DATA 605: Assignment 04

EigenShoes

Philip Tanofsky

20 February 2021

EigenShoes

Starting point: https://rpubs.com/R-Minator/eigenshoes

```
# Libraries
library(jpeg)
library(EBImage)
library(OpenImageR)
```

Use of Graphics

Add graphics to the data set.

```
# Prepare for Image Processing

# Set constant for number of images to be processed
num_of_images <- 17

# Directory containing the images
img_dir <- "/Users/philiptanofsky/Documents/School/CUNY/MSDS/Courses/DATA605/Week04/jpg/"

# Read in the list of file names ending in ".jpg" from the directory
# These are images provided by the professor
files <- list.files(img_dir, pattern="\\.jpg")[1:num_of_images]

#files: Contains the list of jpg file names
```

View Shoes Function

```
# Set Adjustment Parameters (default values)
height <- 1200
width <- 2500
scale <- 20
colors <- 3 # Constant represents R,G,B colors of the jpg images

plot_jpeg <- function(path, add=FALSE) {
```

```r
  # Read the file
  jpg <- readJPEG(path, native=T)
  # Get the resolution, [x, y]
  res <- dim(jpg)[2:1]
  # Initialize any empty plot are 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])
  }
}
```

Load the Data into an Array

```r
# Based on the default values above
# Creating an arry with dimensions[17, 60, 125, 3] with all zeroes to start
img_arr <- array(rep(0, length(files) * height/scale * width/scale * colors),
           dim=c(length(files), height/scale, width/scale, colors))

# Loop through the image paths
for (i in 1:num_of_images) {
  temp <- resize(readJPEG(paste0(img_dir, files[i])),
                 height/scale,
                 width/scale)
  # Load the scaled images into the image array declared above
  img_arr[i,,,] <- array(temp, dim=c(1, height/scale, width/scale, colors))
}
```

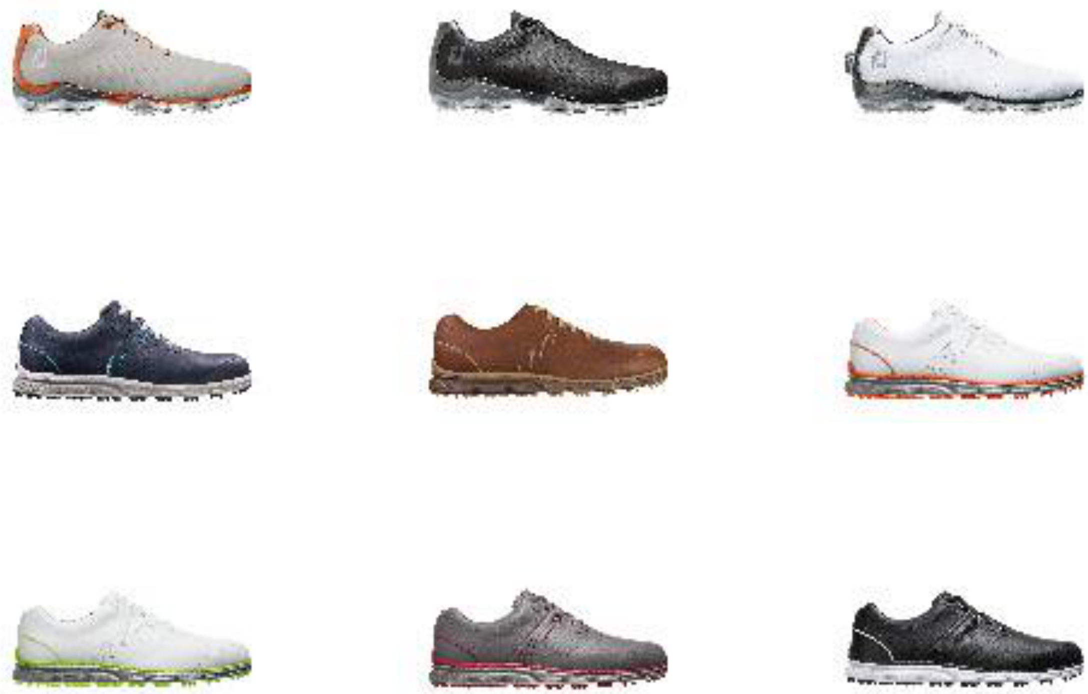
Vectorize

```r
# Create matrix of image count rows (17) and image array dimensions as columns (382500)
flat <- matrix(0, num_of_images, prod(dim(img_arr)))

# Loop through images
```

```r
for (i in 1:num_of_images) {
  #newim <- readJPEG(paste0(img_dir, files[i])) # Extra line of code
  # From image array, pull vector for color Red for given image
  r <- as.vector(img_arr[i,,,1])
  # From image array, pull vector for color Green for given image
  g <- as.vector(img_arr[i,,,2])
  # From image array, pull vector for color Blue for given image
  b <- as.vector(img_arr[i,,,3])
  # Take transpose of the color vectors to make it fit the row length of flat matrix
  flat[i,] <- t(c(r, g, b))
}
# Take transpose of the flat matrix
shoes <- as.data.frame(t(flat))

#dim(shoes)
#382500     17
# As expected, now >300k rows with 17 columns
```

Actual Plots

```r
# Plot the original images of the Shoes
par(mfrow=c(3,3))
# mai: set margin in inches
par(mai=c(.3, .3, .3, .3))
# Loop through image array and call the plot_jpeg function
for (i in 1:num_of_images) {
  plot_jpeg(writeJPEG(img_arr[i,,,]))
}
```



Get Eigen components from correlation structure

```r
scaled <- scale(shoes, center=TRUE, scale=TRUE)
#scaled object contains rows:382500     cols:17
# Clever way to pull the mean of each column using the scale function
mean.shoe <- attr(scaled, "scaled:center") # saving for classification
# Clever way to pull the standard deviation of each column using the scale function
std.shoe <- attr(scaled, "scaled:scale") # saving for classification ... later
```

Calculate Covariance (Correlation)

```r
# Create correlation matrix based on the number of images, in this case 17x17
sigma_ <- cor(scaled)

#sigma_
```

Get the eigencomponents

```
# Compute the eigenvalues and eigenvectors based on the correlation matrix
myeigen <- eigen(sigma_)
cumsum(myeigen$values) / sum(myeigen$values)
```

```
##  [1] 0.6928202 0.7940449 0.8451073 0.8723847 0.8913841 0.9076338 0.9216282
##  [8] 0.9336889 0.9433872 0.9524455 0.9609037 0.9688907 0.9765235 0.9832209
## [15] 0.9894033 0.9953587 1.0000000
```

Eigen shoes

```
# Why was 5 selected?
scaling <- diag(myeigen$values[1:2]^(-1/2)) / (sqrt(nrow(scaled)-1))
eigenshoes <- scaled %*% myeigen$vectors[,1:2] %*% scaling
imageShow(array(eigenshoes[,1], c(height/scale, width/scale, colors)))
```

```
# Dupe from above with different column count ... remove as needed
scaling <- diag(myeigen$values[1:17]^(-1/2)) / (sqrt(nrow(scaled)-1))
eigenshoes <- scaled %*% myeigen$vectors[,1:17] %*% scaling
imageShow(array(eigenshoes[,1], c(height/scale, width/scale, colors)))
```

Generate Principal Components

Transform the images

```r
# Generate variables
height <- 1200
width <- 2500
scale <- 20
# Start with the image array of the original images
newdata <- img_arr

# Convert the dimensions of the array to n x m, instead of n x m x o x p
dim(newdata) <- c(length(files), height*width*colors/scale^2)

# Transpose the n x m array, and then calculate the principal components
mypca <- princomp(t(as.matrix(newdata)), scores=TRUE, cor=TRUE)
# contains 17 components
mypca
```

```
## Call:
## princomp(x = t(as.matrix(newdata)), cor = TRUE, scores = TRUE)
##
## Standard deviations:
##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6    Comp.7    Comp.8
## 3.4319009 1.3118000 0.9316975 0.6809679 0.5683219 0.5255886 0.4877556 0.4528049
##    Comp.9   Comp.10   Comp.11   Comp.12   Comp.13   Comp.14   Comp.15   Comp.16
## 0.4060420 0.3924175 0.3791956 0.3684830 0.3602187 0.3374253 0.3241916 0.3181866
##   Comp.17
## 0.2808942
##
##  17  variables and  22500 observations.
```

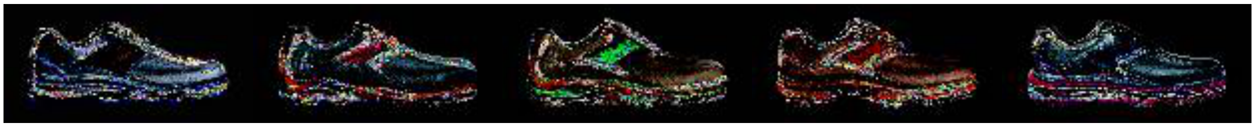
```r
dim(mypca$scores)
```

```
## [1] 22500    17
```

Eigenshoes

Generate Eigenshoes

```r
mypca2 <- t(mypca$scores)
dim(mypca2) <- c(length(files), height/scale, width/scale, colors)
par(mfrow=c(5,5))
par(mai=c(.001, .001, .001, .001))
# Plot the eigenshoes only
for (i in 1:num_of_images) {
  plot_jpeg(writeJPEG(mypca2[i,,,], bg="white")) # Complete without reduction
}
```



Variance Capture

```
a <- round(mypca$sdev[1:num_of_images]^2 / sum(mypca$sdev^2), colors)
cumsum(a)
```

```
##  Comp.1  Comp.2  Comp.3  Comp.4  Comp.5  Comp.6  Comp.7  Comp.8  Comp.9 Comp.10
##   0.693   0.794   0.845   0.872   0.891   0.907   0.921   0.933   0.943   0.952
## Comp.11 Comp.12 Comp.13 Comp.14 Comp.15 Comp.16 Comp.17
##   0.960   0.968   0.976   0.983   0.989   0.995   1.000
```