John Mazon
DATA 605
Assignment#4

Eigen Shoes

This markdown file using PCA (Principal Component Analysis) to analyze the varibility of a series of related color images, in this case pictures of shoes.

Load required packages.

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.3.3     ✓ purrr   0.3.4
## ✓ tibble  3.0.6     ✓ dplyr   1.0.4
## ✓ tidyr   1.1.2     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(EBImage)
```

```
##
## Attaching package: 'EBImage'
```

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

```
library(jpeg)
library(OpenImageR)
```

```
##
## Attaching package: 'OpenImageR'
```

```
## The following objects are masked from 'package:EBImage':
##
##     readImage, writeImage
```

```
library(DT)

# EBImage is a BioConductor package and has non-standard installation.  If not installed on the
 local machine, run the following in the console:

# if (!requireNamespace("BiocManager", quietly = TRUE))
#     install.packages("BiocManager")
#
# BiocManager::install("EBImage")
```

The shoe jpg files are in a sub-directory named “eigen.” The following code block gathers the filenames and counts the number of images.

```
files <- list.files("eigen",pattern="\\.jpg")
numFiles <- length(files)
```

The images are 1200 by 2500 pixels (3 layers). Here we set the native dimensions and a scale reduction factor.

```
height <- 1200
width <- 2500
scale <- 20 # Resolution reduction factor
```

Next is the definition of a function for plotting a single jpg image.

```
plot_jpeg <- function(path, add = FALSE) {
  jpg = readJPEG(path, native = T) # read the file
  res = dim(jpg)[2:1] # get the resolution, [x, y]
  # initialize an empty plot area if add == FALSE
  if (!add) {
    plot(1, 1, xlim = c(1, res[1]), ylim = c(1, res[2]),
         asp = 1, type = 'n', xaxs = 'i', yaxs = 'i', xaxt = 'n', yaxt = 'n',
         xlab = '', ylab = '', bty = 'n')
  }
  rasterImage(jpg, 1, 1, res[1], res[2])
}
```

Read the jpg files and store them in a single array (im).

```
# Array "im" is 4 dimensional with indices for shoe, image rows, image columns and image layers
im <- array(0, dim = c(numFiles, height / scale, width / scale, 3))

# flat is a matrix with numFiles rows and columns equal to the product of resized
# jpg image dimensions (height * width * 3)
flat <- matrix(0, numFiles, prod(dim(im)))

# Read in the shoe jpgs, then immediately resize them (reducing resolution by factor of "scale")
for (i in 1:numFiles){
  temp <- resize(readJPEG(paste0("eigen/", files[i])), height / scale, width / scale)
  im[i,,,] <- array(temp, dim=c(1, height / scale, width / scale, 3))

    # Create single vectors for each color layer
  r <- as.vector(im[i,,,1])
  g <- as.vector(im[i,,,2])
  b <- as.vector(im[i,,,3])

  # Concatenate the three layers into a single row of "flat"
  flat[i,] <- t(c(r, g, b))
}

# Shoes is a dataframe that holds the image data.  Each column is a shoe.
shoes<- as.data.frame(t(flat))
```

Create a paneled plot of the original shoe images “as is” (albeit with reduced resolution).

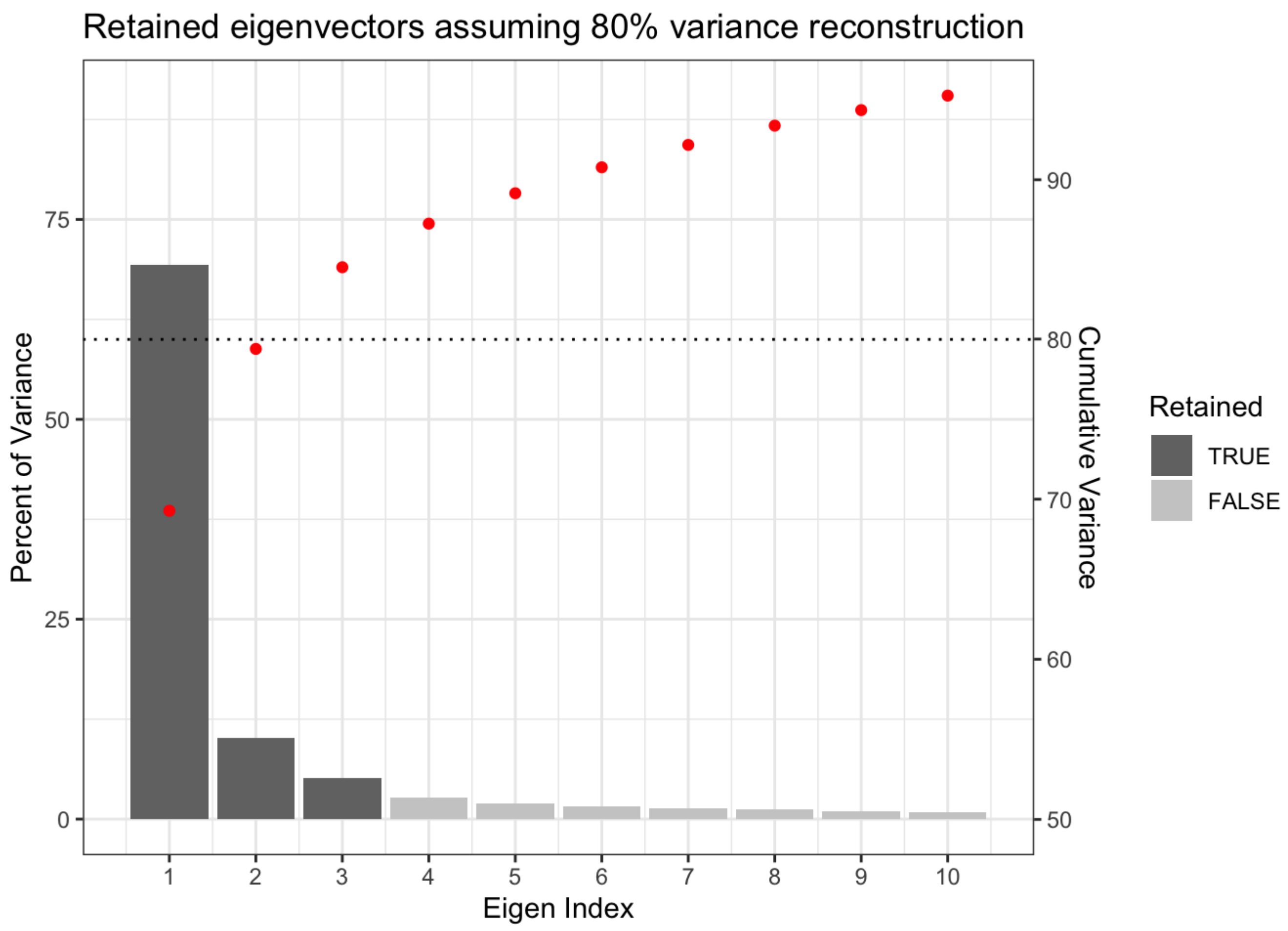
```
par(mfrow = c(3, 3)) # 3 rows, 3 columns
par(mai = c(.3,.3,.3,.3)) # set margins on subplots
for (i in 1:numFiles) {
  plot_jpeg(writeJPEG(im[i, , ,]))
}
```

The next section of code centers and scales the individual shoe images, then forms a covariance matrix (covariance of image i with image j) and computes its eigenvalues/vectors. The eigenvectors represent a set of orthogonal images. The eigenvalue associated with each is a measure of the variance explained by that image.

```r
# Centering and scaling of columns in "shoes." Centers and scales are saved.
scaled <- scale(shoes, center = TRUE, scale = TRUE)
mean.shoe <- attr(scaled, "scaled:center")
std.shoe <- attr(scaled, "scaled:scale")

# Compute the covariance matrix of the centered and scaled shoe matrix
Sigma <- cor(scaled)

# Perform the eigen decomposition
myeigen <- eigen(Sigma)
varTotal <- cumsum(myeigen$values) / sum(myeigen$values)
mindex <- min(which(varTotal > 0.80))
```

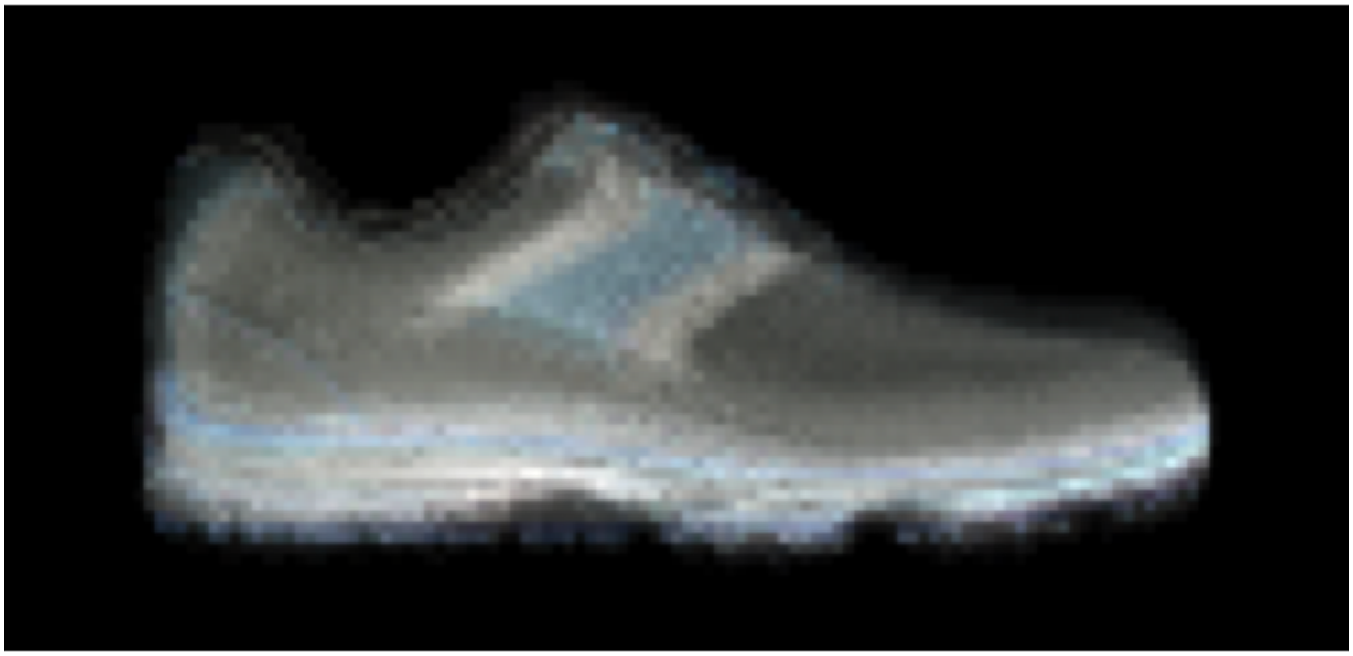
Display the variance explained by the eigenvectors in a Pareto-style plot

```r
# Make a Pareto-style plot of eigenvalue index and cumulative variance
varList <- tibble(`Eigen Index` = seq_along(myeigen$values),
                  `Percent of Variance` = myeigen$values / sum(myeigen$values) * 100,
                  `Cumulative Variance` = (varTotal * 100 - 50) * 2,
                   Retained = factor(`Eigen Index` <= mindex,
                                     levels = c("TRUE", "FALSE"))) %>%
  filter(`Eigen Index` <= 10)

theme_set(theme_bw())
ggplot(varList, aes(x = `Eigen Index`, y = `Percent of Variance`)) +
  geom_col(aes(fill = Retained)) +
  geom_point(aes(y = `Cumulative Variance`), col = "red") +
  geom_hline(yintercept = 60, lty = 3) +
  scale_y_continuous(sec.axis = sec_axis(trans = ~. / 2 + 50, name = "Cumulative Variance")) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5, 10.5)) +
  scale_fill_manual(values = c("TRUE" = "gray45", "FALSE" = "gray80")) +
  labs(title = "Retained eigenvectors assuming 80% variance reconstruction")
```



Reconstruct the images using the first few eigenvectors (however many are required to capture 80% of the total variance.)

```
scaling <- diag(myeigen$values[1:mindex]^(-1/2)) / (sqrt(nrow(scaled) - 1))
eigenshoes <- scaled %*% myeigen$vectors[,1:mindex] %*% scaling

# Display the first reconstructed shoe
imageShow(array(eigenshoes[,1], dim = c(height / scale, width / scale, 3)))
```



The next step is to use princomp to compute the principal component decomposition of the shoe images.

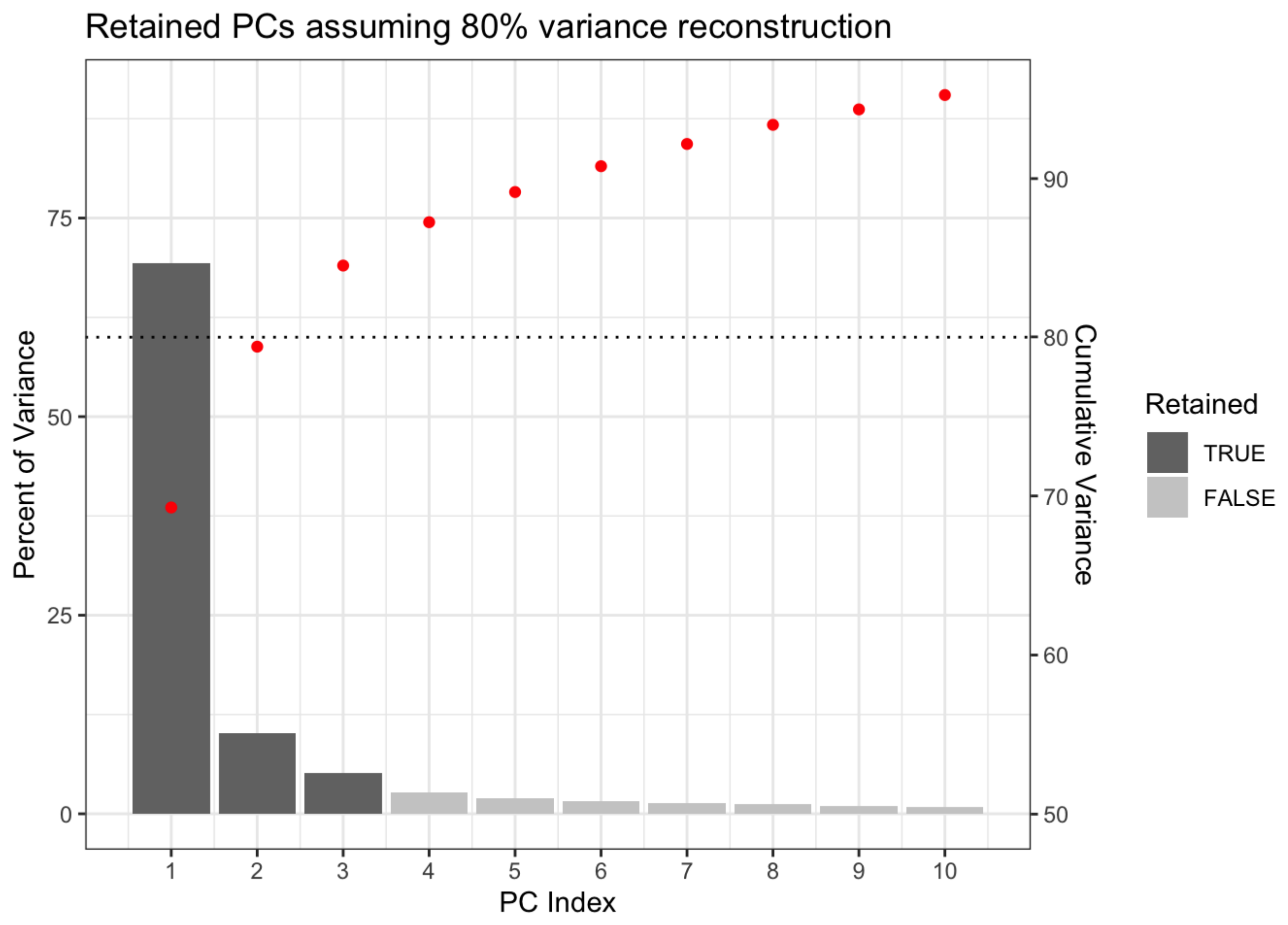
```
newdata <- im

# Resetting the dimensions collapses the height/width/layer dimensions into a single vector)
# This creates a matrix with rows corresponding to individual shoe images
dim(newdata) <- c(numFiles, height * width * 3 / scale^2)

# princomp computes the PC decomposition.  We use the transpose so that the individual
# PCs will have the dimensions of the shoe images
mypca <- princomp(t(as.matrix(newdata)), scores = TRUE, cor = TRUE)
```

Let’s plot out the individual principal components, i.e., “eigen shoes.”

```
# Transpose so that rows correspond to individual PC images
mypca2 <- t(mypca$scores)

# Reset dimensions to convert rows back to jpg arrays (3 layer)
dim(mypca2) <- c(numFiles, height / scale, width / scale, 3)

par(mfrow = c(5, 4))
par(mai = c(.001, .001, .001, .001))
for (i in 1:numFiles) plot_jpeg(writeJPEG(mypca2[i,,,], bg = "white"))
```

Repeat the variance computations from before.

```
varTotal <- cumsum(mypca$sdev^2) / sum(mypca$sdev^2)
mindex <- min(which(varTotal > 0.80))
varList <- tibble(`PC Index` = seq_along(mypca$sdev),
                  `Percent of Variance` = mypca$sdev^2 / sum(mypca$sdev^2) * 100,
                  `Cumulative Variance` = (varTotal * 100 - 50) * 2,
                   Retained = factor(`PC Index` <= mindex,
                                     levels = c("TRUE", "FALSE"))) %>%
  filter(`PC Index` <= 10)

theme_set(theme_bw())
ggplot(varList, aes(x = `PC Index`, y = `Percent of Variance`)) +
  geom_col(aes(fill = Retained)) +
  geom_point(aes(y = `Cumulative Variance`), col = "red") +
  geom_hline(yintercept = 60, lty = 3) +
  scale_y_continuous(sec.axis = sec_axis(trans = ~. / 2 + 50, name = "Cumulative Variance")) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5, 10.5)) +
  scale_fill_manual(values = c("TRUE" = "gray45", "FALSE" = "gray80")) +
  labs(title = "Retained PCs assuming 80% variance reconstruction")
```



Create the new data set. This is the product of the shoe image with each of the retained eigenvectors. These "coordinates" capture > 80% of the total variation within the full dataset. This produces considerable data reduction as only three eigenvectors are kept, each the size of one of the original images.

```
x = t(t(eigenshoes) %*% scaled)
df <- tibble(Shoe = 1:numFiles)
wgts <- map_dfc(1:mindex, function(i) tibble(V = round(x[,i], 2)))
```

```
## New names:
## * V -> V...1
## * V -> V...2
## * V -> V...3
```

```
colnames(wgts) <- map_chr(1:mindex, function(i) paste0("V", i))
df <- bind_cols(df, wgts)
datatable(df)
```

Show 10 entries Search:

	Shoe	V1	V2	V3
1	1	-533.93	-48.38	-81.33
2	2	-544.35	186.36	-54.64

	Shoe	V1	V2	V3
3	3	-419.18	-280.11	-141.62
4	4	-507.59	247.58	-78.4
5	5	-535.98	193.86	-35.13
6	6	-445.07	-282.14	-243.88
7	7	-471.29	-261.05	-212.76
8	8	-551.32	112.46	-157.67
9	9	-476.03	316.47	-101.86
10	10	-535.7	218.56	15.24

Showing 1 to 10 of 17 entries

Previous 1 2 Next