Module 2 Exercises

Filipp Krasovsky

5/23/2021

1. (10 points) 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) # Use ?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?
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?
c. Develop a strategy for handling missing data, either by eliminating predictors or imputation.

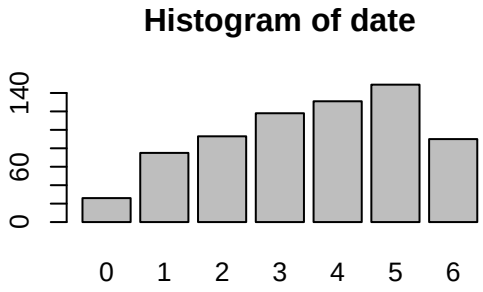
```
#load in our dataset
library(mlbench)
```

```
## Warning: package 'mlbench' was built under R version 4.0.5
```

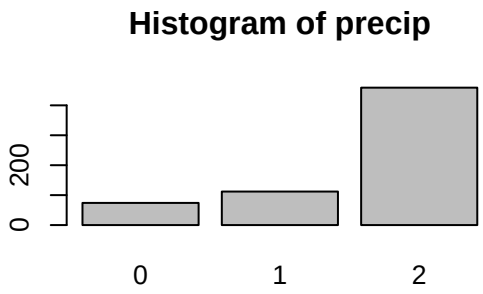
```
data(Soybean)
soy.df = as.data.frame(Soybean)
```

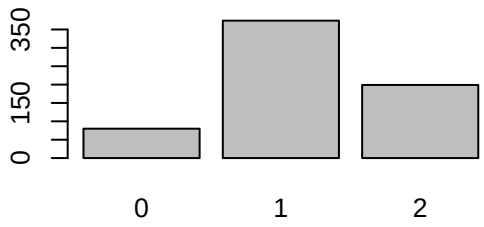
```
#split up into predictors and classes
predictors = subset(soy.df,select=-c(Class))
classes    = subset(soy.df,select=c(Class))
```

```
par(mfrow=c(2,2))
for(i in names(predictors)){
  plot_title = paste("Histogram of",i)
  plot(predictors[i],main=plot_title)
}
```

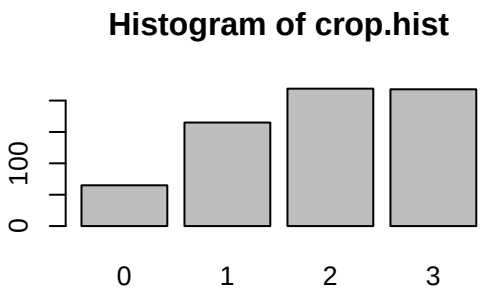

Histogram of date
0
60
140
0
1
2
3
4
5
6

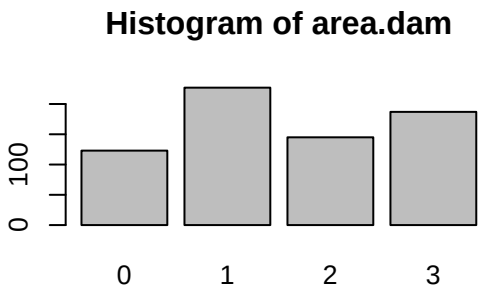

Histogram of plant.stand
0
150
350
0
1


Histogram of precip
0
200
0
1
2

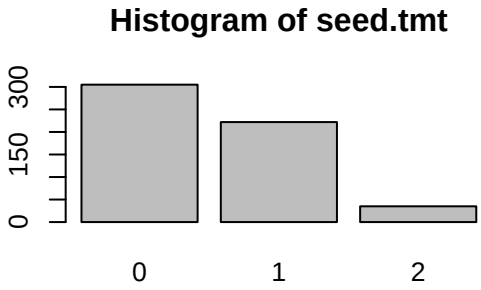

Histogram of temp
0
150
350
0
1
2

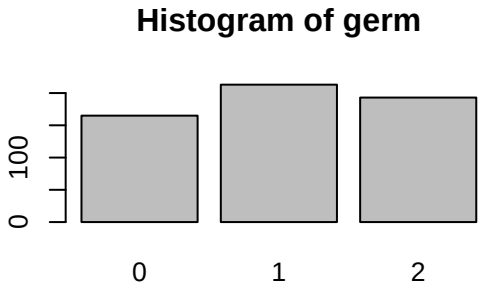

Histogram of hail
0
200
0
1


Histogram of crop.hist
0
100
0
1
2
3


Histogram of area.dam
0
100
0
1
2
3

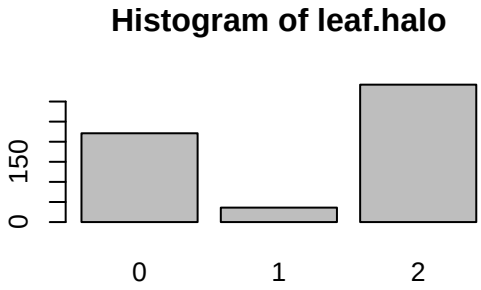

Histogram of sever
0
150
0
1
2

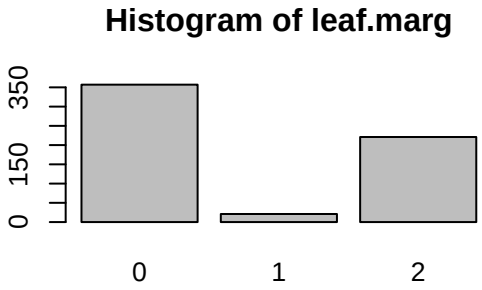

Histogram of seed.tmt
0
150
300
0
1
2

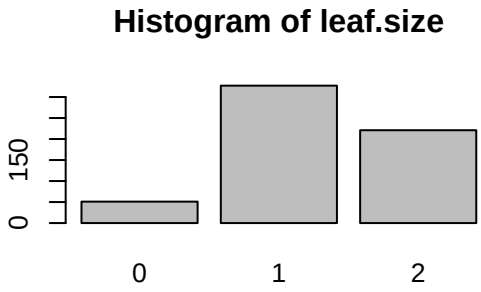

Histogram of germ
0
100
0
1
2

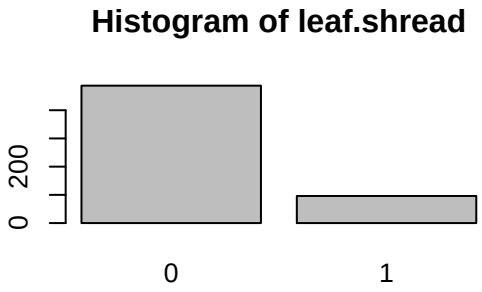

Histogram of plant.growth
0
200
0
1

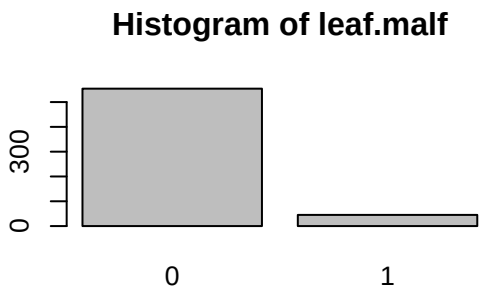

Histogram of leaves
0
300
600
0
1

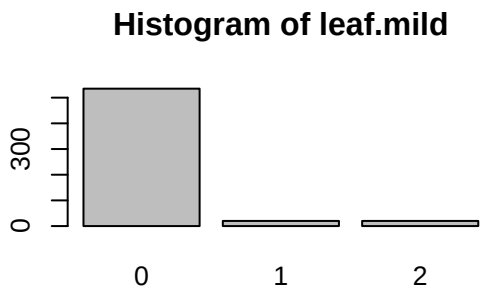

Histogram of leaf.halo
0
150
0
1
2

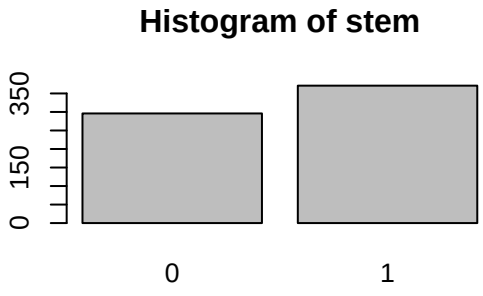

Histogram of leaf.marg
0
150
350
0
1
2


Histogram of leaf.size
0
150
0
1
2

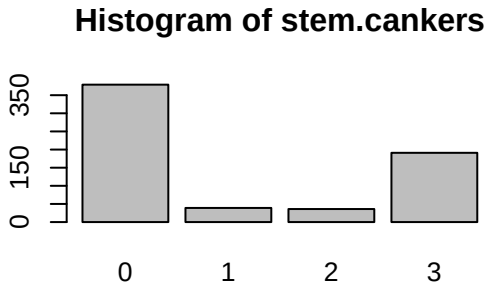

Histogram of leaf.shread
0
200
0
1

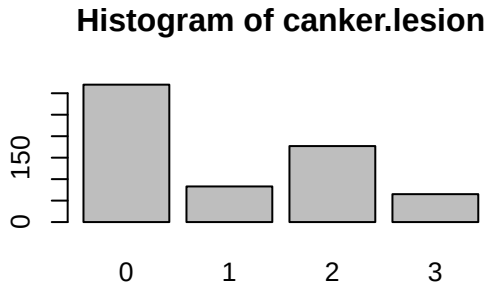

Histogram of leaf.malf
0
300
0
1

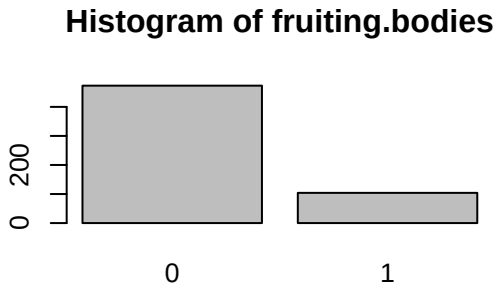

Histogram of leaf.mild
0
300
0
1
2

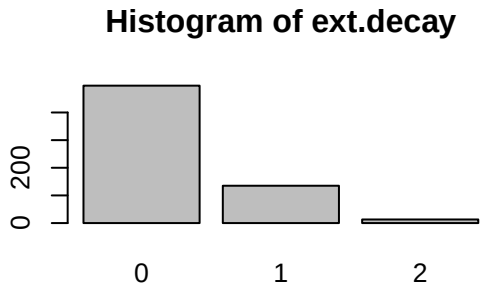

Histogram of stem
0
150
350
0
1

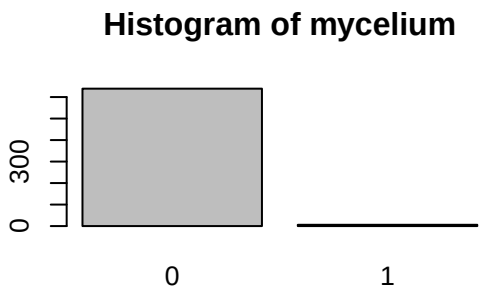

Histogram of lodging
0
200
500
0
1

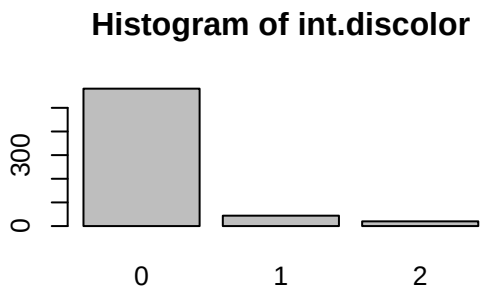

Histogram of stem.cankers
0
150
350
0
1
2
3


Histogram of canker.lesion
0
150
0
1
2
3

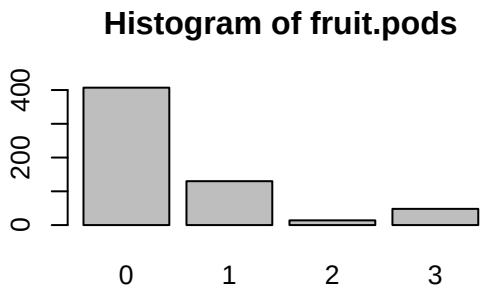

Histogram of fruiting.bodies
0
200
0
1

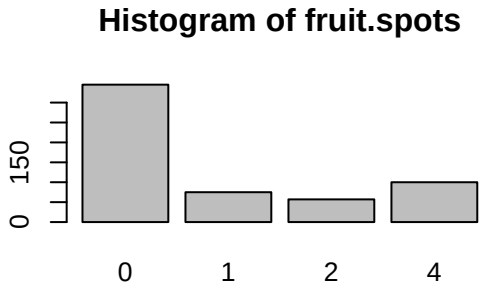

Histogram of ext.decay
0
200
0
1
2

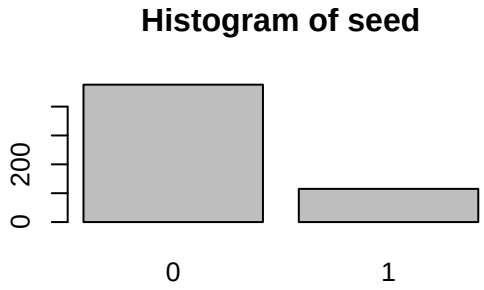

Histogram of mycelium
0
300
0
1

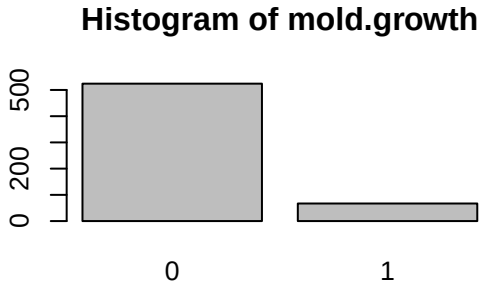

Histogram of int.discolor
0
300
0
1
2

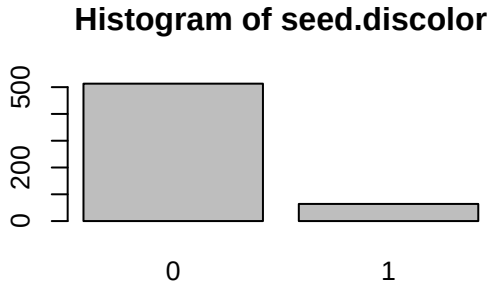

Histogram of sclerotia
0
300
0
1

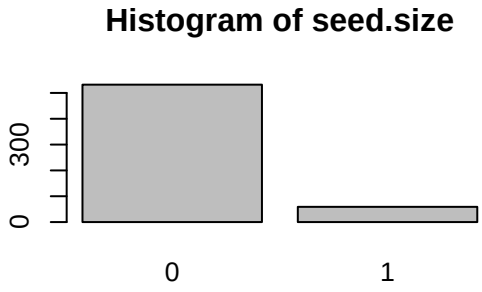

Histogram of fruit.pods
0
200
400
0
1
2
3

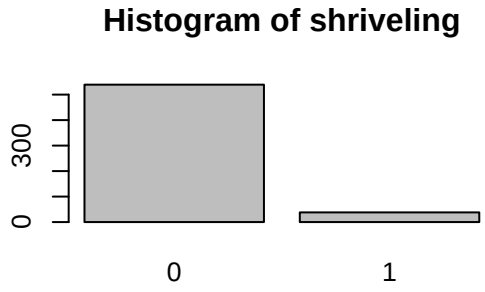

Histogram of fruit.spots
0
150
0
1
2
4

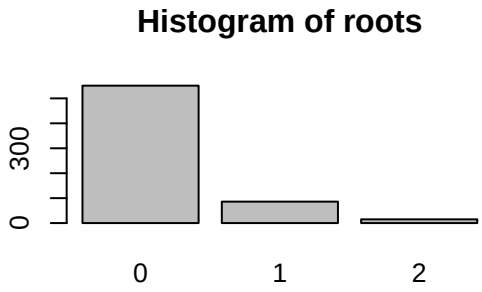

Histogram of seed
0
200
0
1


Histogram of mold.growth
0
200
500
0
1


Histogram of seed.discolor
0
200
500
0
1







It seems that a considerable amount of predictors might be degenerate or near zero variance predictors. In order to verify this, we operate on the two following rules of thumb:

1. the fraction of unique values over n is low (<10%)
2. the ratio of the most frequent value to the second most frequent value is greater than 20.

These assumptions are modified slightly in the nearZeroVar() function that verifies which distributions are degenerate.

```
require(caret)
```

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 4.0.5
```

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

```
## Loading required package: ggplot2
```

```
degenerates = caret::nearZeroVar(predictors)
print(names(predictors)[degenerates])
```

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

In conclusion, leaf.mild, myeclium, and sclerotia are all near-zero variance predictors.

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?

```
#check for missing values
nas = sapply(soy.df, function(x) round(sum(is.na(x))/nrow(soy.df),2))
nas[order(nas,decreasing=TRUE)]
```

```
##            hail           sever        seed.tmt         lodging           germ
##            0.18            0.18            0.18            0.18           0.16
##       leaf.mild fruiting.bodies     fruit.spots   seed.discolor     shriveling
##            0.16            0.16            0.16            0.16           0.16
##     leaf.shread            seed     mold.growth       seed.size      leaf.halo
##            0.15            0.13            0.13            0.13           0.12
##       leaf.marg       leaf.size       leaf.malf      fruit.pods         precip
##            0.12            0.12            0.12            0.12           0.06
##    stem.cankers   canker.lesion       ext.decay        mycelium    int.discolor
##            0.06            0.06            0.06            0.06           0.06
##       sclerotia     plant.stand           roots            temp      crop.hist
##            0.06            0.05            0.05            0.04           0.02
##    plant.growth            stem           Class            date       area.dam
##            0.02            0.02            0.00            0.00           0.00
##          leaves
##            0.00
```

It appears that all above variables ranging from hail to fruit.pods have over a 10% rate of missing values, with hail through shirveling having the highest likelihood. It's also curious that some of these seem to have similar probabilities - breaking this down numerically presents the following findings:

```
#check for missing values
nas = sapply(soy.df, function(x) sum(is.na(x)))
nas[order(nas,decreasing=TRUE)]
```

```
##            hail           sever        seed.tmt         lodging           germ
##             121             121             121             121            112
##       leaf.mild fruiting.bodies     fruit.spots   seed.discolor     shriveling
##             108             106             106             106            106
##     leaf.shread            seed     mold.growth       seed.size      leaf.halo
##             100              92              92              92             84
##       leaf.marg       leaf.size       leaf.malf      fruit.pods         precip
##              84              84              84              84             38
##    stem.cankers   canker.lesion       ext.decay        mycelium    int.discolor
##              38              38              38              38             38
##       sclerotia     plant.stand           roots            temp      crop.hist
##              38              36              31              30             16
##    plant.growth            stem            date       area.dam          Class
##              16              16               1               1              0
##          leaves
##               0
```

fruiting.bodies, fruit.spots, seed.discolor, and shriveling all have the exact same NA count while seed, seed.size, and mold.growth experience the same phenomenon. We also observe this for leaf.halo, leaf.marg.

leaf.size. leaf.malf, and fruit.pods, as well as some other instances. However, we will not pursue any computation to determine how spurious this claim is, and will move straight into identifying class dependency.

We start with a high level observation of which classes have the most of any predictors with missing data.

```
soy.na = soy.df[rowSums(is.na(soy.df))>0,]
na.tbl = table(soy.na$Class)
na.tbl = na.tbl[order(na.tbl,decreasing=TRUE)]

na.tbl = as.data.frame(na.tbl)
na.tbl = subset(na.tbl,Freq > 0)
na.tbl
```

```
##                          Var1 Freq
## 1            phytophthora-rot   68
## 2                2-4-d-injury   16
## 3 diaporthe-pod-&-stem-blight   15
## 4               cyst-nematode   14
## 5            herbicide-injury    8
```

It appears, on face value, that only 5 classes out of 19 comprise the entirety of missing values. From here, we also observe that phytophthora-rot constitutes the most missing cases possible. Given that there seems to be a class dependency for missing values, we can also move forward with identifying the class dependency for each predictor. However, it's important to note that for each of these five classes, the number of rows with missing values coincides with the number of class instances - that is, all of these class values have missing data in every data point.

```
for (i in c(2:length(soy.na))){
  predictor.vals = soy.na[,i]
  predictor.name = names(soy.na)[i]
  predictor.table = as.data.frame(table(soy.na$Class,predictor.vals,useNA = "always"))
  predictor.table = subset(predictor.table,Freq>0 & is.na(predictor.vals),select=-c(predictor.vals))

  print(predictor.table )
}
```

```
##             Var1 Freq
## 141 2-4-d-injury    1
##                          Var1 Freq
## 41                2-4-d-injury   16
## 49               cyst-nematode   14
## 50 diaporthe-pod-&-stem-blight    6
##                Var1 Freq
## 61     2-4-d-injury   16
## 69    cyst-nematode   14
## 74 herbicide-injury    8
##             Var1 Freq
## 61  2-4-d-injury   16
## 69 cyst-nematode   14
##                          Var1 Freq
## 41                2-4-d-injury   16
## 49               cyst-nematode   14
## 50 diaporthe-pod-&-stem-blight   15
```

```
## 54            herbicide-injury    8
## 56            phytophthora-rot   68
##           Var1 Freq
## 81 2-4-d-injury   16
##           Var1 Freq
## 81 2-4-d-injury    1
##                           Var1 Freq
## 61               2-4-d-injury   16
## 69              cyst-nematode   14
## 70 diaporthe-pod-&-stem-blight   15
## 74            herbicide-injury    8
## 76            phytophthora-rot   68
##                           Var1 Freq
## 61               2-4-d-injury   16
## 69              cyst-nematode   14
## 70 diaporthe-pod-&-stem-blight   15
## 74            herbicide-injury    8
## 76            phytophthora-rot   68
##                           Var1 Freq
## 61               2-4-d-injury   16
## 69              cyst-nematode   14
## 70 diaporthe-pod-&-stem-blight    6
## 74            herbicide-injury    8
## 76            phytophthora-rot   68
##           Var1 Freq
## 41 2-4-d-injury   16
## [1] Var1 Freq
## <0 rows> (or 0-length row.names)
##                           Var1 Freq
## 69              cyst-nematode   14
## 70 diaporthe-pod-&-stem-blight   15
## 76            phytophthora-rot   55
##                           Var1 Freq
## 69              cyst-nematode   14
## 70 diaporthe-pod-&-stem-blight   15
## 76            phytophthora-rot   55
##                           Var1 Freq
## 69              cyst-nematode   14
## 70 diaporthe-pod-&-stem-blight   15
## 76            phytophthora-rot   55
##                           Var1 Freq
## 41               2-4-d-injury   16
## 49              cyst-nematode   14
## 50 diaporthe-pod-&-stem-blight   15
## 56            phytophthora-rot   55
##                           Var1 Freq
## 49              cyst-nematode   14
## 50 diaporthe-pod-&-stem-blight   15
## 56            phytophthora-rot   55
##                           Var1 Freq
## 61               2-4-d-injury   16
## 69              cyst-nematode   14
## 70 diaporthe-pod-&-stem-blight   15
## 74            herbicide-injury    8
```

```
## 76            phytophthora-rot   55
##           Var1 Freq
## 41 2-4-d-injury   16
##                           Var1 Freq
## 41               2-4-d-injury   16
## 49              cyst-nematode   14
## 50 diaporthe-pod-&-stem-blight   15
## 54            herbicide-injury    8
## 56            phytophthora-rot   68
##                Var1 Freq
## 81     2-4-d-injury   16
## 89    cyst-nematode   14
## 94 herbicide-injury    8
##                Var1 Freq
## 81     2-4-d-injury   16
## 89    cyst-nematode   14
## 94 herbicide-injury    8
##                Var1 Freq
## 41     2-4-d-injury   16
## 49    cyst-nematode   14
## 54 herbicide-injury    8
## 56 phytophthora-rot   68
##                Var1 Freq
## 61     2-4-d-injury   16
## 69    cyst-nematode   14
## 74 herbicide-injury    8
##                Var1 Freq
## 41     2-4-d-injury   16
## 49    cyst-nematode   14
## 54 herbicide-injury    8
##                Var1 Freq
## 61     2-4-d-injury   16
## 69    cyst-nematode   14
## 74 herbicide-injury    8
##                Var1 Freq
## 41     2-4-d-injury   16
## 49    cyst-nematode   14
## 54 herbicide-injury    8
##                Var1 Freq
## 81     2-4-d-injury   16
## 96 phytophthora-rot   68
##                Var1 Freq
## 81     2-4-d-injury   16
## 89    cyst-nematode   14
## 94 herbicide-injury    8
## 96 phytophthora-rot   68
##                Var1 Freq
## 41     2-4-d-injury   16
## 54 herbicide-injury    8
## 56 phytophthora-rot   68
##                Var1 Freq
## 41     2-4-d-injury   16
## 54 herbicide-injury    8
## 56 phytophthora-rot   68
```

```
##                Var1 Freq
## 41     2-4-d-injury   16
## 49    cyst-nematode   14
## 54 herbicide-injury    8
## 56 phytophthora-rot   68
##                Var1 Freq
## 41     2-4-d-injury   16
## 54 herbicide-injury    8
## 56 phytophthora-rot   68
##                Var1 Freq
## 41     2-4-d-injury   16
## 49    cyst-nematode   14
## 54 herbicide-injury    8
## 56 phytophthora-rot   68
##                           Var1 Freq
## 61                2-4-d-injury   16
## 70 diaporthe-pod-&-stem-blight   15
```

While operationally, this doesn't provide us with much insight, it does confirm the heavy class imbalance in missing values concentrated in the five classes aforementioned.

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

We can use utilize both strategies in tandem: Leaf.mild, myeclium, and sclerotia are all near-zero variance predictors. We can begin by considering eliminating them.

Because we are primarily working with categorical data, looking at correlation doesn't provide much value.

```
#eliminate
predictors.v1 = subset(predictors,select=-c(leaf.mild,mycelium,sclerotia))
#observe our missing values again
nas = sapply(predictors.v1, function(x) sum(is.na(x)))
nas[order(nas,decreasing=TRUE)]
```

```
##            hail           sever        seed.tmt         lodging            germ
##             121             121             121             121             112
## fruiting.bodies     fruit.spots   seed.discolor      shriveling     leaf.shread
##             106             106             106             106             100
##            seed     mold.growth       seed.size       leaf.halo       leaf.marg
##              92              92              92              84              84
##       leaf.size       leaf.malf      fruit.pods          precip    stem.cankers
##              84              84              84              38              38
##   canker.lesion       ext.decay    int.discolor     plant.stand           roots
##              38              38              38              36              31
##            temp       crop.hist    plant.growth            stem            date
##              30              16              16              16               1
##        area.dam          leaves
##               1               0
```

the leaves variable has no missing values, leaving us with only 31 problematic variables. From here, we are best off using modal or median imputation for the rest of the dataset - removing up to 121 rows of data introduces an incredibly large amount of bias into our modeling efforts, so we have to rely as much as possible on existing data.

2. (10 points) The caret package contains a QSAR data set from Mente and Lombardo (2005). Here, the ability of a chemical to permeate the blood-brain barrier was experimentally determined for 208 compounds. 134 descriptors were measured for each compound.

a. Start R and use these commands to load the data: library(caret) data(BloodBrain) # use ?BloodBrain to see more details The numeric outcome is contained in the vector logBBB while the predic- tors are in the data frame bbbDescr.
b. Do any of the individual predictors have degenerate distributions?
c. Generally speaking, are there strong relationships between the predictor data? If so, how could correlations in the predictor set be reduced? Does this have a dramatic effect on the number of predictors available for modeling?

```
#a. load data
library(caret)
data(BloodBrain)
```

Variables negative, peoe_vsa2.1 and 3.1, a_acid, vsa_acid, frac.anion7, and alert are degenerate variables.

```
#b. identifying degenerate distributions
names(bbbDescr)[nearZeroVar(bbbDescr)]
```

```
## [1] "negative"     "peoe_vsa.2.1" "peoe_vsa.3.1" "a_acid"       "vsa_acid"
## [6] "frac.anion7." "alert"
```

Next, we find relationships in the predictor set. Given the number of variables we have to analyze, it would be more convenient to create a findCorrelation() search of the corelogram of all predictors. We look at absolute correlation for simplicity, and define 0.7 as a problematic point of interest for our colinearity problem:

```
predictors.cor <- cor(bbbDescr)
highCor <- caret::findCorrelation(x=predictors.cor,cutoff=0.7,verbose = FALSE)
numVars <- length(bbbDescr)
p_colinear <- length(highCor)/numVars
print(p_colinear)
```

```
## [1] 0.5597015
```

almost 56% of our predictors experience a potential colinearity problem, which creates a massive impediment to properly modeling our response variable. We can attempt to fix this with centering and scaling:

```
trans <- preProcess(bbbDescr,method=c("center","scale"))
df <- predict(trans,bbbDescr)
cor.df <- (cor(df))
highCor <- caret::findCorrelation(x=cor.df,cutoff=0.7,verbose = FALSE)
numVars <- length(bbbDescr)
p_colinear <- length(highCor)/numVars
print(p_colinear)
```

```
## [1] 0.5597015
```

Centering and Scaling doesn't affect our problem at all - a BoxCox transformation was omitted from the report due to the fact it increased the number of correlations altogether. Given this, we can conduct PCA to look for improvements:

```
trans <- preProcess(bbbDescr,method=c("center","scale","pca"))
df <- predict(trans,bbbDescr)
cor.df <- (cor(df))
highCor <- caret::findCorrelation(x=cor.df,cutoff=0.7,verbose = FALSE)
numVars <- length(bbbDescr)
p_colinear <- length(highCor)/numVars
print(p_colinear)
```

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

PCA transformation vastly reduces the size of our predictor set while considerably removing the amount of colinearity present, but ultimately absolves us of any interpretability.

4. (15 points) Consider the permeability data set described in Sect. 1.4. of the textbook. The objective for these data is to use the predictors to model compounds' permeability.

a. What data splitting method(s) would you use for these data? Explain.

b. Using tools described in this chapter, provide code for implementing your approach(es).

c. Because we have a small sample size and sparse predictors, a single test sample is unadvised because every sample may be needed for modeling and we run the risk of precision loss. Because the response is highly skewed, we may be interested in looking at nonrandom sampling. The chapter advises us that in the instance of clinical data, breaking down by disease stage may be a good means of generalizing - similarly, we can break down data by the severity of the injury type to gain some insight. As a result, we may want to pursue k-fold cross validation or repeated k-fold cross validation.

d. Code:

```
require(AppliedPredictiveModeling)
```

```
## Loading required package: AppliedPredictiveModeling
```

```
## Warning: package 'AppliedPredictiveModeling' was built under R version 4.0.5
```

```
data(permeability)
#we use the fingerprints dataset
```

```
require(caret)
#create data partition for training
set.seed(1)
trainingRows <- createDataPartition(permeability,p = 0.80, list=FALSE)
trainPredict <- fingerprints[trainingRows,]
trainClass   <- permeability[trainingRows,]
testPredict  <- fingerprints[-trainingRows,]
testClass    <- permeability[-trainingRows,]

#k-fold cross validation
cvSplits <- createFolds(trainClass,k=10,returnTrain = TRUE)

#get first 90% of data
```

```
cvPredictors1 <- trainPredict[cvSplits$Fold01,];
cvResponse1   <- trainClass[cvSplits$Fold01];

#(insert model here...)

#repeated k-fold cross validation  (3)
rcv = createMultiFolds(trainClass,k=10,times=3)

#get first 90% of data for first set of folds, etc.
rcv1 <- trainPredict[rcv$Fold01.Rep1,]
rcvClasses1 <- trainClass[rcv$Fold01.Rep1]

#insert model here
```

We can also combine some of these steps with the train() function, with a naive bayes example:

```
require(caret)
train_control <- trainControl(method = "repeatedcv",
                              number = 10, repeats = 3)

dataset = as.data.frame(fingerprints)
dataset$permeability = permeability

model <- train(permeability~., data = dataset,
               trControl = train_control, method = "glm")
```

5. (20 points) Partial least squares was used to model the yield of a chemical manufacturing process (Sect. 1.4). The data can be found in the AppliedPredictiveModeling package and can be loaded using library(AppliedPredictiveModeling) data(ChemicalManufacturingProcess)

The objective of this analysis is to find the number of PLS components that yields the optimal R2 value. PLS models with 1 through 10 components were each evaluated using five repeats of 10-fold cross-validation and the results are presented in the following table:

a. Using the "one-standard error" method, what number of PLS components provides the most parsimonious model?

b. Compute the tolerance values for this example. If a 10 % loss in R2 is acceptable, then what is the optimal number of PLS components?

c. Several other models with varying degrees of complexity were trained and tuned and the results are presented in Figure below. If the goal is to select the model that optimizes R2, then which model(s) would you choose, and why?

d. Prediction time, as well as model complexity are other factors to consider when selecting the optimal model(s). Given each model's prediction time, model complexity, and R2 estimates, which model(s) would you choose, and why?

e. 3 components are optimal.

f. If a 10% loss in accuracy is acceptable, and the best value is 54.5% (or 0.545), then any value higher than 49.5% is acceptable. With that, we can assert that 2 components are optimal.

g. For R-squared maximization, we would choose the SVM because it has the highest range of R-squared values.We might also consider boosted linear regression as a runner up.

h. we would choose boosted linear regression because it is the only one whose confidence interval overlaps with that of the svm performance but has a very low complexity compared to Svm.

6. (20 points) Brodnjak-Vonina et al. (2005) develop a methodology for food laboratories to determine the type of oil from a sample. In their procedure, they used a gas chromatograph (an instrument that separates chemicals in a sample) to measure seven different fatty acids in an oil. These measurements would then be used to predict the type of oil in a food sample. To create their model, they used 96 samples of seven types of oils.

These data can be found in the caret package using data(oil). The oil types are contained in a factor variable called oilType. The types are pumpkin (coded as A), sunflower (B), peanut (C), olive (D), soybean (E), rapeseed (F) and corn (G).

a. Use the sample function in base R to create a completely random sample of 60 oils. How closely do the frequencies of the random sample match the original samples? Repeat this procedure several times to understand the variation in the sampling process.
b. Use the caret package function createDataPartition to create a stratified random sample. How does this compare to the completely random samples?
c. With such a small sample size, what are the options for determining performance of the model? Should a test set be used?
d. One method for understanding the uncertainty of a test set is to use a confidence interval. To obtain a confidence interval for the overall accu- racy, the based R function binom.test can be used. It requires the user to input the number of samples and the number correctly classified to calculate the interval. For example, suppose a test set sample of 20 oil samples was set aside and 76 were used for model training. For this test set size and a model that is about 80 % accurate (16 out of 20 correct), the confidence interval would be computed using binom.test(16, 20) Exact binomial test data: 16 and 20 number of successes = 16, number of trials = 20, p-value = 0.01182 alternative hypothesis: true probability of success is not equal to 0.5 95 percent confidence interval: 0.563386 0.942666 sample estimates: probability of success 0.8 In this case, the width of the 95% confidence interval is 37.9%. Try different samples sizes and accuracy rates to understand the trade-off between the uncertainty in the results, the model performance, and the test set size.

```
#load in our caret package
require(caret)
require(e1071)
```

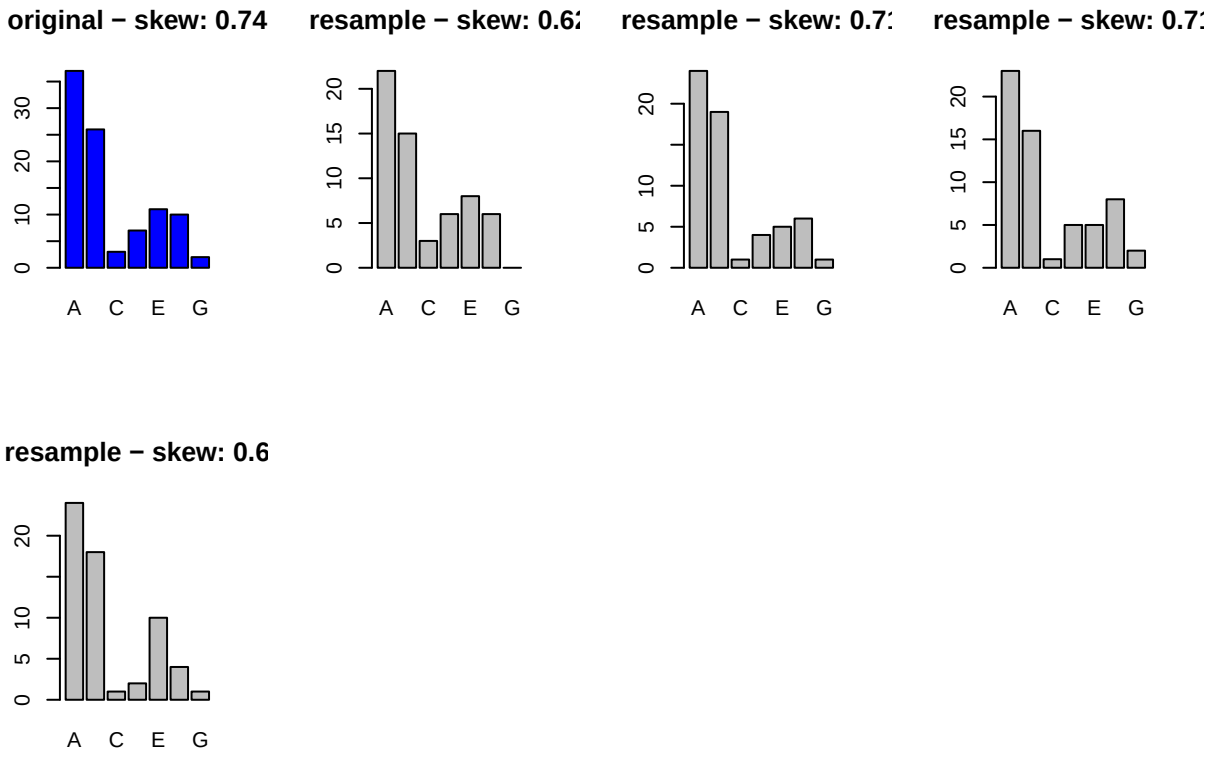
```
## Loading required package: e1071
```

```
data(oil)

par(mfrow=c(2,4))
oil.sk = round(skewness(table(oilType)),2)
plot(oilType,main=paste("original - skew:",oil.sk),col='blue')

set.seed(20)

for (i in c(1:4)){
  #get n=60 samples
  n <- sample(oilType,size=60)
  n.sk = round(skewness(table(n)),2)
  plot(n,main=paste("resample - skew:",n.sk))
}
```



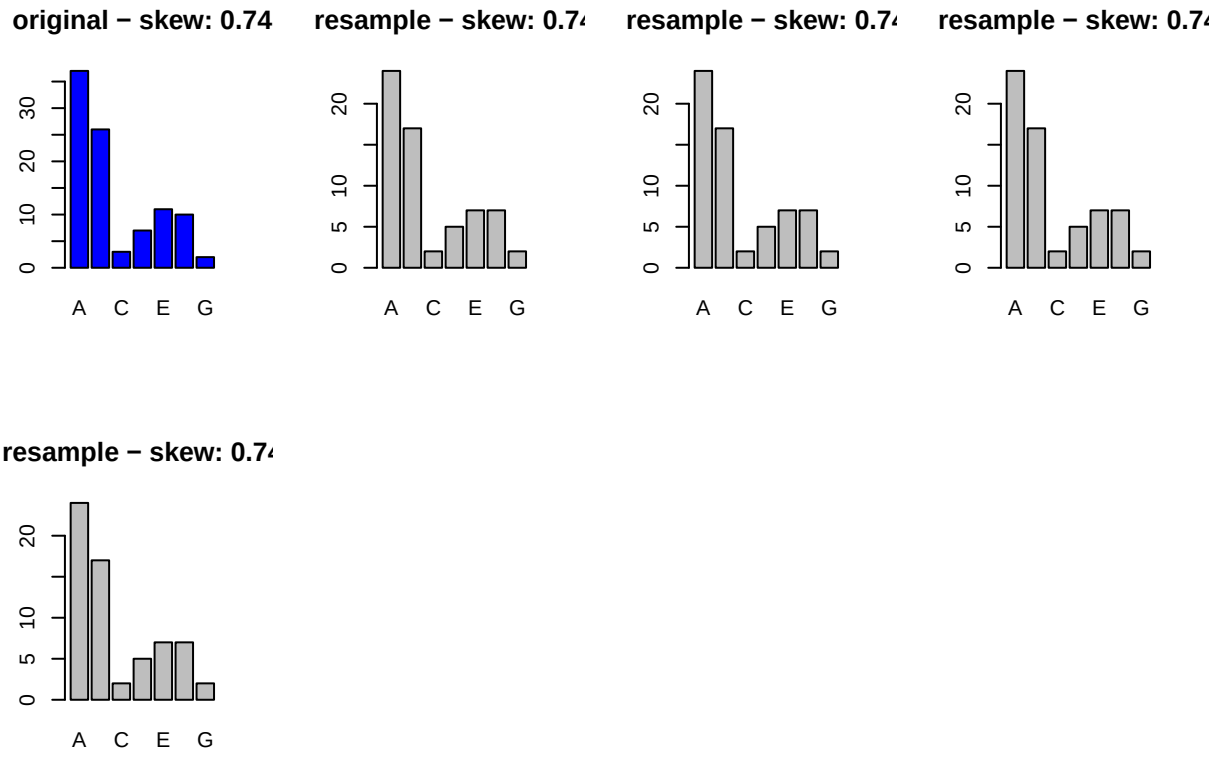
From resampling four times, we find that the most consistent overlap in sampling is for oil types A and B, with the most variation in all other types. Despite these variations (especially in skewness), most distributions tend to maintain a somewhat similar shape to the original dataset.

b. Use the caret package function createDataPartition to create a stratified random sample. How does this compare to the completely random samples?

```
par(mfrow=c(2,4))
oil.sk = round(skewness(table(oilType)),2)
plot(oilType,main=paste("original - skew:",oil.sk),col='blue')

set.seed(20)

for (i in c(1:4)){
  set.seed(i)
  #get n=60 samples
  n <- createDataPartition(oilType,p=0.625,list=FALSE)
  n <- oilType[n]
  n.sk = round(skewness(table(n)),2)
  plot(n,main=paste("resample - skew:",n.sk))
}
```



Stratified resampling creates an almost perfect rendition of the original dataset.

c. With such a small sample size, what are the options for determining performance of the model? Should a test set be used?

Given how small our sample size is, a testing set would be ill-advised since a test set would be characterized by enough uncertainty that each test set would produce different results. A better alternative for handling model performance would be to use leave-one-out cross validation, which would reduce the randomness and bias inherent in smaller datasetts.

d. One method for understanding the uncertainty of a test set is to use a confidence interval. To obtain a confidence interval for the overall accuracy, the based R function binom.test can be used. It requires the user to input the number of samples and the number correctly classified to calculate the interval. For example, suppose a test set sample of 20 oil samples was set aside and 76 were used for model training. For this test set size and a model that is about 80 % accurate (16 out of 20 correct), the confidence interval would be computed using

binom.test(16, 20)

Exact binomial test data: 16 and 20 number of successes = 16, number of trials = 20, p-value = 0.01182 alternative hypothesis: true probability of success is not equal to 0.5 95 percent confidence interval: 0.563386 0.942666 sample estimates: probability of success 0.8 In this case, the width of the 95% confidence interval is 37.9%. Try different samples sizes and accuracy rates to understand the trade-off between the uncertainty in the results, the model performance, and the test set size.

```
#there are two scales on which to adjust: accuracy and sample size.
#we will set a boundary of accuracy for 0.5 - 0.7 and one for sample size from 10 to 15

accuracy = seq(0.5,0.7,0.1)
samples  = seq(15,10, -1)
v <- c()

for (i in accuracy)
{
  for (j in samples)
  {
    x = (binom.test(round(j*i),j))
    uncertainty = x$conf.int[2] - x$conf.int[1]
    v<-rbind(v,c(i,j,uncertainty))
  }
}
```

From here, we can observe the head and tial of our vector of accuracy, sample size, and uncertainty range:

```
head(v)
```

```
##      [,1] [,2]      [,3]
## [1,]  0.5   15 0.5214719
## [2,]  0.5   14 0.5392789
## [3,]  0.5   13 0.5564221
## [4,]  0.5   12 0.5781107
## [5,]  0.5   11 0.5987183
## [6,]  0.5   10 0.6258279
```

```
tail(v)
```

```
##       [,1] [,2]      [,3]
## [13,]  0.7   15 0.4979552
## [14,]  0.7   14 0.4971459
## [15,]  0.7   13 0.5233413
## [16,]  0.7   12 0.5518784
## [17,]  0.7   11 0.5495248
## [18,]  0.7   10 0.5857133
```

A face-value overview demonstrates that increasing accuracy while holding sample size constant reduces uncertainty in the model, which increasing sample size and holding accuracy constant decreases uncertainty.