Solution, Tutorial, PCA

Benoit Liquet[*1]

[1]Macquarie University

*benoit.mates-weiland@mq.edu.au

Contents

1 Prediction using PCA 2

2 Data compression using SVD 2

3 Solution PCA for prediction 4

4 Solution Data compression using SVD 12

1 Prediction using PCA

In this question we will use the dataset `Breast_cancer.RData` which contains quantitative information from digitized images of a diagnostic test (fine needle aspirate (FNA) test on breast mass) for the diagnosis of breast cancer. The variables describe characteristics of the cell nuclei present in the image. The mean, standard error and "worst' " or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

```
load("Breast_cancer.RData")
```

(a) Run a PCA model on the variables `radius _mean`, `texture_mean`,`perimeter_mean`, `area_mean`, `smoothness_mean`, `compactness_mean`,`concavity_mean`, `concave.points_mean`, `symmetry_mean`, `fractal_dimension_mean`. We will use the function `prcomp()`.

The prcomp() function will include in its results

- sdev - the standard deviations of the principal components

- rotation - the matrix of variable **loadings** for the components

- x - the scaled matrix of data times the factor loadings (known as scores)

(b) We now can see how much variance is explained by the components. We will use some functions of the package `factoextra` to plot some of the results. The plot representing the variance explained by the components is called `Scree Plot`. Provide the scree plot for this analysis. Comment your results.

(c) Provide the circle-correlation plot for the 3 first components. Comments your results.

(d) Use the eigen value decomposition of the appropriate matrix to get the same result as the `prcomp` function.

(e) Use the singular value decomposition of the appropriate matrix to get the same result as the `prcomp` function.

(f) Project the samples on the two first components. Annotate the sample using the `diagnosis` variables ("Malignant", "Benin").

(g) Use the tree first principal components as predictors in a logistic model for the variable diagnosis. Use the `caret` package and a K-fold cross validation ($K = 10$) approach for providing the sensitivity, specificity and AUC values.

2 Data compression using SVD

In this question we will use the SVD to compress an image. The following code enables us to read a jpeg image file into an array with 3 channels (Red-Green-Blue, RGB).

```
if (!"jpeg" %in% installed.packages()) install.packages("jpeg")
# Read image file into an array with three channels (Red-Green-Blue, RGB)
mates <- jpeg::readJPEG("pool_color.jpg")
```

We then get 3 channels:

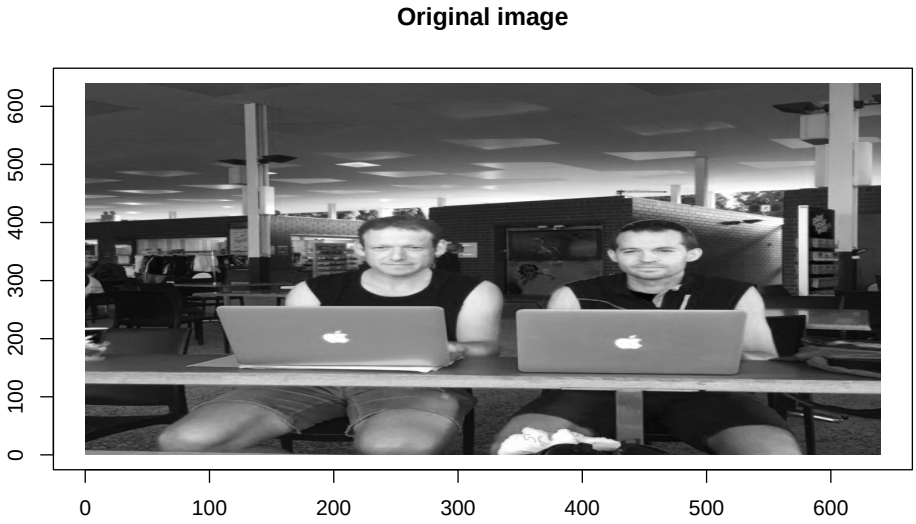
```
r <- mates[, , 1] ; g <- mates[, , 2] ; b <- mates[, , 3]
```

This function enables to plot an image from an array

```
plot.image <- function(pic, main = "") {
  h <- dim(pic)[1] ; w <- dim(pic)[2]
  plot(x = c(0, h), y = c(0, w), type = "n", xlab = "", ylab = "", main = main)
  rasterImage(pic, 0, 0, h, w)
}
```

For simplicity we will only use the red channel information. This will correspond to a black and white image. An image is a matrix of pixels with values that represent the intensity of the the pixel, where $0 = white$ and $1 = black$.

```
plot.image(r, "Original image")
```



This image is a (640×640) pixels

```
dim(r)
[1] 640 640
```

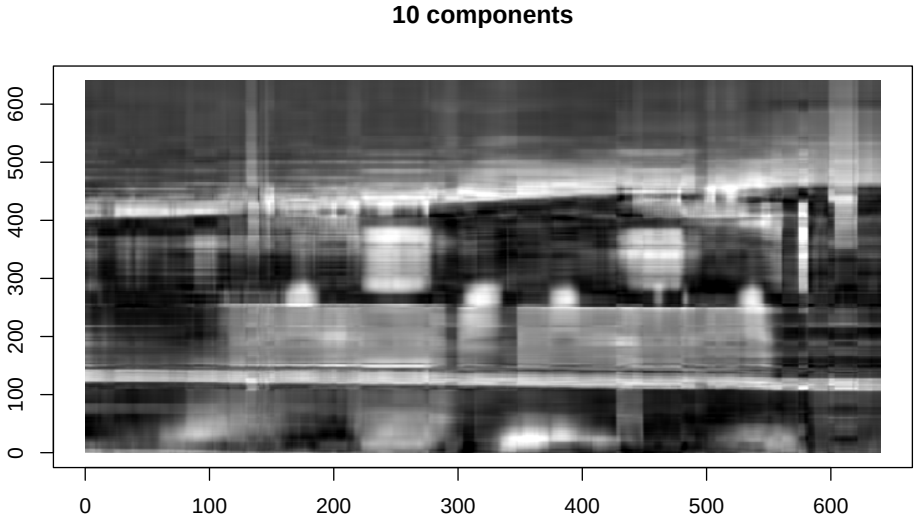
Using the SVD properties (see lecture) we can try to compress the image by using only 10 components.

```
nb.comp <- 10
r.svd <- svd(r)
d <- r.svd$d[1:nb.comp]
v <- r.svd$v[,1:nb.comp]
u <- r.svd$u[,1:nb.comp]
```

Using the first 10 right and left singular vector of the matrix we can reconstruct the image using:

```
img.compressed <- u %*% diag(d) %*% t(v)
img.compressed[img.compressed < 0] <- 0
img.compressed[img.compressed > 1] <- 1
```

```
plot.image(img.compressed,"10 components")
```



(a) Explore different number of component to see the effect on the compressed image. For example 10, 30, 50 and 640.

(b) Produce the image above with 100 components but using the colored photo.

(c) If you have time, explore the same procedure on a photo on your choice

3 Solution PCA for prediction

(a) PCA using prcomp

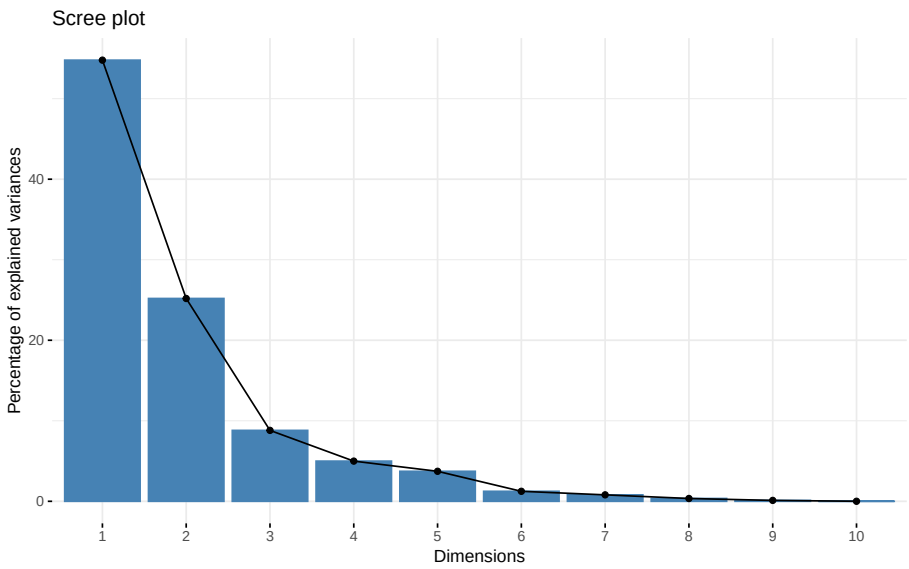
```
load("Breast_cancer.RData")
Breast.10vars <- Breast_cancer_data[,c("radius_mean", "texture_mean",
                     "perimeter_mean", "area_mean",
                     "smoothness_mean", "compactness_mean",
                     "concavity_mean", "concave.points_mean",
                     "symmetry_mean", "fractal_dimension_mean")]
```

```
pca <- prcomp(Breast.10vars, scale=TRUE) #pca with scaled variables
```

(b) The plot representing the variance explained by the components is called Scree Plot.

```
library(factoextra)
summary(pca) #variance explained
Importance of components:
                          PC1    PC2     PC3    PC4     PC5     PC6     PC7
Standard deviation     2.3406 1.5870 0.93841 0.7064 0.61036 0.35234 0.28299
Proportion of Variance 0.5479 0.2519 0.08806 0.0499 0.03725 0.01241 0.00801
Cumulative Proportion  0.5479 0.7997 0.88779 0.9377 0.97495 0.98736 0.99537
                           PC8     PC9    PC10
Standard deviation     0.18679 0.10552 0.01680
Proportion of Variance 0.00349 0.00111 0.00003
Cumulative Proportion  0.99886 0.99997 1.00000
```

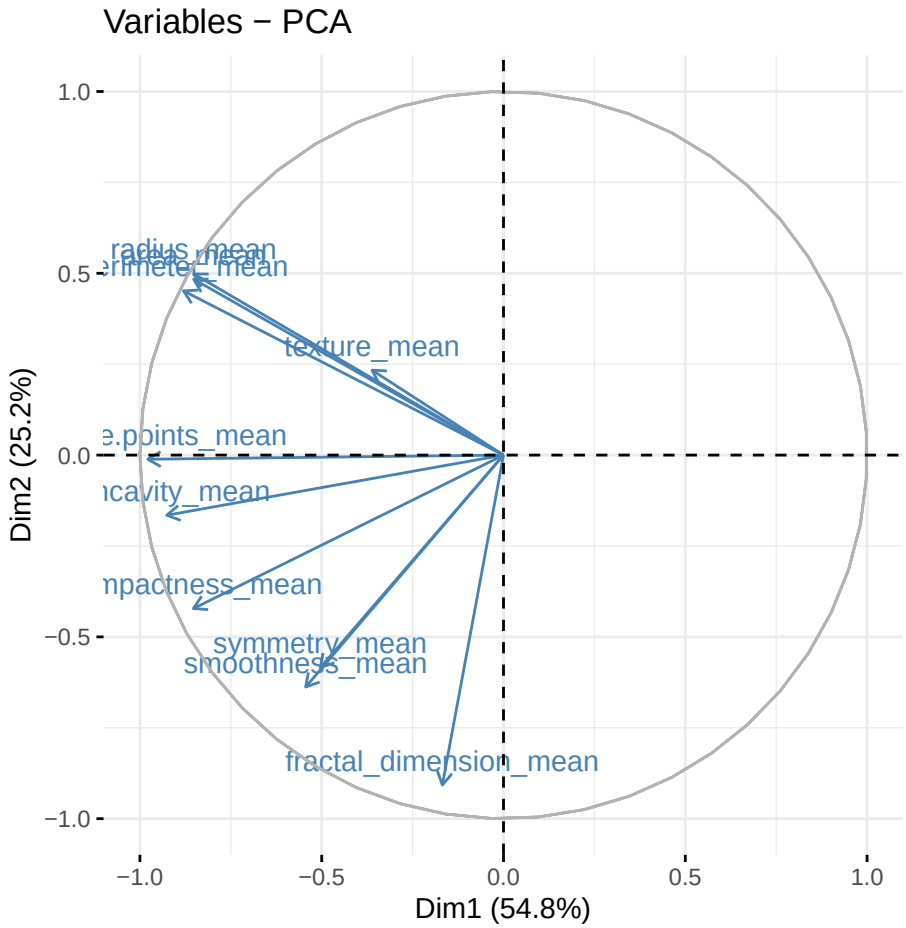
```
fviz_eig(pca)
```



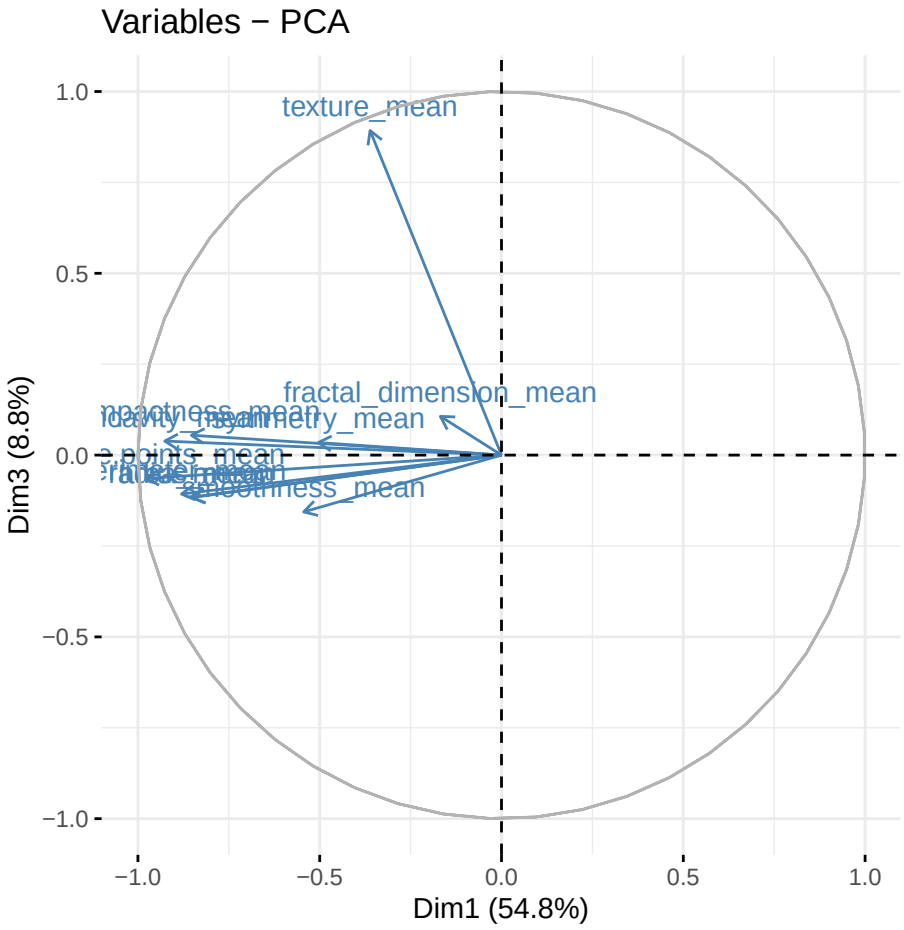
The first 2 components explain approximately 80% of the total variation and 3 components, almost 90%. The number of components that one wants to retain is problem specific and it is a trade-off between information and low-dimensionality.

(c) Provide the circle-correlation plot for the 3 first components

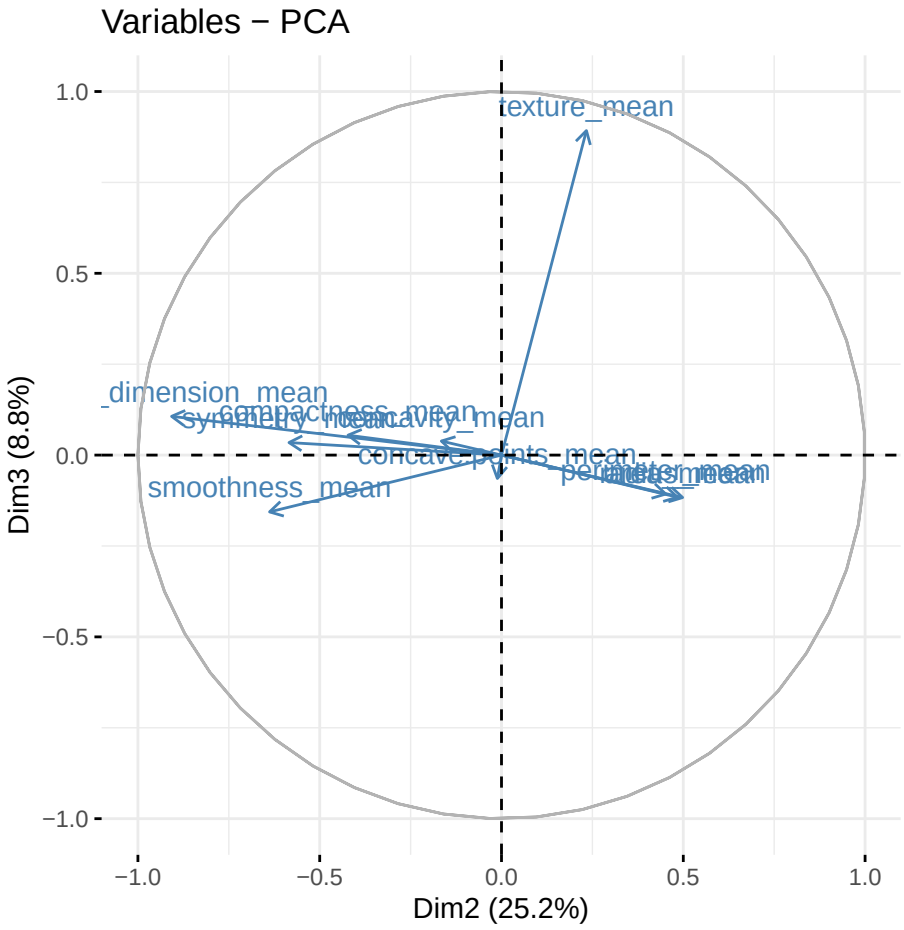
```
par(mfrow=c(1,3))
 fviz_pca_var(pca, col.var = "steelblue")  # comp 1 vs 2
```




```
fviz_pca_var(pca, col.var = "steelblue",axes=c(1,3)) # comp 1 vs 3
```




```
fviz_pca_var(pca, col.var = "steelblue",axes=c(2,3)) # comp 2 vs 3
```



We can see in the plot above that among the 3 components, fractal_dimension_mean contributes mostly to component 2 whereas e.points_mean is contributing to component 1.

(d) using the `eigen()` function that computes eigenvectors and eigenvalues for a matrix.

```
#first scale the data
data.scaled <- apply(Breast.10vars, 2, scale) #apply scale to columns
#get the covariance matrix
cov.X <- cov(data.scaled)

#Get the eigenvalues and eigenvectors
#of the covariance matrix
ev<- eigen(cov.X)

#The sqrt of the eigenvalues are the std
#deviations of the compontents
sqrt(ev$values)  #equal to pca$sdev
 [1] 2.34063837 1.58704555 0.93841099 0.70640600 0.61035989 0.35233755
 [7] 0.28299348 0.18678810 0.10552469 0.01680196
pca$sdev
 [1] 2.34063837 1.58704555 0.93841099 0.70640600 0.61035989 0.35233755
 [7] 0.28299348 0.18678810 0.10552469 0.01680196
#And the eigenvectors are the principal components.
```

```
ev$vector #equal to pca$rotation (up to the sign)
             [,1]         [,2]        [,3]         [,4]         [,5]
 [1,] -0.36393793 -0.313929073 -0.12442759  0.029558858 -0.031067022
 [2,] -0.15445113 -0.147180909  0.95105659  0.008916084 -0.219922761
 [3,] -0.37604434 -0.284657885 -0.11408360  0.013458069 -0.005945081
 [4,] -0.36408585 -0.304841714 -0.12337786  0.013442682 -0.019341222
 [5,] -0.23248053  0.401962324 -0.16653247 -0.107802033 -0.843745292
 [6,] -0.36444206  0.266013147  0.05827786 -0.185700413  0.240182967
 [7,] -0.39574849  0.104285968  0.04114649 -0.166653523  0.312533244
 [8,] -0.41803840  0.007183605 -0.06855383 -0.072983951 -0.009180198
 [9,] -0.21523797  0.368300910  0.03672364  0.892998475  0.112888068
[10,] -0.07183744  0.571767700  0.11358395 -0.349331790  0.264878077
              [,6]        [,7]         [,8]         [,9]         [,10]
 [1,] -0.264180150  0.04418839  0.084834062 -0.474425305  0.6690714888
 [2,] -0.032206572 -0.02055748 -0.007126797 -0.004212629 -0.0002497826
 [3,] -0.237819464  0.08336923  0.089258879 -0.380167210 -0.7404905337
 [4,] -0.331707454 -0.26118796  0.144609749  0.747347357  0.0323589585
 [5,]  0.062225368 -0.01129197  0.170503128 -0.005847386 -0.0036904058
 [6,]  0.005271104  0.80380484  0.063980134  0.218732407  0.0527527802
 [7,]  0.601467155 -0.36713629  0.449573315 -0.081170670  0.0103668020
 [8,]  0.265613395 -0.14131308 -0.850918762  0.022024652  0.0037475480
 [9,] -0.061957003 -0.04790201  0.016455606 -0.009067850 -0.0014669472
[10,] -0.567918997 -0.34521359 -0.065259461 -0.129667491 -0.0070573477
```

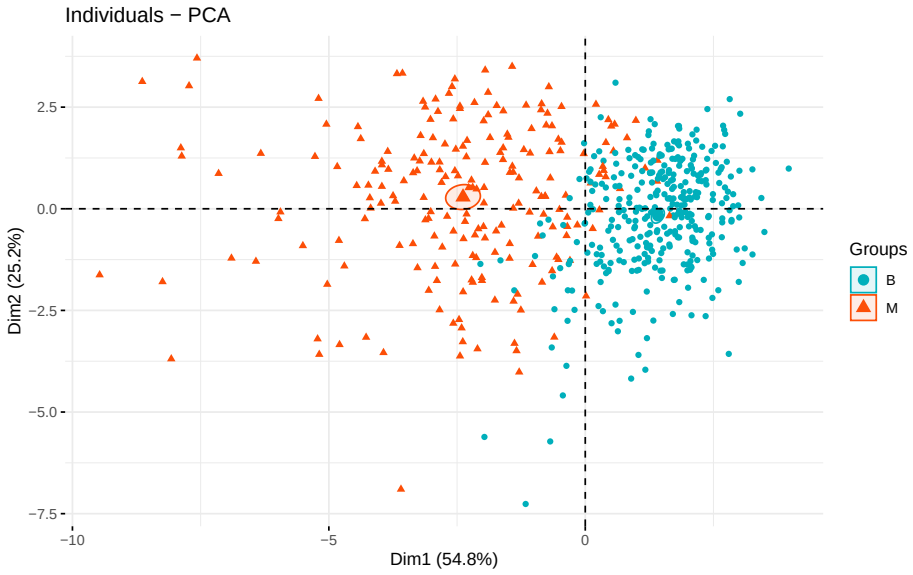
(e) We can also use the SVD of the scaled data to get the same results

```
svd.X <- svd(data.scaled)
svd.X$v[,1:2] # the two first components loading equal to ev$vector[,1:2]
             [,1]         [,2]
 [1,] -0.36393793  0.313929073
 [2,] -0.15445113  0.147180909
 [3,] -0.37604434  0.284657885
 [4,] -0.36408585  0.304841714
 [5,] -0.23248053 -0.401962324
 [6,] -0.36444206 -0.266013147
 [7,] -0.39574849 -0.104285968
 [8,] -0.41803840 -0.007183605
 [9,] -0.21523797 -0.368300910
[10,] -0.07183744 -0.571767700
svd.X$d # equal to sqrt((n-1)*ev$values)
 [1] 55.7838505 37.8236608 22.3649150 16.8355980 14.5465549  8.3971729
 [7]  6.7445131  4.4516741  2.5149437  0.4004369
n <- dim(data.scaled)[1]
sqrt((n-1)*ev$values)
 [1] 55.7838505 37.8236608 22.3649150 16.8355980 14.5465549  8.3971729
 [7]  6.7445131  4.4516741  2.5149437  0.4004369
```

(f) We can visualize the samples on the first two components

```
groups  <-  Breast_cancer_data$diagnosis

fviz_pca_ind(pca,
```

```
                col.ind = groups, # color by groups
                palette = c("#00AFBB",  "#FC4E07", "blue", "red"),
                addEllipses = TRUE, # Concentration ellipses
                ellipse.type = "confidence",
                legend.title = "Groups",
                repel = FALSE,
                label="none",
                axes=c(1,2)
                )
```



(g) We now use some of the principal components as predictors in a logistic model for the variable diagnosis.

```
library(caret)
trctrl <- trainControl(method = "repeatedcv",
                       number = 10,
                       preProcOptions =list(pcaComp = 3),
                       classProbs = TRUE,
                       summaryFunction = twoClassSummary)

breast.glm <- train(diagnosis ~ radius_mean + texture_mean +
                      perimeter_mean + area_mean +
                      smoothness_mean + compactness_mean +
                      concavity_mean + concave.points_mean +
                      symmetry_mean + fractal_dimension_mean,
                    data = Breast_cancer_data,
                    method = "glm",
                    family=binomial,
                    trControl = trctrl,
                    preProcess=c("center", "scale", "pca"), #uses PCA
                    metric="ROC")

breast.glm
```

```
Generalized Linear Model

569 samples
 10 predictor
  2 classes: 'B', 'M'

Pre-processing: centered (10), scaled (10), principal component
 signal extraction (10)
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 512, 513, 512, 513, 512, 513, ...
Resampling results:

  ROC        Sens       Spec
  0.9834192  0.9551587  0.8818182
```

The prediction accuracy is excellent with three components. We can get the coefficients for these components but they do not have an obvious interpretation.

```
summary(breast.glm)

Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9444  -0.1798  -0.0479   0.0283   3.2499

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.5911     0.2022  -2.923  0.00347 **
PC1          -2.7029     0.2882  -9.379  < 2e-16 ***
PC2           0.8271     0.1509   5.481 4.22e-08 ***
PC3           0.7430     0.2144   3.465  0.00053 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 751.44  on 568  degrees of freedom
Residual deviance: 172.10  on 565  degrees of freedom
AIC: 180.1

Number of Fisher Scoring iterations: 8
```

We check this result:

```
model.glm <- glm(factor(groups) ~ pca$x[,1]+pca$x[,2]+pca$x[,3],family=binomial())
summary(model.glm)

Call:
glm(formula = factor(groups) ~ pca$x[, 1] + pca$x[, 2] + pca$x[,
    3], family = binomial())
```

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9444  -0.1798  -0.0479   0.0283   3.2499

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.5911     0.2022  -2.923  0.00347 **
pca$x[, 1]   -2.7029     0.2882  -9.379  < 2e-16 ***
pca$x[, 2]    0.8271     0.1509   5.481 4.22e-08 ***
pca$x[, 3]    0.7430     0.2144   3.465  0.00053 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 751.44  on 568  degrees of freedom
Residual deviance: 172.10  on 565  degrees of freedom
AIC: 180.1

Number of Fisher Scoring iterations: 8
```

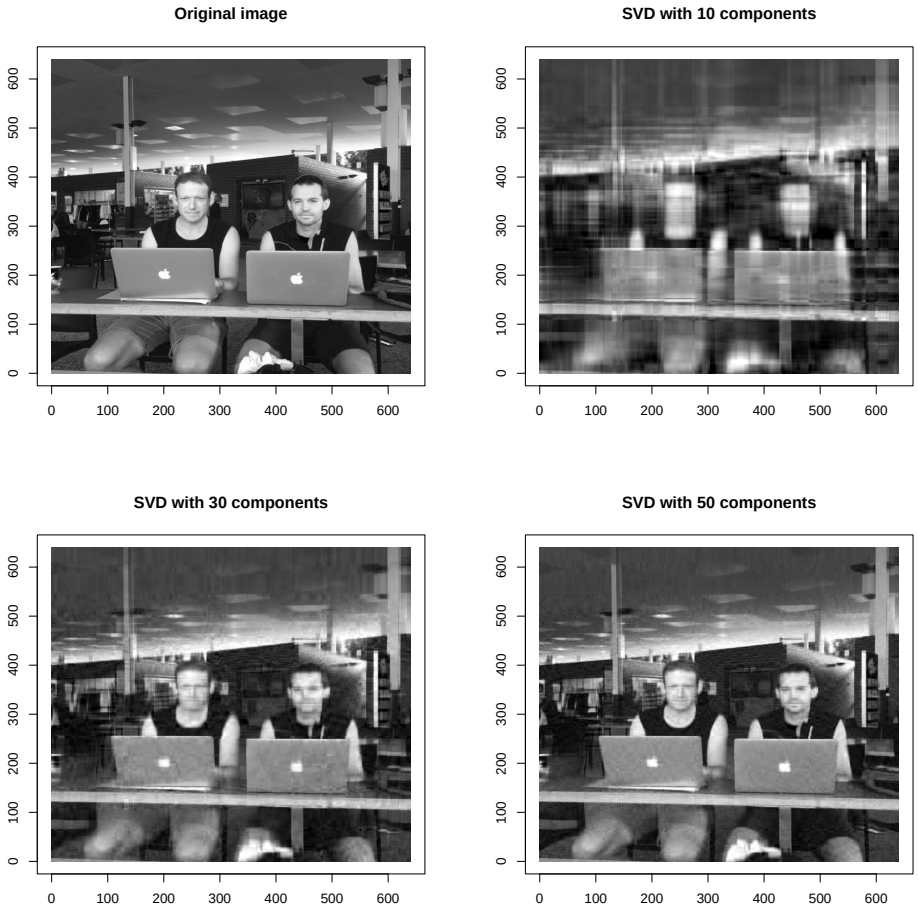
4 Solution Data compression using SVD

Better to wrap it into a function

```
compress.image <- function(rgb.svds, nb.comp) {
  # nb.comp (number of components) should be less than min(dim(img[,,1])),
  svd.lower.dim <- lapply(rgb.svds, function(i) list(d = i$d[1:nb.comp],
                                                     u = i$u[, 1:nb.comp],
                                                     v = i$v[, 1:nb.comp]))
  img <- sapply(svd.lower.dim, function(i) {
    img.compressed <- i$u %*% diag(i$d) %*% t(i$v)
  }, simplify = 'array')
  img[img < 0] <- 0
  img[img > 1] <- 1
  return(list(img = img, svd.reduced = svd.lower.dim))
}
```

```
par(mfrow = c(2, 2))
plot.image(r, "Original image")

rgb.svds <- list(r.svd)

p <- 10 ; plot.image(compress.image(rgb.svds, p)$img[,,1],
                     paste("SVD with", p, "components"))

p <- 30 ; plot.image(compress.image(rgb.svds, p)$img[,,1],
                     paste("SVD with", p, "components"))
```

```
p <- 50 ; plot.image(compress.image(rgb.svds, p)$img[,,1],
                     paste("SVD with", p, "components"))
```

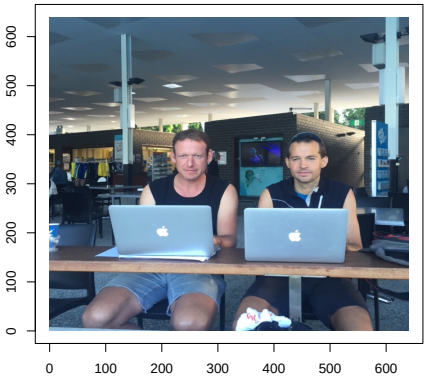


(b) on a colored image. First we perform full SVD of each channel

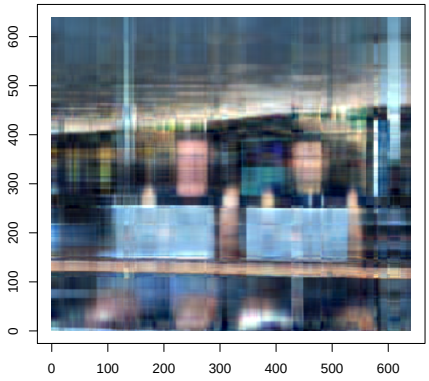
```
r <- mates[, , 1] ; g <- mates[, , 2] ; b <- mates[, , 3]
r.svd <- svd(r) ; g.svd <- svd(g) ;b.svd <- svd(b)
```

```
par(mfrow = c(2, 2))
plot.image(mates, "Original image")

rgb.svds <- list(r.svd,g.svd,b.svd)

p <- 10 ; plot.image(compress.image(rgb.svds, p)$img,
                     paste("SVD with", p, "components"))

p <- 30 ; plot.image(compress.image(rgb.svds, p)$img,
                     paste("SVD with", p, "components"))


p <- 50 ; plot.image(compress.image(rgb.svds, p)$img,
                     paste("SVD with", p, "components"))
```

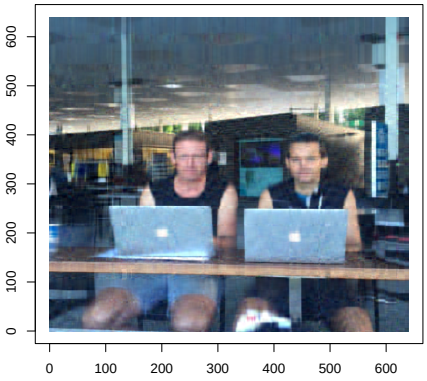
Original image



SVD with 10 components



SVD with 30 components



SVD with 50 components