Principal Component Analysis on Imaging

R Programming

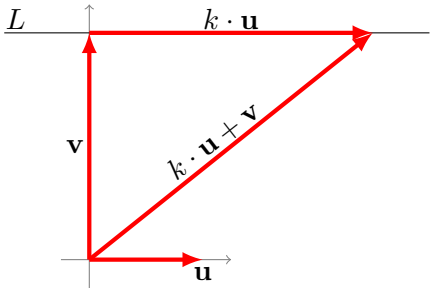
25 of December 2014

Al-Ahmadgaid B. Asaad
alstated@gmail.com

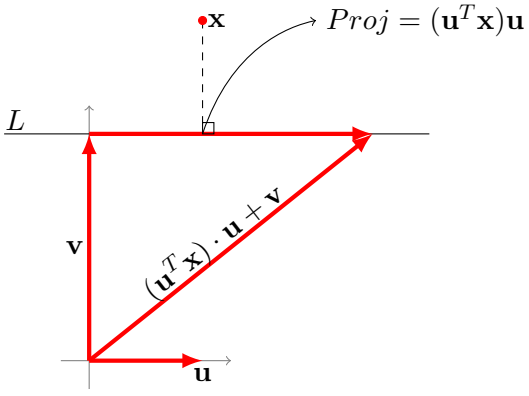
Ever wonder what's the mathematics behind face recognition on most gadgets like digital camera and smartphones? Well for most part it has something to do with statistics. One statistical tool that is capable of doing such feature is the Principal Component Analysis (PCA). In this post, however, we will not do (sorry to disappoint you) face recognition as we reserve this for future post while I'm still doing research on it. Instead, we go through its basic concept and use it for data reduction on spectral bands of the image using R.

1 Let's view it mathematically

Consider a line L in a parametric form described as a set of all vectors $k \cdot \mathbf{u} + \mathbf{v}$ parameterized by $k \in \mathbb{R}$, where $\mathbf{v}$ is a vector orthogonal to a normalized vector $\mathbf{u}$. Below is the graphical equivalent of the statement:



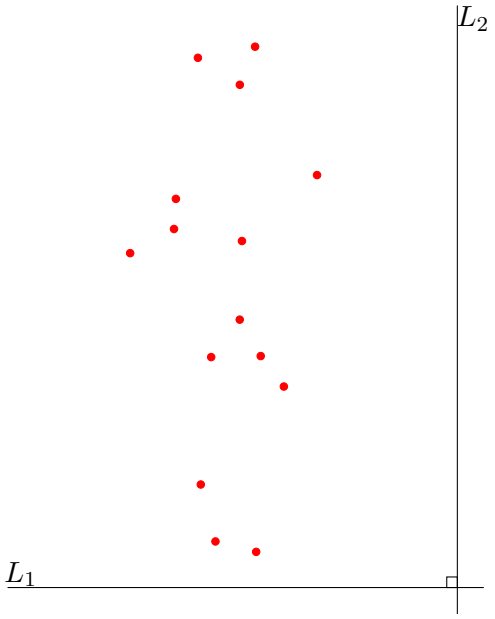
So if given a point $\mathbf{x} = [x_1, x_2]^T$, the orthogonal projection of this point on the line L is given by $(\mathbf{u}^T\mathbf{x})\mathbf{u} + \mathbf{v}$. Graphically, we mean



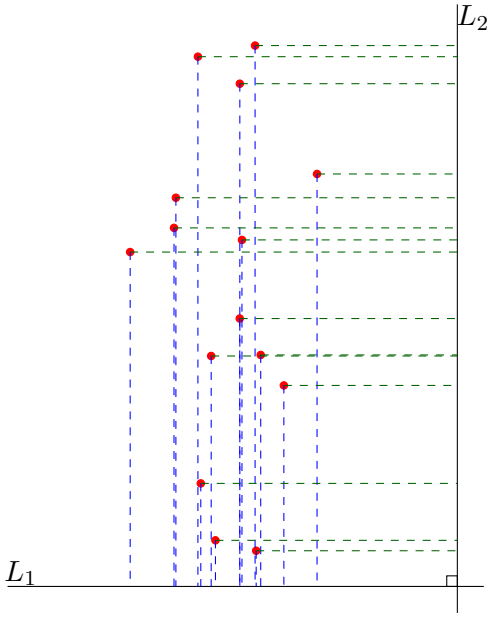
Proj is the projection of the point $\mathbf{x}$ on the line, where the position of it is defined by the scalar $\mathbf{u}^T\mathbf{x}$. Therefore, if we consider $\mathbf{X} = [X_1, X_2]^T$ be a random vector, then

the random variable $Y = \mathbf{u}^T\mathbf{X}$ describes the variability of the data on the direction of the normalized vector $\mathbf{u}$. So that Y is a linear combination of $X_i, i = 1, 2$. *The principal component analysis identifies a linear combinations of the original variables* $\mathbf{X}$ *that contain most of the information, in the sense of variability, contained in the data. The general assumption is that useful information is proportional to the variability. PCA is used for data dimensionality reduction and for interpretation of data. (Ref 1. Bajorski, 2012)*

To better understand this, consider two dimensional data set, below is the plot of it along with two lines (L_1 and L_2) that are orthogonal to each other:



If we project the points orthogonally to both lines we have,

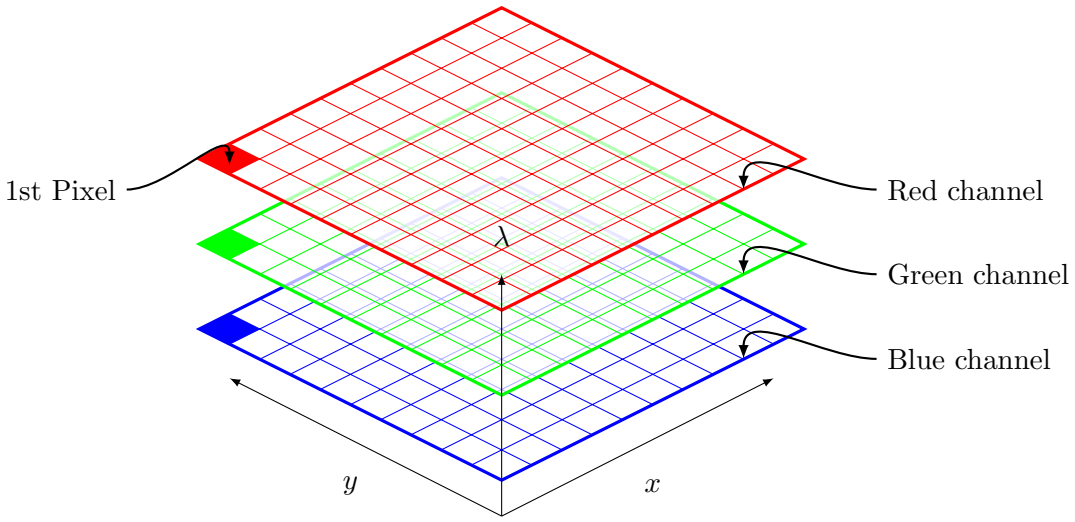


So that if normalized vector $\mathbf{u}_1$ defines the direction of L_1, then the variability of the points on L_1 is described by the random variable $Y_1 = \mathbf{u}_1^T\mathbf{X}$. Also if $\mathbf{u}_2$ is a normalized vector that defines the direction of L_2, then the variability of the points on this line is described by the random variable $Y_2 = \mathbf{u}_2^T\mathbf{X}$. The first principal component is one with maximum variability. So in this case, we can see that Y_2 is more variable than Y_1, since the points projected on L_2 are more dispersed than in L_1. In practice, however, the linear combinations $Y_i = \mathbf{u}_i^T\mathbf{X}, i = 1, 2, \cdots, p$ is maximized sequentially so that Y_1 is the linear combination of the first principal component, Y_2 is the linear combination of the second principal component, and so on. Further, the estimate of the direction vector $\mathbf{u}$ is simply the normalized eigenvector $\mathbf{e}$ of the variance-covariance matrix $\mathbf{\Sigma}$ of the original variable $\mathbf{X}$. And the variability explained by the principal component is the corresponding eigenvalue λ. For more details on theory of PCA refer to (Bajorski, 2012) at Reference 1 below.

As promised we will do dimensionality reduction using PCA. We will use the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) data from (Barjorski, 2012), you can use other locations of AVIRIS data that can be downloaded here. However, since for most cases the AVIRIS data contains thousands of bands so for simplicity we will stick with the data given in (Bajorski, 2012) as it was cleaned reducing to 152 bands only.

2 What is spectral bands?

In imaging, spectral bands refer to the third dimension of the image usually denoted as λ. For example, RGB image contains red, green and blue bands as shown below along with the first two dimensions x and y that define the resolution of the image.



These are few of the bands that are visible to our eyes, there are other bands that are not visible to us like infrared, and many other in electromagnetic spectrum. That is why in most cases AVIRIS data contains huge number of bands each captures different characteristics of the image. Below is the proper description of the data.

3 Data

The Airborne Visible/Infrared Imaging Spectrometer (AVIRIS), is a sensor collecting spectral radiance in the range of wavelengths from 400 to 2500 nm. It has been flown on various aircraft platforms, and many images of the Earths surface are available. A 100 by 100 pixel AVIRIS image of an urban area in Rochester, NY, near the Lake Ontario shoreline is shown below. The scene has a wide range of natural and man-made material including a mixture of commercial/warehouse and residential neighborhoods, which adds a wide range of spectral diversity. Prior to processing, invalid bands (due to atmospheric water absorption) were removed, reducing the overall dimensionality to 152 bands. This image has been used in Bajorski et al. (2004) and Bajorski (2011a, 2011b). The first 152 values in the AVIRIS Data represent the spectral radiance values (a spectral curve) for the top left pixel. This is followed by spectral curves of the pixels in the first row, followed by the next row, and so on. (Ref. 1 Bajorski, 2012)

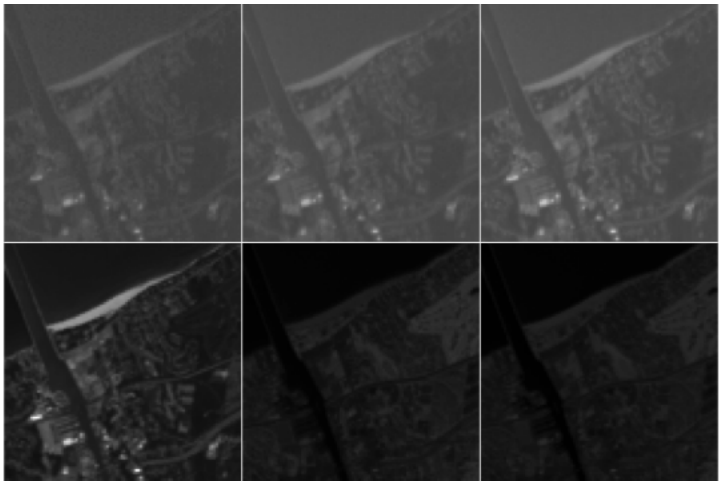
To load the data, run the following code:

```
# Download the data
web <- "https://raw.githubusercontent.com/alstat/Analysis-with-
    Programming/master/2014/R/Principal%20Component%20Analysis%20on
    %20Imaging/AVIRIS_Data.txt"
text <- download.file(web, destfile = "/tmp/dat.txt", method = "
    curl")

# Scan the data
dat <- scan("/tmp/dat.txt", skip = 1)
dat.mat <- matrix(dat, ncol = 152, byrow = T)

# Assign the pixel values to an array that defines the image
img <- array(0, c(100, 100, 152))
for(i in 1:100) {
  for(j in 1:100){
    n = (i - 1) * 100 + j
    img[i, j, ] <- dat.mat[n, ]
  }
}

library(EBImage)

# Function for contrast
contr <- function(x) {
  out <- (x - min(x) + 1) / (max(x) - min(x) + 1)
  out
}

# Display the 1st to 3rd, and 30th, 60th, and 100th bands
display(contr(img[,, c(1:3, 30, 60, 100)]), all = T, meth = 'r')
```

Above code uses EBImage package, and can be installed from my previous post

4 Why do we need to reduce the dimension of the data?

Before we jump in to our analysis, in case you may ask why? Well sometimes it's just difficult to do analysis on high dimensional data, especially on interpreting it.



This is because there are dimensions that aren't significant (like redundancy) which adds to our problem on the analysis. So in order to deal with this, we remove those nuisance dimension and deal with the significant one.

To perform PCA in R, we use the function `princomp` as seen below:

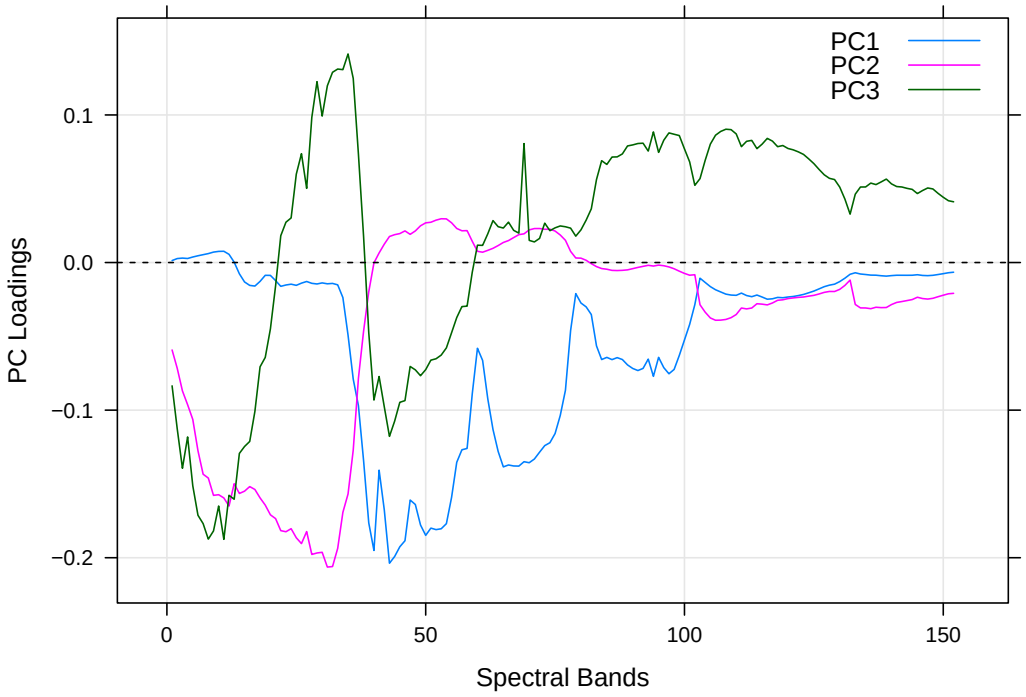
```
# Perform the principal components analysis
pc <- princomp(dat.mat)
str(pc)

# OUTPUT
List of 7
 $ sdev     : Named num [1:152] 10121 4631 632 477 342 ...
  ..- attr(*, "names")= chr [1:152] "Comp.1" "Comp.2" "Comp.3" "
      Comp.4" ...
 $ loadings: loadings [1:152, 1:152] 0.00136 0.0027 0.00301 0.00273
      0.00375 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:152] "Comp.1" "Comp.2" "Comp.3" "Comp.4" ...
 $ center   : num [1:152] 3428 3735 3878 3497 3623 ...
 $ scale    : num [1:152] 1 1 1 1 1 1 1 1 1 1 ...
 $ n.obs    : int 10000
 $ scores   : num [1:10000, 1:152] 15339 15385 15404 15354 15192 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:152] "Comp.1" "Comp.2" "Comp.3" "Comp.4" ...
 $ call     : language princomp(x = dat.mat)
 - attr(*, "class")= chr "princomp"
```

The structure of `princomp` consist of a list shown above, we will give description to selected outputs. Others can be found in the documentation of the function by executing `?princomp`.

- `sdev` - standard deviation, the square root of the eigenvalues λ of the variance-covariance matrix $\mathbf{\Sigma}$ of the data, `dat.mat`;
- `loadings` - eigenvectors $\mathbf{e}$ of the variance-covariance matrix $\mathbf{\Sigma}$ of the data, `dat.mat`;

- `scores` - the principal component scores.

Recall that the objective of PCA is to find for a linear combination $Y = \mathbf{u}^T\mathbf{X}$ that will maximize the variance $Var(Y)$. So that from the output, the estimate of the components of $\mathbf{u}$ is the entries of the `loadings` which is a matrix of eigenvectors, where the columns corresponds to the eigenvectors of the sequence of principal components, that is if the first principal component is given by $Y_1 = \mathbf{u}_1^T\mathbf{X}$, then the estimate of $\mathbf{u}_1$ which is $\mathbf{e}_1$ (eigenvector) is the set of coefficients obtained from the first column of the `loadings`. The explained variability of the first principal component is the square of the first standard deviation `sdev`, the explained variability of the second principal component is the square of the second standard deviation `sdev`, and so on. Now let's interpret the loadings (coefficients) of the first three principal components. Below is the plot of this,

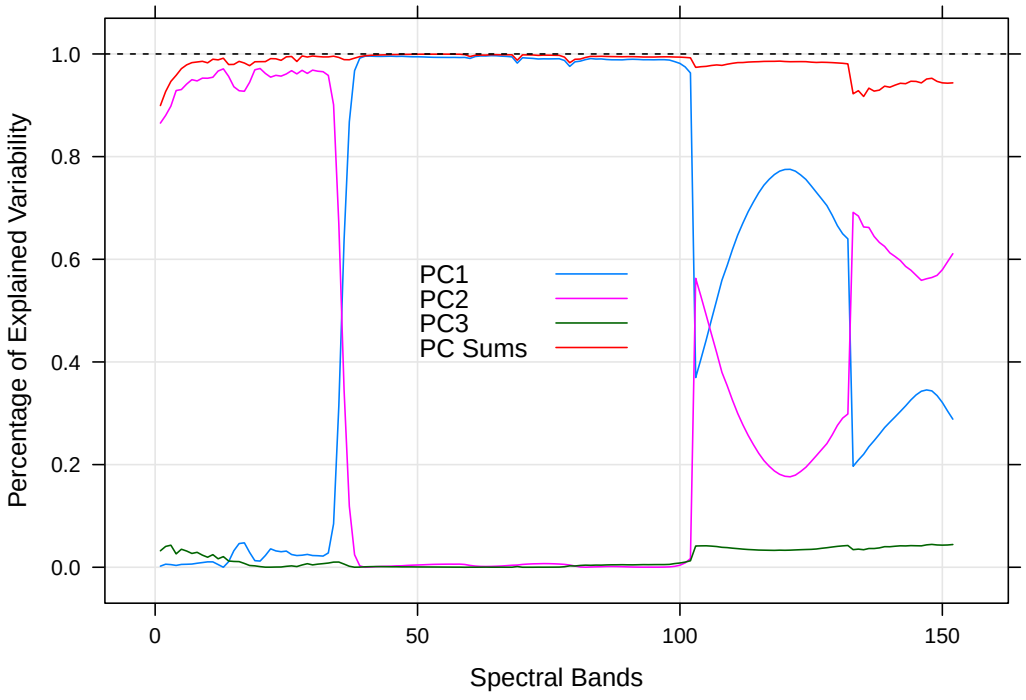



```
library(lattice)
library(reshape2)
pc.load <- cbind(pc$loadings[, 1:3])
colnames(pc.load) <- c("PC1", "PC2", "PC3")
pc.df <- melt(pc.load)
xyplot(value ~ Var1, data = pc.df, group = Var2, type = "l",
       ylab = "PC Loadings", xlab = "Spectral Bands",
       auto.key = list(corner = c(0.98, 0.98), points = FALSE,
           lines = TRUE),
       panel = function (x, y, ...) {
         panel.grid(h = -1, v = -1)
         panel.xyplot(x, y, ...)
         panel.abline(h = 0, lty = "dashed")
       })
```

Base above, the coefficients of the first principal component (PC1) are almost all negative. A closer look, the variability in this principal component is mainly explained by the weighted average of radiance of the spectral bands 35 to 100. Analogously, PC2 mainly represents the variability of the weighted average of radiance of spectral bands 1 to 34. And further, the fluctuation of the coefficients of PC3 makes it difficult to tell on which bands greatly contribute on its variability. Aside from examining the loadings, another way to see the impact of the PCs is through the *impact plot* where the *impact curve* $\sqrt{\lambda_j}\mathbf{e}_j$ are plotted, I want you to explore that. Moving on, let's investigate the percent of variability in X_i explained by the jth principal component, below is the formula of this,

$$\frac{\lambda_j \cdot e_{ij}^2}{s_{ii}},$$

where s_{ii} is the estimated variance of X_i. So that below is the percent of explained variability in X_i of the first three principal components including the cumulative percent variability (sum of PC1, PC2, and PC3),




```r
# Compute the percentage of variability explained by the
# the PC1, PC2, and PC3, and their sum
PC1 <- (pc$sdev[1] ^ 2) * (pc$loadings[, 1] ^ 2) / diag(var(dat.mat
    ))
PC2 <- (pc$sdev[2] ^ 2) * (pc$loadings[, 2] ^ 2) / diag(var(dat.mat
    ))
PC3 <- (pc$sdev[3] ^ 2) * (pc$loadings[, 3] ^ 2) / diag(var(dat.mat
    ))
PCSums <- PC1 + PC2 + PC3
pcVar <- melt(cbind(PC1, PC2, PC3, "PC Sums" = PCSums))
```

```
xyplot(value ~ Var1, data = pcVar, group = Var2, type = "l",
       ylab = "Percentage of Explained Variability", xlab = "
           Spectral Bands",
       auto.key = list(corner = c(0.45, 0.5), points = FALSE, lines
            = TRUE),
       panel = function (x, y, ...) {
         panel.grid(h = -1, v = -1)
         panel.xyplot(x, y, ...)
         panel.abline(h = 1, lty = "dashed")
       })
```

For the variability of the first 33 bands, PC2 takes on about 90 percent of the explained variability as seen in the above plot. And still have great contribution further to 102 to 152 bands. On the other hand, from bands 37 to 100, PC1 explains almost all the variability with PC2 and PC3 explain 0 to 1 percent only. The sum of the percentage of explained variability of these principal components is indicated as orange line in the above plot, which is the cumulative percent variability.

To wrap up this section, here is the percentage of the explained variability of the first 10 PCs. Above variability were obtained by noting that the variability explained by

PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
82.057	17.176	0.320	0.182	0.094	0.065	0.037	0.029	0.014	0.005

the principal component is simply the eigenvalue (square of the ¡code¿sdev¡/code¿) of the variance-covariance matrix $\mathbf{\Sigma}$ of the original variable $\mathbf{X}$, hence the percentage of variability explained by the jth PC is equal to its corresponding eigenvalue λ_j divided by the overall variability which is the sum of the eigenvalues, $\sum_{j=1}^{p} \lambda_j$, as we see in the following code,

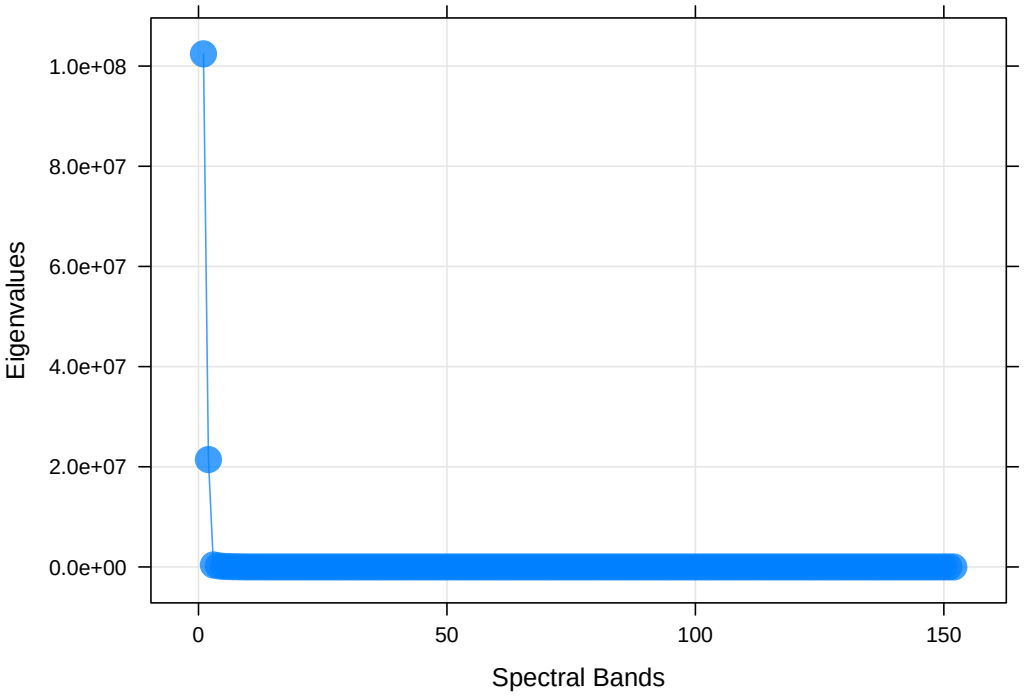
```
round((as.numeric(pc$sdev) ^ 2) / sum(as.numeric(pc$sdev) ^ 2) *
    100, 3)
```

5 Stopping Rules

Given the list of percentage of variability explained by the PCs in Table 1, how many principal components should we take into account that would best represent the variability of the original data? To answer that, we introduce the following stopping rules that will guide us on deciding the number of PCs:

- Scree plot;
- Simple fare-share;
- Broken-stick; and,
- Relative broken-stick.

The scree plot is the plot of the variability of the PCs, that is the plot of the eigenvalues. Where we look for an elbow or sudden drop of the eigenvalues on the plot, hence for our example we have



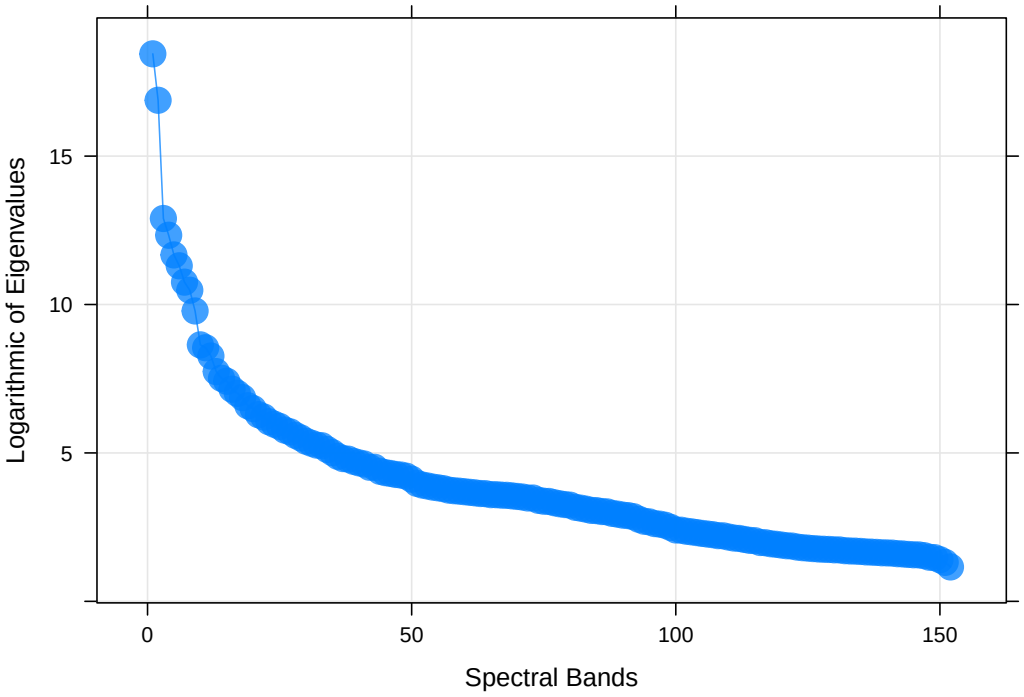

```
xyplot((pc$sdev ^ 2) ~ 1:152, pch = 20, cex = 3, alpha = 0.75, type
    = "b",
       xlab = "Spectral Bands", ylab = "Eigenvalues",
       panel = function (x, y, ...) {
         panel.grid(h = -1, v = -1)
         panel.xyplot(x, y, ...)
       })
```

Therefore, we need return the first two principal components based on the elbow shape. However, if the eigenvalues differ by order of magnitude, it is recommended to use the logarithmic scale which is illustrated below,

```
xyplot(log(pc$sdev ^ 2) ~ 1:152, pch = 20, cex = 3, alpha = 0.75,
   type = "b",
       xlab = "Spectral Bands", ylab = "Logarithmic of Eigenvalues"
           ,
       panel = function (x, y, ...) {
         panel.grid(h = -1, v = -1)
         panel.xyplot(x, y, ...)
       })
```

Unfortunately, sometimes it won't work as we can see here, it's just difficult to determine where the elbow is. The succeeding discussions on the last three stopping rules are based on (Bajorski, 2012). The *simple fair-share* stopping rule identifies the largest k such that λ_k is larger than its fair share, that is larger than $(\lambda_1 + \lambda_2 + \cdots + \lambda_p)/p$. To illustrate this, consider the following:

```
# Simple fair-share
which((pc$sdev ^ 2) > (sum(pc$sdev ^ 2)) / length(pc$sdev))

# OUTPUT
```




```
Comp.1 Comp.2
     1      2
```

Thus, we need to stop at second principal component.

If one was concerned that the above method produces too many principal components, a *broken-stick rule* could be used. The rule is that it identifies the principal components with largest k such that $\lambda_j/(\lambda_1 + \lambda_2 + \cdots + \lambda_p) > a_j$, for all $j \leq k$, where

$$a_j = \frac{1}{p}\sum_{i=j}^{p}\frac{1}{i}, \quad j = 1, \cdots, p.$$

Let's try it,

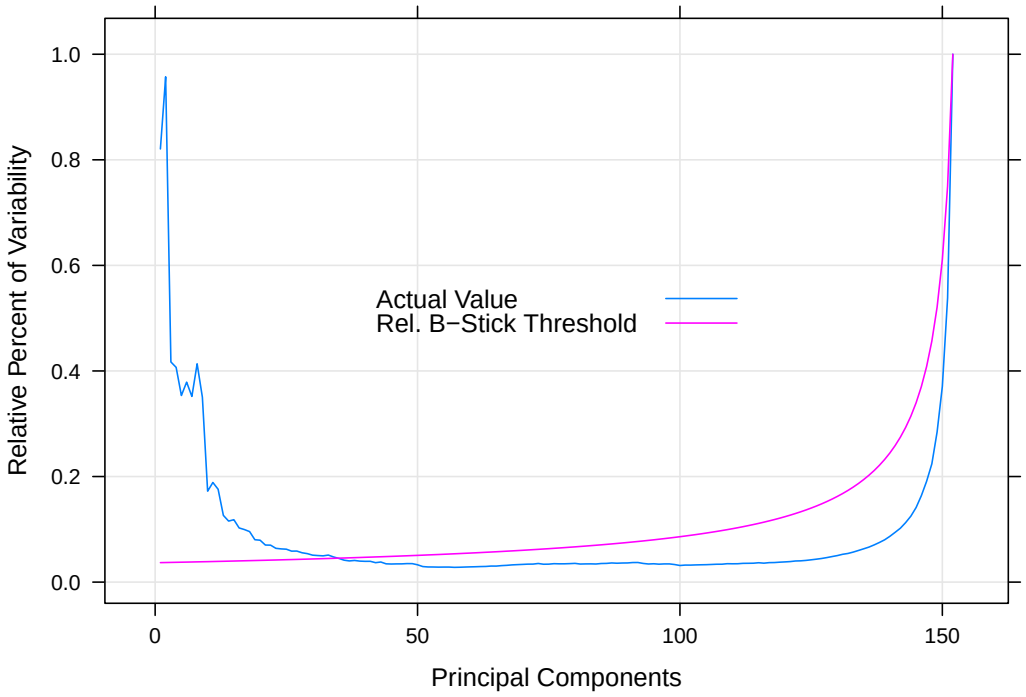
```
# Broken-stick function
brStick <- function (x) {
  m <- 0
  out <- matrix(NA, ncol = 2, nrow = length(x))
  colnames(out) <- c("% of Variability", "B-Stick Threshold")
  for (i in 1:length(x)) {
    for (k in i:length(x)) {
      m <- m + ((1 / length(x)) * (1 / k))
    }
    out[i, ] <- c((x[i] / sum(x)) * 100, m * 100)
    m <- 0
  }
```

```
  return(list("Use PCs:" = which(out[, 1] > out[, 2]), Table = out)
      )
}

# Apply the function to the data
brStick(pc$sdev ^ 2)[["Use PCs:"]]

# OUTPUT
[1] 1 2
```

Above result coincides with the first two stopping rule. The draw back of simple fair-share and broken-stick rules is that it do not work well when the eigenvalues differ by orders of magnitude. In such case, we then use the *relative broken-stick* rule, where we analyze λ_j as the first eigenvalue in the set $\lambda_j \geq \lambda_{j+1} \geq \cdots \geq \lambda_p$, where $j < p$. The dimensionality k is chosen as the largest value such that $\lambda_j/(\lambda_j + \cdots + \lambda_p) > b_j$, for all $j \leq k$, where

$$b_j = \frac{1}{p-j+1} \sum_{i=1}^{p-j+1} \frac{1}{i}.$$

Applying this to the data we have,



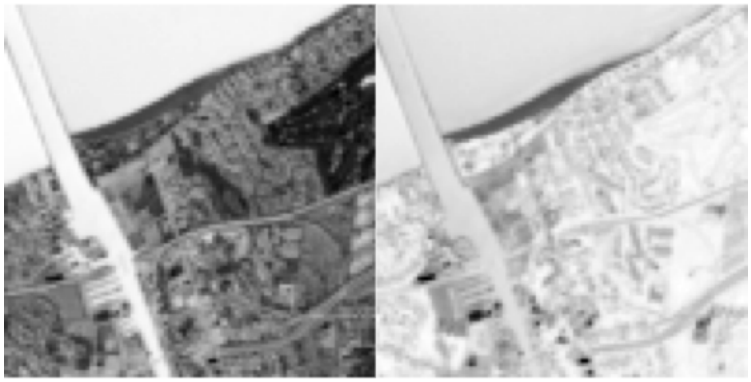

```
library(reshape2)
rbrStick <- function (x, plot = FALSE) {
  m <- 0
  out <- matrix(NA, ncol = 2, nrow = length(x))
  colnames(out) <- c("Actual Value", "Rel. B-Stick Threshold")
  for (i in 1:length(x)) {
    for (k in 1:(length(x) - i + 1)) {
      m <- m + ((1 / (length(x) - i + 1)) * (1 / k))
```

```
    }
    out[i, ] <- c(x[i] / sum(x[i:length(x)]), m)
    m <- 0
  }
  }
  if (plot == TRUE) {
    rbr.df <- melt(out)
    p <- xyplot(value ~ Var1, group = Var2, data = rbr.df, type = "
        l",
                xlab = "Principal Components", ylab = "Relative
                    Percent of Variability",
                auto.key = list(corner = c(0.5, 0.5), points =
                    FALSE, lines = TRUE),
                panel = function(x, y, ...) {
                  panel.grid(h = -1, v = -1)
                  panel.xyplot(x, y, ...)
                })
    show(p)
  }
  return(list("Use PCs:" = which(out[, 1] > out[, 2]), Table = out)
      )
}

# Apply the function to the data
rbrStick(pc$sdev ^ 2, plot = TRUE)[["Use PCs:"]]
```

According to the numerical output, the first 34 principal components are enough to represent the variability of the original data.

6 Principal Component Scores

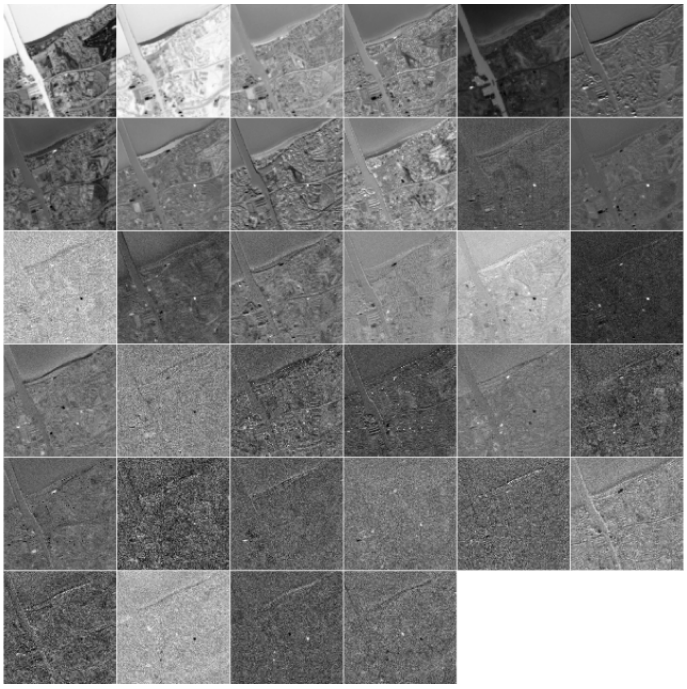
The principal component scores is the resulting new data set obtained from the linear combinations $Y_j = \mathbf{e}_j(\mathbf{x} - \bar{\mathbf{x}}), j = 1, \cdots, p$. So that if we use the first three stopping rules, then below is the scores (in image) of PC1 and PC2,



```
# PC Scores
library(EBImage)
img <- array(0, c(100, 100, 152))
for(i in 1:100) {
  for(j in 1:100){
```

```
    n = (i - 1) * 100 + j
    img[i, j, ] <- as.vector(pc$scores[n, ])
  }
}

# Function for contrast
hcont <- function(x) {
  out <- (x - min(x) + 1) / (max(x) - min(x) + 1)
  out
}

img1 <- array(0, c(100, 100, 152))
for (i in 1:152) img1[,, i] <- hcont(img[,, i])

# Display the image
display(img1[,, c(1, 2)], all = T, meth = 'r')
```

If we base on the relative broken-stick rule then we return the first 34 PCs, and below is the corresponding scores (in image).



```
# PC Scores
library(EBImage)
img <- array(0, c(100, 100, 152))
for(i in 1:100) {
  for(j in 1:100){
    n = (i - 1) * 100 + j
    img[i, j, ] <- as.vector(pc$scores[n, ])
  }
}

# Function for contrast
hcont <- function(x) {
  out <- (x - min(x) + 1) / (max(x) - min(x) + 1)
  out
```

```
}

img1 <- array(0, c(100, 100, 152))
for (i in 1:152) img1[,, i] <- hcont(img[,, i])

# Display the image
display(img1[,, c(1:34)], all = T, meth = 'r')
```

7 Residual Analysis

Of course when doing PCA there are errors to be considered unless one would return all the PCs, but that would not make any sense because why would someone apply PCA when you still take into account all the dimensions? An overview of the errors in PCA without going through the theory is that, the overall error is simply the excluded variability explained by the kth to pth principal components, $k > j$.

References

Bajorski, P. (2012). *Statistics for Imaging, Optics, and Photonics*. John Wiley & Sons, Inc..

Labels

Data Mining,Image Analysis,LaTeX,Machine Learning,R,