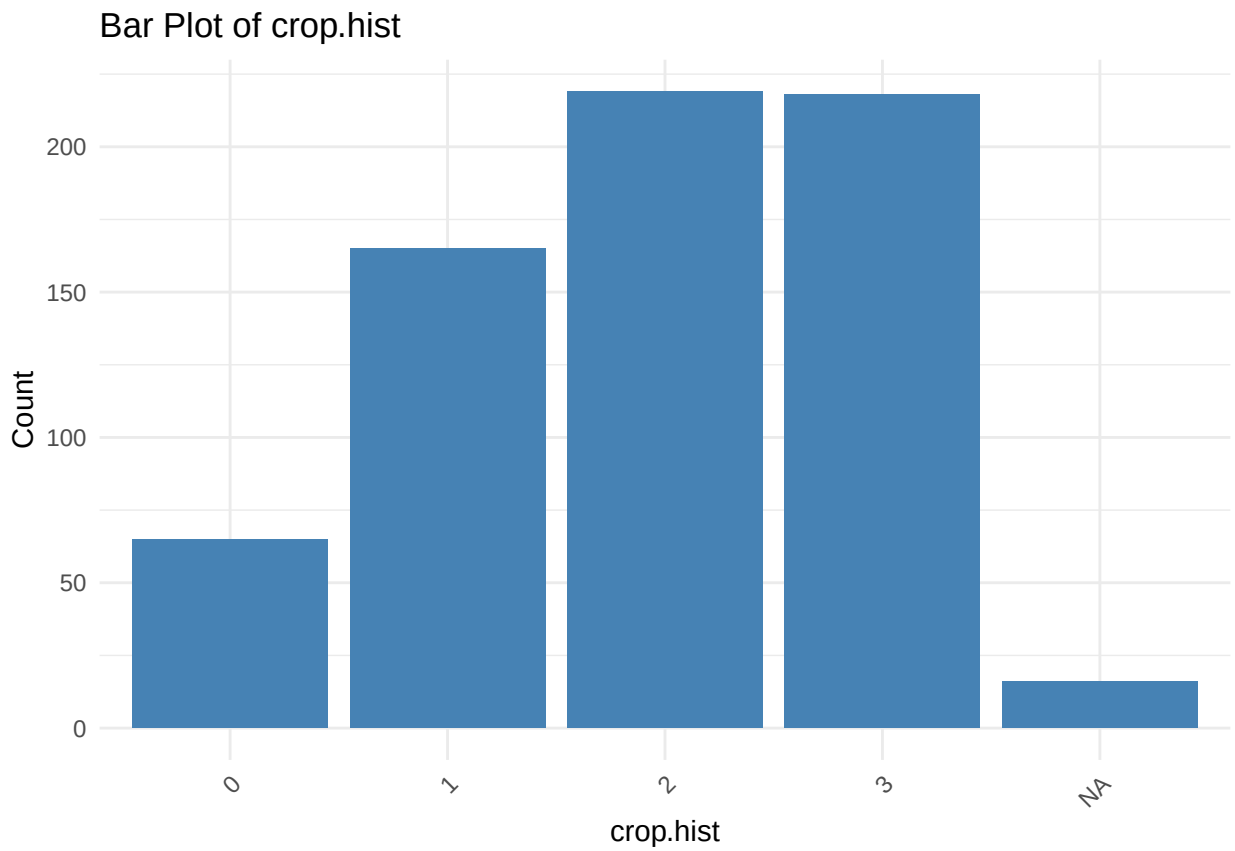

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



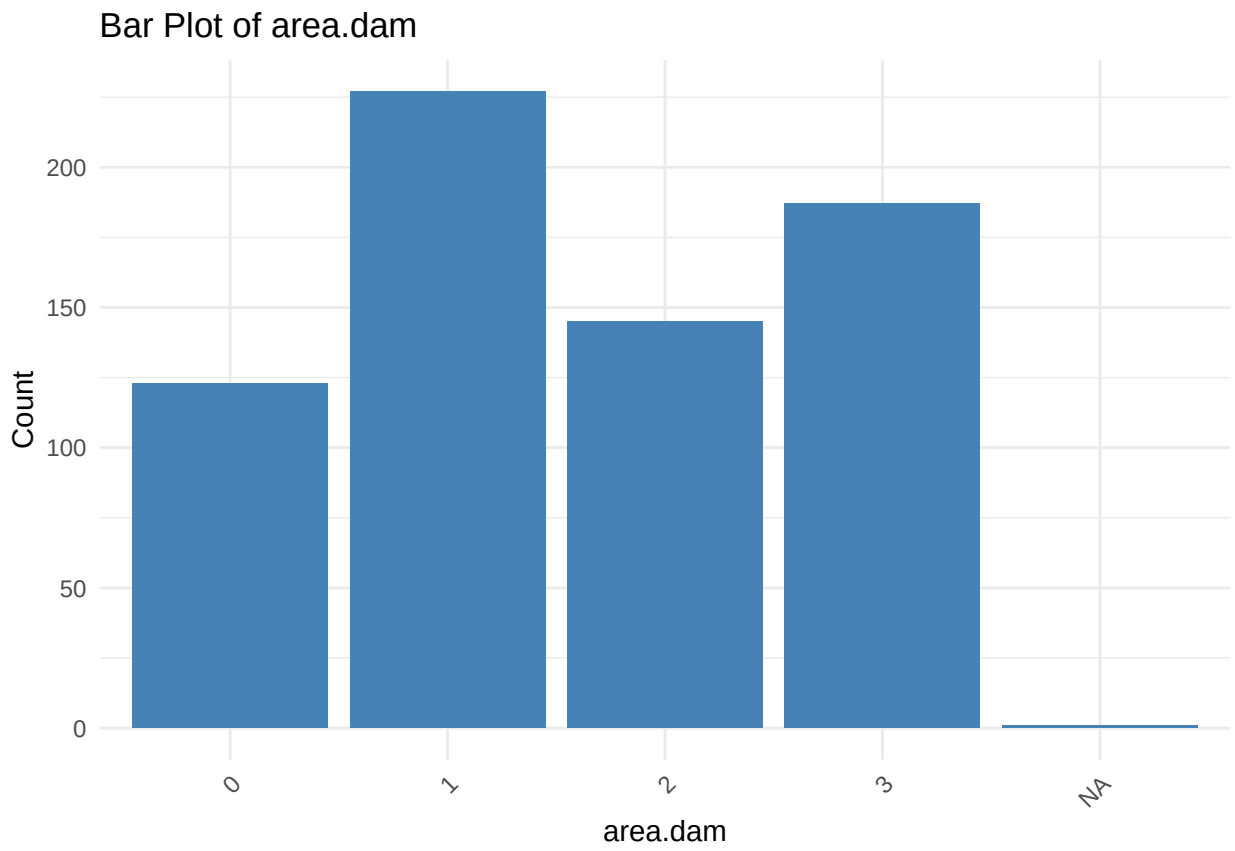

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



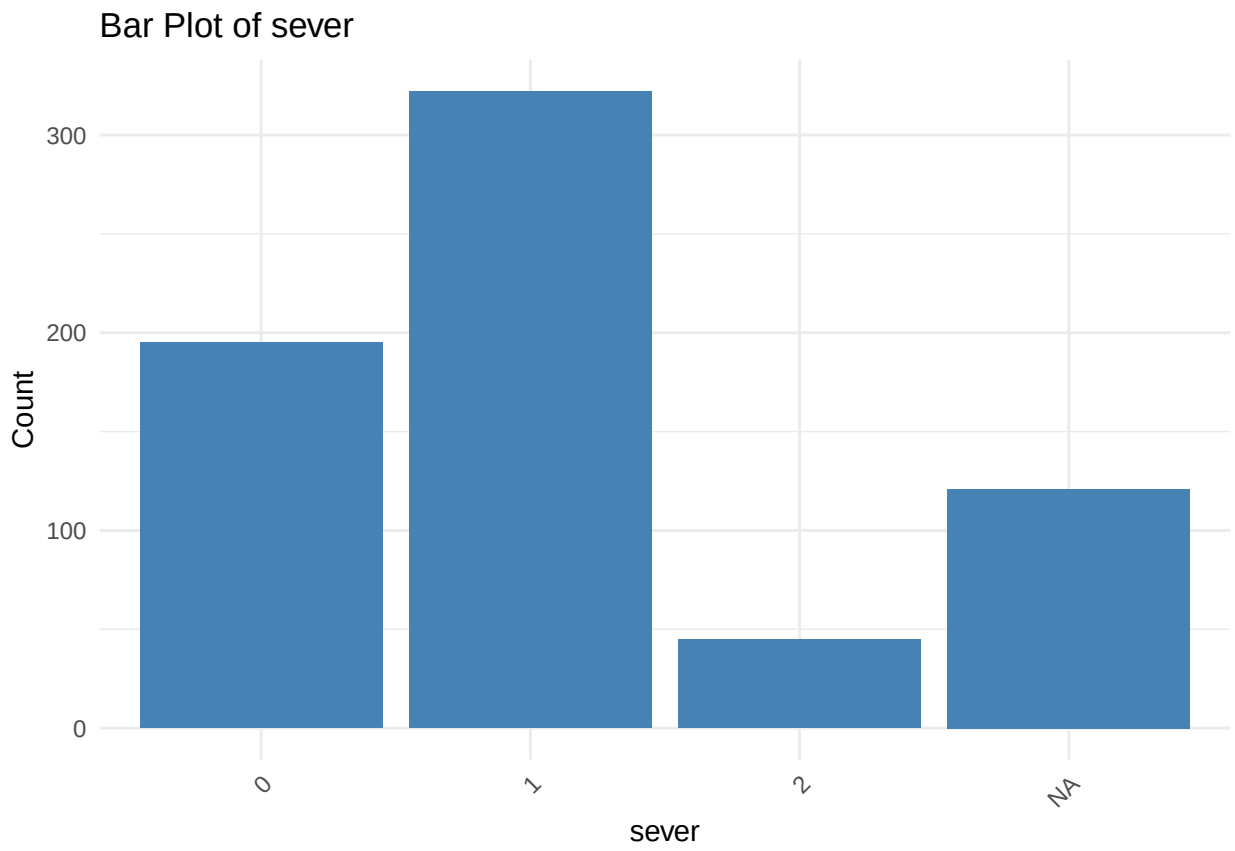

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



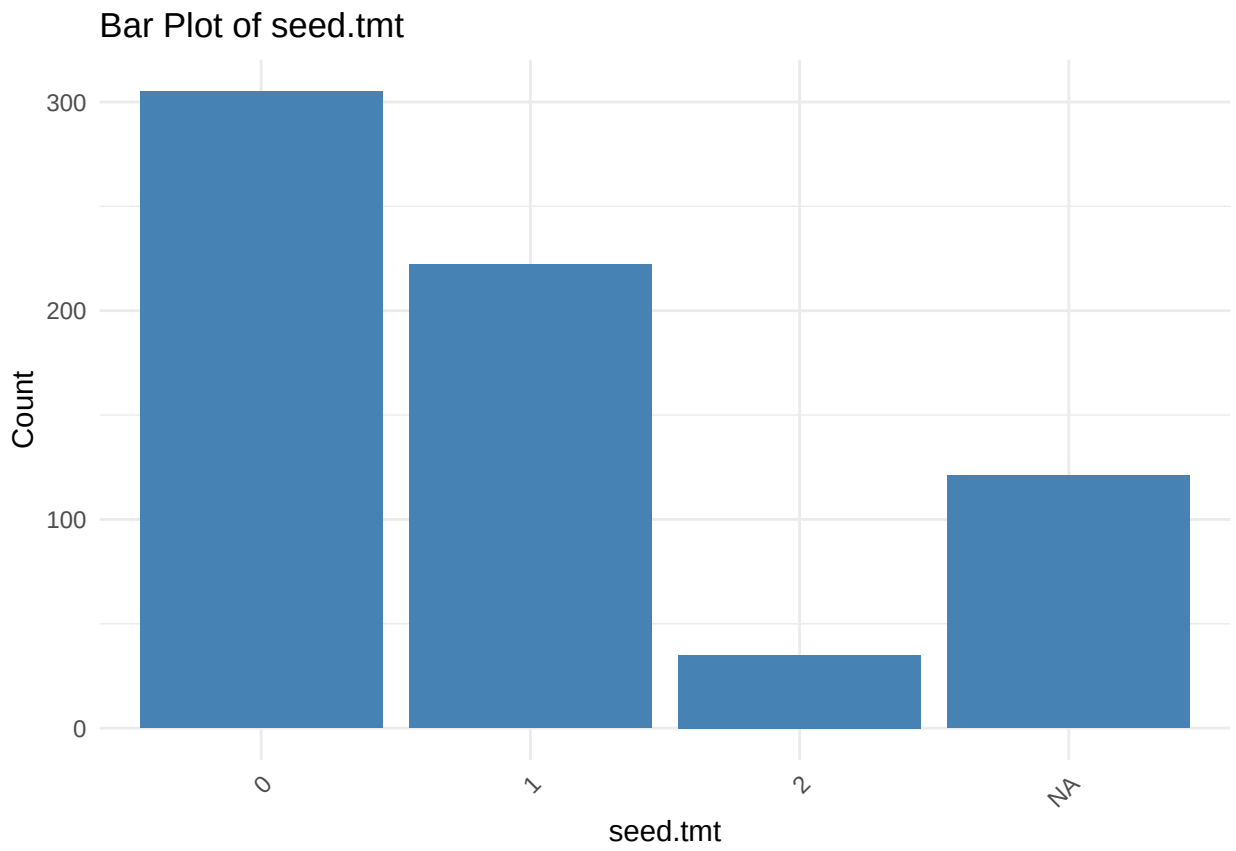

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



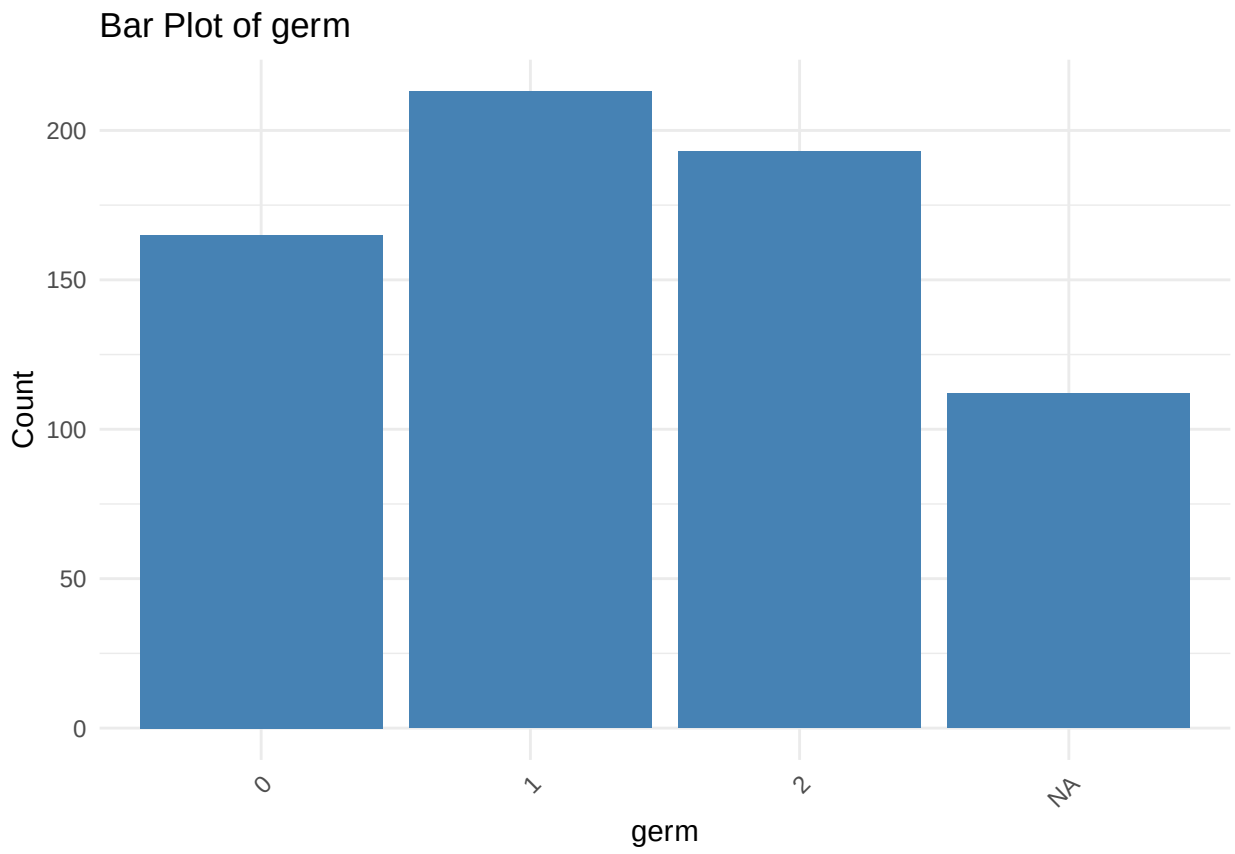

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



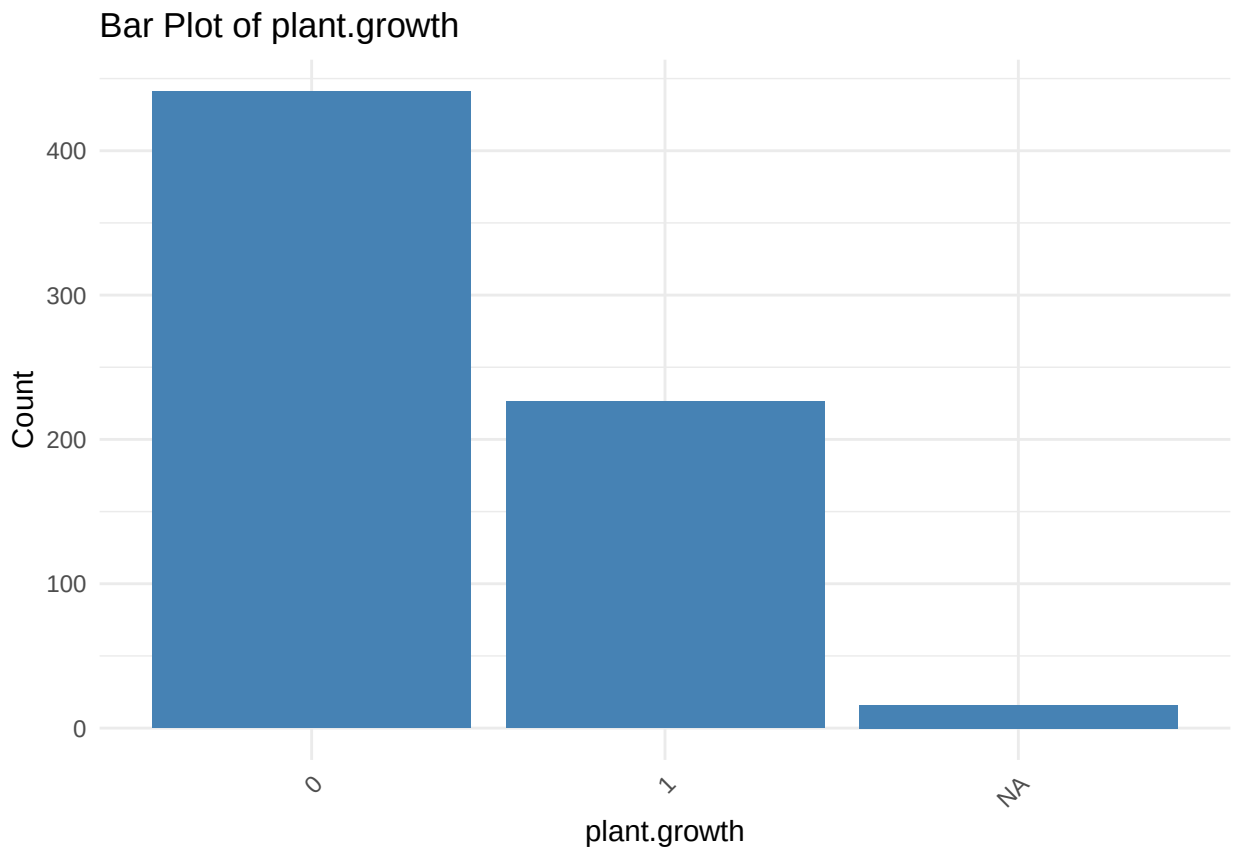

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



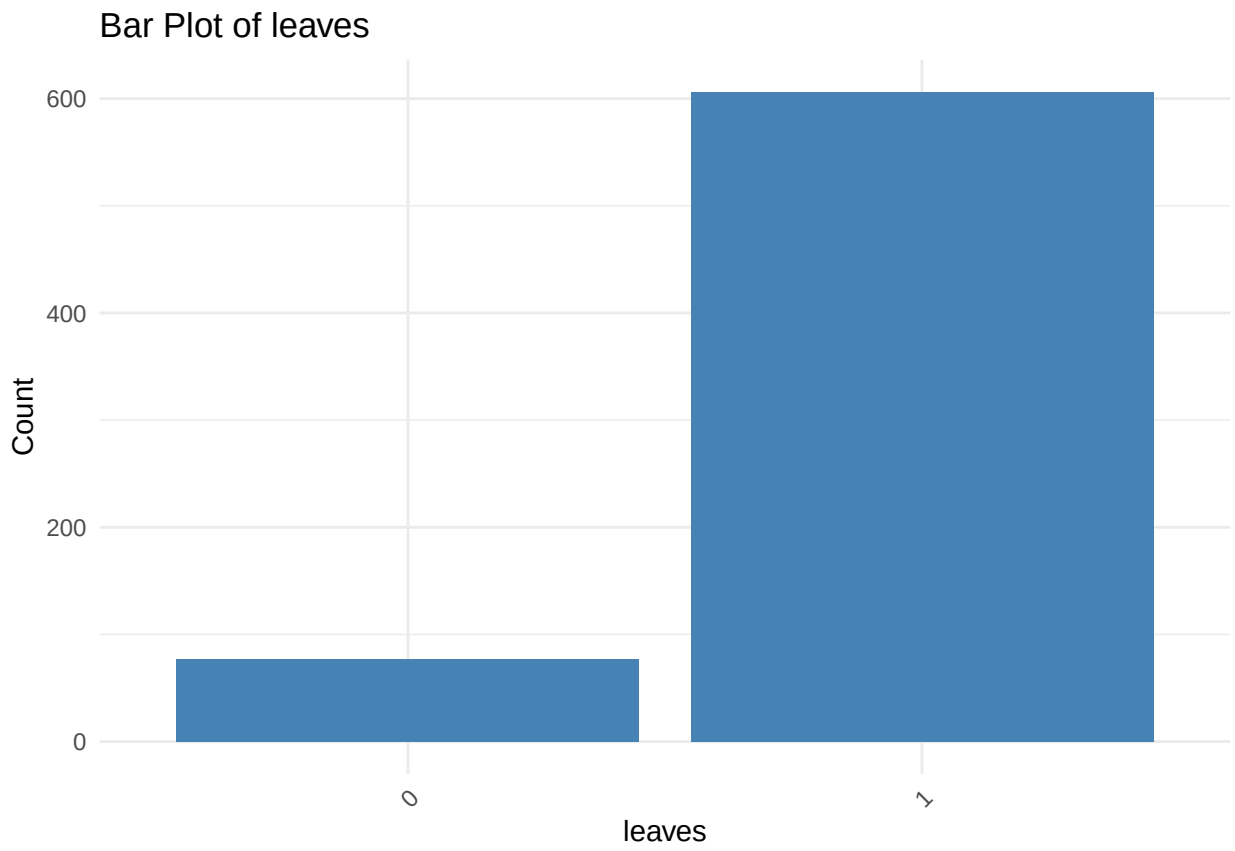

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



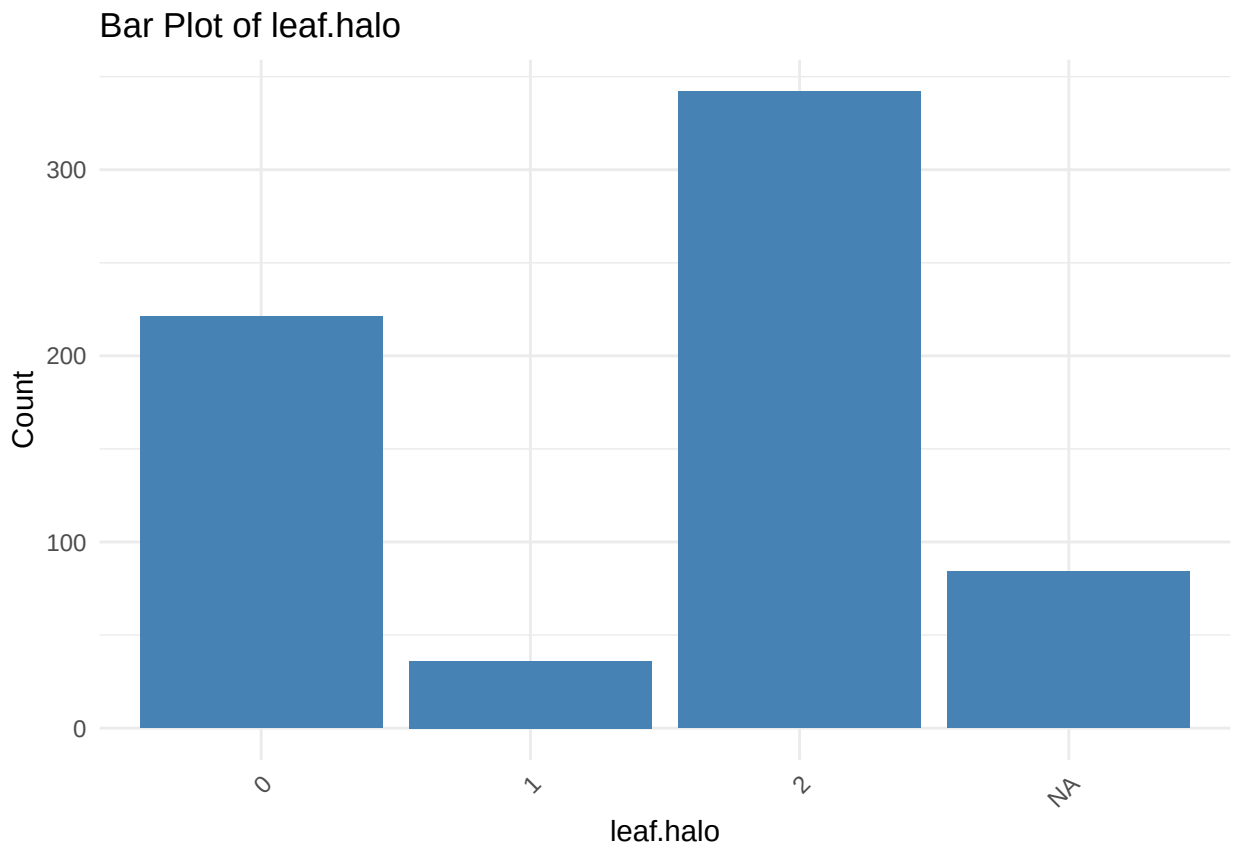

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



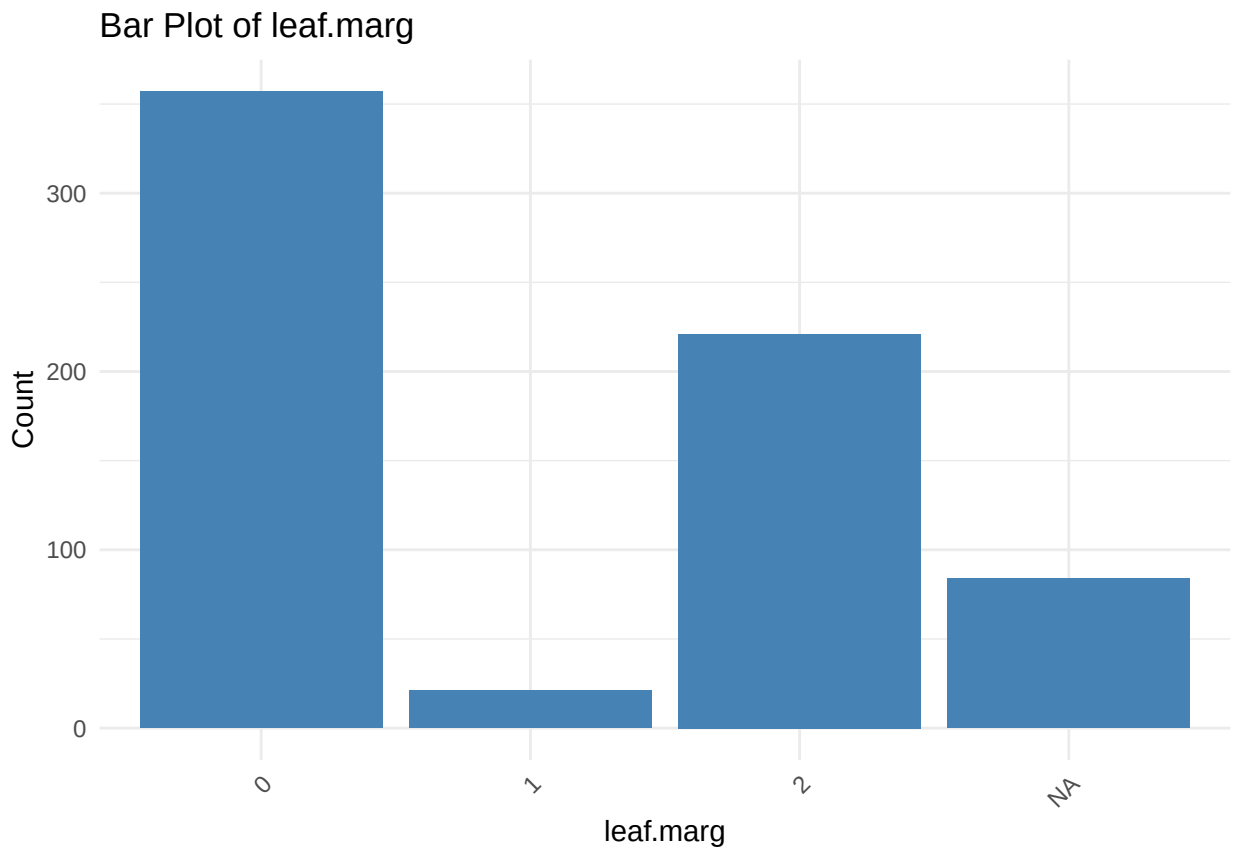

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



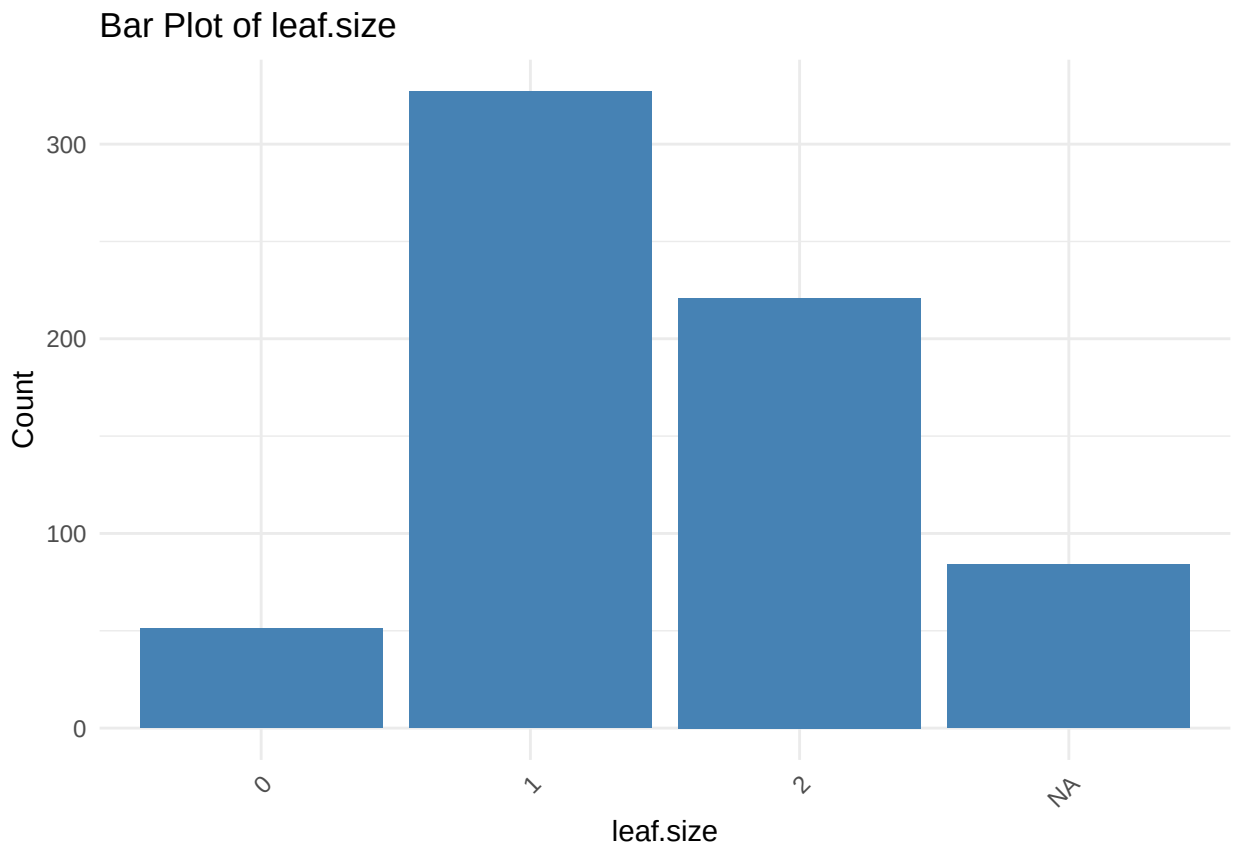

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



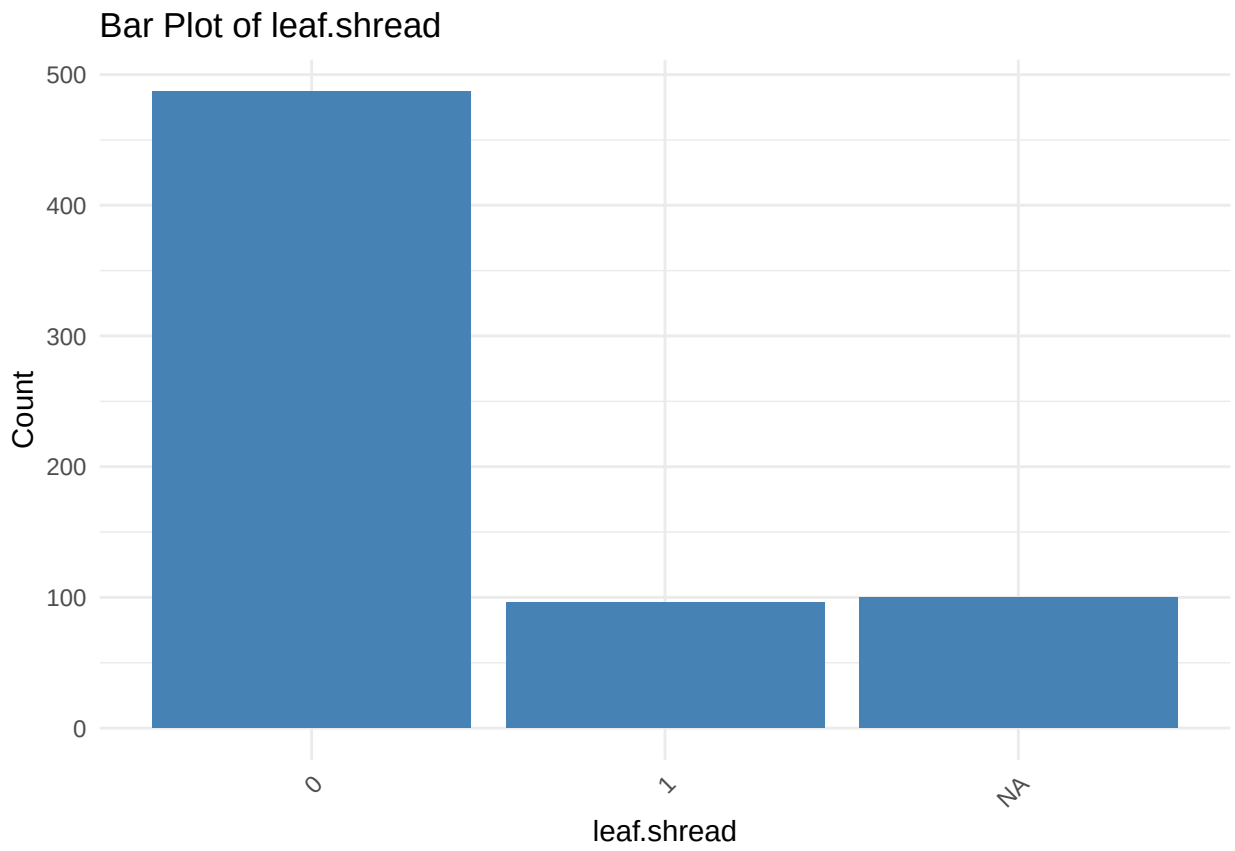

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



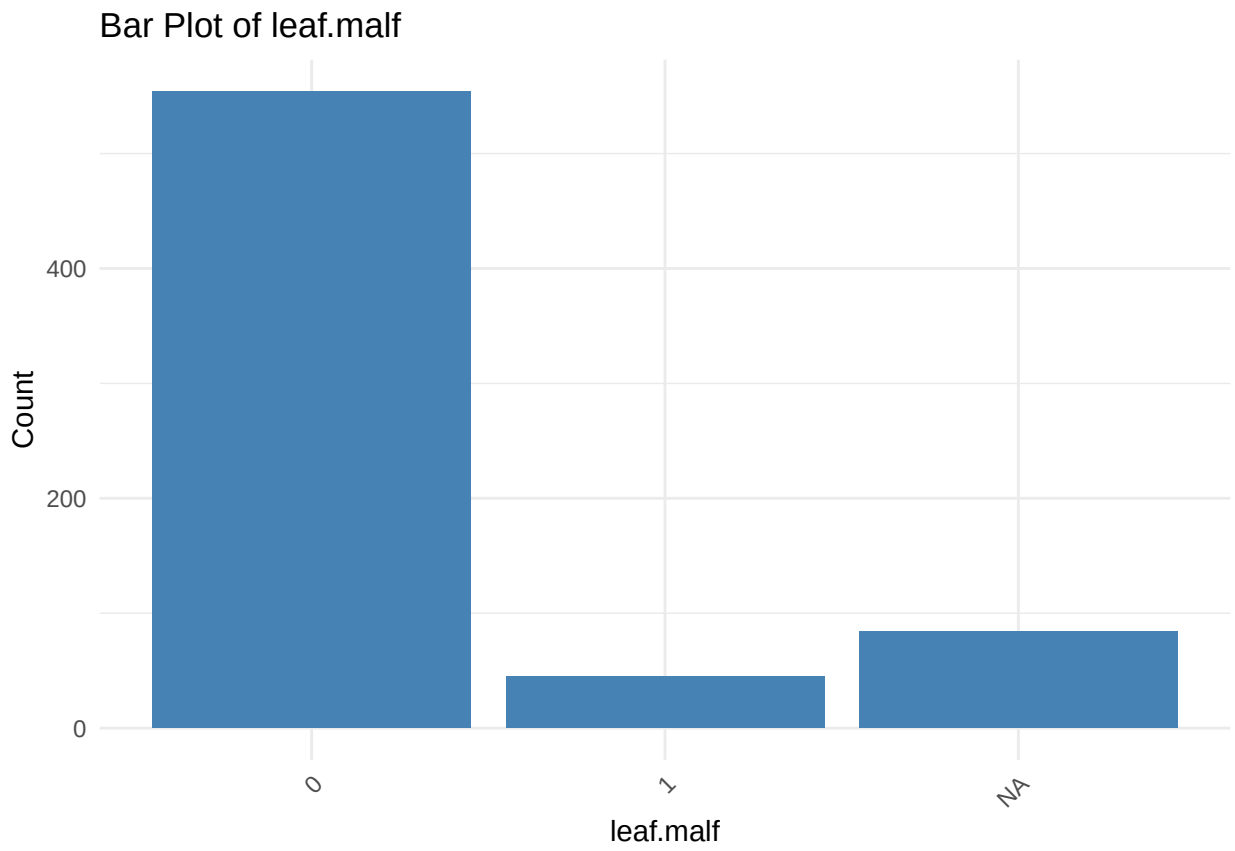

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



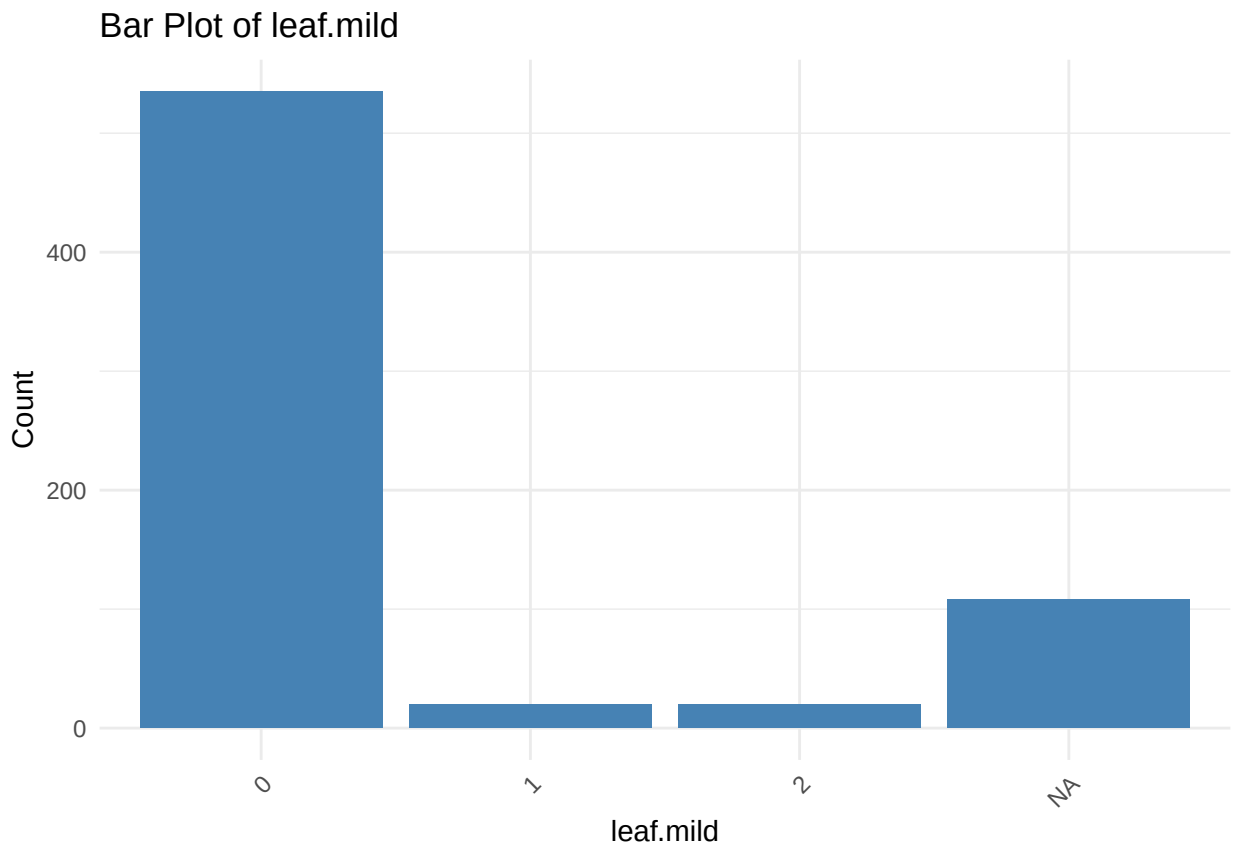

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



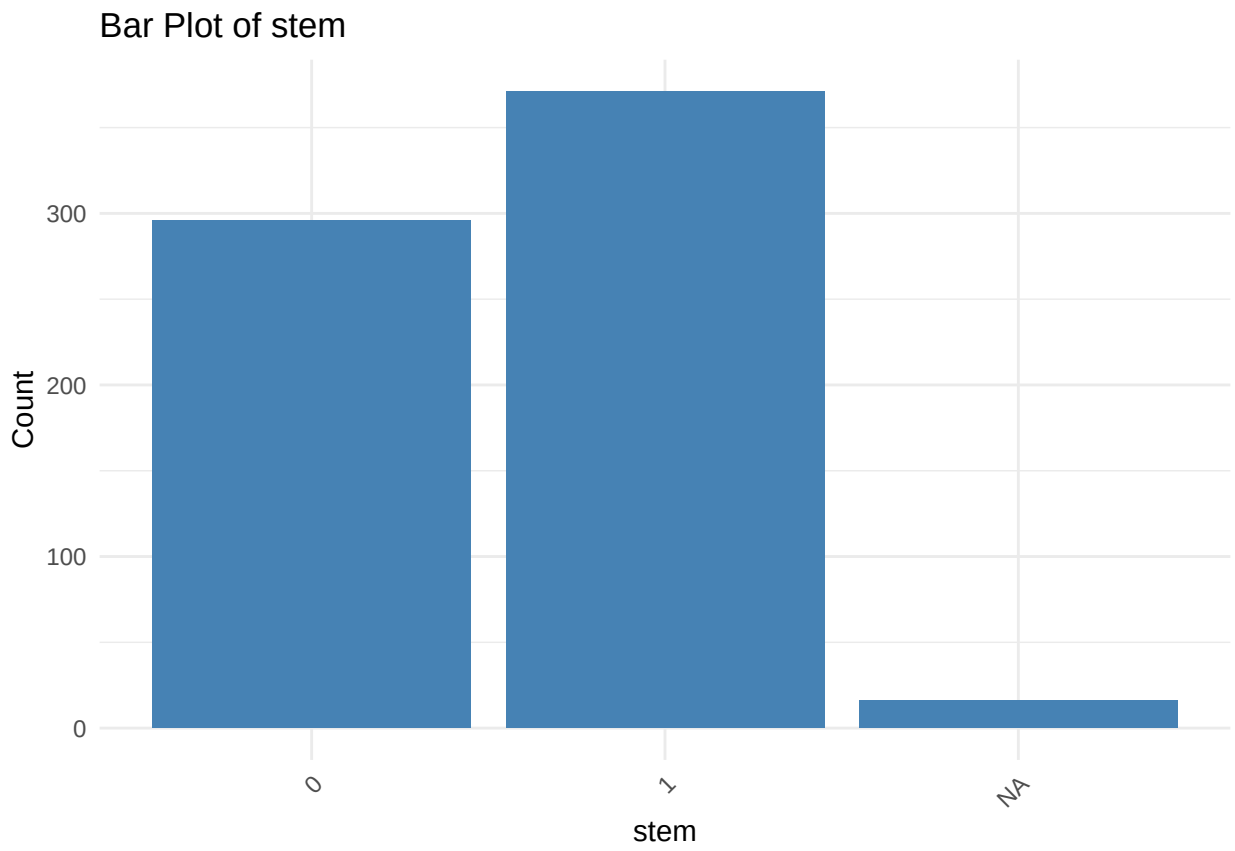

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



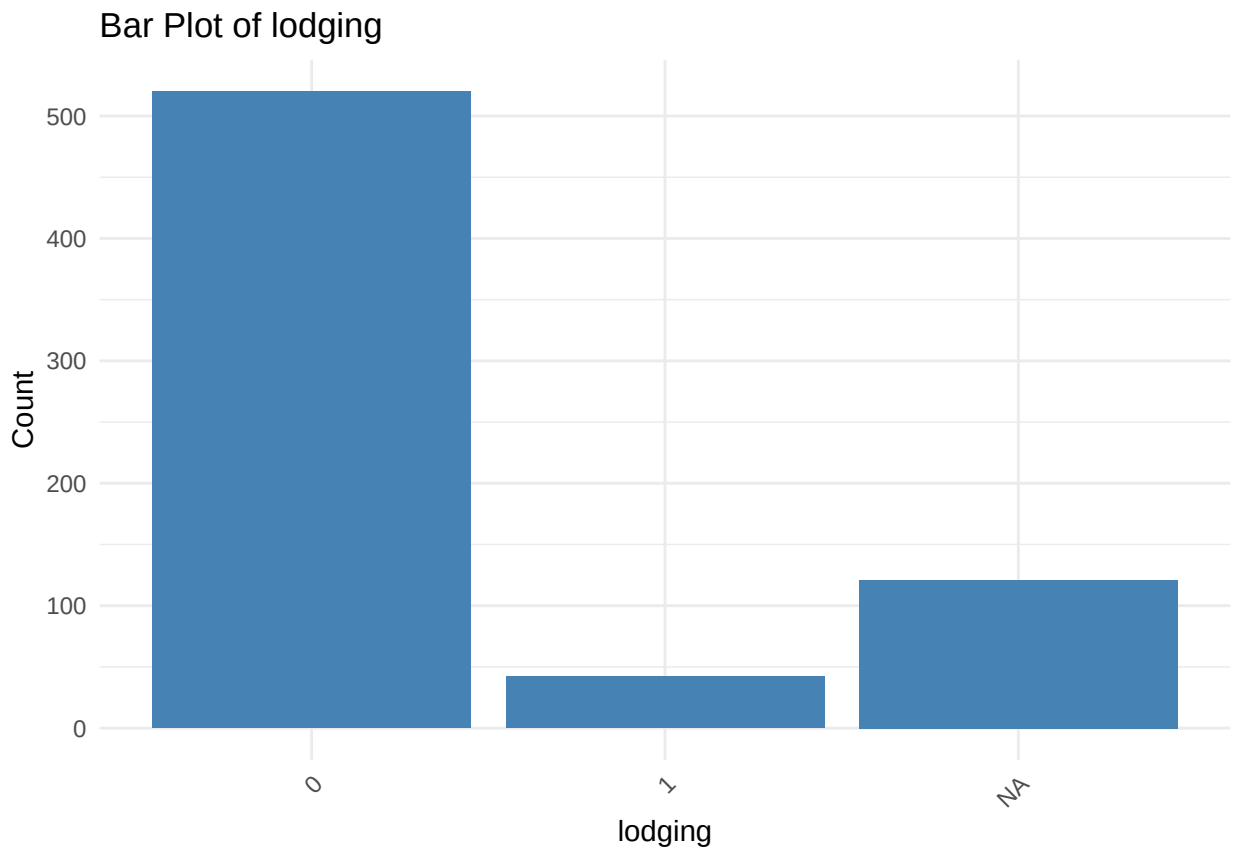

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



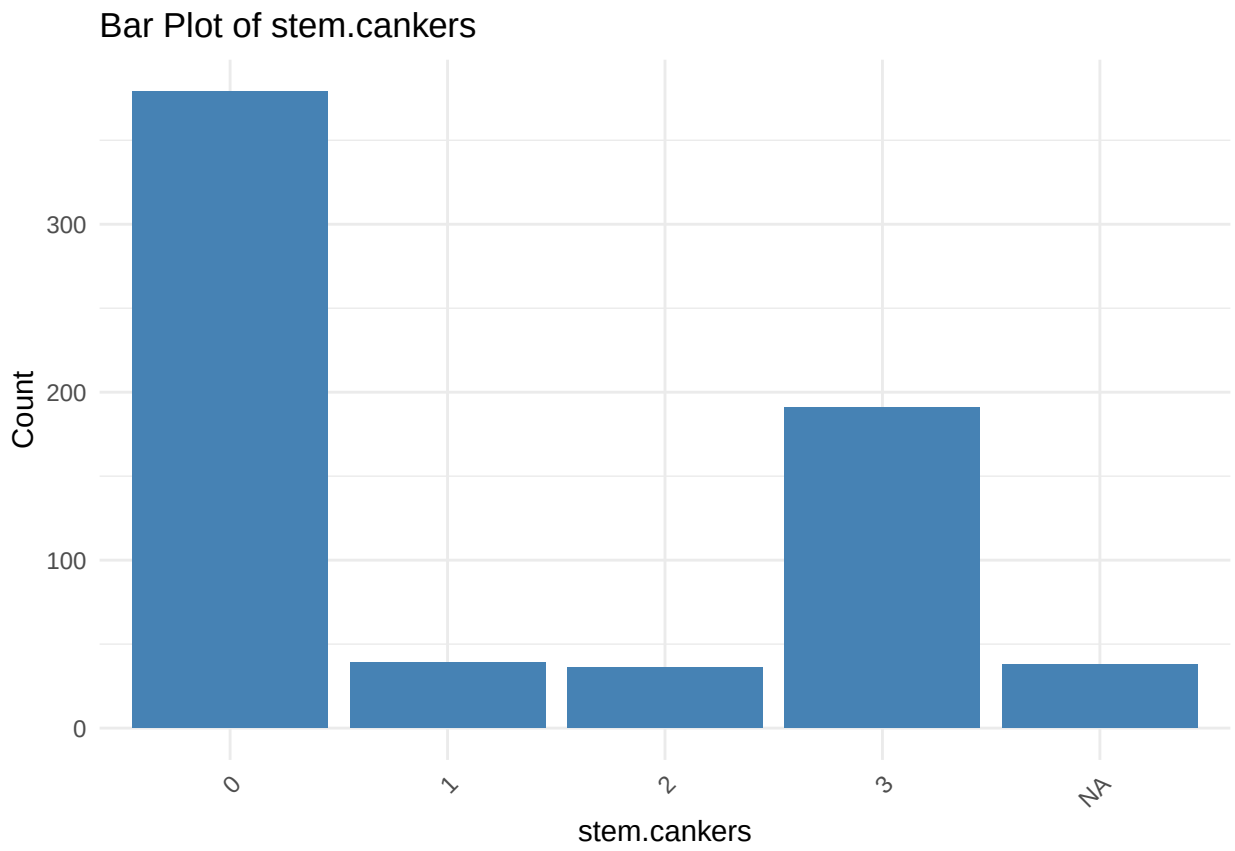

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



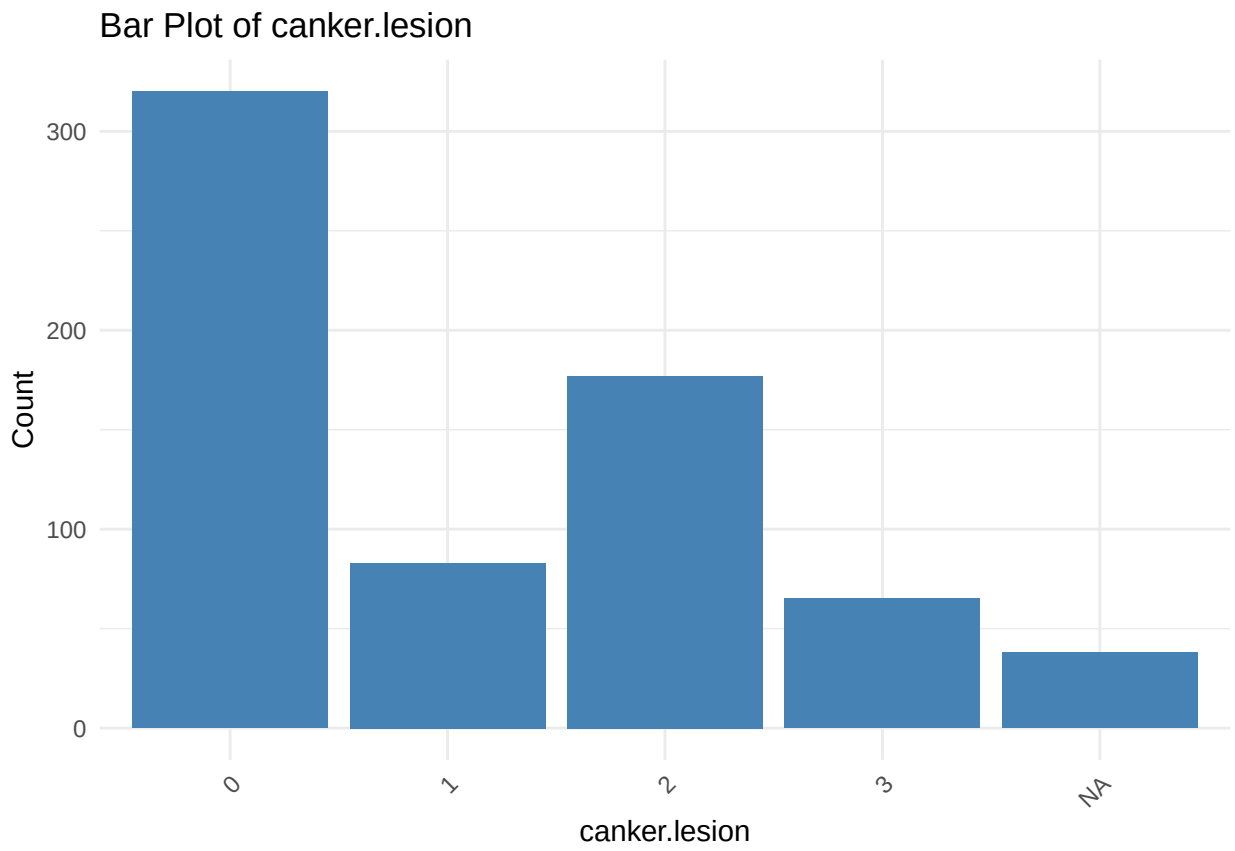

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```




```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```

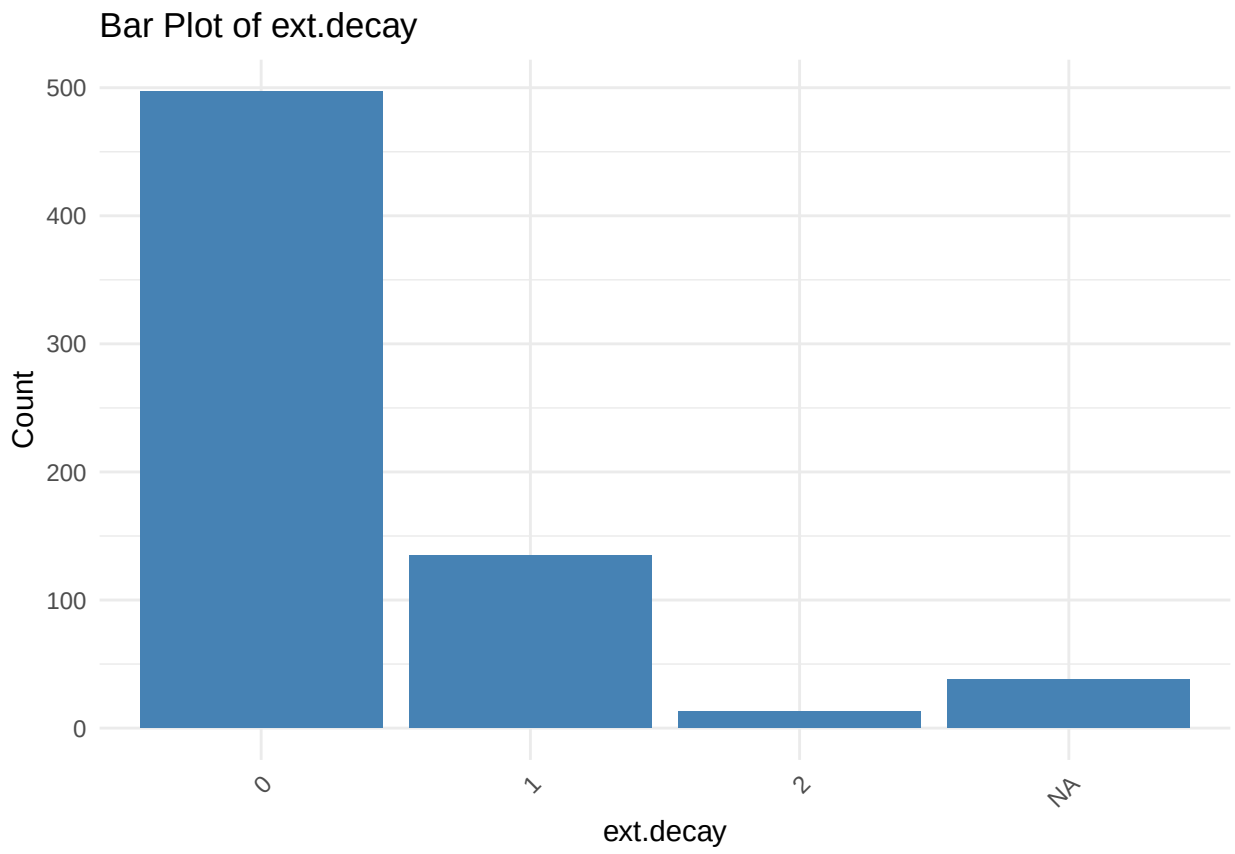



```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```

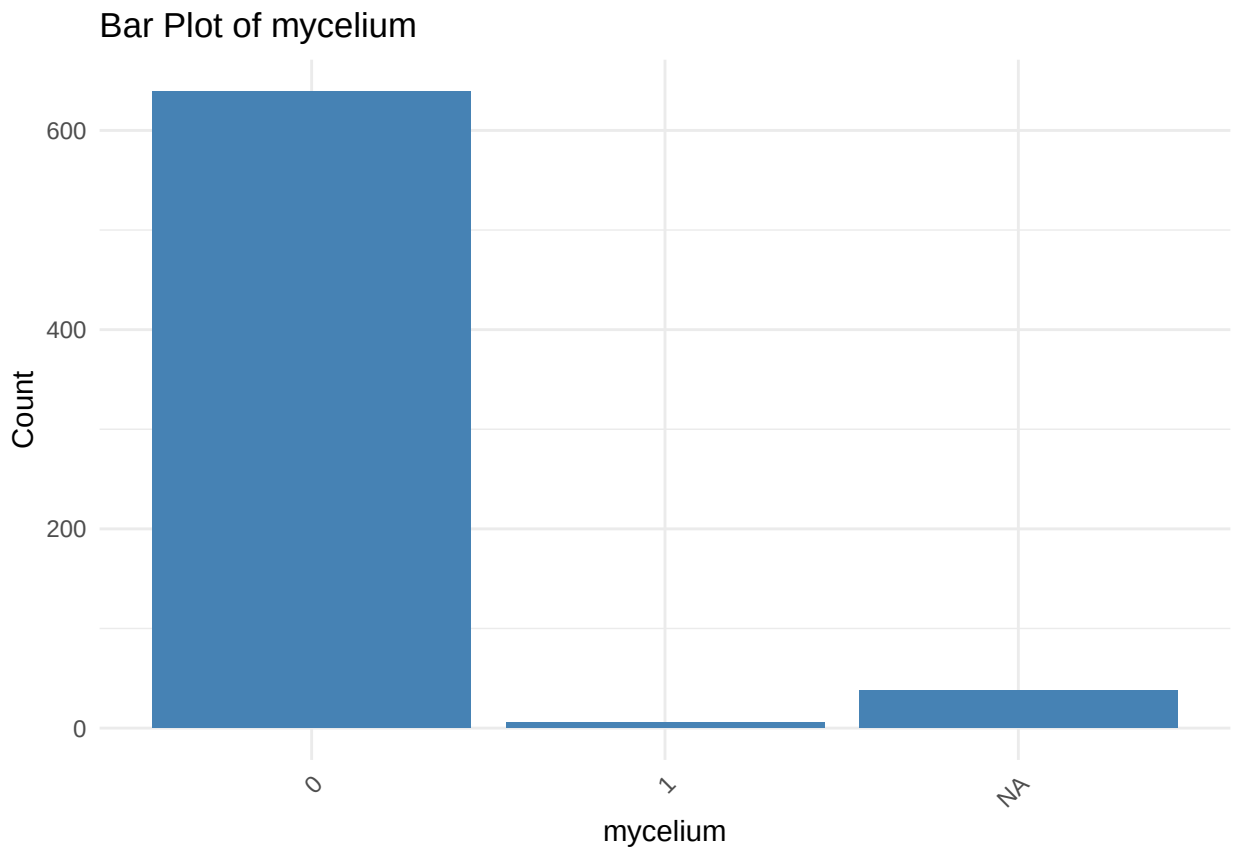



```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```

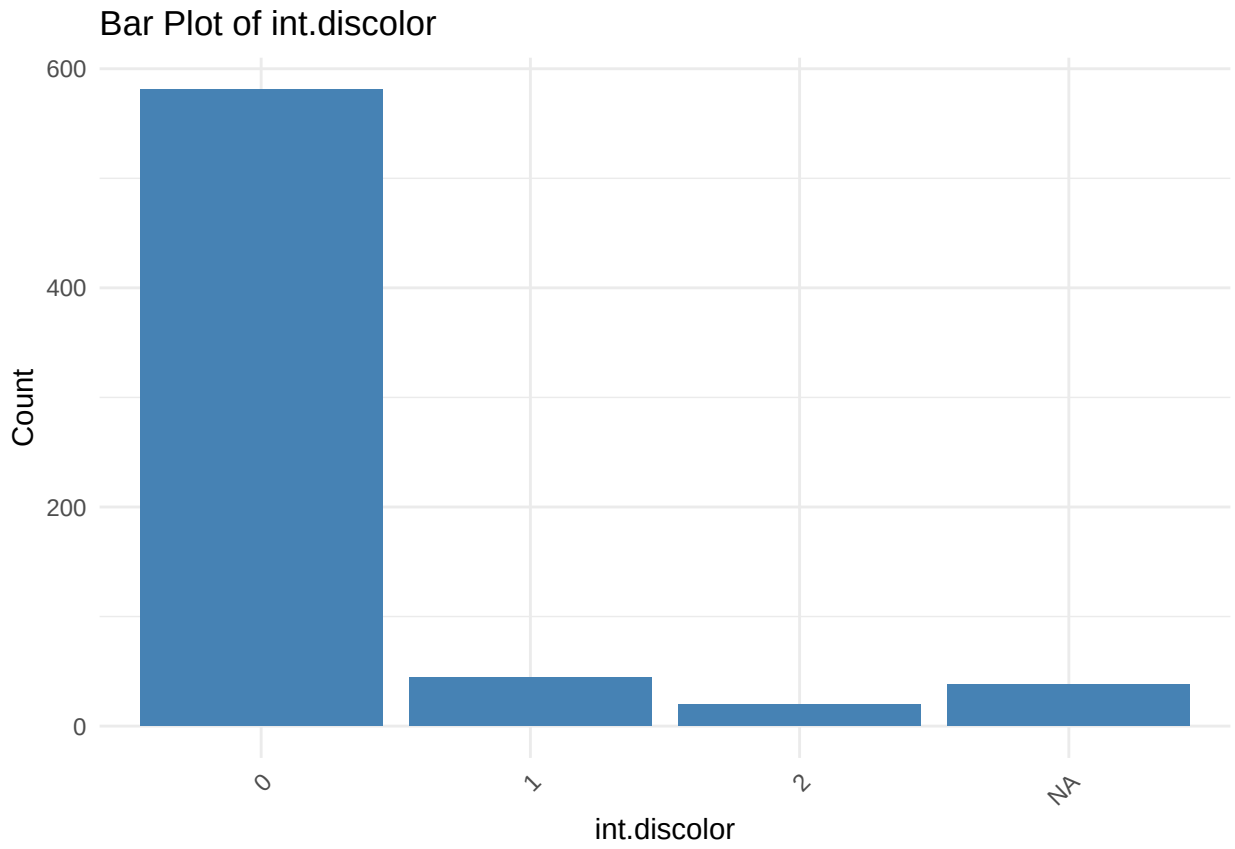

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



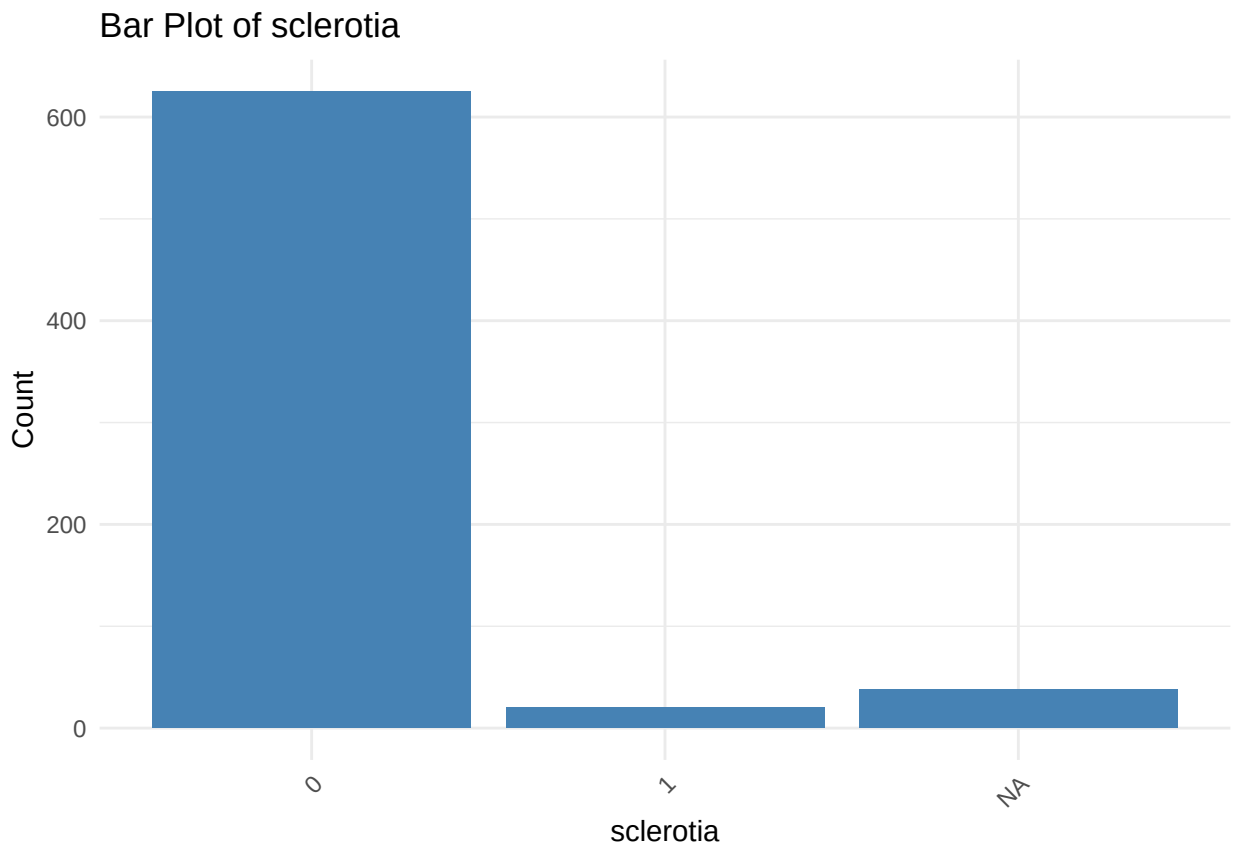

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



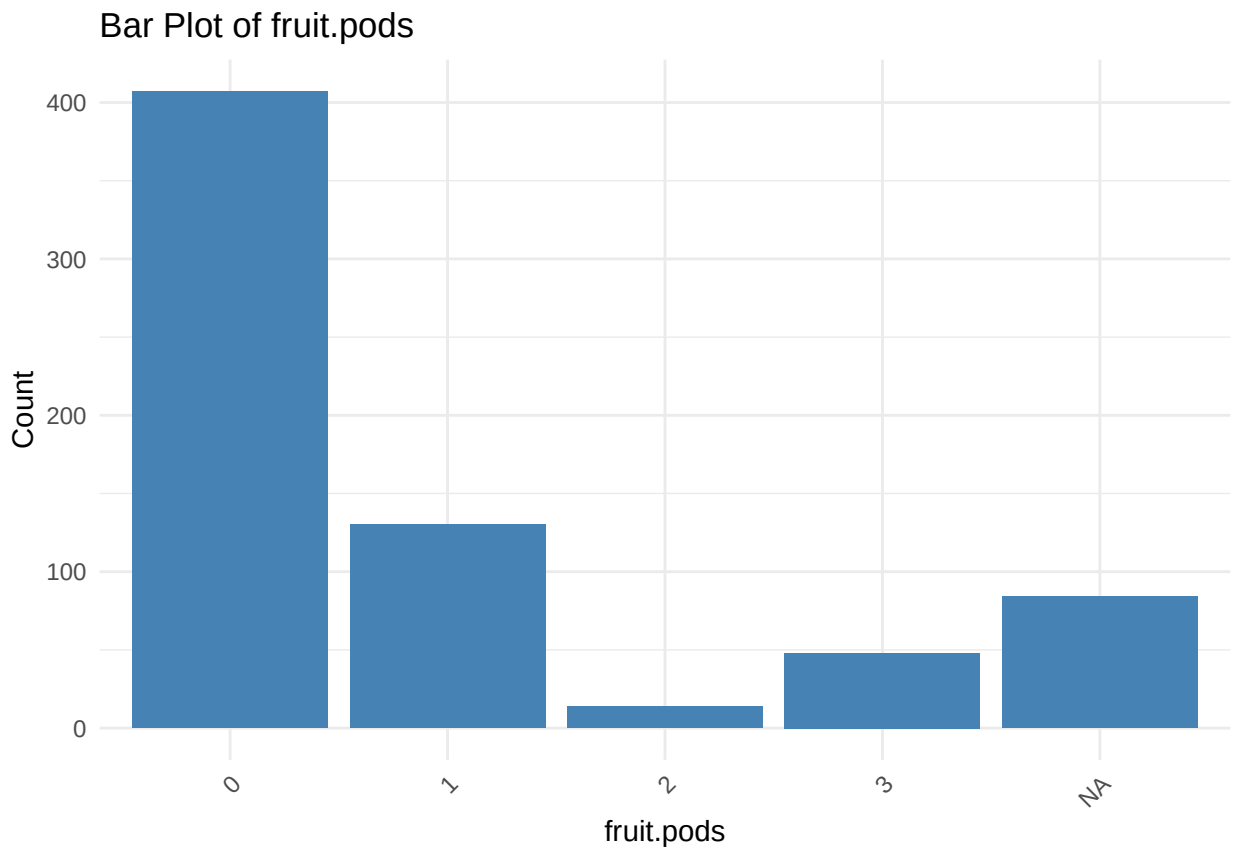

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



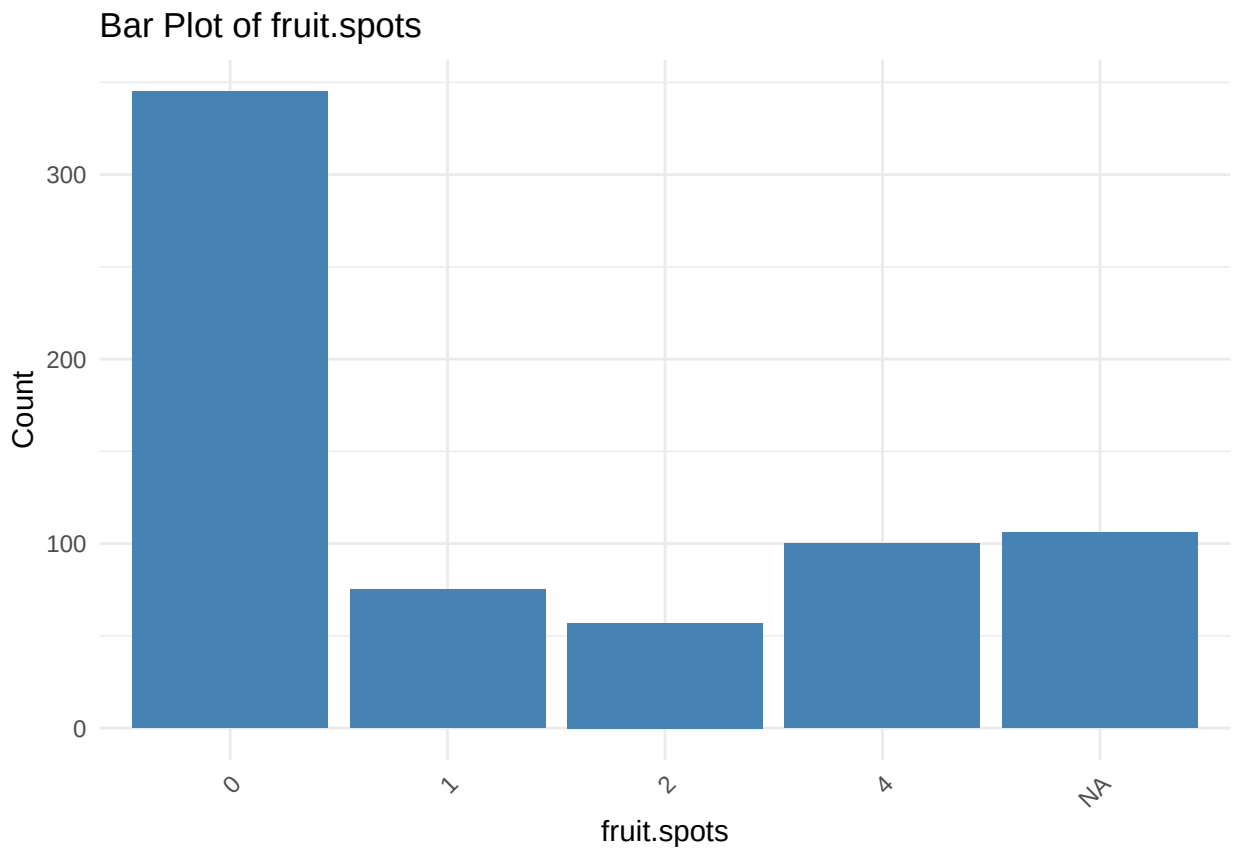

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



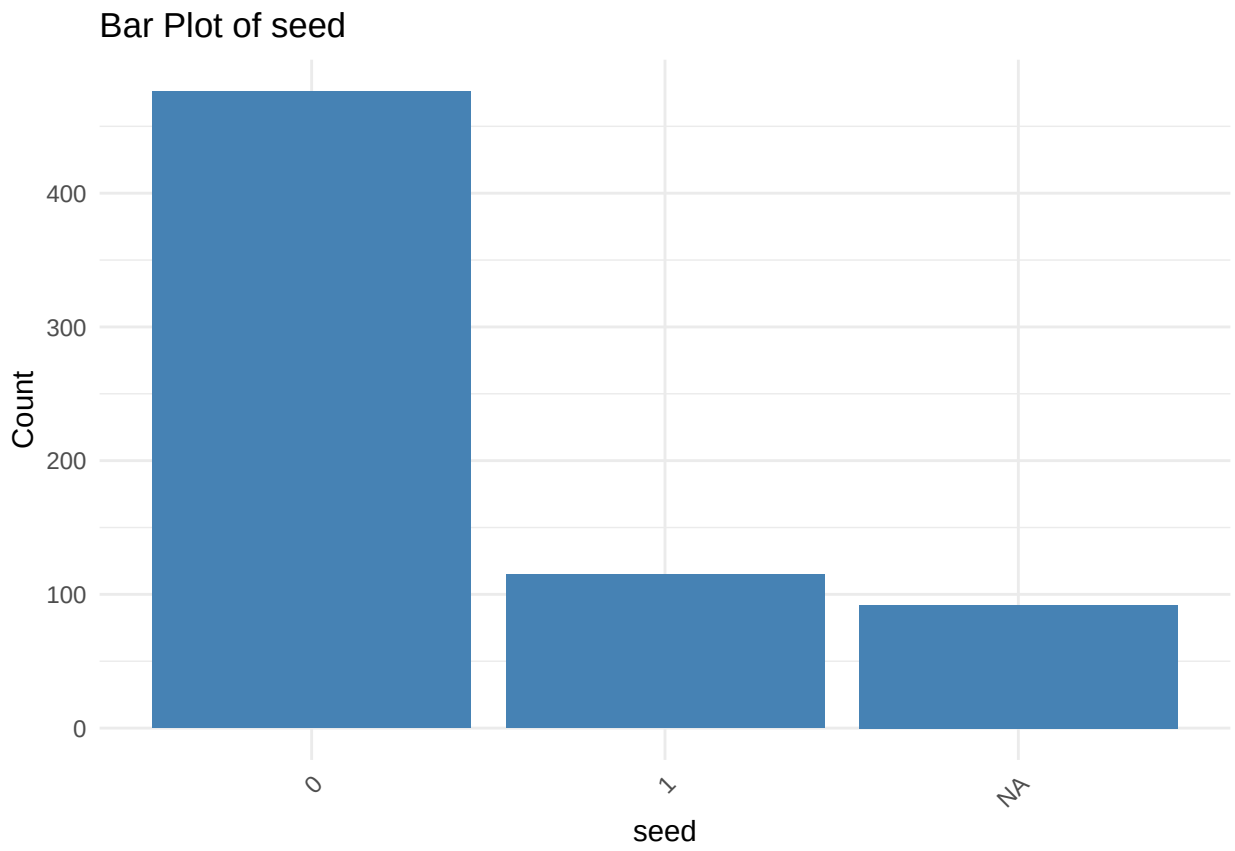

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



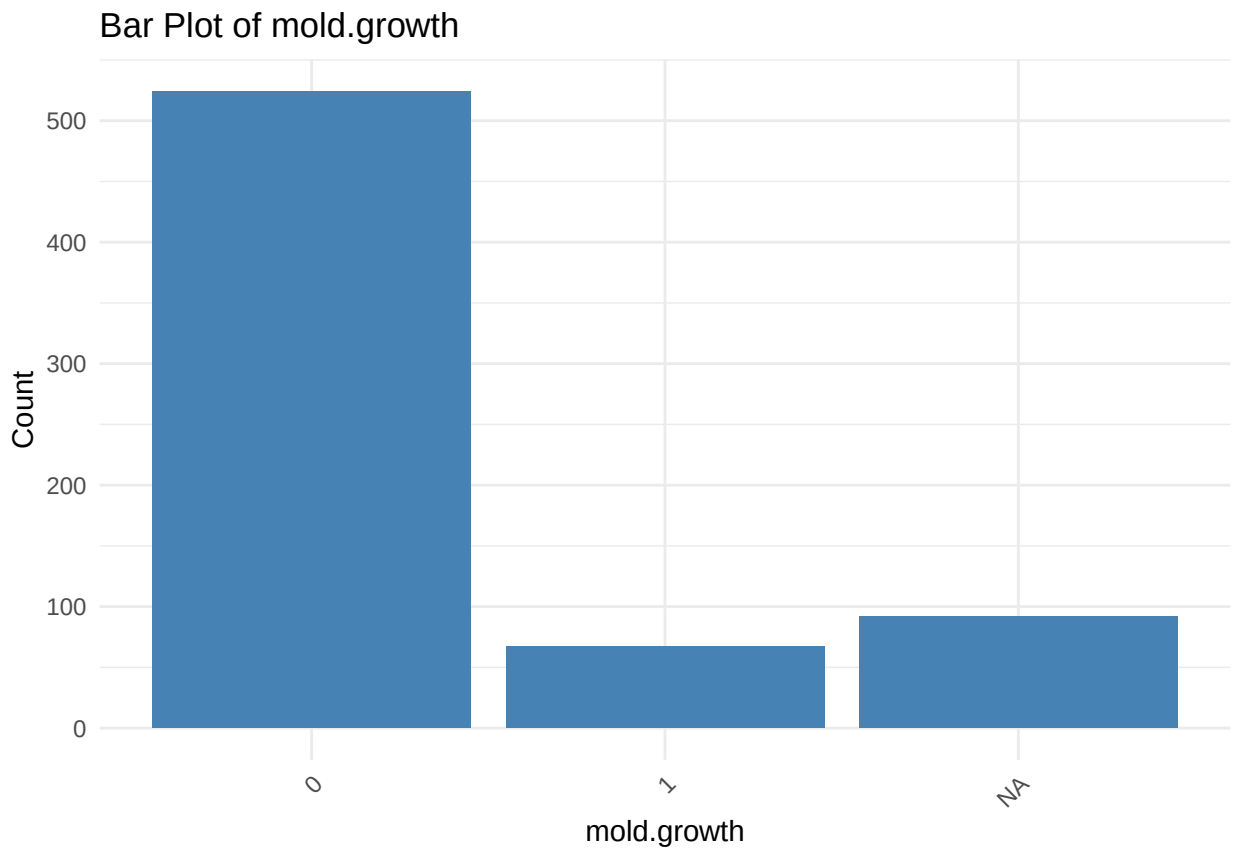

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



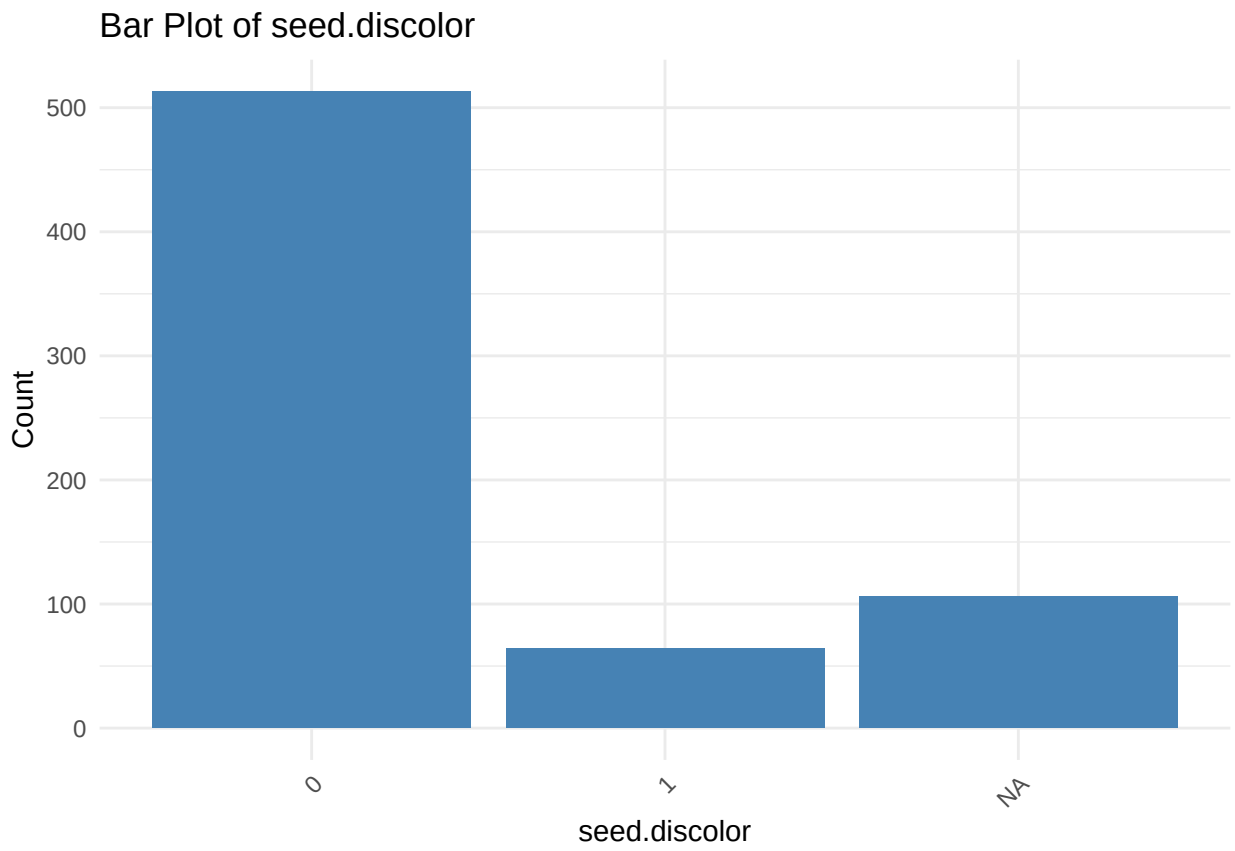

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



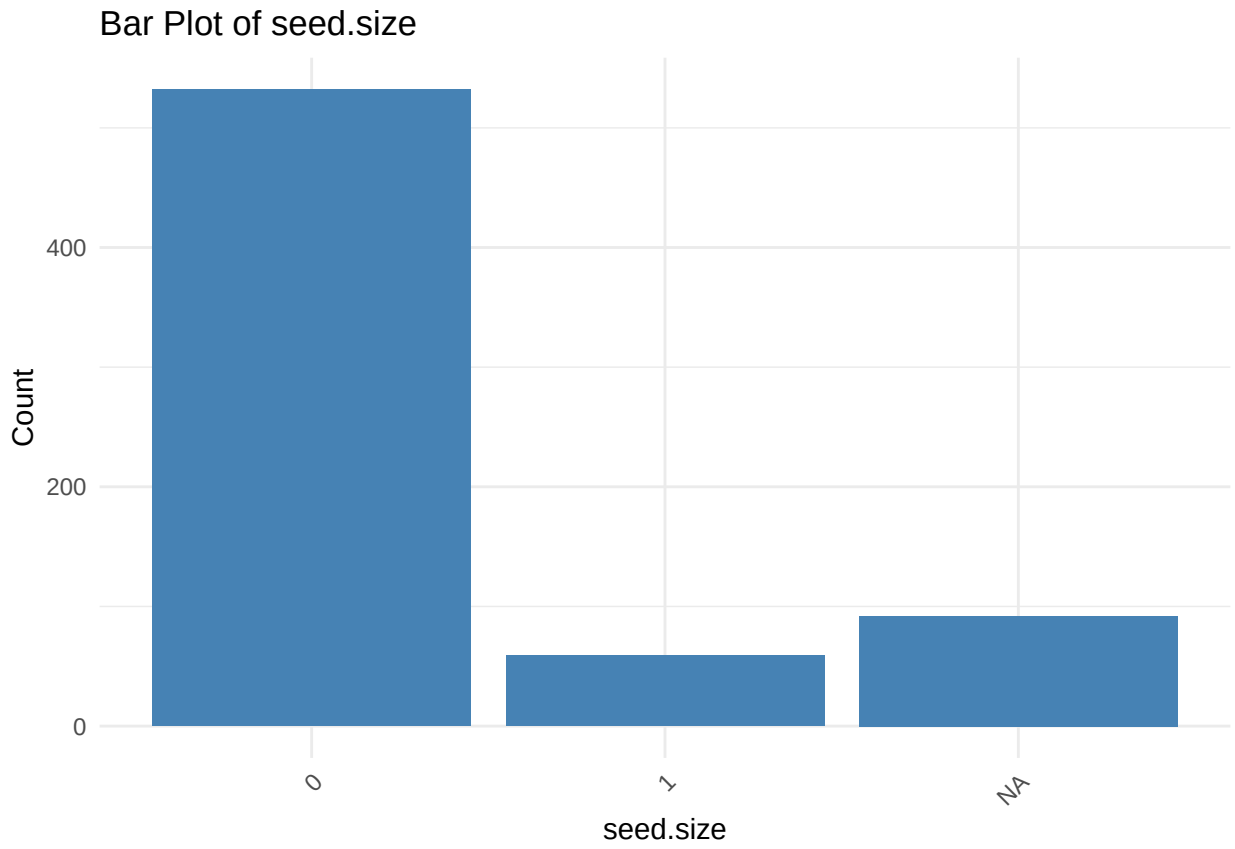

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



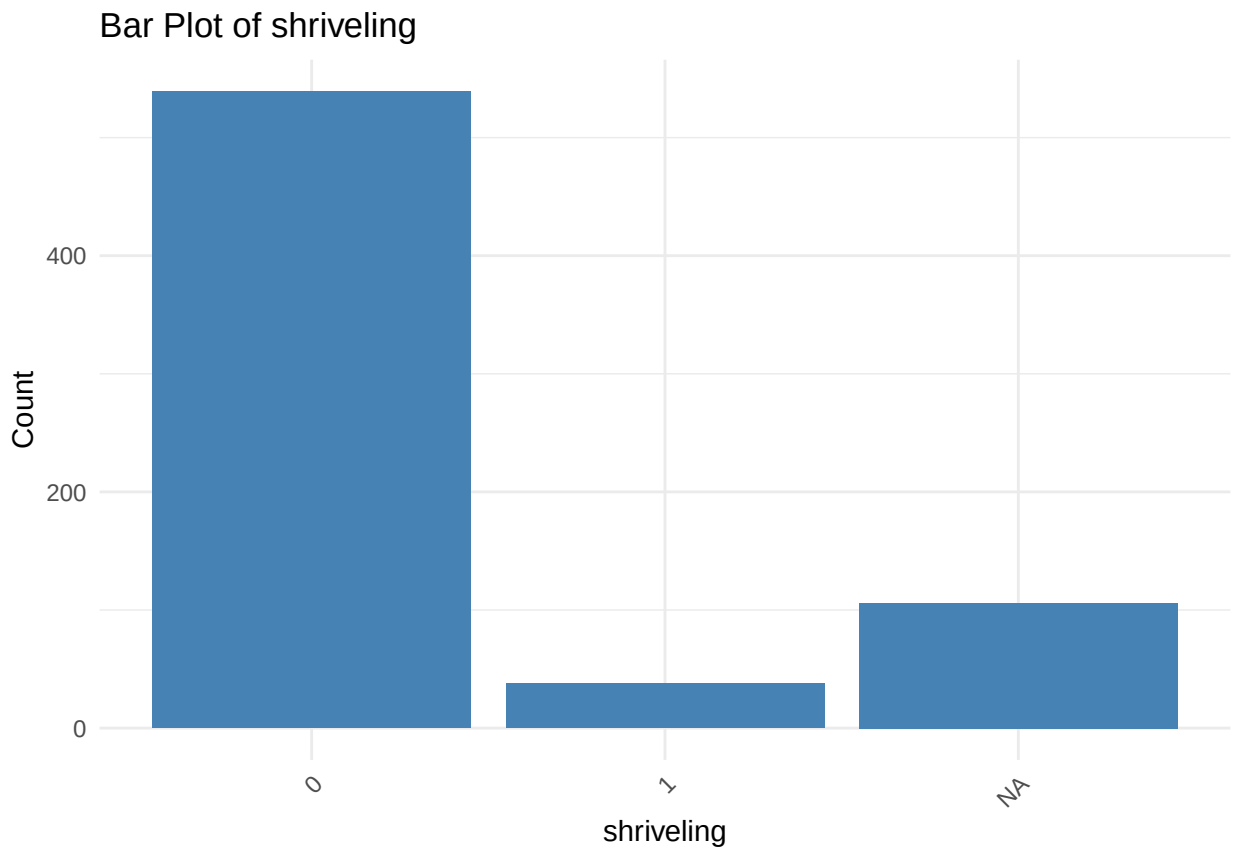

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



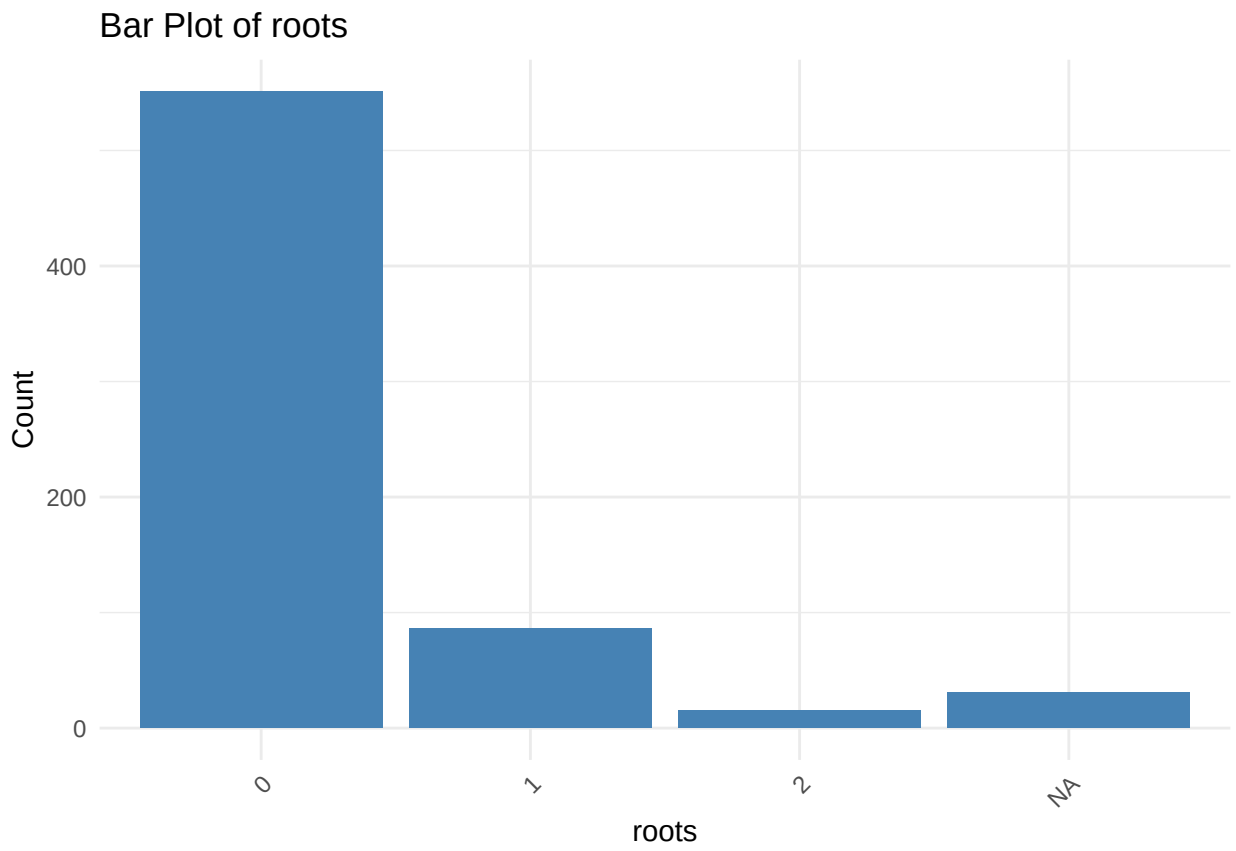

```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```




```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```




```
## Warning: Use of ‘predictors[[predictor]]‘ is discouraged.
## i Use ‘.data[[predictor]]‘ instead.
```



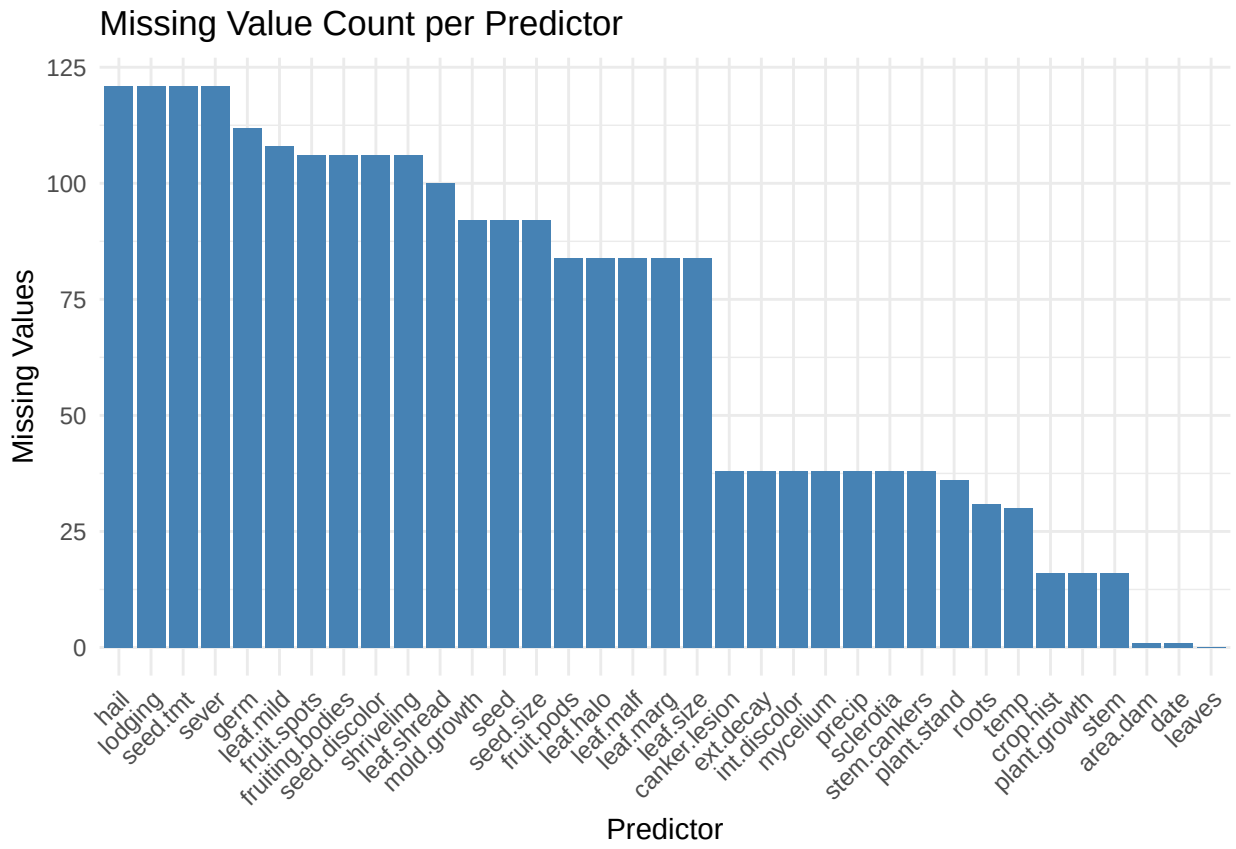

**b. Roughly 18% of the data are missing. Are there particular predictors that are more likely to be missing? Is the pattern of missing data related to the classes?**

- In the plots, we can see there are a lot of missing values. Hail, lodging, seed.tmt, and sever have most missing values.
- I think, yes, based on the structure of the Soybean dataset, the pattern of missing data is related to the class labels.

```
predictors <- Soybean |> select(-Class)

# Count NAs per predictor
missing_df <- data.frame(
  Predictor = names(predictors),
  Missing_Count = sapply(predictors, function(x) sum(is.na(x)))
)

ggplot(missing_df, aes(x = reorder(Predictor, -Missing_Count), y = Missing_Count)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  theme_minimal() +
  labs(title = "Missing Value Count per Predictor",
       x = "Predictor", y = "Missing Values") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
library(naniar)
```

```
## Warning: package 'naniar' was built under R version 4.4.3
```

```
# Count % missing per column
sort(colMeans(is.na(Soybean)), decreasing = TRUE)
```

```
##           hail          sever       seed.tmt        lodging          germ
##    0.177159590    0.177159590    0.177159590    0.177159590    0.163982430
##      leaf.mild fruiting.bodies     fruit.spots seed.discolor     shriveling
##    0.158125915    0.155197657    0.155197657    0.155197657    0.155197657
##    leaf.shread           seed    mold.growth       seed.size      leaf.halo
##    0.146412884    0.134699854    0.134699854    0.134699854    0.122986823
##      leaf.marg      leaf.size      leaf.malf      fruit.pods         precip
##    0.122986823    0.122986823    0.122986823    0.122986823    0.055636896
##   stem.cankers   canker.lesion      ext.decay       mycelium   int.discolor
##    0.055636896    0.055636896    0.055636896    0.055636896    0.055636896
##      sclerotia     plant.stand          roots           temp      crop.hist
##    0.055636896    0.052708638    0.045387994    0.043923865    0.023426061
##   plant.growth           stem           date       area.dam          Class
##    0.023426061    0.023426061    0.001464129    0.001464129    0.000000000
##         leaves
##    0.000000000
```

```
vis_miss(Soybean, sort_miss = TRUE)
```