Eigenpets

Load up the eigenpets data.

```
dog <- read.csv("~data/dog.csv", header=FALSE)
cat <- read.csv("~data/cat.csv", header=FALSE)
dim(dog)
```

```
## [1] 4096   80
```

```
dim(cat)
```

```
## [1] 4096   80
```

For data analysis, we want each image to be in a row. We also want 1 large matrix.

```
dog <- t(dog)
cat <- t(cat)
pet <- rbind(dog, cat)
```

We want to look at these images

```
disp_pet <- function(img_vector)  {
  cmap <- gray.colors(256, start=0, end=1)
  img <- matrix(img_vector, nrow=64, ncol=64, byrow=TRUE)
  image(img[,64:1], col=cmap, axes=FALSE, asp=1)
}
disp_pet(dog[1,])
```

```
disp_pet(dog[2,])
```

```
disp_pet(dog[3,])
```

```
disp_pet(cat[1,])
```

```
disp_pet(cat[2,])
```

```
disp_pet(cat[3,])
```

```
disp_pet(dog[77,])
```

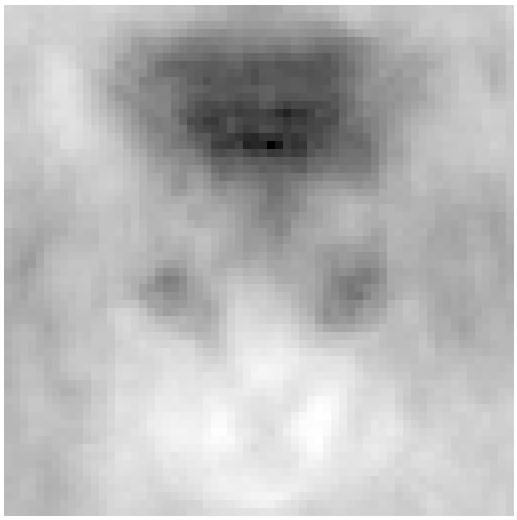
Compute the average cat and the averag dog

```
mean_cat <- colMeans(cat)
good_dog <- colMeans(dog)
disp_pet(mean_cat)
```

```
disp_pet(good_dog)
```

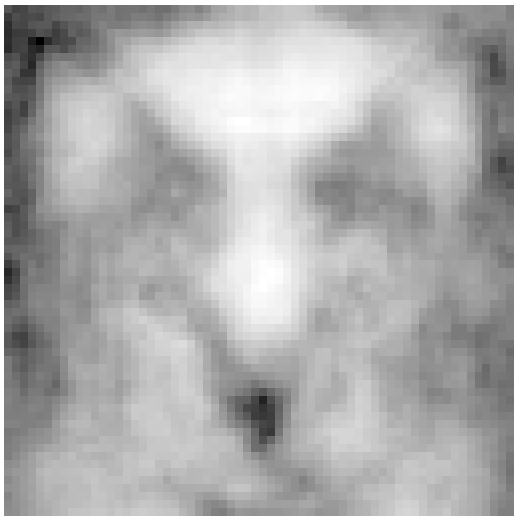
Calculate the difference between cats and dogs

```
diff <- mean_cat - good_dog
disp_pet(diff)
```



The Average Pet

```
disp_pet(colMeans(pet))
```



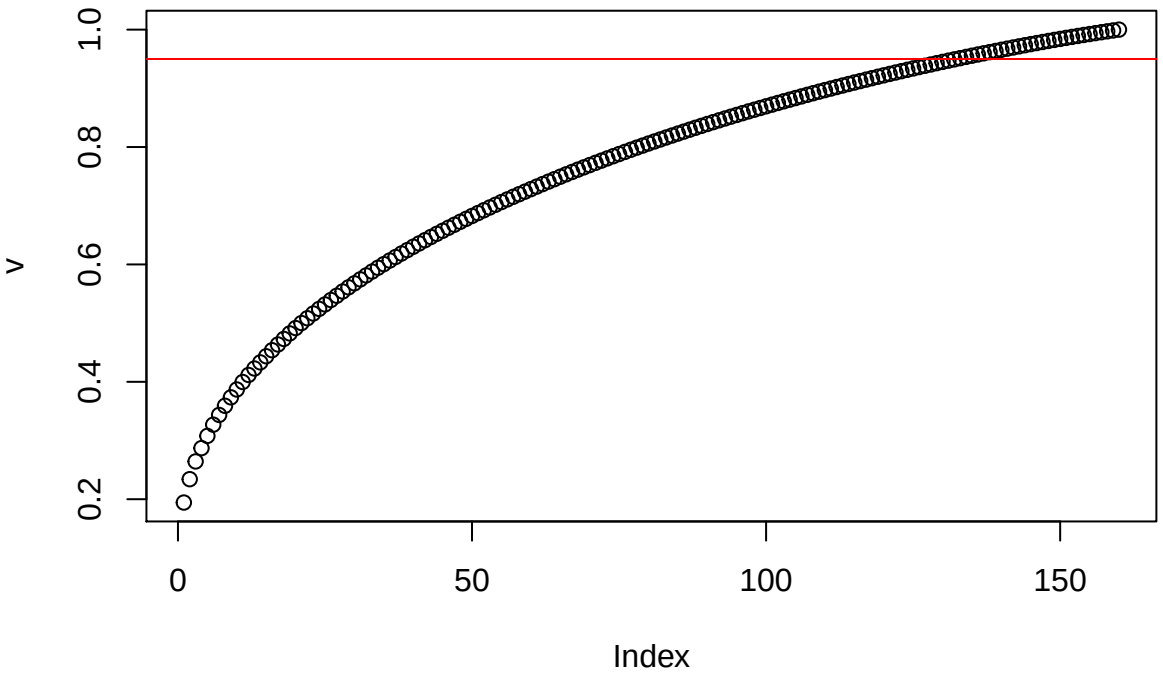
Use SVD to perform PCA

```
pet_svd <- svd(pet)
dim(pet_svd$v)
```

```
## [1] 4096  160
```

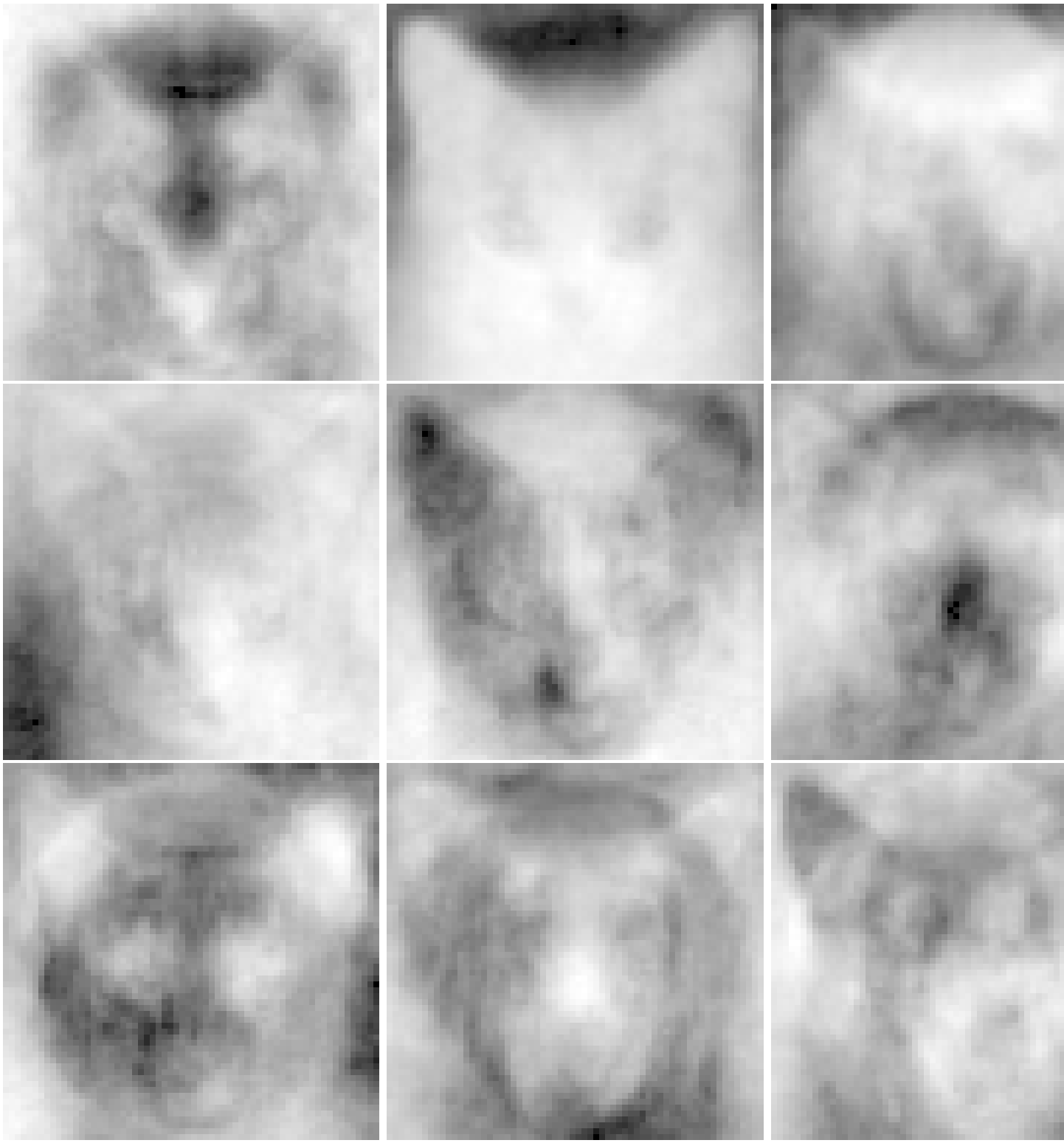
Variance described by each component

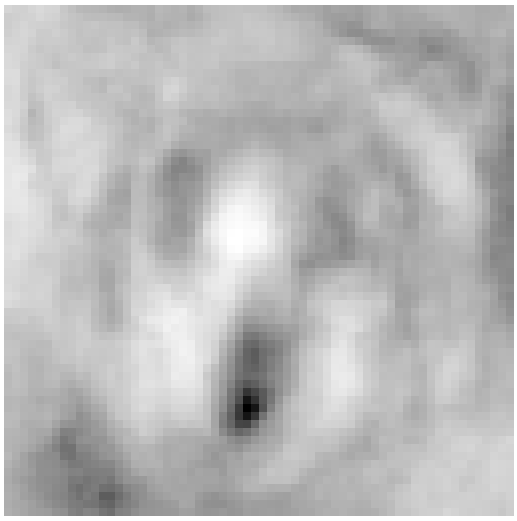
```
v <- cumsum(pet_svd$d)/sum(pet_svd$d)
plot(v)
abline(h=0.95, col="red")
```



Look at the eigenvectors

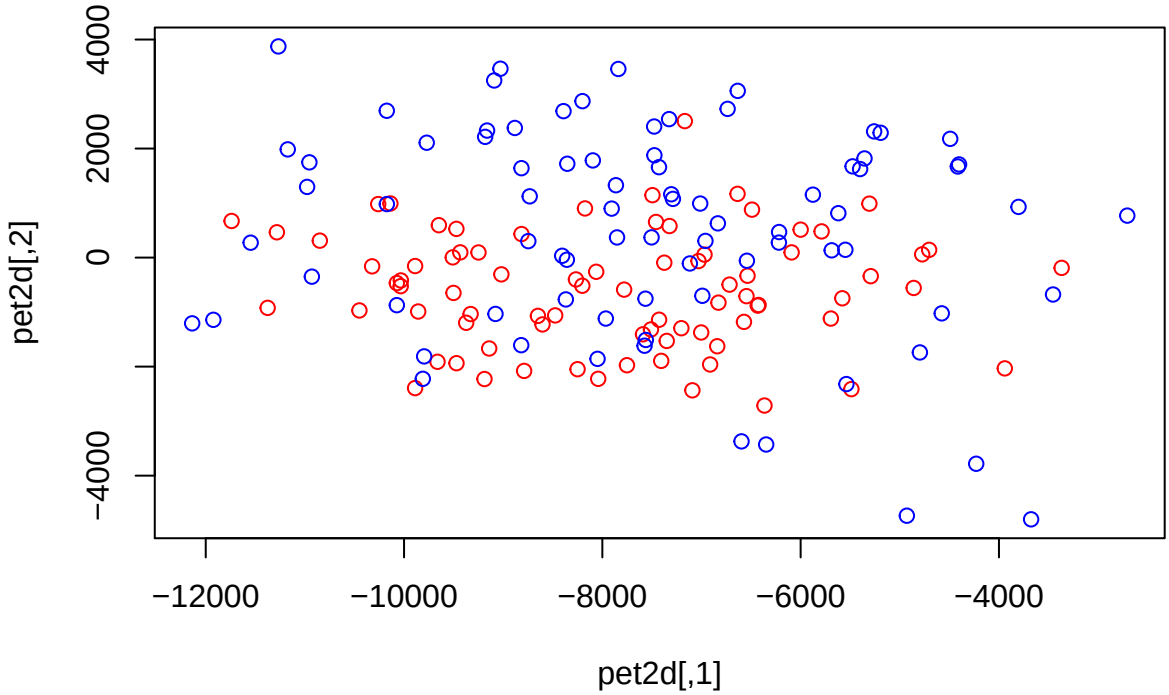
```
for(i in 1:10) {
  disp_pet(pet_svd$v[,i])
}
```





Project the pets into an optimal 2d space

```
w2<-pet_svd$v[,1:2]
pet2d <- pet %*% w2
plot(pet2d, col=ifelse(1:160<=80, "red", "blue"))
```



Let's look at the animals in the 2d space

```
dim(pet2d)
```

```
## [1] 160   2
```

Let's do this analysis with training and test set and then classify

```
scat <- cat[sample(nrow(cat)),]
sdog <- dog[sample(nrow(dog)),]
cat_training <- scat[1:60,]
```

```
dog_training <- sdog[1:60,]
pet_training <- rbind(cat_training, dog_training)
cat_test <-scat[61:80, ]
dog_test <-sdog[61:80, ]
pet_test <- rbind(cat_test, dog_test)
training_class <- as.factor(ifelse(1:120<=60, "cat", "dog"))
test_class<-as.factor(ifelse(1:40<=20, "cat", "dog"))
training_class
```

```
##   [1] cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat
##  [19] cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat
##  [37] cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat
##  [55] cat cat cat cat cat cat dog dog dog dog dog dog dog dog dog dog dog dog dog dog
##  [73] dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog
##  [91] dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog
## [109] dog dog dog dog dog dog dog dog dog dog dog dog
## Levels: cat dog
```

```
test_class
```

```
##  [1] cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat cat
## [20] cat dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog
## [39] dog dog
## Levels: cat dog
```

```
disp_pet(sdog[1,])
```

```
disp_pet(scat[1,])
```

Perform PCA and extract the first two components

```
pet_pca <- svd(pet_training)
w2 <- pet_pca$v[,1:2]
```

Project the training and test data into the 2d PCA space

```
pet_training_2d <- pet_training %*% w2
pet_test_2d <- pet_test %*% w2
```

Perform knn for 3 neighbors

```
library(class)
predicted_class <- knn(train=pet_training_2d, test=pet_test_2d, cl=training_class, k=3)
predicted_class
```

```
##  [1] cat cat cat dog cat cat cat cat dog dog dog dog cat cat dog cat cat dog dog
## [20] cat dog dog dog dog cat dog cat dog cat cat dog cat cat cat dog dog dog cat
## [39] dog dog
## Levels: cat dog
```

Count the correct answers

```
correct <- sum(predicted_class == test_class)
correct / length(test_class)
```

```
## [1] 0.6
```

Again without PCA

```
predicted_class <- knn(train=pet_training, test=pet_test, cl=training_class, k=3)
sum(predicted_class==test_class) / length(predicted_class)
```

```
## [1] 0.85
```