SUPPORT VECTOR MACHINE CLASSIFICATION

Contents

1 Loading libraries and required data

```
 setwd("C:/Users/97156/OneDrive/Desktop/SVM")
# Loading files and libraries
options(repos = c(CRAN = "https://cloud.r-project.org"))
install.packages("bookdown")
```

```
## package 'bookdown'  successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\97156\AppData\Local\Temp\RtmpUH3WUP\downloaded_packages
```

```
library(ggplot2)
library(dplyr)
library(e1071)
library(caret)
library(pROC)
library(gridExtra)
library(bookdown)

# Importing the dataset
class_dataset <- read.csv("class.csv")
zoo_dataset <- read.csv("zoo.csv")

#Viewing the dataset
head(class_dataset)
```

```
##   Class_Number Number_Of_Animal_Species_In_Class Class_Type
## 1            1                               41     Mammal
## 2            2                               20       Bird
## 3            3                                5    Reptile
## 4            4                               13       Fish
## 5            5                                4  Amphibian
## 6            6                                8        Bug
##
## 1 aardvark, antelope, bear, boar, buffalo, calf, cavy, cheetah, deer, dolphin, elephant, fruitbat,
## 2
## 3
## 4
## 5
## 6
```

```
head(zoo_dataset)
```

```
##   animal_name hair feathers eggs milk airborne aquatic predator toothed
## 1    aardvark    1        0    0    1        0       0        1       1
## 2    antelope    1        0    0    1        0       0        0       1
## 3        bass    0        0    1    0        0       1        1       1
## 4        bear    1        0    0    1        0       0        1       1
## 5        boar    1        0    0    1        0       0        1       1
## 6     buffalo    1        0    0    1        0       0        0       1
```

```
##   backbone breathes venomous fins legs tail domestic catsize class_type
## 1        1        1        0    0    4    0        0       1          1
## 2        1        1        0    0    4    1        0       1          1
## 3        1        0        0    1    0    1        0       0          4
## 4        1        1        0    0    4    0        0       1          1
## 5        1        1        0    0    4    1        0       1          1
## 6        1        1        0    0    4    1        0       1          1
```

```
# Renaming the columns of class_data set and sub-setting with relevant columns
colnames(class_dataset) <- c("class_type",
                             "no_of_animal_species_in_class",
                             "class_name",
                             "animal_name"
)
class_dataset <- select(class_dataset,
                        class_type,
                        class_name)
class_dataset <- class_dataset[, c("class_type", "class_name")]

# Displaying the head of the modified dataset
head(zoo_dataset, 3)
```

```
##   animal_name hair feathers eggs milk airborne aquatic predator toothed
## 1    aardvark    1        0    0    1        0       0        1       1
## 2    antelope    1        0    0    1        0       0        0       1
## 3        bass    0        0    1    0        0       1        1       1
##   backbone breathes venomous fins legs tail domestic catsize class_type
## 1        1        1        0    0    4    0        0       1          1
## 2        1        1        0    0    4    1        0       1          1
## 3        1        0        0    1    0    1        0       0          4
```

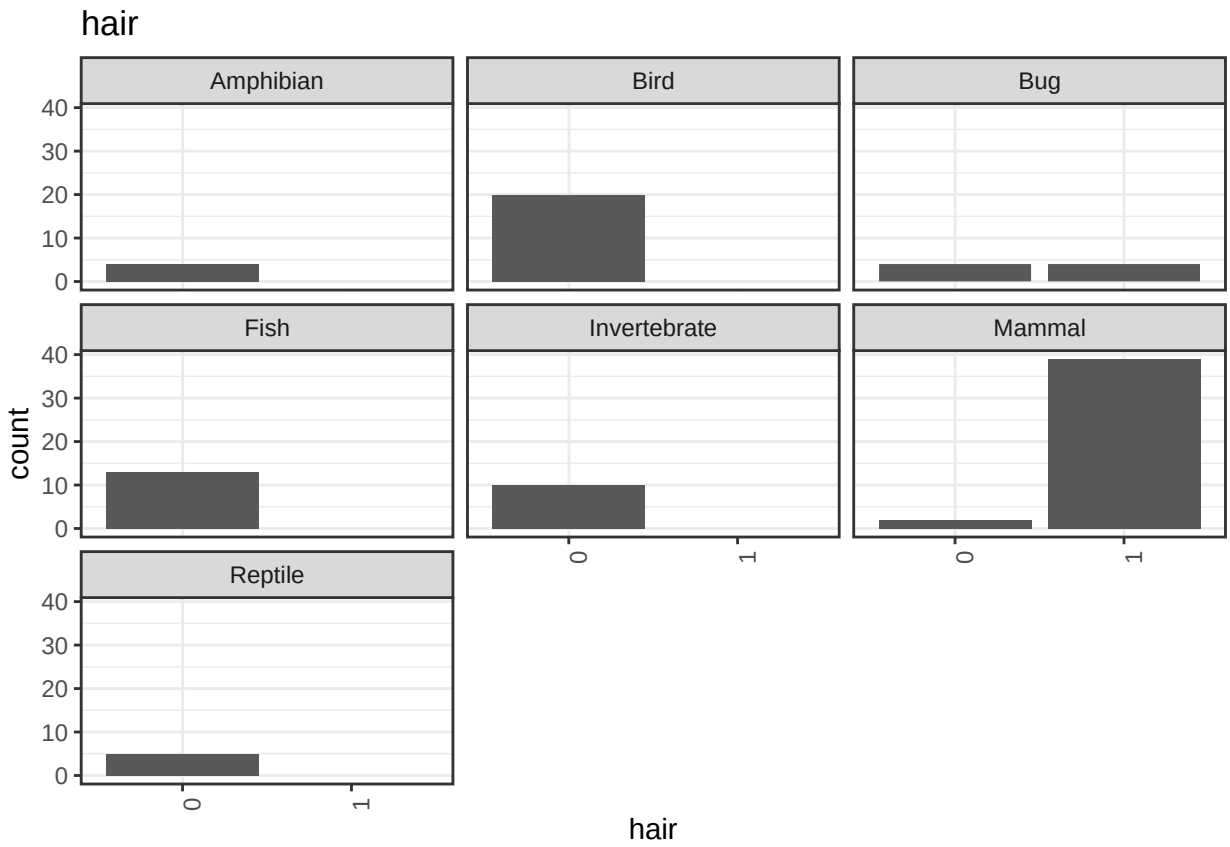
```
# Merging datasets based on class_type
zoo_dataset$class_type <- as.factor(zoo_dataset$class_type)
zoo_dataset <- merge(x = zoo_dataset, y = class_dataset, by = "class_type", all.x = TRUE)

# Displaying the head of the merged dataset
head(zoo_dataset, 3)
```

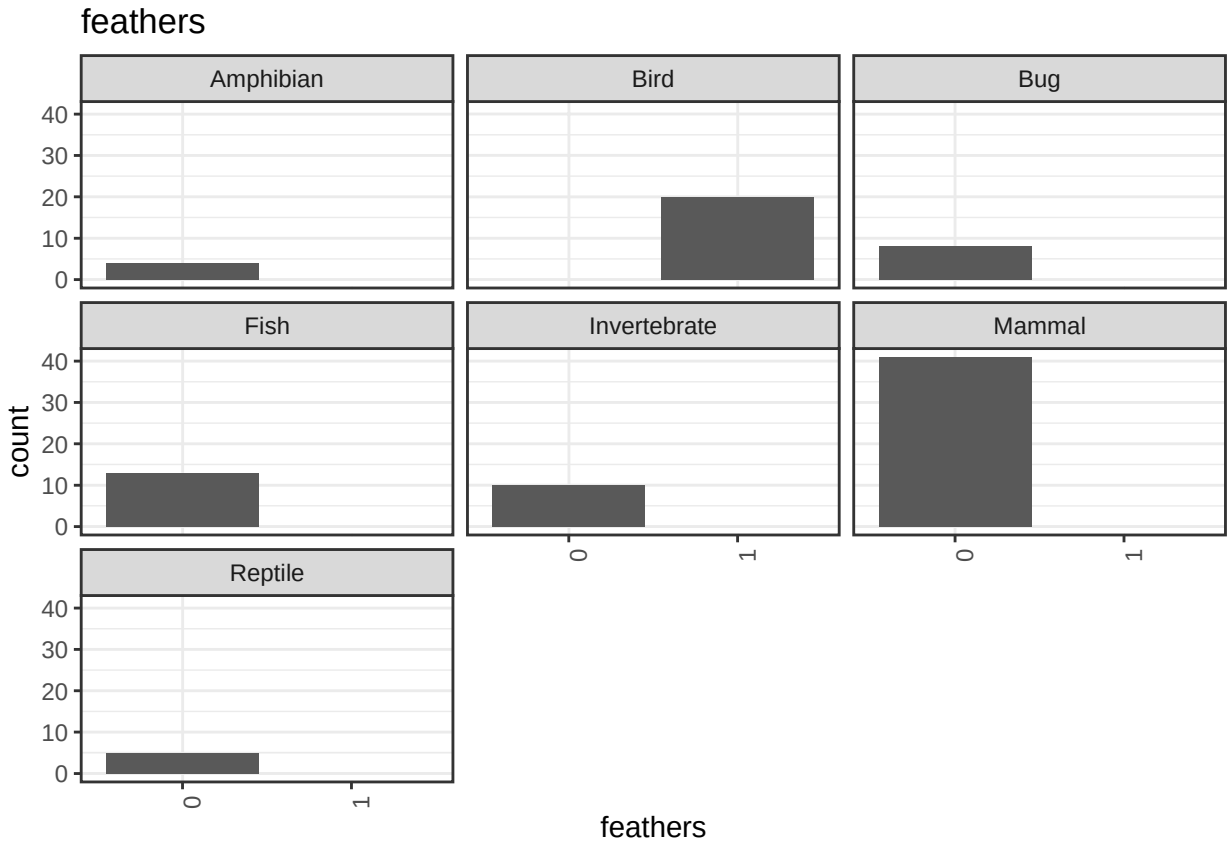
```
##   class_type animal_name hair feathers eggs milk airborne aquatic predator
## 1          1    aardvark    1        0    0    1        0       0        1
## 2          1    antelope    1        0    0    1        0       0        0
## 3          1     giraffe    1        0    0    1        0       0        0
##   toothed backbone breathes venomous fins legs tail domestic catsize class_name
## 1       1        1        1        0    0    4    0        0       1     Mammal
## 2       1        1        1        0    0    4    1        0       1     Mammal
## 3       1        1        1        0    0    4    1        0       1     Mammal
```

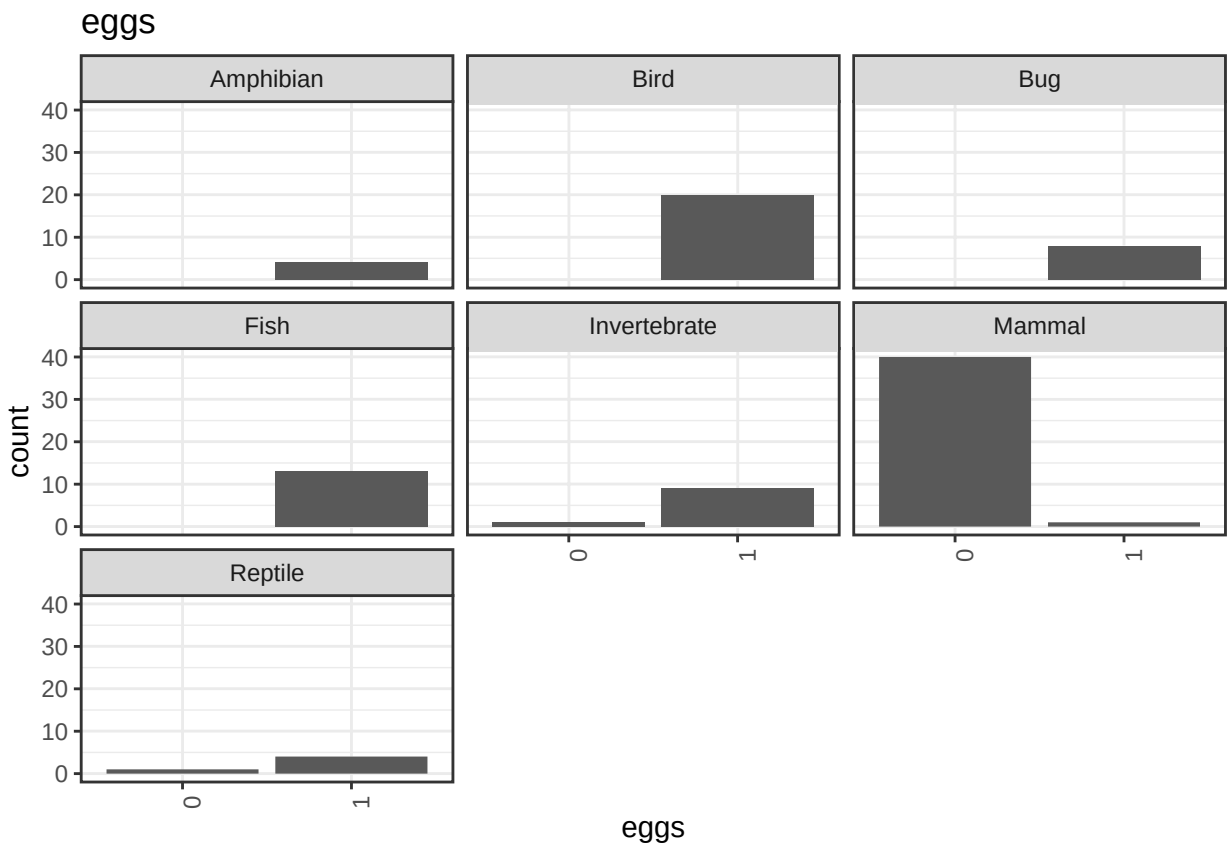
```
# Removing unnecessary objects from the environment
rm(class_dataset)
```

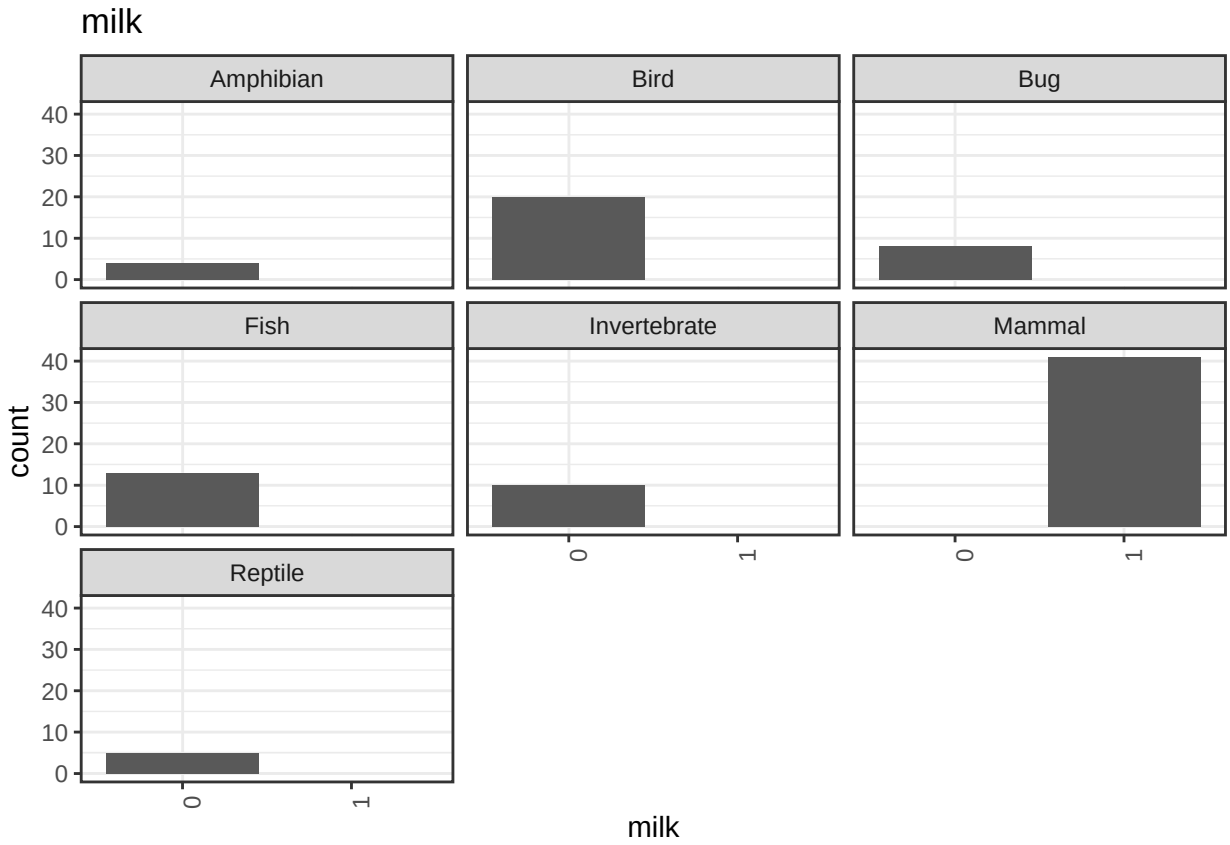
2 Visualizing the Dataset

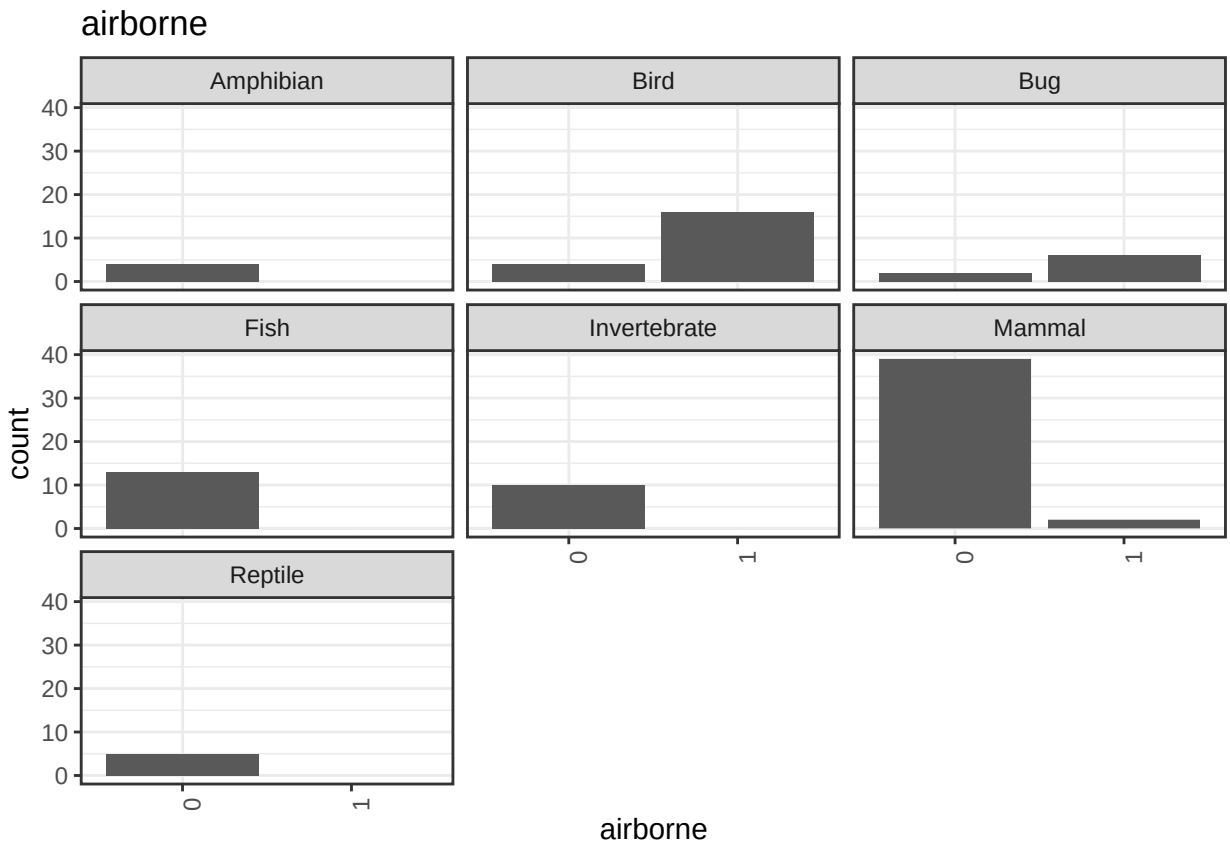
```
#plotting the count of unique values in the selected columns
plot_data <- function(df, i) {
  data <- data.frame(x = df[[i]])
  plot1 <- ggplot(data = data, aes(x = factor(x))) +stat_count()
    +
    xlab(colnames(df)[i]) +
    ggtitle(colnames(df)[i]) +
    theme_bw() +
    facet_wrap(~df[[19]]) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
  return(plot1)
}
#generating multiple plots itrating over specified columns and arranging the plots generated
lets.plot <- function(df, fun, ii, ncol = 3) {
  plot2 <- list()
  for (i in ii) {
    plot1 <- fun(df = df, i = i)
    plot2 <- c(plot2, list(plot1))
  }
  do.call("grid.arrange", c(plot2, ncol=ncol))
}

convert.to.numeric <- function(df, lst) {
  for (i in lst) {
    df[[i]] <- as.numeric(df[[i]])
  }
  return(df)
}

# Identifying Integer Columns in zoo_dataset & converting to numeric and plotting specified columns

int.list <- names(zoo_dataset)[which(sapply(zoo_dataset, is.integer))]
zoo_dataset <- convert.to.numeric(zoo_dataset, int.list)
rm(int.list)
for(i in seq(3, 18, 1)){
  lets.plot(zoo_dataset,fun=plot_data,ii=i,ncol=1)
}
```

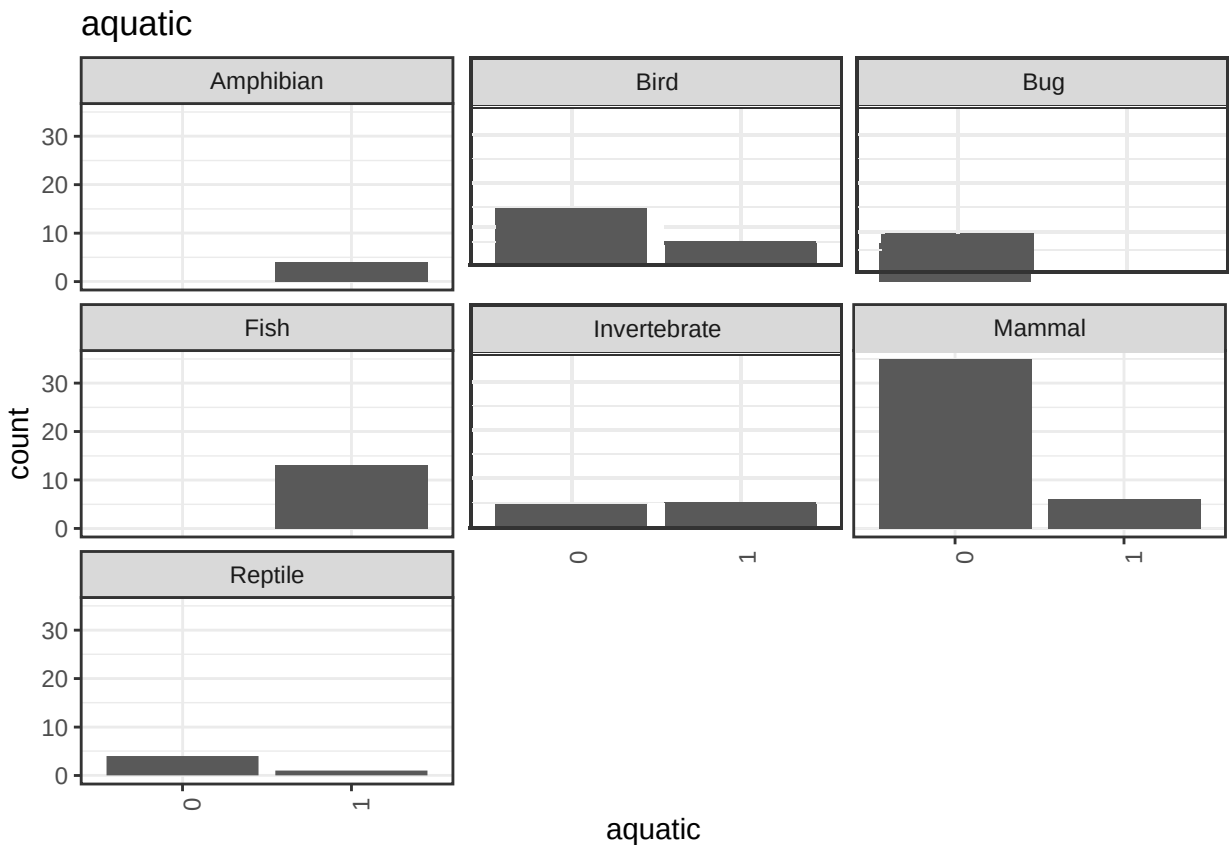

hair
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
40
30
20
10
0
0
1
hair

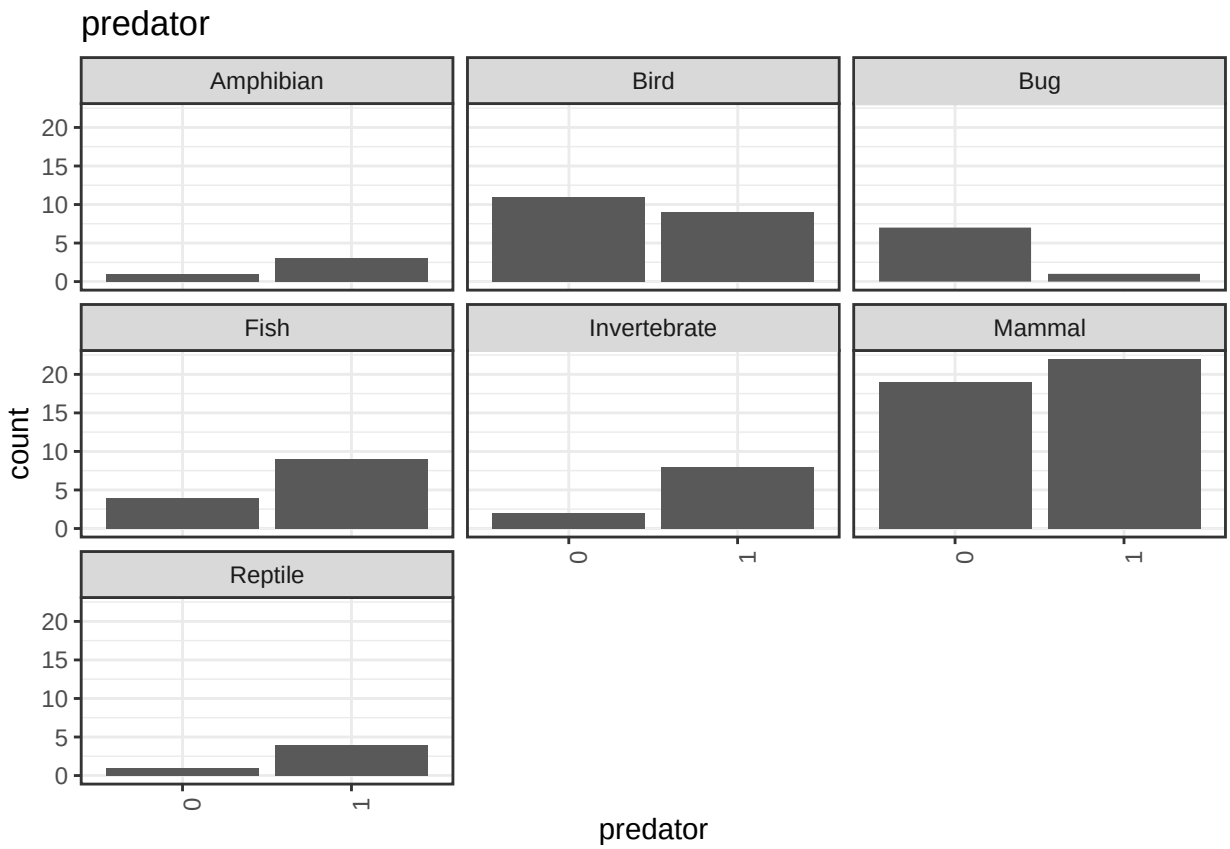

feathers
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
0
10
20
30
40
0
1
feathers

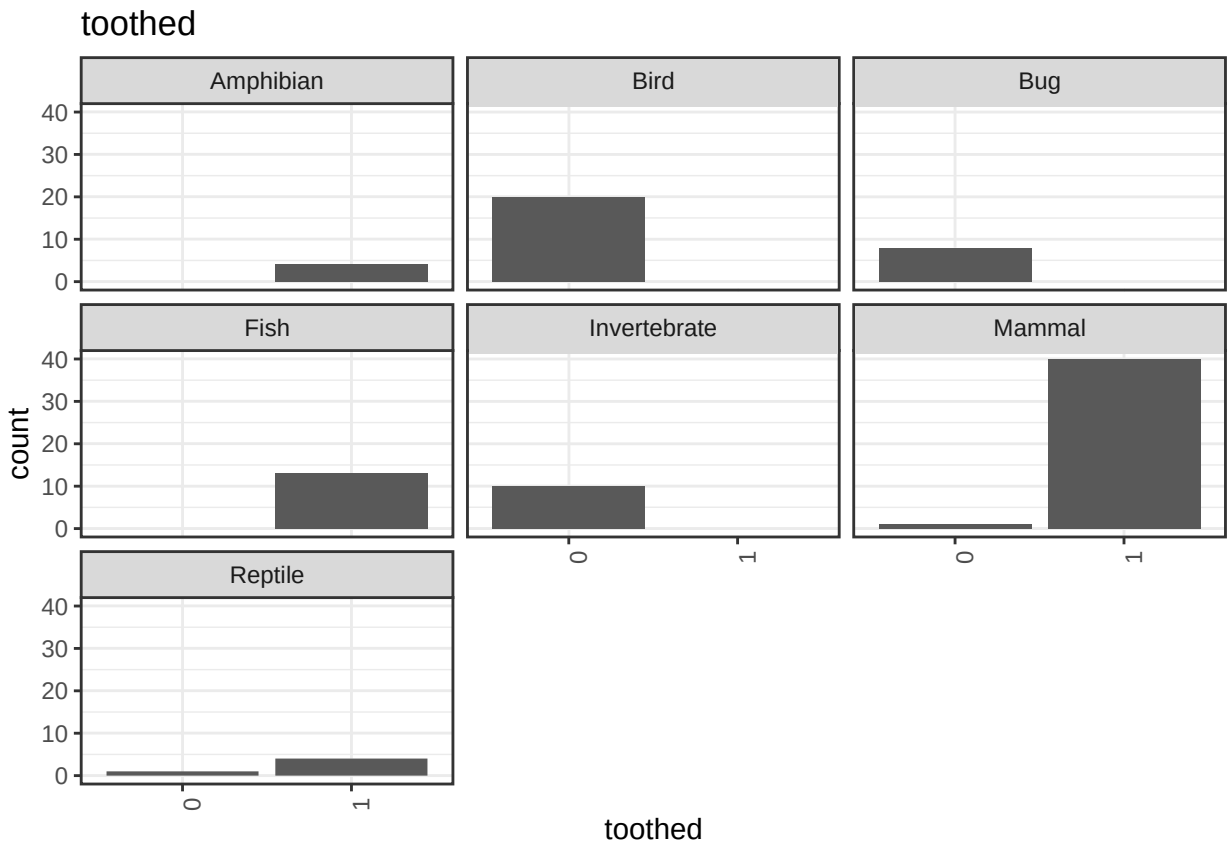

eggs
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
0
10
20
30
40
0
1
eggs

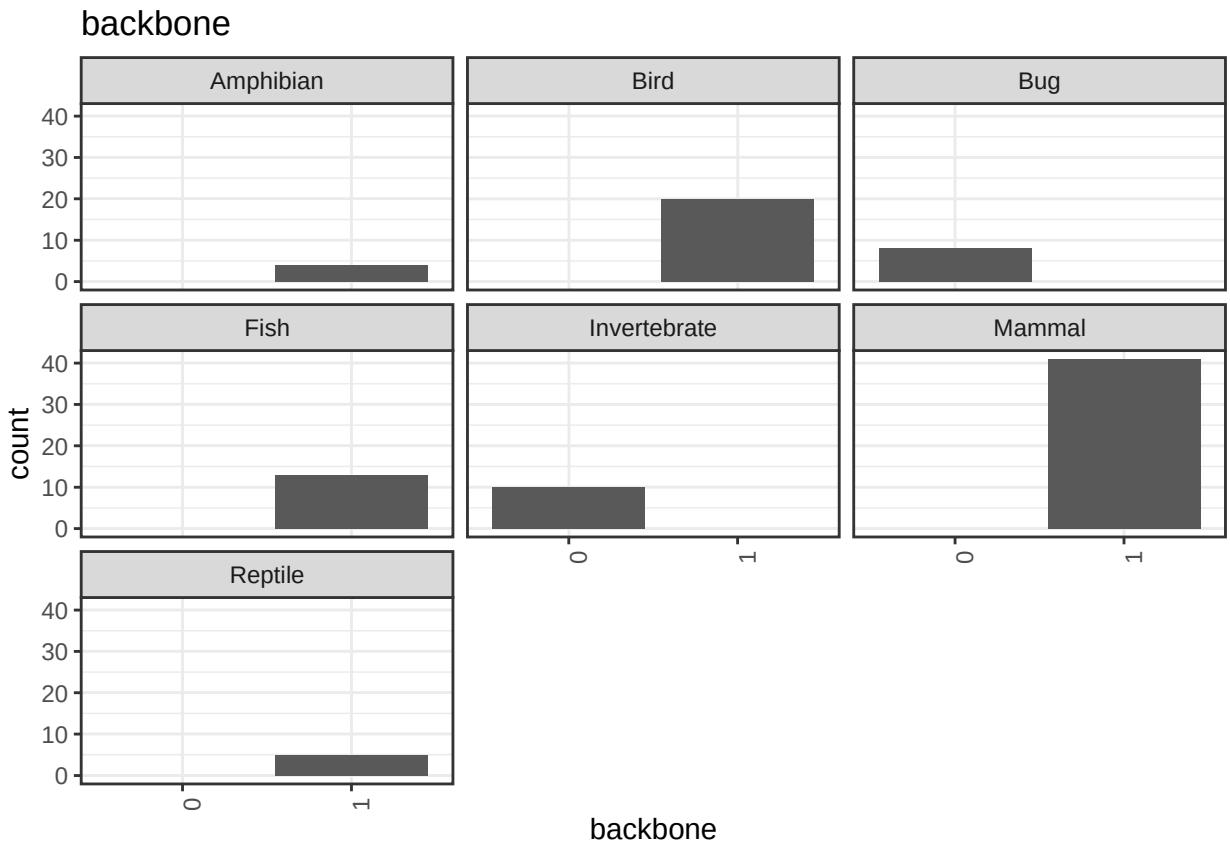

milk
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
0
10
20
30
40
0
1
milk

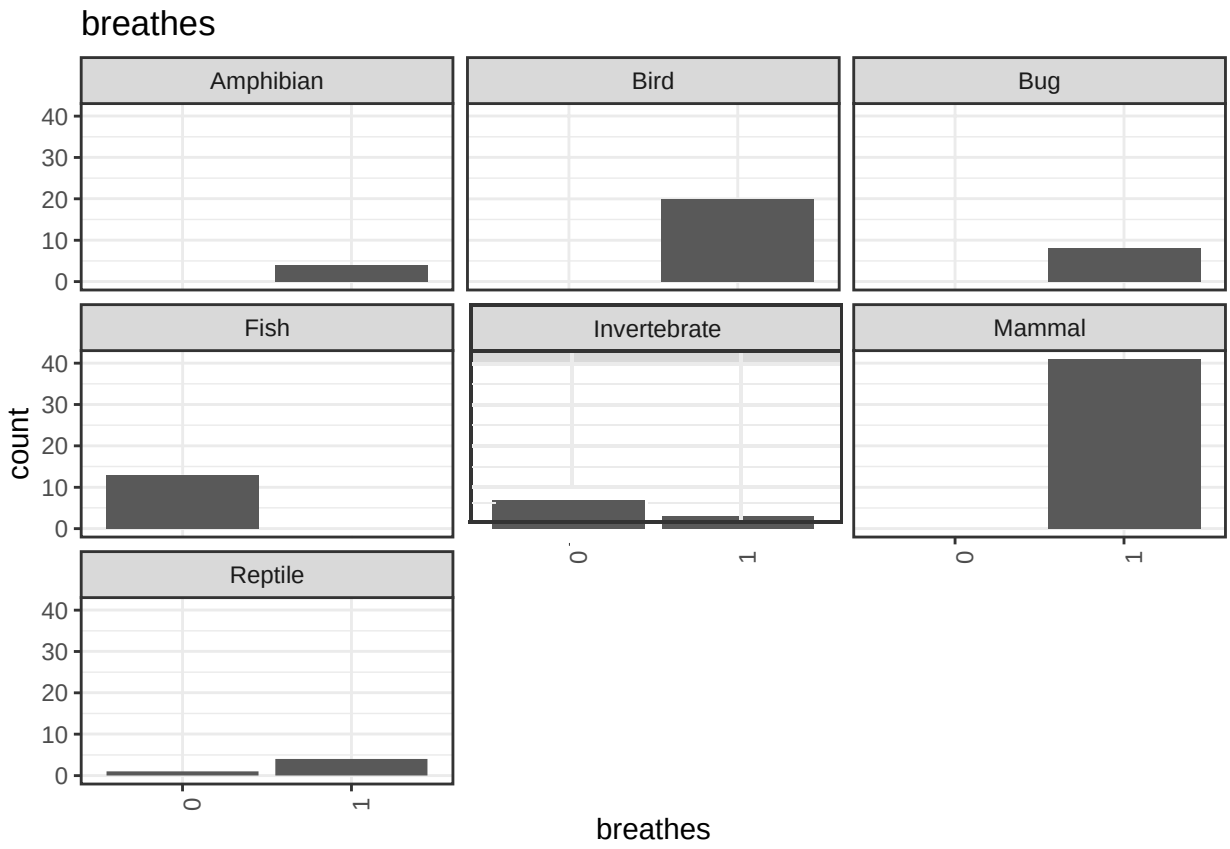

airborne
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
0
10
20
30
40
0
1
airborne

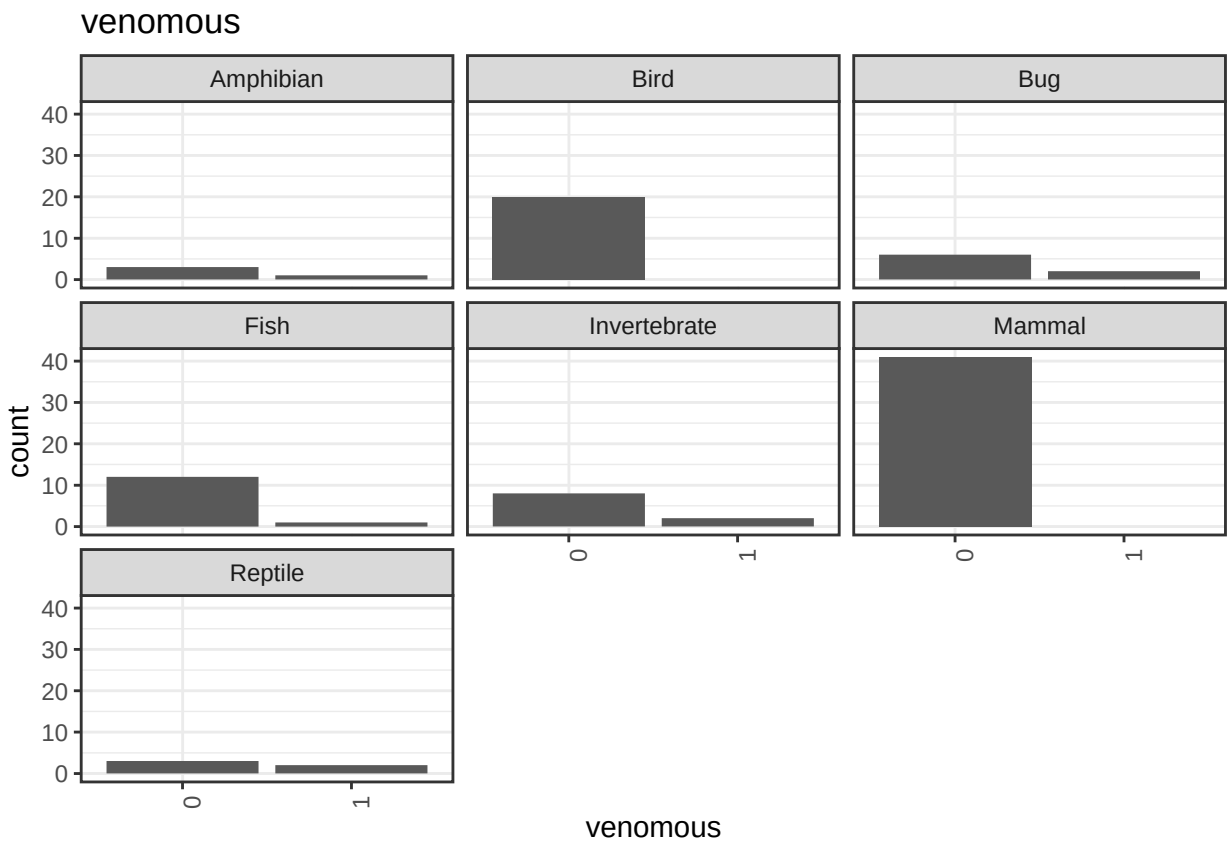

aquatic
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
30
20
10
0
0
1
aquatic

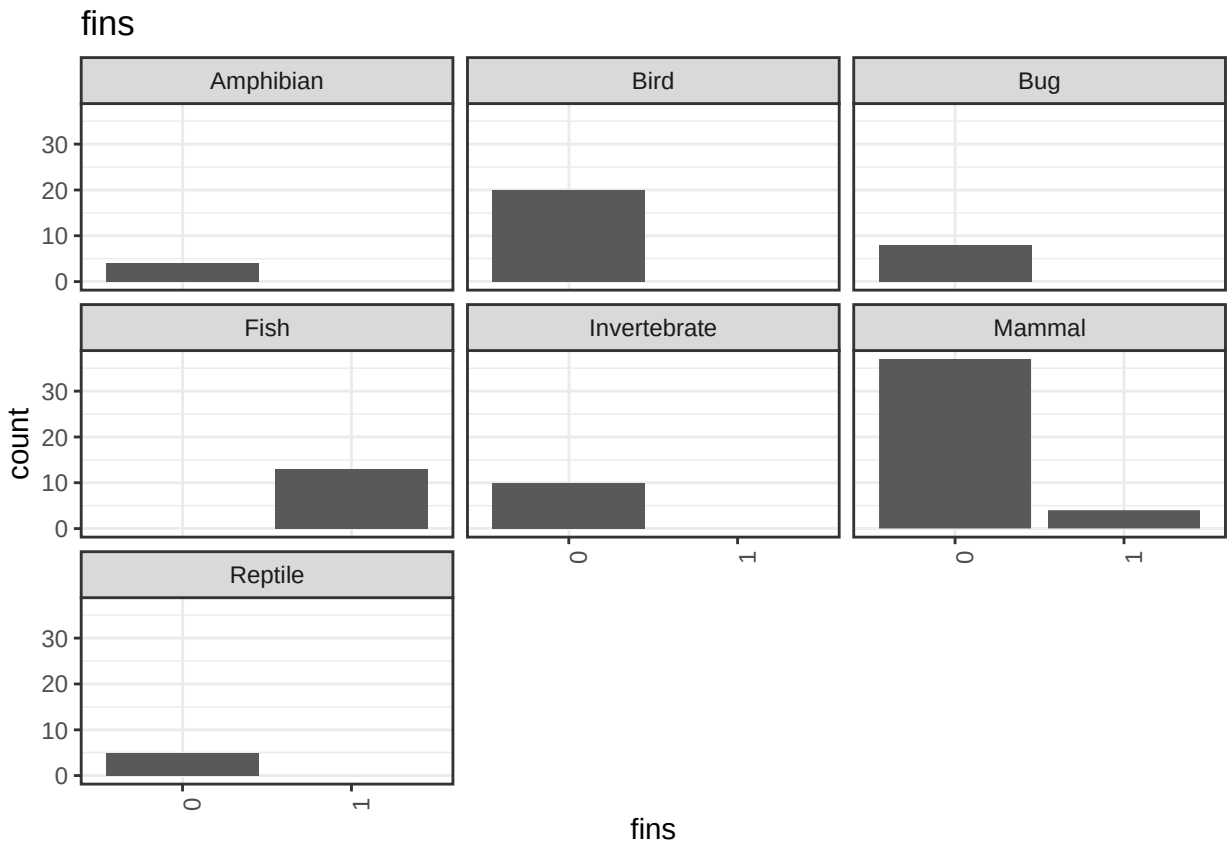

predator
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
0
5
10
15
20
0
1
predator

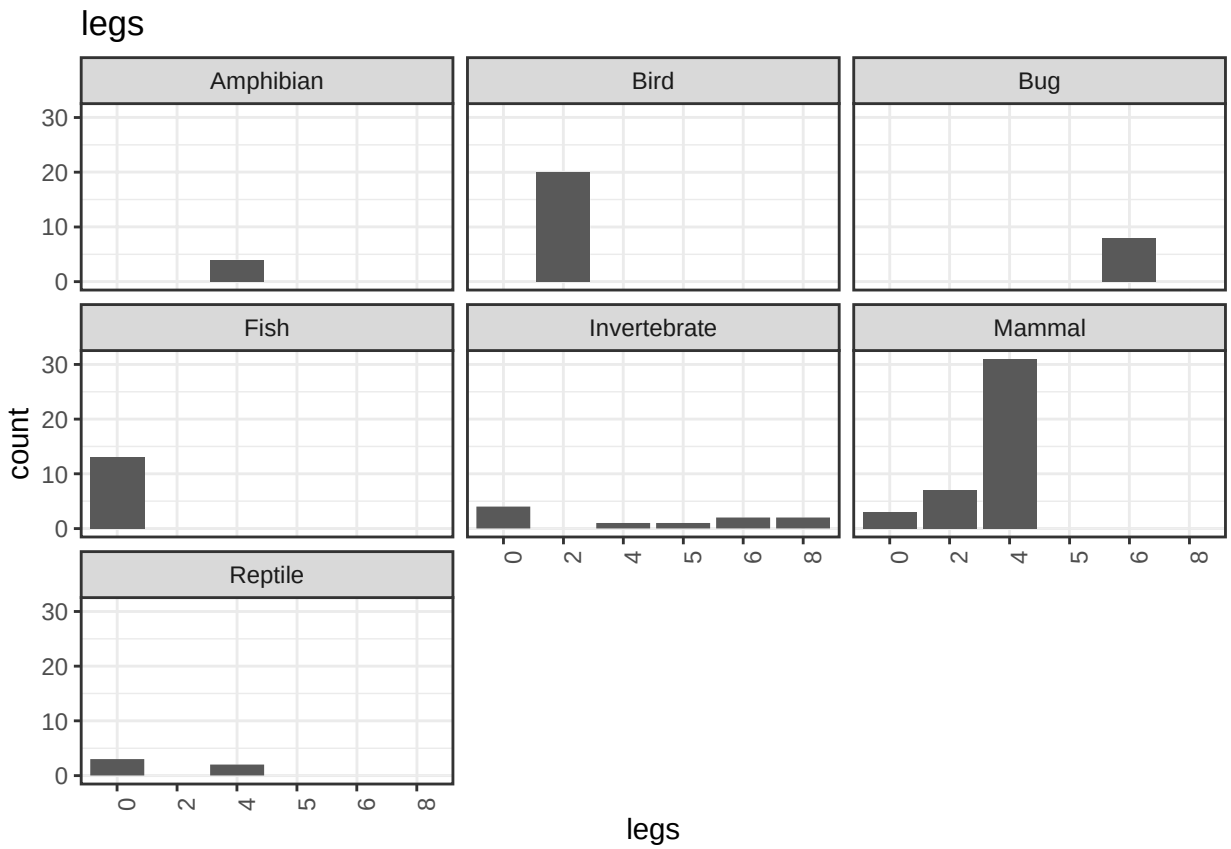

toothed
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
0
10
20
30
40
0
1
toothed

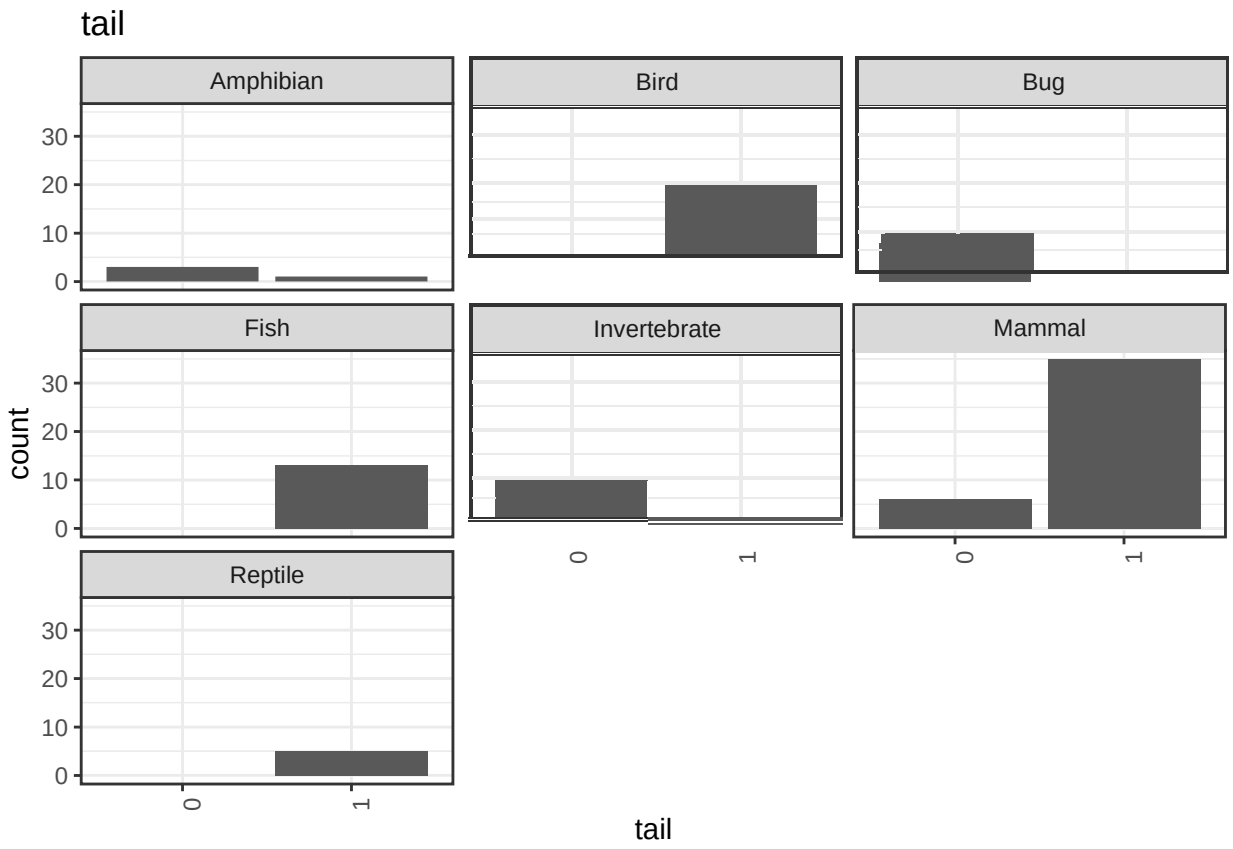

backbone
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
0
10
20
30
40
0
1
backbone

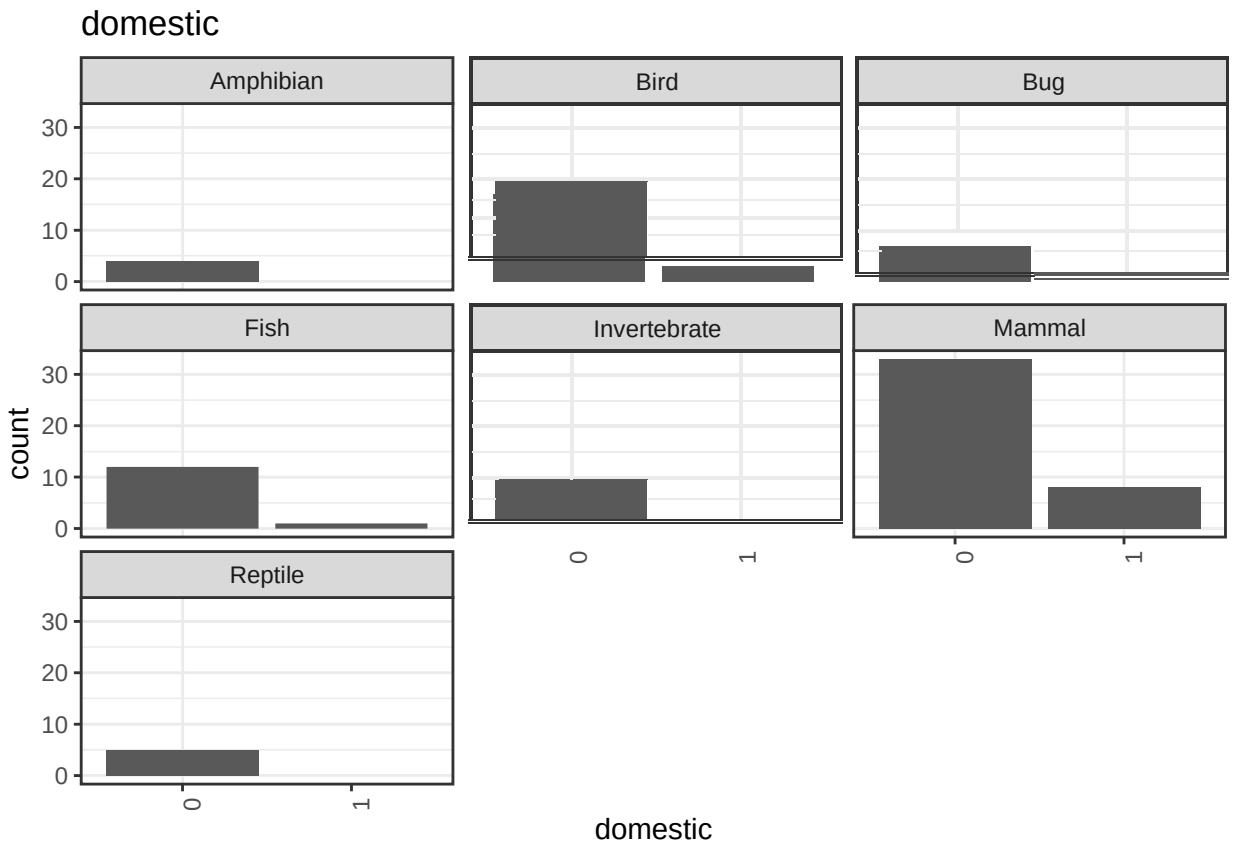

breathes
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
0
10
20
30
40
0
1
breathes

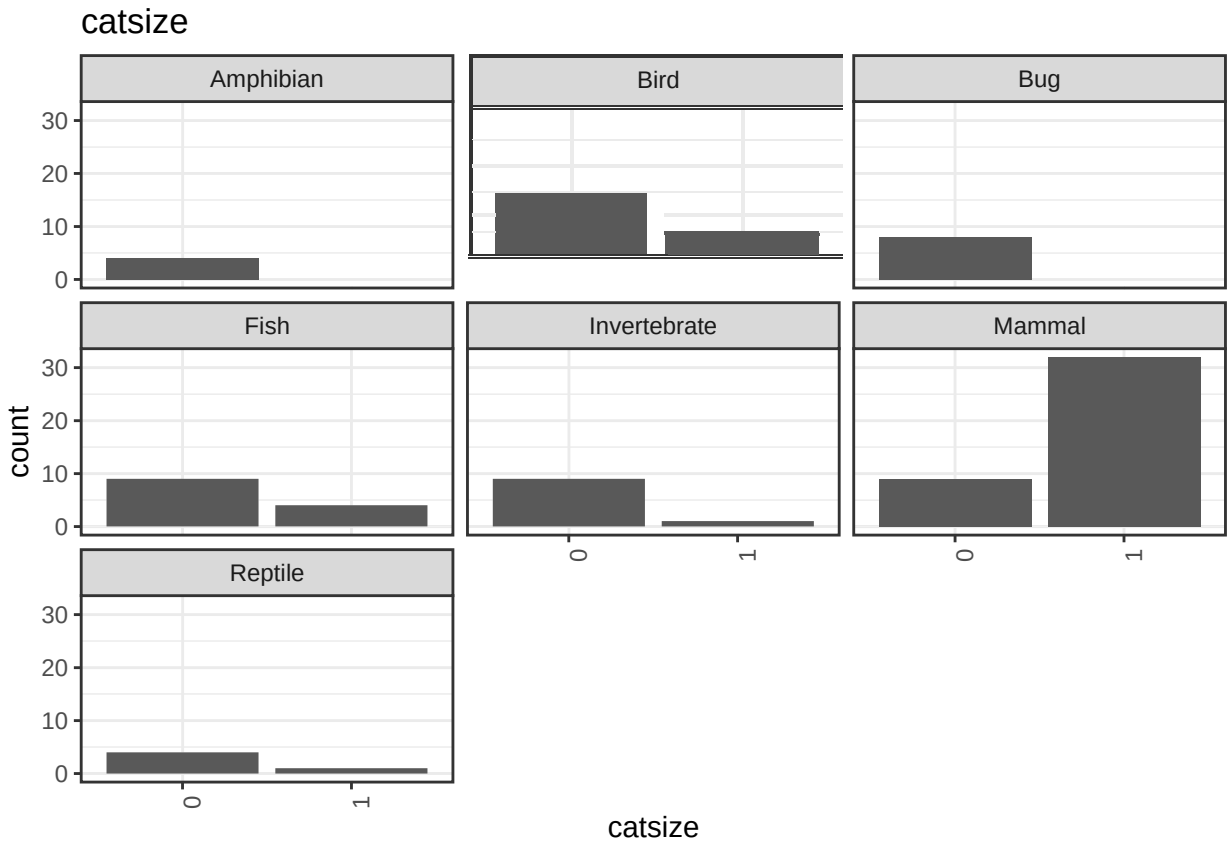

venomous
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
0
10
20
30
40
0
1
venomous


fins
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
30
20
10
0
0
1
fins


legs
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
30
20
10
0
0
2
4
5
6
8
legs


tail
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
30
20
10
0
0
1
tail


domestic
Amphibian
Bird
Bug
Fish
Invertebrate
Mammal
Reptile
count
30
20
10
0
0
1
domestic



3 Analysis of the Visualization of Numeric Data Distribution

The above code Visualses different attributes of animals in the zoo_dataset across various categories.This visualisation helps in understanding the distribution or patterns within the dataset. The code generates column chart for each numeric column (columns 3 to 18) in the zoo_dataset data frame. Histograms are useful for understanding the distribution of numerical variables, including their central tendency, spread, and shape.

Faceting by Categorical Variable: Each histogram is faceted by the categories in the 19th column of the zoo_dataset dataframe. Faceting allows us to compare the distribution of numeric variables across different categories of a categorical variable.

Identification of Outliers and Data Patterns: By visualizing the distribution of numeric variables, the code helps identify potential outliers or unusual patterns in the data. Outliers or unexpected patterns may indicate data quality issues or interesting insights about the dataset.

Comparison Across Categories: Faceting the histograms by a categorical variable enables comparison of the distribution of numeric variables across different groups. This comparison can reveal differences or similarities in data patterns between groups, providing insights into potential relationships or differences within the dataset.

Overall, the analysis provided by the code allows for a visual exploration of the numeric variables in the zoo_dataset data frame and facilitates the identification of patterns, outliers, and relationships within the data.

4 Data Partitioning and Training/Test Set Creation

```
# Setting up training (70%) and testing (30%) datasets
intraining <- createDataPartition(y = zoo_dataset$class_type, p = 0.7, list = FALSE)
train_batch <- zoo_dataset[intraining, ]
test_batch <- zoo_dataset[-intraining, ]

# Displaying class distribution in training and testing datasets
cat("----- Training batch---------- ")
```

```
## ----- Training batch ------
```

```
table(train_batch$class_type)
```

```
##
##  1  2  3  4  5  6  7
## 29 14  4 10  3  6  7
```

```
cat("----- Testing batch -------- ")
```

```
## ----- Testing batch -----
```

```
table(test_batch$class_type)
```

```
##
##  1  2  3  4  5  6  7
## 12  6  1  3  1  2  3
```

```
# Preparing training data
train_batch <- select(train_batch, -animal_name, -class_name)
train_x <- select(train_batch, -class_type)
train_y <- train_batch$class_type
```

The above output displays the class distribution in the training and testing batches after splitting the dataset.Based on the results, the most frequent class type in both the training and testing batches is class 1. Following class 1, class 2 and class 4 exhibit the highest frequencies.

5 Building the SVM classification model using radial kernel

```
set.seed(501)
svm_model <- svm(class_type ~., data = train_batch, kernel = "radial")

# Displaying the model summary
summary(svm_model)
```

```
##
## Call:
## svm(formula = class_type ~ ., data = train_batch, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  43
##
##  ( 12 8 4 4 3 6 6 )
##
##
## Number of Classes:  7
##
## Levels:
##  1 2 3 4 5 6 7
```

```r
# Preparing testing data
test_x <- select(test_batch, -class_type, -animal_name, -class_name)
test_y <- test_batch$class_type

# Making predictions on the test data set
test_pred <- predict(svm_model, test_x)
#Evaluating the SVM model using confusion matrix
confusionMatrix(test_pred, test_y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4  5  6  7
##          1 12  0  0  0  0  0  0
##          2  0  6  0  0  0  0  0
##          3  0  0  1  0  1  0  0
##          4  0  0  0  3  0  0  0
##          5  0  0  0  0  0  0  0
##          6  0  0  0  0  0  2  0
##          7  0  0  0  0  0  0  3
##
## Overall Statistics
##
##                Accuracy : 0.9643
##                  95% CI : (0.8165, 0.9991)
##     No Information Rate : 0.4286
##     P-Value [Acc > NIR] : 1.906e-09
##
##                   Kappa : 0.9517
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity            1.0000   1.0000  1.00000   1.0000  0.00000  1.00000
## Specificity            1.0000   1.0000  0.96296   1.0000  1.00000  1.00000
## Pos Pred Value         1.0000   1.0000  0.50000   1.0000      NaN  1.00000
## Neg Pred Value         1.0000   1.0000  1.00000   1.0000  0.96429  1.00000
## Prevalence             0.4286   0.2143  0.03571   0.1071  0.03571  0.07143
## Detection Rate         0.4286   0.2143  0.03571   0.1071  0.00000  0.07143
## Detection Prevalence   0.4286   0.2143  0.07143   0.1071  0.00000  0.07143
## Balanced Accuracy      1.0000   1.0000  0.98148   1.0000  0.50000  1.00000
##                      Class: 7
## Sensitivity            1.0000
## Specificity            1.0000
## Pos Pred Value         1.0000
## Neg Pred Value         1.0000
## Prevalence             0.1071
## Detection Rate         0.1071
## Detection Prevalence   0.1071
## Balanced Accuracy      1.0000
```

6 Analysis of SVM classification model using confusion matrix

1.A SVM classification model was fitted on the training dataset to predict the classes of animals (Dependent variable) based on the features (independent variable). For this model the radial kernel was used which is effective for handling non-linear decision boundaries, and is commonly used in SVM classification models.

2.The results show that the cost for the fitted model is 1.Cost is a parameter to penalize misclassifications by controlling the trade-off between maximizing the margin and minimizing the classification error.

3.The total number of support vectors used by the fitted SVM model was 45.This number refers to the data points from the training set that lie closest to the decision boundary and are thus critical for defining the boundary between different classes.

4.The results also show the breakdown of the number of support vectors (45) for each class as follow(note that the number of support vector can vary each time the code is executed as the partitioning of data to test and training set is random): (12 4 8 7 3 6 5); Each number corresponds to a class label, and the sequence matches the order of class labels in the dataset. Then, the first number (12) represents the numberof support vectors for class 1 (Mammal), the second number (4) represents the number for class 2 (Bird), and so on.Having a significant number of support vectors can indicate a more complex decision boundary,which may lead to a model that is more susceptible to overfitting.

5.The confusion matrix presents the performance of the SVM model on the test dataset. The Reference columns represent the actual classes in the test dataset, and the Prediction rows represents the predicted classes generated by the classification SVM. The diagonal of the confusion matrix represents the number of animals that were correctly classified, while off-diagonal elements represent misclassifications. Here the confusion matrix shows there were missclassifications in class 7. Other than all animals in the test set were classified correctly which shows a good performance of the model.

7 Overal statistics:

1. The overall accuracy of the SVM model is 96.43%. This means that approximately 96.43% of the animals in the test dataset were classified correctly by the model. The confidence interval for this accuracy is (0.8165, 0.9991). The interval indicates that with 95% confidence the true accuracy of the model falls within this interval.

2.No Information Rate (NIR) indicates the proportion of the largest class in the test dataset which is 42.86%. This is the accuracy that would be achieved by always predicting the majority class without using model.

3.The p-value associated with the comparison of the model's accuracy to the No Information Rate is 1.906e-09 , which is very low. This indicates strong evidence against the null hypothesis that the model's accuracy is equal to the No Information Rate. So, using a model is significantly informative for predicting the class of animals.

4.The kappa statistic measures the agreement between the observed accuracy and the expected accuracy (chance agreement). A kappa value of 0.9516 indicates very strong agreement between the model's predictions and the actual classes, beyond what would be expected by chance alone.

5.McNemar's test is used to compare the observed accuracy of the model to the expected accuracy under the null hypothesis of equal predictive ability for two classes. In this case, since we have more than two classes the p-value for McNemar's test is not available (NA).

8 Analysing the perfomance of the fitted SVM model for each class

1. The sensitivity measures the proportion of actual positive cases that were correctly identified by the model. The sensitivity for classes 1,2,3,4,5 and 6 was 1 indicating that the model correctly identified all positive cases for these classes of animals. However, this value is .6 classes 6.

2.The specificity measures the proportion of actual negative cases that were correctly identified by the model. The specificity for classes 1,2,3,4,5 and 7 was 1 and means that the model correctly identified all negative cases for that class. However, this value is .96 for class 6.

3.Positive Predictive Value represents the proportion of predicted positive cases that were correctly classified. The PPV is 1.0 for classes 1,2,3,4,5 and 6 indicating that all predicted positive cases were correct.This value was .96 for class 7.

4.Negative Predictive Value (NPV) is the proportion of predicted negative cases that were correctly classified. The NPVs are 1.0 for classes 1,2,3,4,5 and 6 meaning that all predicted negative cases were correct for these classes.

5. Prevalence is proportion of positive cases in the dataset for each class which has the highest value for class 1 and lowest value for classes 4 and 5.

6.Detection Rate is the proportion of cases that were correctly identified by the model for each class and has the highest value for class 1 i.e. 0.43 and lowest value for class 5 i.e. 0.000.

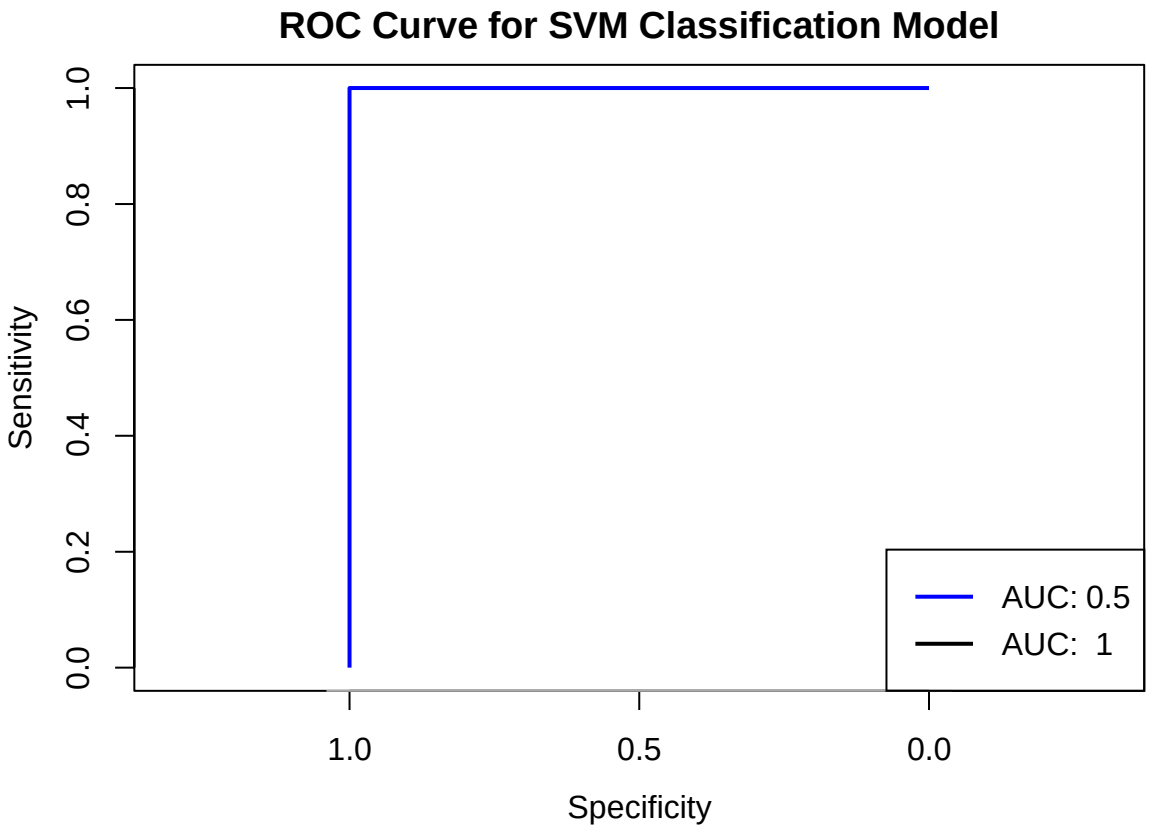
7. Detection Prevalence is the proportion of cases that the model predicted to be positive for each class.

8.Balanced Accuracy is the average of sensitivity and specificity, providing a balanced assessment of the model's performance across classes. The output shows high balanced accuracy for classes 1,2,3,4 and 5,but this proportion is low for classes 6 and 7

Overall, the SVM model demonstrates strong performance across most classes, with some exceptions. Further analysis may be needed to understand the reasons behind such discrepancies and to improve the model's performance for all classes.

9 Evaluating SVM model using the Receiver Operating Characteristic(ROC) Curve

```r
test_pred_prob<-as.numeric(test_pred)

# Creating a ROC curve

roc_curve <- roc(test_y, test_pred_prob)

# Plot the ROC curve
plot(roc_curve, main = "ROC Curve for SVM Classification Model", col = "blue", lwd = 2)

# Add labels and legend
legend("bottomright", legend = c("AUC: 0.5", paste("AUC: ", round(auc(roc_curve), 2))), col = c("blue",
```



Interpretation of the ROC Curve

The ROC curve plots the True Positive Rate (Sensitivity) against the False Positive Rate (1 - Specificity) for different threshold values.The curve illustrates the trade-off between sensitivity and specificity. A diagonal line (from [0,0] to [1,1]) represents random guessing, where the area under the curve (AUC) is 0.5. The closer the ROC curve is to the upper-left corner of the plot, the better the classifier's performance.

1. Area Under the Curve (AUC): The AUC represents the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.A perfect classifier will have an AUC of 1, indicating perfect discrimination between the two classes.An AUC of 0.5 suggests that the classifier has no discriminatory power and performs no better than random chance.Generally, an AUC above 0.5 indicates better-than-random performance, with higher values indicating better discrimination.

2.Analysis of the Results: Evaluate the AUC: Check the AUC value obtained from the ROC curve. A higher AUC suggests better model performance.

3.Sensitivity and Specificity: Assess how well the model balances sensitivity (true positive rate) and specificity (true negative rate). A good model maintains high sensitivity while keeping specificity high as well.

4.Threshold Selection: The ROC curve can help in selecting an appropriate threshold based on the desired balance between sensitivity and specificity, depending on the specific application's requirements.

5. Performance Comparison: Compare the AUC of the current model with other models or baseline models to determine which classifier performs better.If multiple models are evaluated, the one with the highest AUC is generally considered the best performer.

6.Practical Implications: Understanding the ROC curve helps in assessing the trade-offs between sensitivity and specificity, which are crucial in many classification tasks, such as medical diagnosis, fraud detection, and risk assessment. It provides insights into the model's ability to distinguish between different classes and helps in setting appropriate thresholds for decision-making.

In summary, the ROC curve and AUC provide valuable insights into the performance of a binary classi- fier, helping practitioners assess the model's discriminatory power and make informed decisions about its deployment in real-world applications.