ZooDataPCA

Colleen Giannotta

8/15/2020

```
# load libraries
library(ggbiplot)
```

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

```
## Loading required package: plyr
```

```
## Loading required package: scales
```

```
## Loading required package: grid
```

```
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 4.0.2
```

```
##
## Attaching package: 'ggfortify'
```

```
## The following object is masked from 'package:ggbiplot':
##
##     ggbiplot
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.0.2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

This article will explain how to perform principal component analysis (PCA) on a multi-dimensional dataset. The data consists of 101 observations of zoo animals with 18 features per observation. The features included for each animal are:

1. animal_name

2. hair
3. feathers
4. eggs
5. milk
6. airborne
7. aquatic
8. predator
9. toothed
10. backbone
11. breathes
12. venomous
13. fins
14. legs
15. tail
16. domestic
17. catsize
18. class_type

Features 2-13 and 15-17 have boolean values, legs takes numeric values 2-8, animal_name is the actual name of the animal (a string) and class_type is a numeric value 1-7 indicating mammal, bird, fish, reptile, amphibian, bug, or invertebrate.

Given the large number of features in this dataset, it is difficult to visualize patterns that may exist. We will therefore reduce the dimensions using PCA.

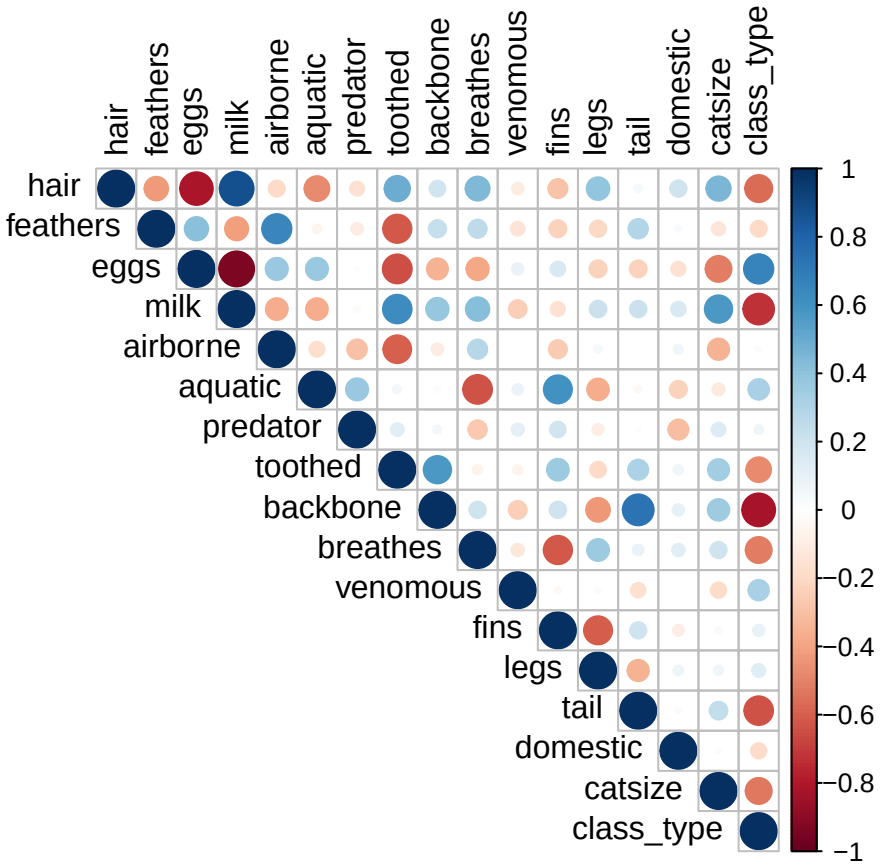
```
# load data
zoo = read.csv("zoo.csv", header = TRUE)
classes = c("Mammal", "Bird", "Reptile", "Fish", "Amphibian", "Bug", "Invertebrate")
zoo$class_name = 0

# add column with class_name
for (i in 1:nrow(zoo)) {
  if(zoo[i,]$class_type==1){
    zoo[i,]$class_name = "Mammal"
  } else if(zoo[i,]$class_type==2){
    zoo[i,]$class_name = "Bird"
  } else if(zoo[i,]$class_type==3){
    zoo[i,]$class_name = "Reptile"
  } else if(zoo[i,]$class_type==4){
    zoo[i,]$class_name = "Fish"
  } else if(zoo[i,]$class_type==5){
    zoo[i,]$class_name = "Amphibian"
  } else if(zoo[i,]$class_type==6){
    zoo[i,]$class_name = "Bug"
  } else if(zoo[i,]$class_type==7){
    zoo[i,]$class_name = "Invertebrate"
  }
}

head(zoo)
```

```
##   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
##   class_name
## 1     Mammal
## 2     Mammal
## 3       Fish
## 4     Mammal
## 5     Mammal
## 6     Mammal
```

Exploratory Data Analysis

```
# assess multicollinearity
corrplot(cor(zoo[,-c(1,19)]), method = "circle", type = "upper", tl.col="black")
```



The corrplot shows that there is strong correlation between certain variables (eggs and hair, backbone and tail, eggs and milk), which may influence the parameters and hypothesis testing for any models we build using this data. PCA can help with multicollinearity, as the data will be transformed into a new space with orthogonal basis vectors, which are linearly independent.

PCA in R from Scratch

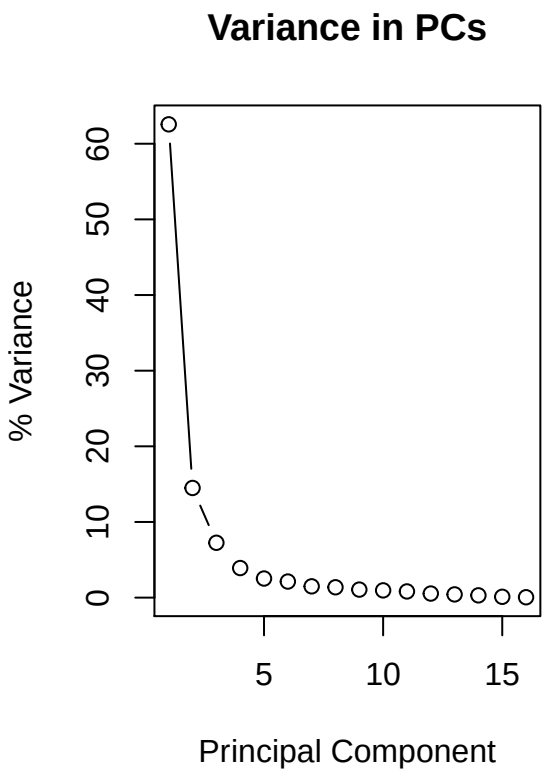
First we will implement each step of PCA from scratch, before using R's prcomp function to perform PCA in one simple step. We begin by calculating the covariance matrix for the zoo data, which holds the varance between each pair of dimensions in our dataset. We can only use numeric data for PCA, so we leave out the animal_name, class_type, and class_name information, as these will be used later for visualization.

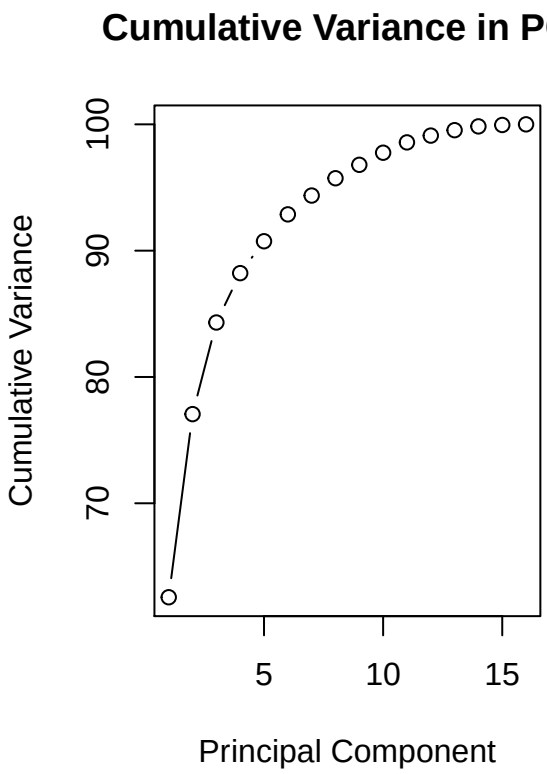
```
# calculate covariance matrix
zoo.cov = cov(zoo[,2:17])
```

Next, we use the covariance matrix to find the eigenvalues and eigenvectors. Eigenvectors are also known as principal components, and these orthogonal vectors are the directions that define the transformed space once we perform PCA. In other words, they form the basis for the transformed space. Because they are orthogonal, we can reduce the impact of multicollinearity in the original dataset. Since the new space will have fewer dimensions than the original data, we can more clearly visualize patterns and perform other analysis on the data. The number of principal components we select will determine the final number of dimensions in our transformed data.

The eigenvalues will tell us how much of the variance is accounted for in each eigenvector. Then we can select a given number of principal components to use in the transformation. The final transformed data will not capture all of the original data, but it will include enough of the patterns in the original data for us to be able to use it effectively.

```
# grid plots
par(mfrow=c(1,2))

# calculate eigenvectors and eigenvalues of covariance matrix
zoo.eigen = eigen(zoo.cov)

# determine proportion of variance accounted for by each principal component (eigenvector)
props = rep(0, length(zoo.eigen$values))
for (i in 1:length(props)) {
  props[i] = zoo.eigen$values[i] / sum(zoo.eigen$values) * 100
}

# plot variance per principal component
plot(x = c(1:length(zoo.eigen$values)), y = props, xlab = 'Principal Component', ylab = '% Variance',
     main = 'Variance in PCs', type = "b")

# save cumulative proportions
cumul.props = rep(0, length(zoo.eigen$values))
for (i in 1:length(props)) {
  cumul.props[i] = sum(props[1:i])
}

# plot cumulative variance in principal components
plot(x = c(1:length(zoo.eigen$values)), y = cumul.props, xlab = 'Principal Component',
     ylab = 'Cumulative Variance', main = 'Cumulative Variance in PCs', type = "b")
```





We see that the first and second components account for over 75% of the variance in the data. Therefore we can use these two principal components to get a picture of patterns that exist in this data.

Now we can form the feature vector, which has the first and second principal components as its columns. This vector will transform our original data into the final dataset, which will have the same number of observations, but only two dimensions instead of 16.

```
# form feature vector
feat.vect = zoo.eigen$vectors[,1:2]
row.names(feat.vect) = colnames(zoo)[2:17]
colnames(feat.vect) = c("PC1", "PC2")
feat.vect
```

```
##                   PC1         PC2
## hair       0.104560005  0.38346999
## feathers -0.041193075 -0.17562243
## eggs     -0.063451200 -0.44032456
## milk       0.061244193  0.45088946
## airborne   0.008718832 -0.21319749
## aquatic  -0.093907817 -0.12603902
## predator -0.029338280  0.01740505
## toothed  -0.041852339  0.39915885
## backbone -0.077852062  0.22151954
## breathes   0.079763337  0.11207428
## venomous   0.002238808 -0.05012928
## fins     -0.113607095  0.02599768
## legs       0.967811203 -0.08205807
```

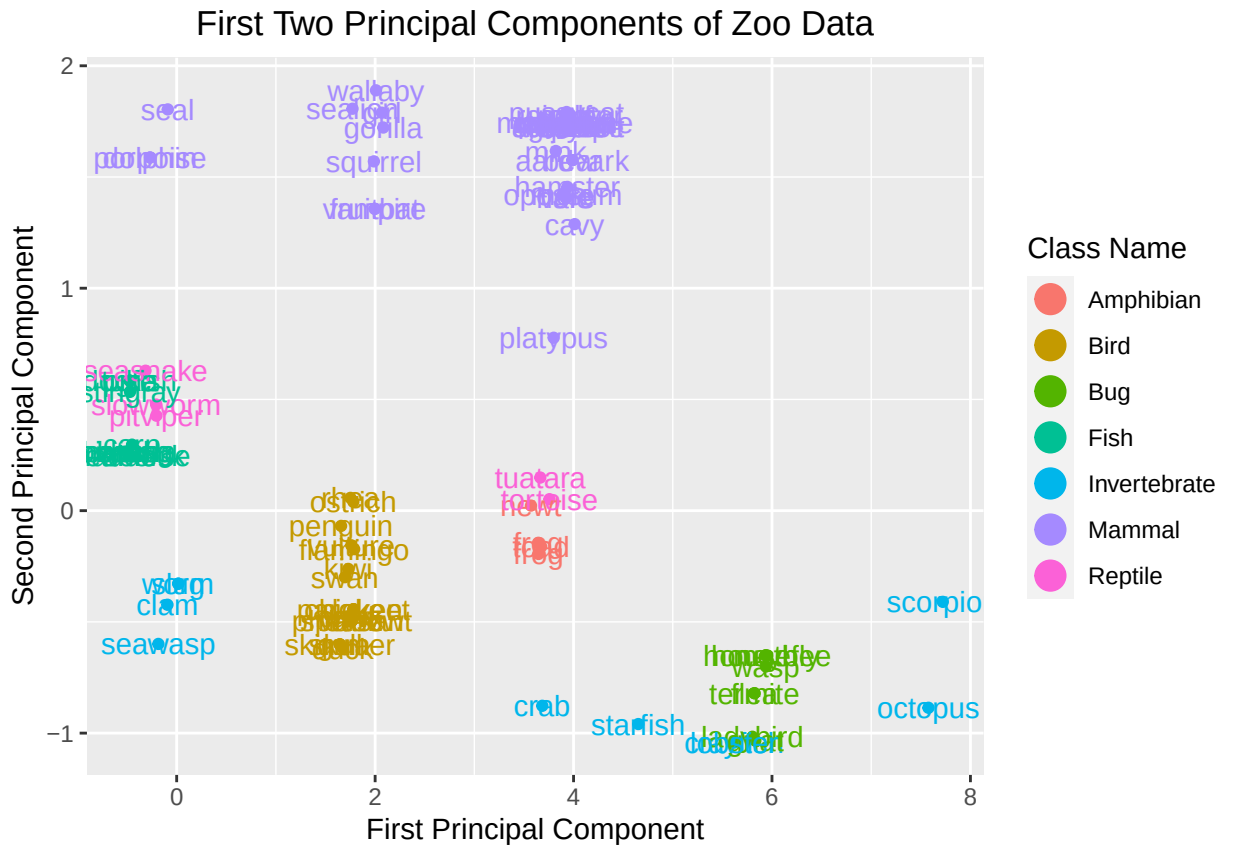
```
## tail     -0.073732679  0.16721638
## domestic  0.014434464  0.04929813
## catsize   0.021172542  0.31938261
```

```r
# use feature vector to create final data
pc1 = as.matrix(zoo[,2:17]) %*% feat.vect[,1]
pc2 = as.matrix(zoo[,2:17]) %*% feat.vect[,2]
pca.data = data.frame(animal_name = zoo[,1], class_name = zoo[,19], pc1, pc2)
head(pca.data)
```

```
##   animal_name class_name        pc1       pc2
## 1    aardvark     Mammal  3.9889422 1.5756675
## 2    antelope     Mammal  3.9445478 1.7254788
## 3        bass       Fish -0.4937415 0.2649339
## 4        bear     Mammal  3.9889422 1.5756675
## 5        boar     Mammal  3.9152095 1.7428839
## 6     buffalo     Mammal  3.9445478 1.7254788
```

Now that we have a dataset with two key dimensions instead of 16, we can plot the data and see what patterns emerge.

```r
# Plot Principal Components
ggplot(pca.data, aes(pc1, pc2, color=class_name)) +
  geom_point()+
  geom_text(aes(label = animal_name), show.legend = FALSE, size = 4) +
  xlab("First Principal Component") +
  ylab("Second Principal Component") +
  ggtitle("First Two Principal Components of Zoo Data")+
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(color="Class Name") +
  guides(colour = guide_legend(override.aes = list(size=5)))
```



It is easier now to see the animals mostly clustered by their class names. Mammals are grouped toward the top of the plot, with a few notable exceptions. Aquatic mammals are further to the left, as are reptiles and fish, indicating that mammals such as seals and dolphins share features with these classes. The platypus, a unique mammal that lays eggs, is off on its own, between mammals and reptiles. Invertebrates are somewhat scattered, depending on which other classes they are most similar to. This plot gives us an interesting take on how the variables included in the original data set can be used to identify similarities among these animals.

PCA in R using prcomp

We will now perform PCA on the zoo dataset using R's prcomp function. This may have slightly different results, as prcomp uses singular value decomposition, instead of calculating the eigenvectors and eigenvalues of the covariance matrix, as we did previously. R also has the princomp function, which uses the same method shown above.

Plotting PCA

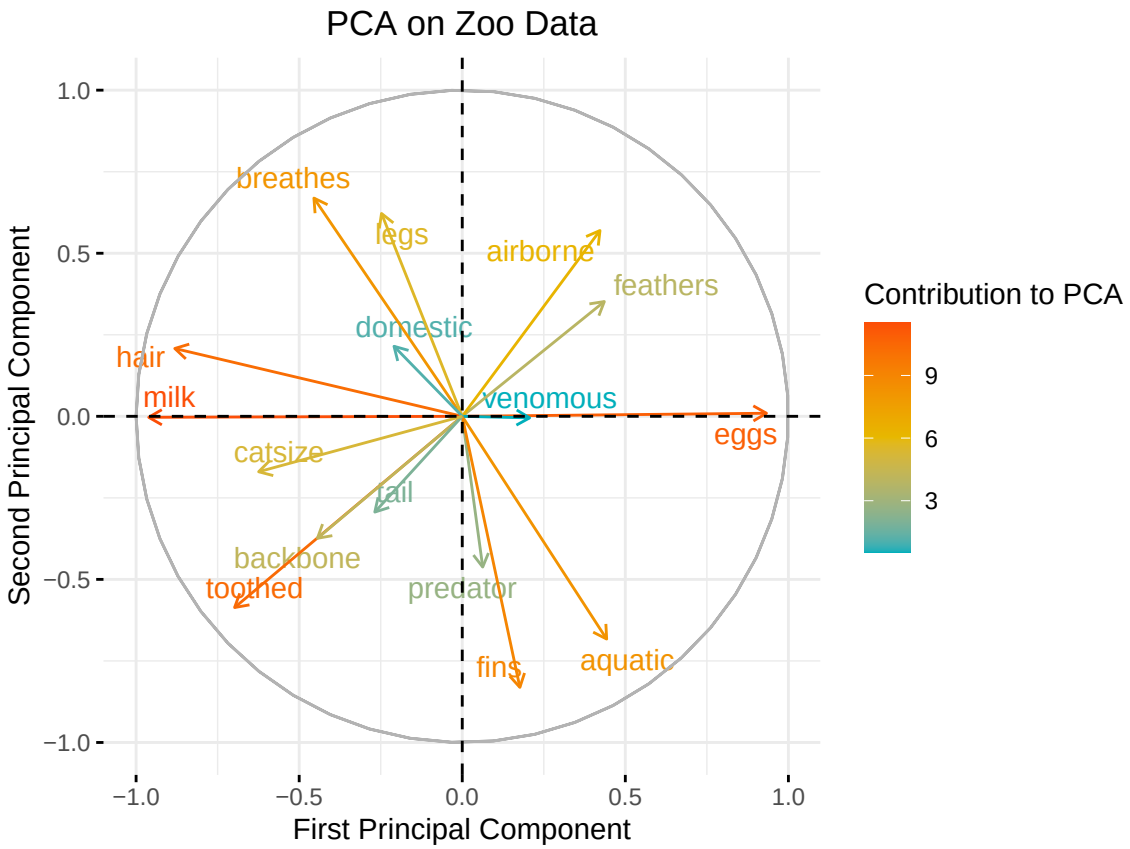
```
# perform PCA and summarize
zoo.pca = prcomp(zoo[,2:17], center = TRUE, scale = TRUE) # leave out animal name and class information
summary(zoo.pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.1612 1.8279 1.5377 1.10948 0.97670 0.86373 0.75021
```

```
## Proportion of Variance 0.2919 0.2088 0.1478 0.07693 0.05962 0.04663 0.03518
## Cumulative Proportion  0.2919 0.5007 0.6485 0.72546 0.78508 0.83170 0.86688
##                            PC8     PC9    PC10    PC11    PC12   PC13    PC14
## Standard deviation     0.71584 0.66917 0.61632 0.52686 0.45905 0.3599 0.34343
## Proportion of Variance 0.03203 0.02799 0.02374 0.01735 0.01317 0.0081 0.00737
## Cumulative Proportion  0.89891 0.92689 0.95063 0.96798 0.98115 0.9892 0.99662
##                           PC15    PC16
## Standard deviation     0.19045 0.13335
## Proportion of Variance 0.00227 0.00111
## Cumulative Proportion  0.99889 1.00000
```

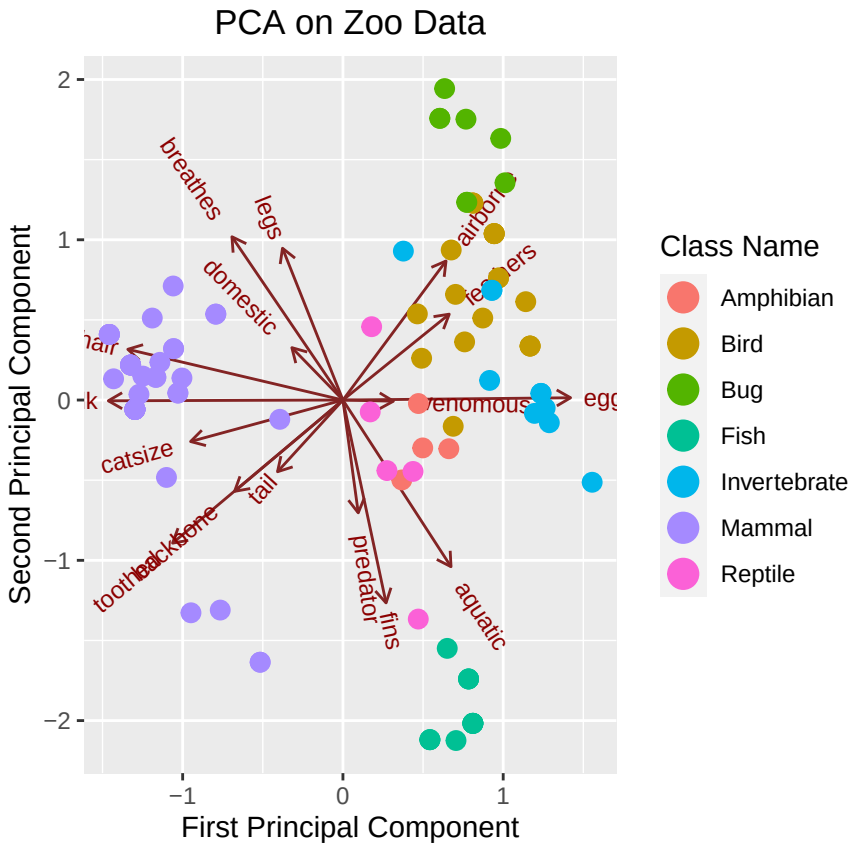
We see that the first and second principal components account for over 50% of the variance in the data. The more principal components we include in our analysis, the more variance in the data we account for, but the more complicated our model will be.

```
# plot feature contributions to principal components
fviz_pca_var(zoo.pca,
             col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE
             )+
  xlab("First Principal Component") +
  ylab("Second Principal Component") +
  ggtitle("PCA on Zoo Data")+
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(color="Contribution to PCA")
```



We see that features milk and eggs contribute the most to the first principal component, while legs and predator contribute more to the second principal component.

```
# plot how different features contribute to principal components
ggbiplot::ggbiplot(zoo.pca, color=zoo$class_name, repel=TRUE) +
  geom_point(aes(colour=zoo$class_name), size = 3) +
  theme(legend.direction ="vertical",
      legend.position = "right") +
  xlab("First Principal Component") +
  ylab("Second Principal Component") +
  ggtitle("PCA on Zoo Data") +
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(color="Class Name") +
  guides(colour = guide_legend(override.aes = list(size=5)))
```

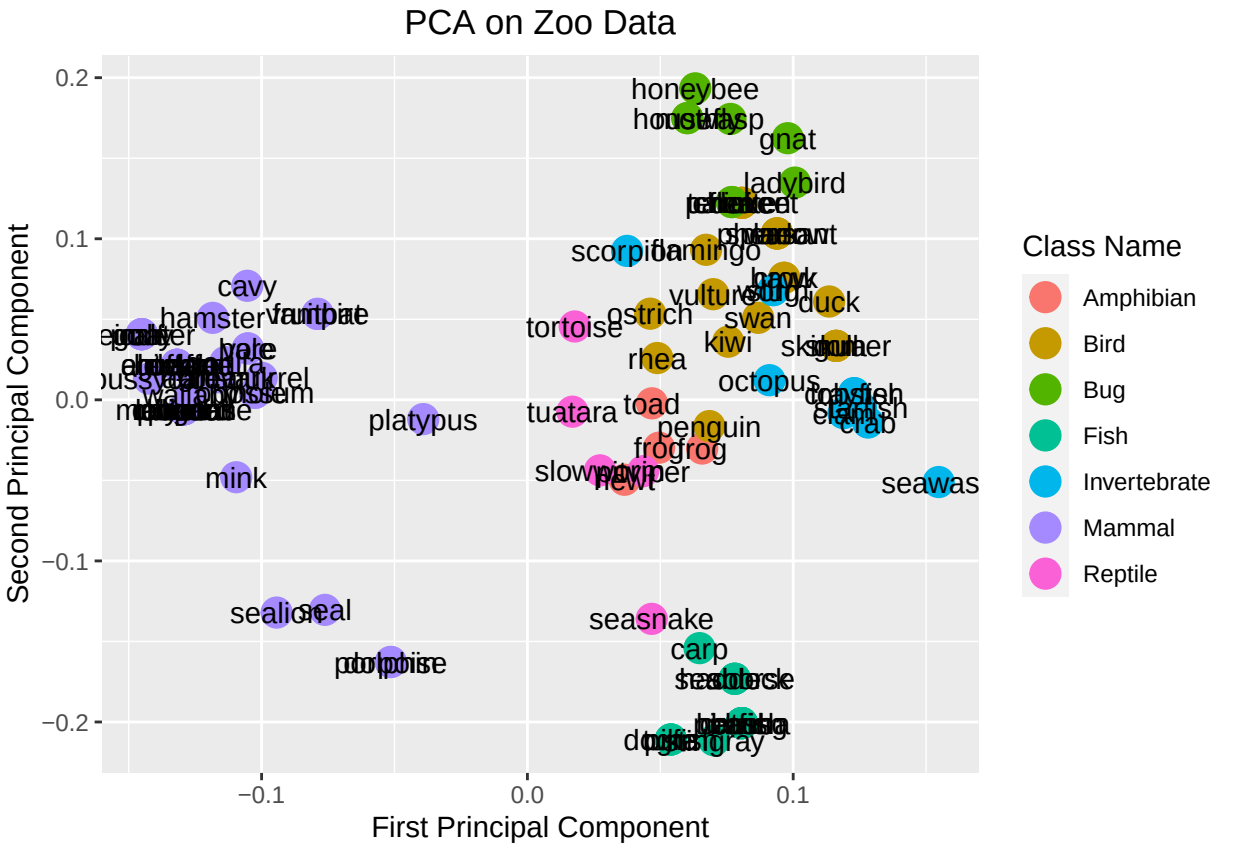


Here is another biplot of the data, with points included for all of the animals. We can see which features were most influential in grouping the animals: mammals have hair and produce milk, while fish and some mammals and reptils are aquatic and bugs and birds are airborne.

```
# plot pca
autoplot(zoo.pca, data = zoo, colour = 'class_name', size=5) +
  geom_text(label=zoo$animal_name)+
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(color="Class Name")+
  ggtitle("PCA on Zoo Data") +
  xlab("First Principal Component") +
```

```
  ylab("Second Principal Component")
```

```
## Warning: `select_()` is deprecated as of dplyr 0.7.0.
## Please use `select()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```



The results of this PCA differ from the PCA we did from scratch in the proportion of variance accounted for by the first two principal components, but the results are similar in showing how the animals are grouped by their features.

Conclusions

The conclusions we can draw from this analysis are not ground-breaking (we all knew that birds and bugs are airborne), but it helps to see the potential for visualizing patterns in larger, higher-dimensional data sets. There are other uses for PCA, such as reducing space needed for storing digital photographs.

References

https://uc-r.github.io/pca http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/118-principal-component-analysis-in-r-prcomp-vs-princomp/