PCA, SVD and Mahalanobis distance

Christopher Gillies

11/16/2017

Multivariate normal distribution

$$f(x) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}(x-\mu)^T \mathbf{\Sigma}^{-1}(x-\mu)\right) \tag{1}$$

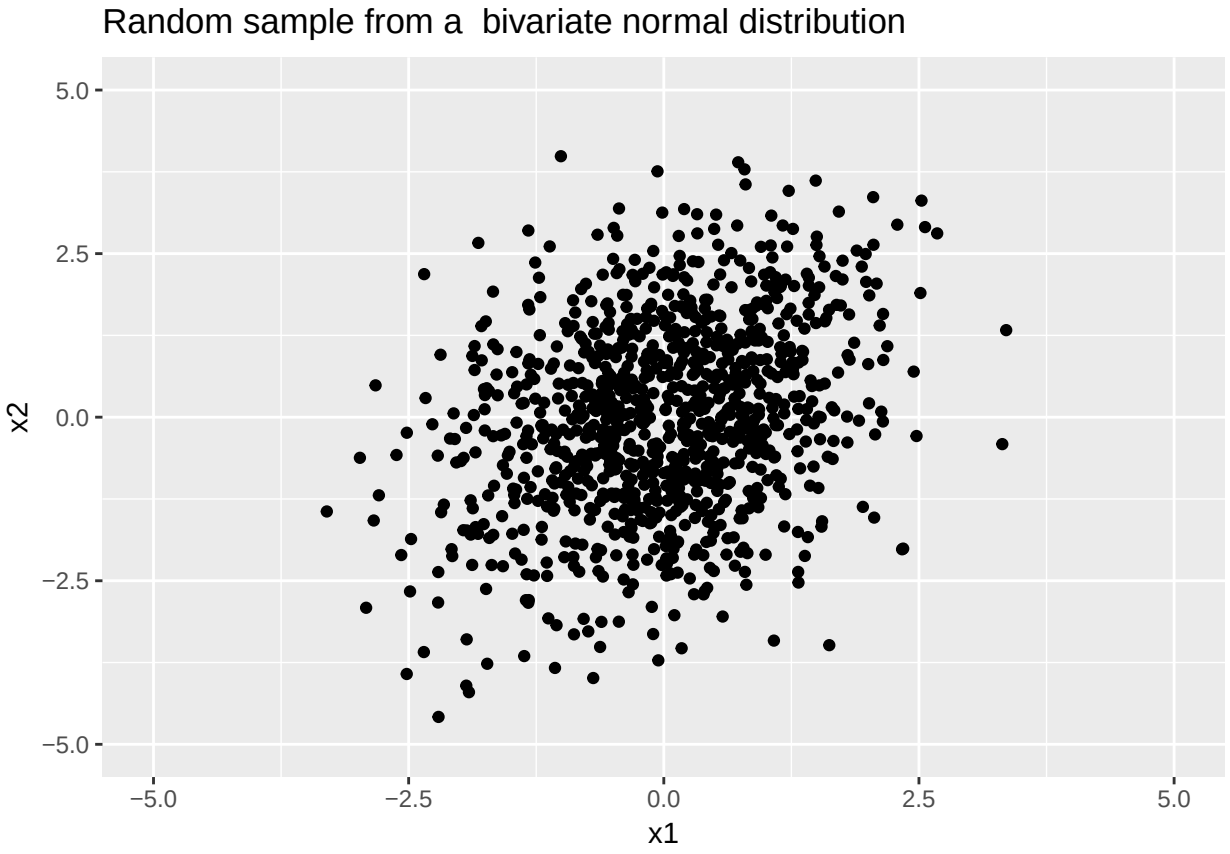
If we assume the data are zero-centered then:

$$f(x) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}(x)^T \mathbf{\Sigma}^{-1}(x)\right) \tag{2}$$

where x is an m-dimensional vector.

Generate a random sample

```
S = matrix(c(1,0.4,0.4,2),ncol=2)
x = mvrnorm(n = 1000, mu=c(0,0), Sigma=S)

ggplot(data.frame()) + geom_point(aes(x=x[,1],y=x[,2])) + scale_x_continuous(limits=c(-5,5))  + scale_y
  ggtitle("Random sample from a  bivariate normal distribution") + xlab("x1") + ylab("x2")
```



Perform PCA

In principal component analysis, we compute the covariance of the features we are studying, and then compute the eigenvectors of this matrix. Let a matrix $X \in \mathbb{R}^{n \times m}$ be a matrix, where we have n observations for m features.

$$X_c = X - \mu_j \tag{3}$$

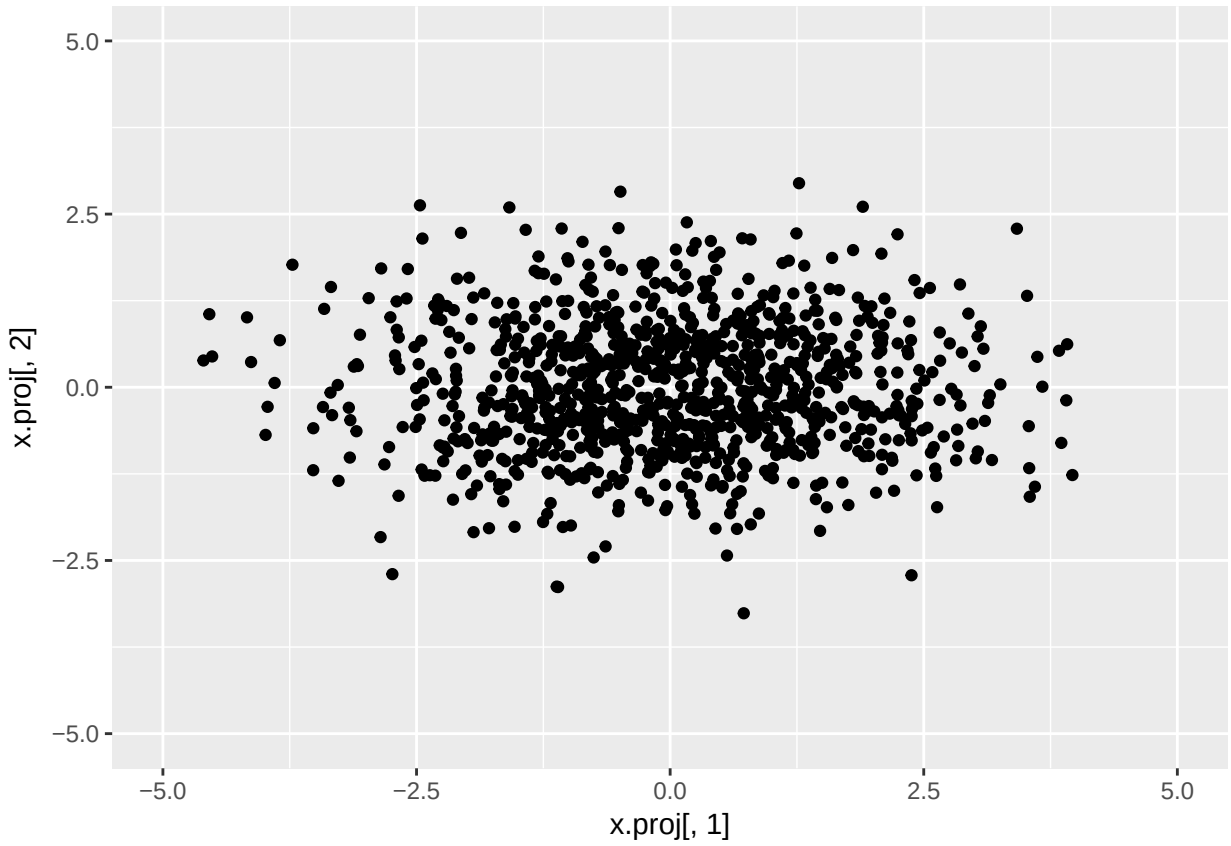
X_c is the centered matrix of X, where we subtract the mean (μ_j) of each column (feature) from the data and $j \in \{1, ..m\}$.

```
S.sample = cov(x)

e.decomp = eigen(S.sample)

e.decomp$vectors %*% diag(e.decomp$values) %*% t(e.decomp$vectors)
```

```
##           [,1]      [,2]
## [1,] 1.0518505 0.4492766
## [2,] 0.4492766 2.1521256
```

```
x.proj = x %*% e.decomp$vectors
ggplot(data.frame()) + geom_point(aes(x=x.proj[,1],y=x.proj[,2])) + scale_x_continuous(limits=c(-5,5))
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
e.decomp$values
```

```
## [1] 2.3122702 0.8917059
```

```
var(x.proj[,1])
```

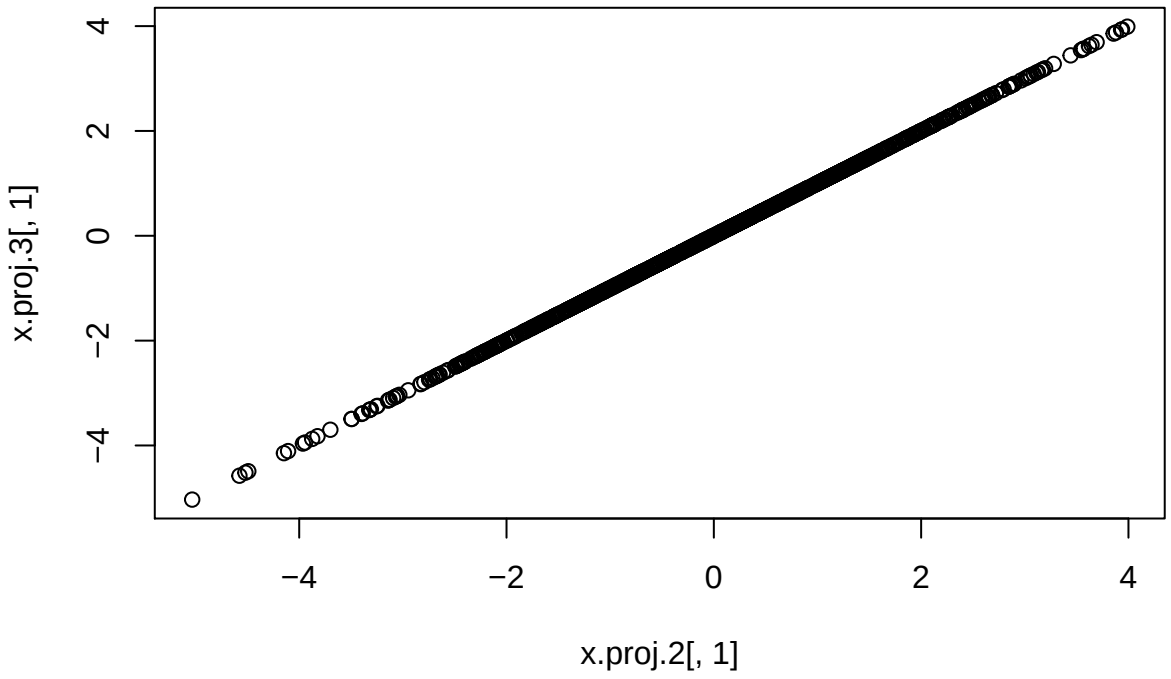
```
## [1] 2.31227
```

```
var(x.proj[,2])
```

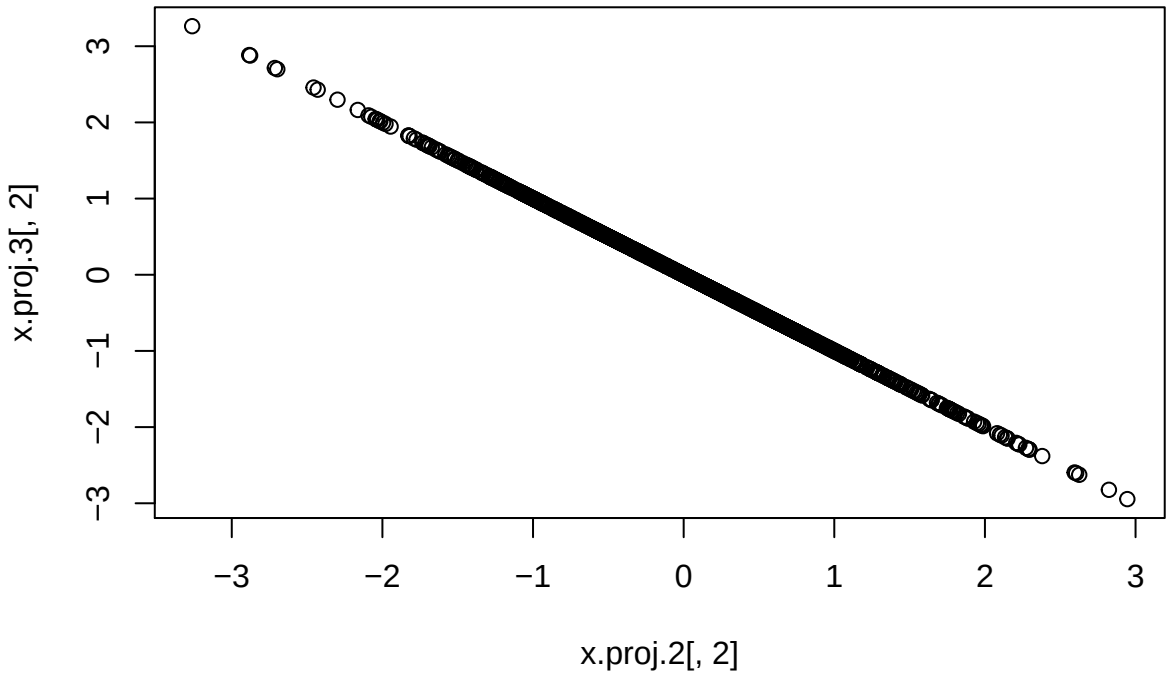
```
## [1] 0.8917059
```

Notice that the variance of the projected data matches the eigenvalues of the covariance matrix of X. The projection of x onto its principal components simply rotates the data.

```
svd.sample.cov = svd(S.sample)


x.center = scale(x,scale = FALSE)

svd.x = svd(x.center)


x.proj.2 = x.center %*% svd.x$v
x.proj.3 = x.center %*% svd.sample.cov$v

plot(x.proj.2[,1],x.proj.3[,1])
```




```
plot(x.proj.2[,2],x.proj.3[,2])
```




```
svd.x$d^2 / (1000 - 1)
```

```
## [1] 2.3122702 0.8917059
```

```
svd.sample.cov$d
```

```
## [1] 2.3122702 0.8917059
```

The same eigenvectors and eigenvalues are computed from the matrix x.center and the covariance matrix. The singlar values σ_i of x and the eigenvalues λ_i of $\mathrm{COV}(X)$ are related as follows:

$$\lambda_i = \frac{\sigma_i^2}{n-1} \tag{4}$$

$$\text{COV}\,[X] = \frac{1}{n-1} X^T X \tag{5}$$

$$X = U\Sigma V^T \tag{6}$$

$$X^T X = (U\Sigma V^T)^T (U\Sigma V^T) \tag{7}$$

$$X^T X = V\Sigma U^T U\Sigma V^T \tag{8}$$

$$X^T X = V\Sigma\Sigma V^T = V\Sigma^2 V^T \tag{9}$$

$$\frac{1}{n-1} X^T X = \frac{1}{n-1} V\Sigma^2 V^T \rightarrow \frac{1}{n-1}\Sigma^2 = \Lambda \tag{10}$$

where Λ is the diagonal matrix of eigenvalues of $\text{COV}\,[X]$. Also note that V is a unitary matrix.

This is the same formula as above for the relationship between the singular values of X and the eigenvalues of its covariance matrix.

What happens if we scale X before running PCA?

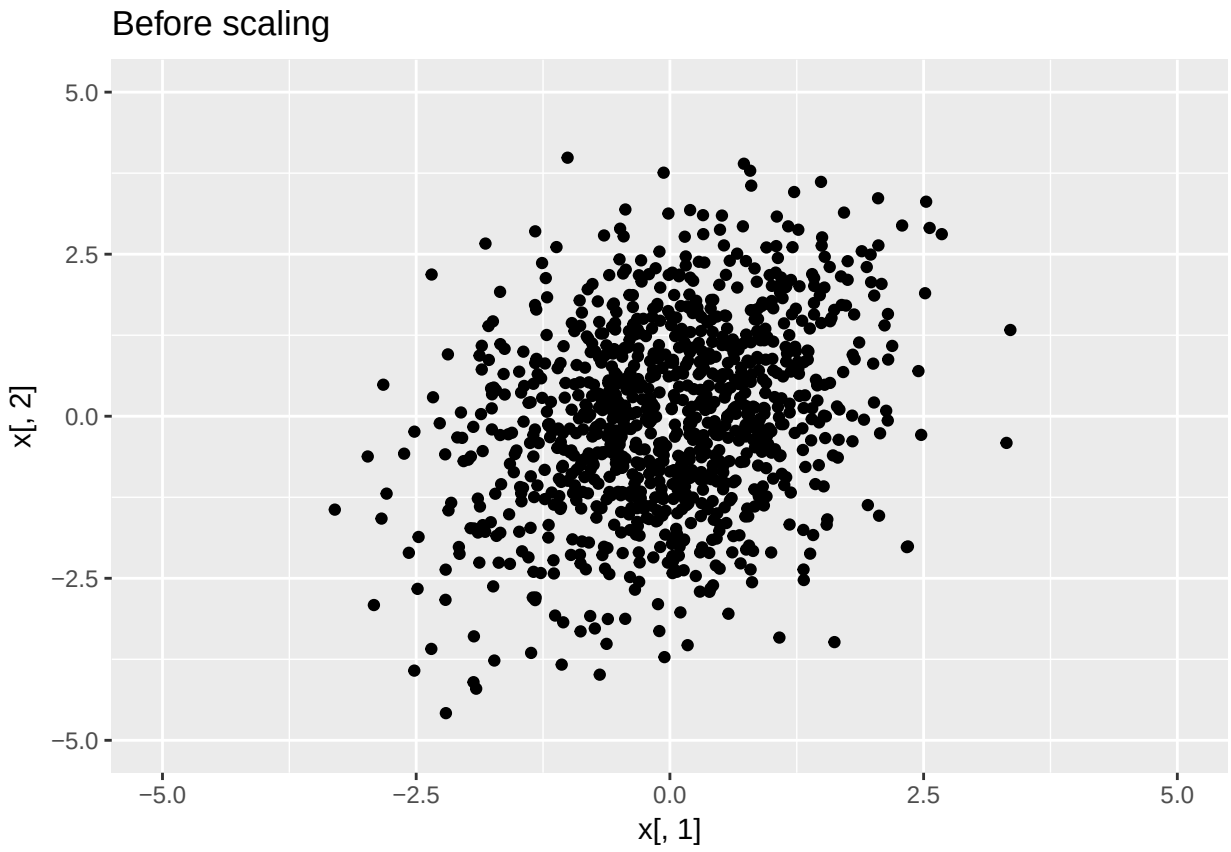
```
x.scaled = scale(x)

cov.scaled.x = cov(x.scaled)
cov.scaled.x
```

```
##          [,1]     [,2]
## [1,] 1.000000 0.298609
## [2,] 0.298609 1.000000
```

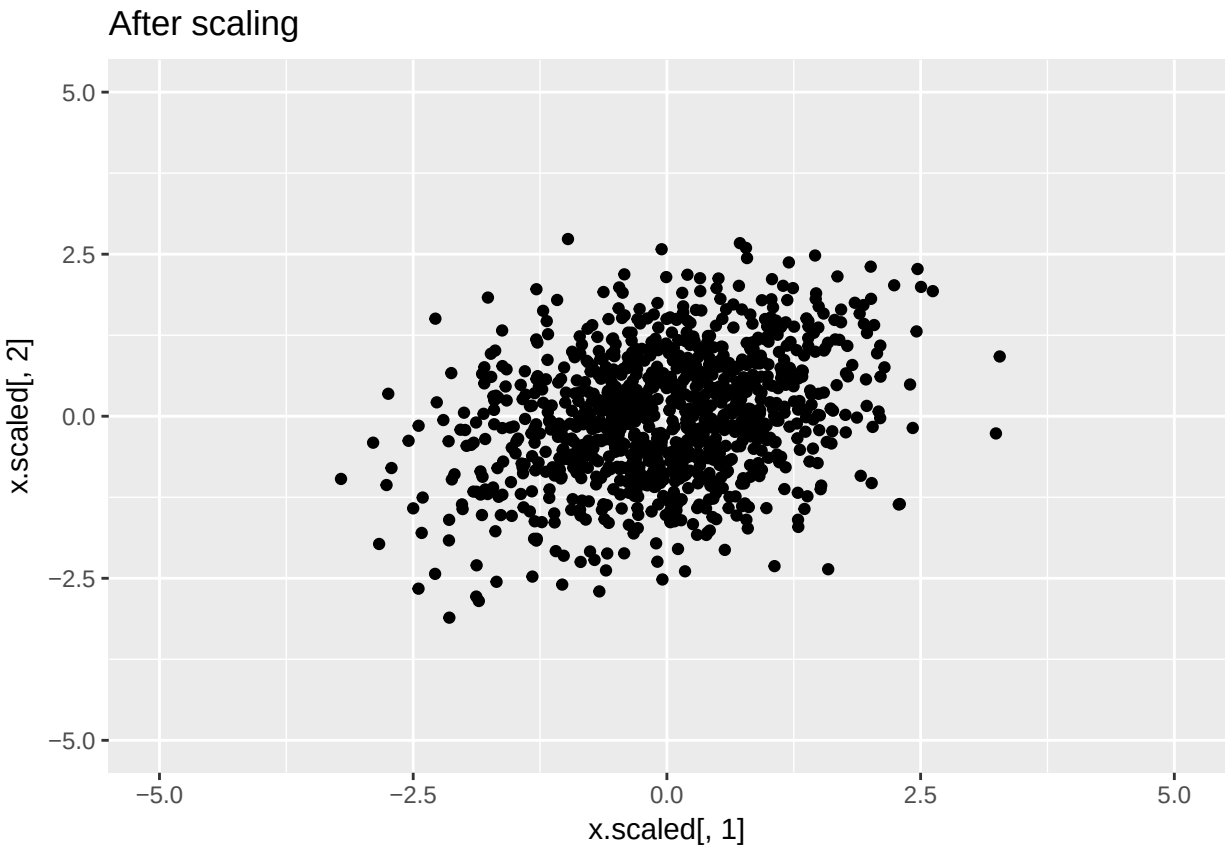
```
S.sample
```

```
##           [,1]      [,2]
## [1,] 1.0518505 0.4492766
## [2,] 0.4492766 2.1521256
```

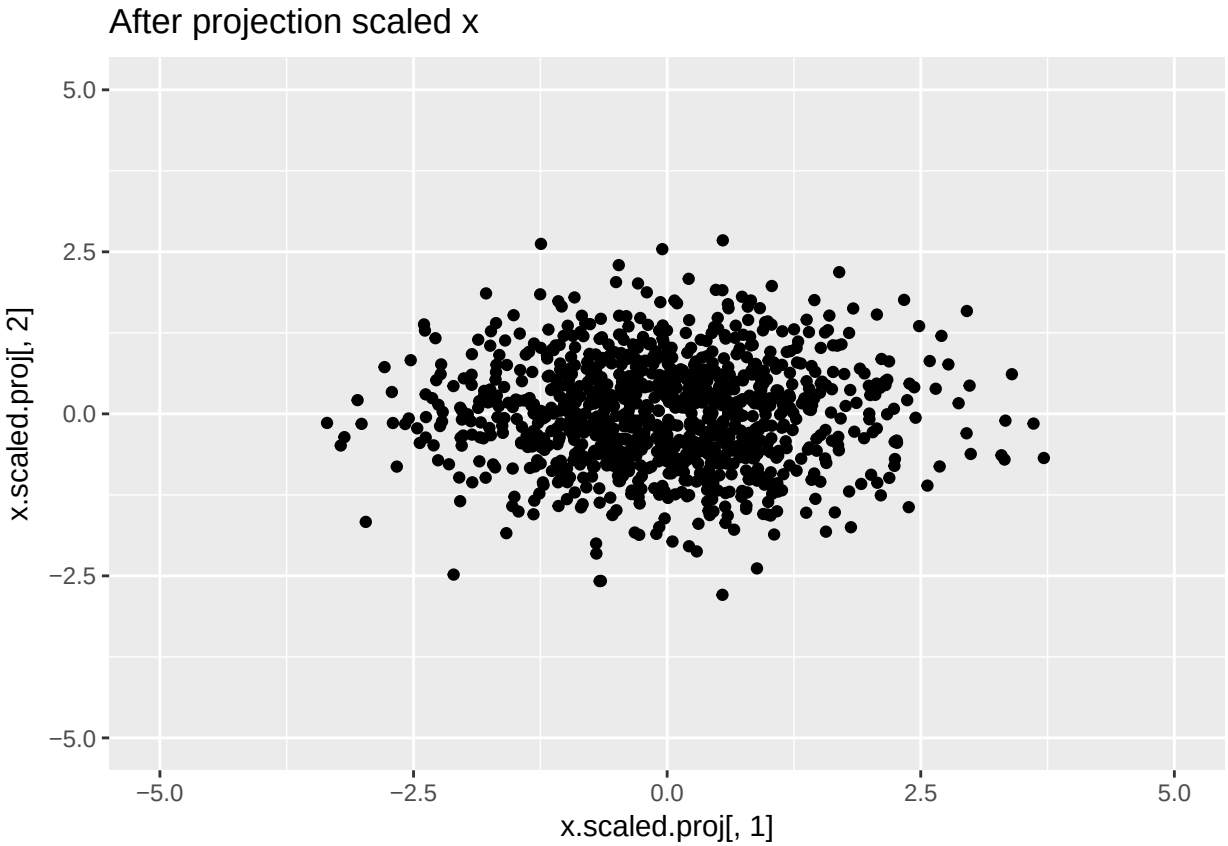
```
ggplot(data.frame()) + geom_point(aes(x=x[,1],y=x[,2])) + scale_x_continuous(limits=c(-5,5))  + scale_y
```




```
ggplot(data.frame()) + geom_point(aes(x=x.scaled[,1],y=x.scaled[,2])) + scale_x_continuous(limits=c(-5,5
```




```
#this is equivalent to the correlation matrix
scaled.svd = svd(cov.scaled.x)

x.scaled.proj = x.scaled %*% scaled.svd$v
ggplot(data.frame()) + geom_point(aes(x=x.scaled.proj[,1],y=x.scaled.proj[,2])) + scale_x_continuous(lim
```




```
svd.sample.cov$d
```

```
## [1] 2.3122702 0.8917059
```

```
scaled.svd$d
```

```
## [1] 1.298609 0.701391
```

It is interesting to note that the angle between the principal axes changes.

$$\cos(\theta) = \frac{v \cdot w}{\|v\|\|x\|} \tag{11}$$

However, we know that the eigenvectors have already been normalized so we can rewrite the equation as:

$$\cos(\theta) = v \cdot w \tag{12}$$

$$\theta = \cos^{-1}(v \cdot w) \tag{13}$$

```
angle = function(v,w) {
  v.norm = v/sqrt(sum(v^2))
  w.norm = w / sqrt(sum(w^2))
  radians = acos(v.norm %*% w.norm)
  radians * 180 / pi
}
angle(c(1,0),-scaled.svd$v[,1])
```

```
##      [,1]
## [1,]   45
```

```
angle(c(1,0),-scaled.svd$v[,2])
```

```
##      [,1]
## [1,]   45
```

```
angle(c(1,0),svd.sample.cov$v[,1])
```

```
##          [,1]
## [1,] 70.38139
```

```
angle(c(1,0),-svd.sample.cov$v[,2])
```

```
##          [,1]
## [1,] 160.3814
```

So the angle between the principal axes changes significantly. In fact for a 2 by 2 matrix the eigenvectors of the correlation matrix are always the same.

Data reconstruction

```
x.projection = svd.sample.cov$v %*% t(x.center)
x.reconstruction = t(svd.sample.cov$v %*% x.projection)
x.reconstruction.partial = t(svd.sample.cov$v %*% rbind(x.projection[1,], 0))

cor(x.reconstruction[,1],x.center[,1])
```

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

```
cor(x.reconstruction[,2],x.center[,2])
```

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

```
cor(x.reconstruction[,1],x.reconstruction.partial[,1])
```

```
## [1] 0.497815
```

```
cor(x.reconstruction[,2],x.reconstruction.partial[,2])
```

```
## [1] 0.976366
```

Notince that the reconstruction works perfectly, and the partial reconstruction works fairly well.

Whitening transform

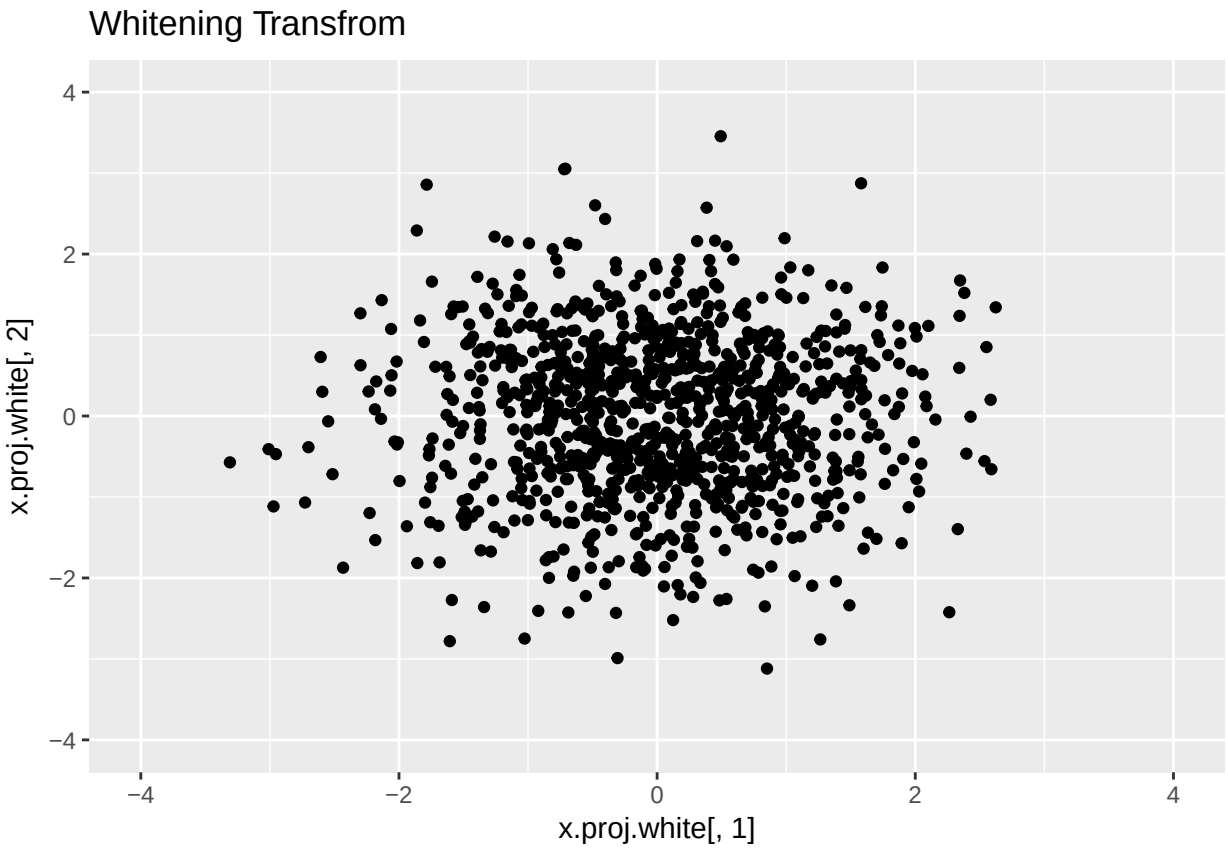
```
x.proj.white = x.center %*% svd.sample.cov$v %*% solve(diag(sqrt(svd.sample.cov$d)))
var(x.proj.white[,1])
```

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

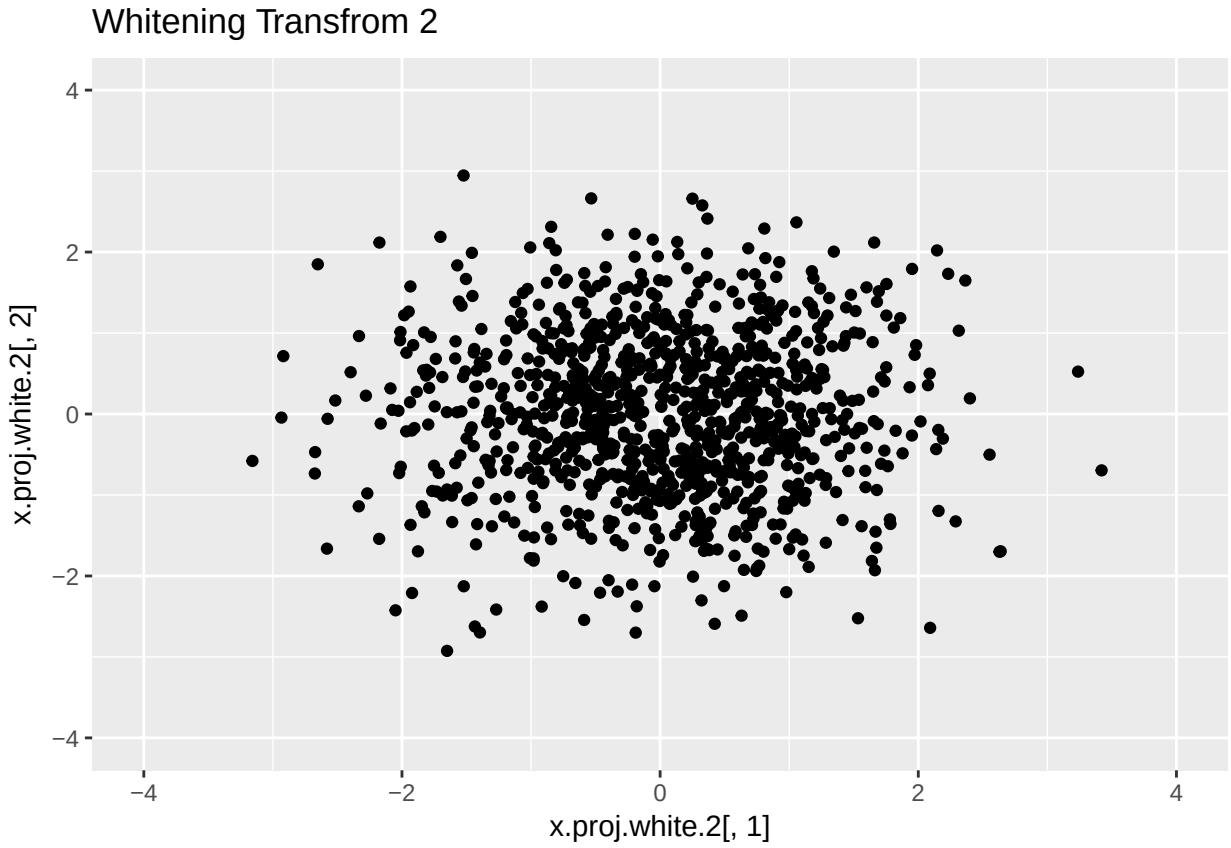
```
var(x.proj.white[,2])
```

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

```
ggplot(data.frame()) + geom_point(aes(x=x.proj.white[,1],y=x.proj.white[,2])) + scale_x_continuous(limi
```




```
w_sqrt = svd.sample.cov$v %*% diag(sqrt(svd.sample.cov$d)) %*% t(svd.sample.cov$v)
w_inv_sqrt = solve(w_sqrt)
w_inv_sqrt.2 = t(svd.sample.cov$v) %*% solve(diag(sqrt(svd.sample.cov$d))) %*% svd.sample.cov$v
x.proj.white.2= x.center %*% w_inv_sqrt.2
ggplot(data.frame()) + geom_point(aes(x=x.proj.white.2[,1],y=x.proj.white.2[,2])) + scale_x_continuous(
```




```
cov(x.proj.white.2)
```

```
##               [,1]          [,2]
## [1,]  1.000000e+00 -6.283154e-17
## [2,] -6.283154e-17  1.000000e+00
```

```
cov(x.proj.white)
```

```
##              [,1]         [,2]
## [1,] 1.000000e+00 1.690772e-17
## [2,] 1.690772e-17 1.000000e+00
```

The above are two different whitening transforms. Please note that:

$$\text{COV}[X] = V\Lambda V^T \tag{14}$$

In a diagonal matrix Λ we can take the square roots by just taking the square root of the diagonal elements

$$\text{COV}[X] = (V\Lambda^{1/2}V^T)^2 = V\Lambda^{1/2}(V^TV)\Lambda^{1/2}V^T = V\Lambda V^T \tag{15}$$

Since, $V^TV = I$. That is $V^T = V^{-1}$.

$$\text{COV}[X]^{1/2} = V\Lambda^{1/2}V^T \tag{16}$$

Now $(ABC)^-1 = C^{-1}B^{-1}A^{-1}$. Therefore

$$\text{COV}[X]^{-1/2} = (V\Lambda^{1/2}V^T)^{-1} = V^{-1}\Lambda^{-1/2}(V^{-1})^-1 = V^T\Lambda^{-1/2}V \tag{17}$$

Since $V^T = V^{-1}$.

Mahalanobis distance^2

The Mahalanobis of a centered vector from the origin is:

$$D_M(x)^2 = x^T S^{-1} x \tag{18}$$

where S is the covariance matrix of the vector x.

```
t(x.center[1,]) %*% solve(S.sample) %*% x.center[1,]
```

```
##          [,1]
## [1,] 3.057022
```

```
sum(x.proj.white[1,]^2)
```

```
## [1] 3.057022
```

```
sum(x.proj.white.2[1,]^2)
```

```
## [1] 3.057022
```

```
x.1.proj.scaled = x.center[1,] %*% svd.sample.cov$v %*% solve(diag(sqrt(svd.sample.cov$d)))
sum(x.1.proj.scaled^2)
```

```
## [1] 3.057022
```

Mahalanobis distance^2 is the equal to the $\|x\|^2$ in the Whitened space.

The norm of a vector

$$\|x\|^2 = \sum x_i^2 = x^T x \tag{19}$$

$$(x^T)^T = x \tag{20}$$

$$D_M(x)^2 = x^T \mathrm{COV}[X]^{-1} x \tag{21}$$

$$\mathrm{COV}[X]^{-1} = (V \Lambda V^T)^{-1} = V^T \Lambda^{-1} V \tag{22}$$

The Mahalanobis distance^2 can be written as

$$D_M(x)^2 = x^T V^T \Lambda^{-1} V x = x^T V^T \Lambda^{-1/2} \Lambda^{-1/2} V x = (x^T V^T \Lambda^{-1/2})(\Lambda^{-1/2} V x) = (\Lambda^{-1/2} V x)^T (\Lambda^{-1/2} V x) \tag{23}$$

$$= (\Lambda^{-1/2} V x)^T (\Lambda^{-1/2} V x) = \|\Lambda^{-1/2} V x\|^2 = \|x^T V^T \Lambda^{-1/2}\|^2 = D_M(x)^2 \tag{24}$$

The final equality $\|x^T V^T \Lambda^{-1/2}\|^2$ shows that the $D_M(x)^2$ is the same as projecting the x onto its principal components, then scaling each axis by the square root of its eigenvalue (if the eigenvalue is the variance then the sqrt(eigenvalue) is like the standard deviation) and finally taking the norm of the scaled projected x. A eigenvalue is the variance of the data projected onto its corresponding eigenvector, so to scale it you divide by the standard deviation.

What about PCA and SVD in the case where there are more variables than observations?

Generate random samples. Assume we have $n = 100$ subjects and $m = 200$ snps.

```
n = 100
m = 1000
z = rbinom(n,size = 2, prob = 0.5)
X = matrix(rep(0,n * m),ncol=n)
snp_afs_0 = runif(m,0.01,0.5)
snp_afs_1 = runif(m,0.25,0.75)
snp_afs_2 = c(snp_afs_0[1:(m/2)],snp_afs_1[(m/2 + 1):m])
for(i in 1:n) {
  x_i = NULL
  if(z[i] == 0) {
    x_i = rbinom(m,size=2,prob=snp_afs_0)
  } else if(z[i] == 1) {
    x_i = rbinom(m,size=2,prob=snp_afs_1)
  } else {
    x_i = rbinom(m,size=2,prob=snp_afs_2)
  }
  X[,i] = x_i
}
```

Standardize by subtracting off the row means and dividing by the standard deviation. Also compute covariance matrix.

```
X_std = t(scale(t(X)))
ind_cov = cov(X_std)
ind_cor_x = cor(X)

#approx covariance matrix
Xt_X = t(X_std) %*% X_std * 1/(n-1)

dim(ind_cov)
```

```
## [1] 100 100
```

```
dim(Xt_X)
```

```
## [1] 100 100
```

```
sign(Xt_X[1:10,1:10]) == sign(ind_cov[1:10,1:10])
```

```
##       [,1]  [,2] [,3]  [,4] [,5] [,6]  [,7]  [,8]  [,9] [,10]
##  [1,] TRUE  TRUE TRUE  TRUE TRUE TRUE  TRUE  TRUE  TRUE  TRUE
##  [2,] TRUE  TRUE TRUE FALSE TRUE TRUE  TRUE  TRUE FALSE  TRUE
##  [3,] TRUE  TRUE TRUE  TRUE TRUE TRUE  TRUE  TRUE  TRUE  TRUE
##  [4,] TRUE FALSE TRUE  TRUE TRUE TRUE  TRUE  TRUE  TRUE  TRUE
##  [5,] TRUE  TRUE TRUE  TRUE TRUE TRUE  TRUE  TRUE  TRUE  TRUE
##  [6,] TRUE  TRUE TRUE  TRUE TRUE TRUE  TRUE  TRUE  TRUE  TRUE
##  [7,] TRUE  TRUE TRUE  TRUE TRUE TRUE  TRUE FALSE  TRUE FALSE
##  [8,] TRUE  TRUE TRUE  TRUE TRUE TRUE FALSE  TRUE  TRUE  TRUE
##  [9,] TRUE FALSE TRUE  TRUE TRUE TRUE  TRUE  TRUE  TRUE  TRUE
## [10,] TRUE  TRUE TRUE  TRUE TRUE TRUE FALSE  TRUE  TRUE  TRUE
```

```
Xt_X[1:10,1:10] / ind_cov[1:10,1:10]
```

```
##           [,1]       [,2]     [,3]      [,4]     [,5]      [,6]      [,7]
##  [1,] 10.89980  68.149867 21.96110 18.162657 18.78374  16.05802  22.48561
##  [2,] 68.14987  11.107380 25.36077 -9.410148 30.69862  26.73879  27.68091
##  [3,] 21.96110  25.360774 10.89818 21.373433 50.04205  30.64854  26.45908
##  [4,] 18.16266  -9.410148 21.37343 10.265644 12.41108  14.15360  28.71394
##  [5,] 18.78374  30.698618 50.04205 12.411082 10.72031  20.30709  17.11011
##  [6,] 16.05802  26.738794 30.64854 14.153603 20.30709  10.65742  20.09903
##  [7,] 22.48561  27.680913 26.45908 28.713937 17.11011  20.09903  12.05147
##  [8,] 15.26777  17.499946 13.83670 11.364970 14.80771  12.11415 -73.12472
##  [9,] 33.95008 -79.140625 22.90666 14.245266 18.85495  26.23400  30.00875
## [10,] 14.96587  15.904237 15.84849 12.083150 17.74304 108.20594  -1.53973
##            [,8]      [,9]     [,10]
##  [1,]  15.26777  33.95008  14.96587
##  [2,]  17.49995 -79.14062  15.90424
##  [3,]  13.83670  22.90666  15.84849
##  [4,]  11.36497  14.24527  12.08315
##  [5,]  14.80771  18.85495  17.74304
##  [6,]  12.11415  26.23400 108.20594
##  [7,] -73.12472  30.00875  -1.53973
##  [8,]  10.17106  24.13391  11.17589
##  [9,]  24.13391  10.70280  14.25806
## [10,]  11.17589  14.25806  10.22376
```

Now let us compare the eigenvalues of each

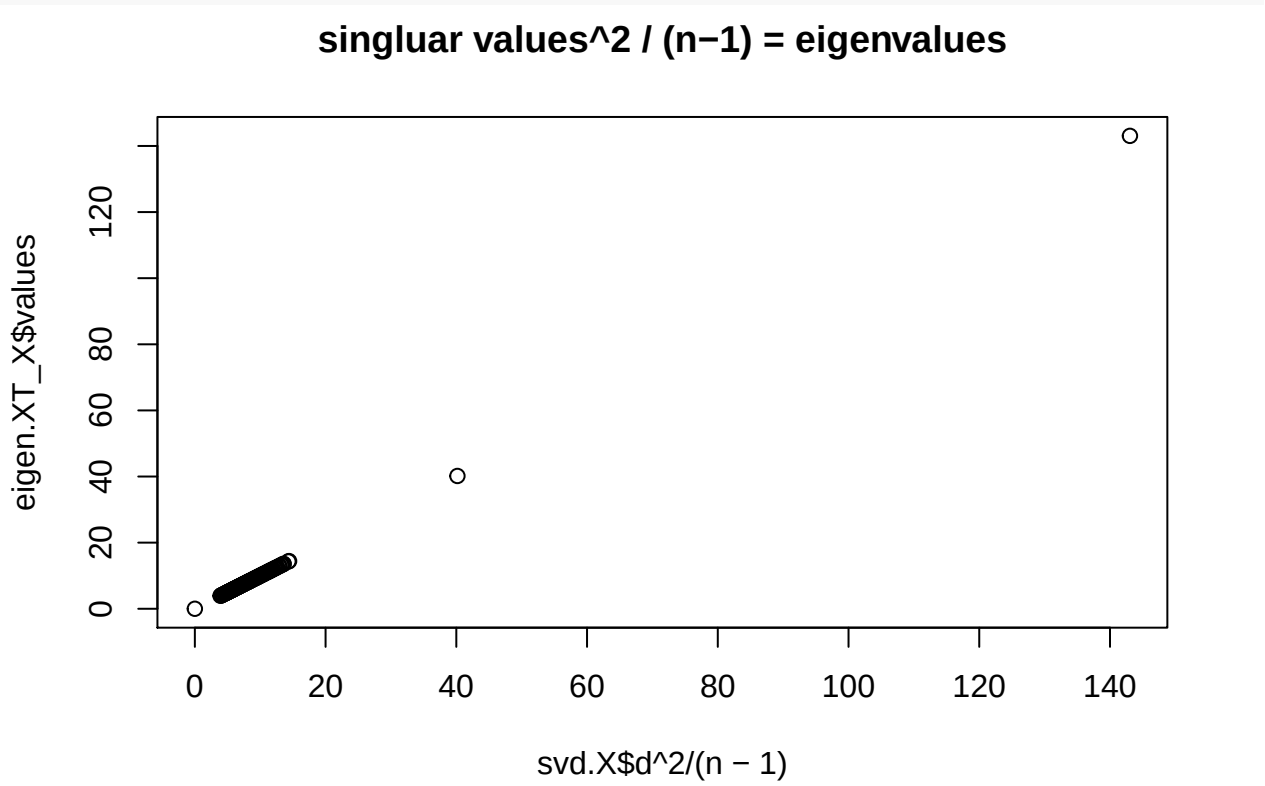
```
eigen.XT_X = eigen(Xt_X)
eigen.ind_cov = eigen(ind_cov)
eigen.ind_cor = eigen(cor(X_std))
eigen.ind_cor_x = eigen(ind_cor_x)
svd.X = svd(X_std)

color = function(x) {
  a_func = function(a) {
    if(a == 0) {
      return("black")
    } else if(a == 1) {
      return("red")
    } else {
      return("blue")
    }
  }
  sapply(x,FUN=a_func)
}

(svd.X$d^2 / (n - 1))[1:10]
```

```
##  [1] 143.05428  40.16840  14.42032  14.31081  13.59077  13.53872  13.33835
##  [8]  13.20337  12.94428  12.86818
```

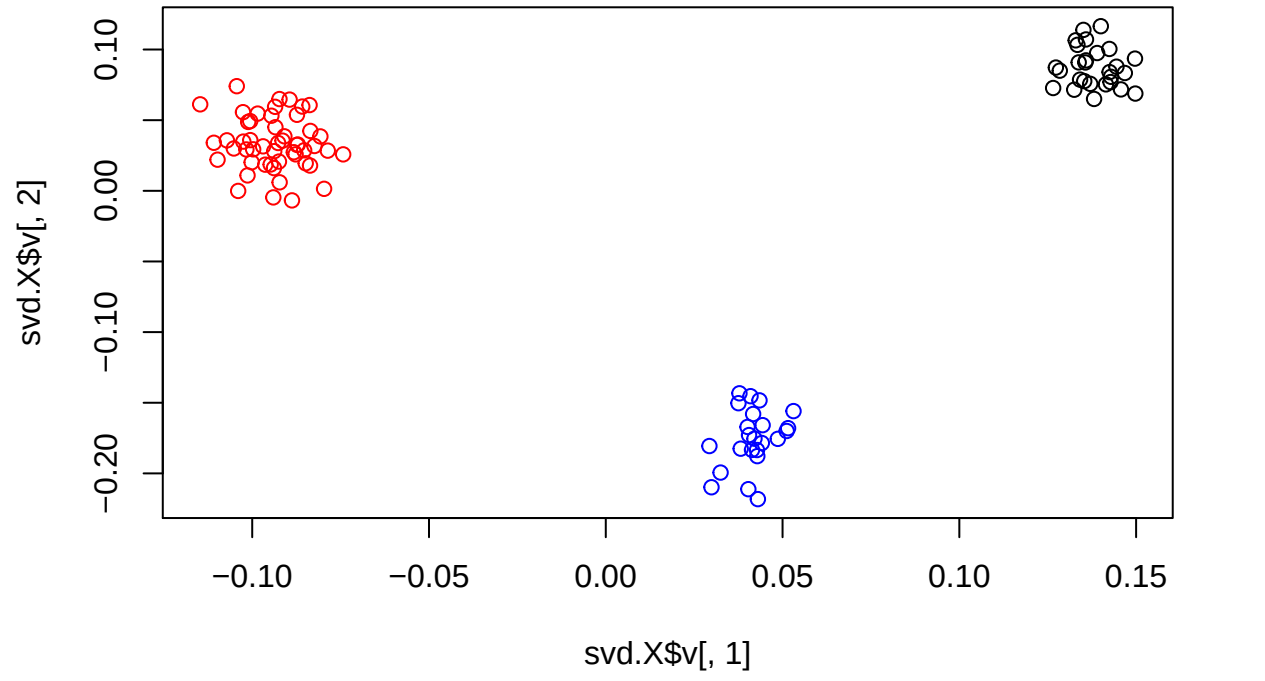
```
eigen.XT_X$values[1:10]
```

```
##  [1] 143.05428  40.16840  14.42032  14.31081  13.59077  13.53872  13.33835
##  [8]  13.20337  12.94428  12.86818
```

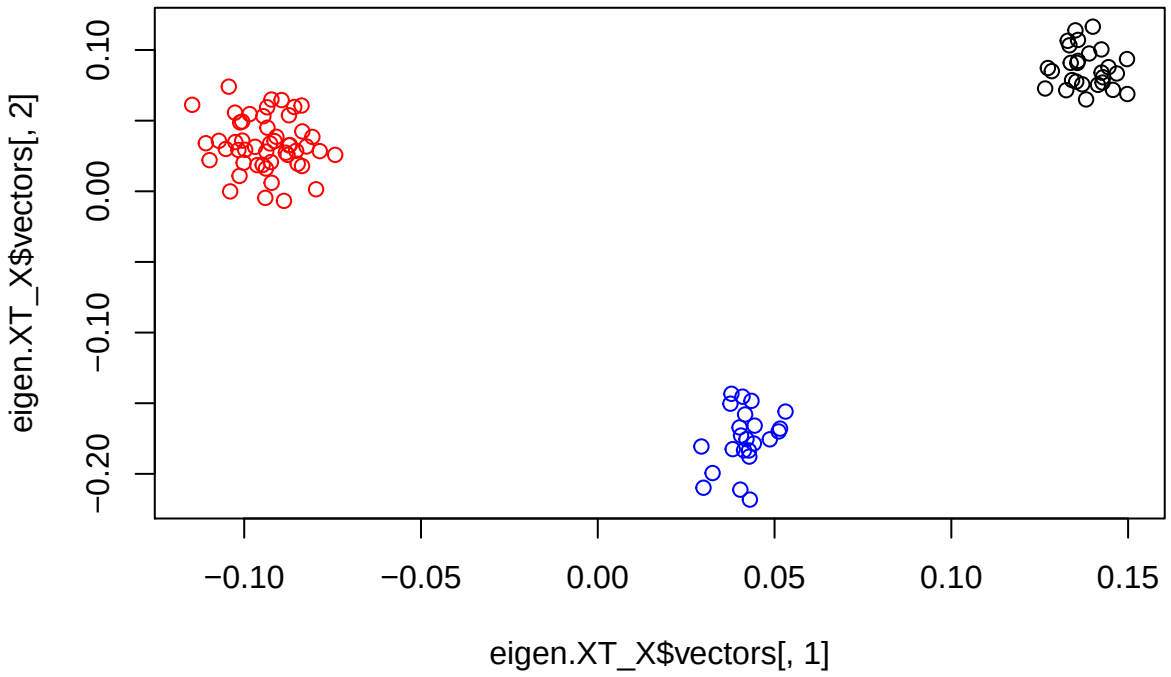
```
plot(svd.X$d^2 / (n - 1),eigen.XT_X$values,main="singluar values^2 / (n-1) = eigenvalues")
```



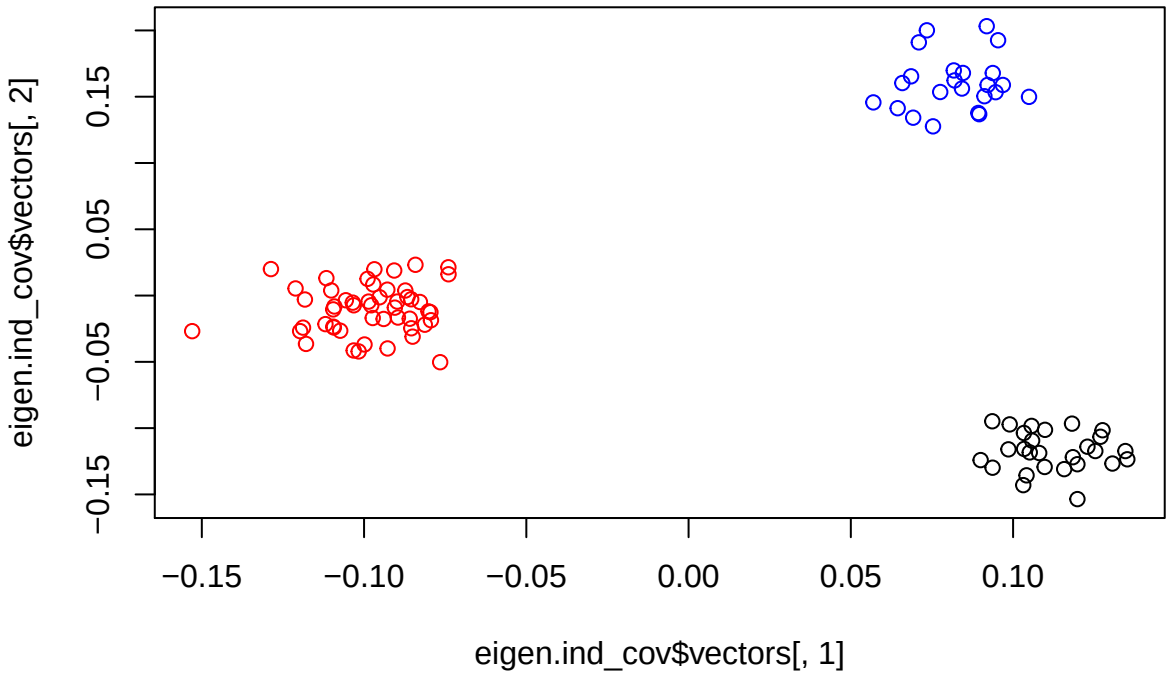

```
plot(svd.X$v[,1],svd.X$v[,2],col=color(z))
```



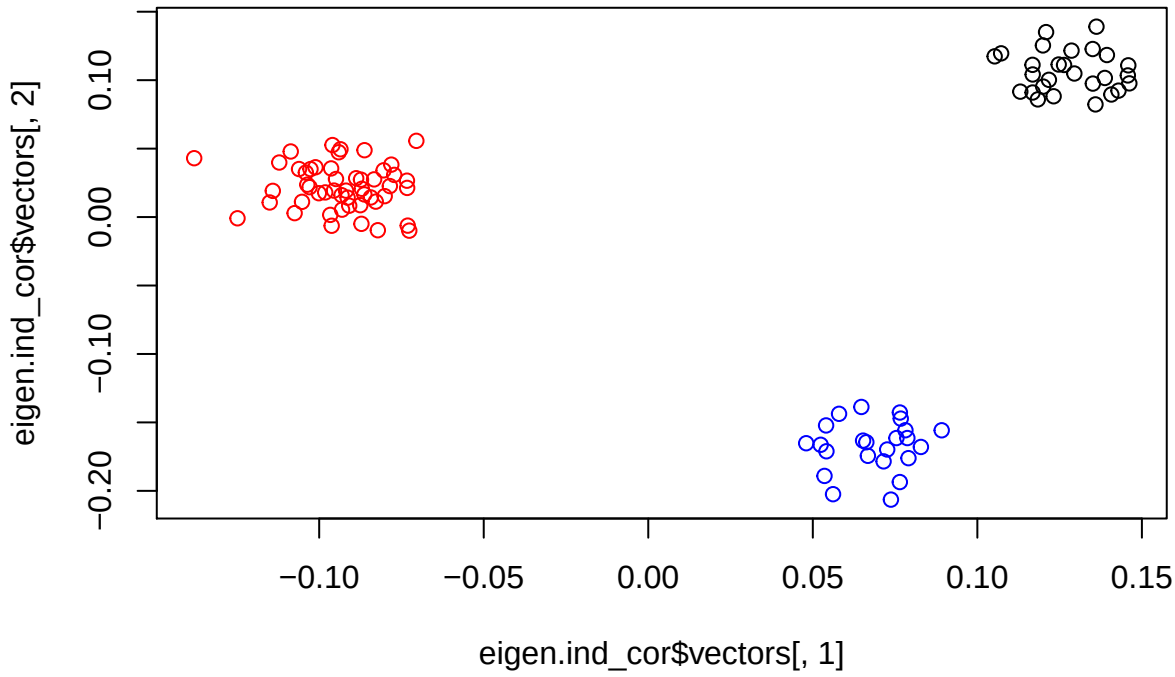

```
plot(eigen.XT_X$vectors[,1],eigen.XT_X$vectors[,2],col=color(z))
```



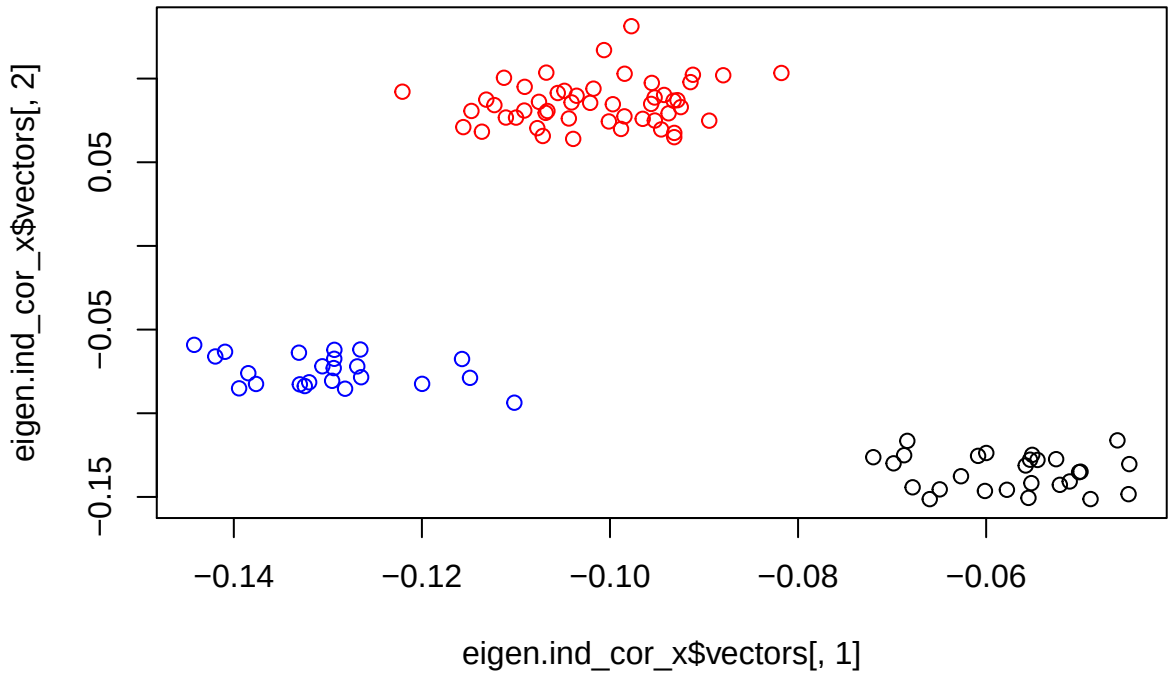

```
plot(eigen.ind_cov$vectors[,1],eigen.ind_cov$vectors[,2],col=color(z))
```



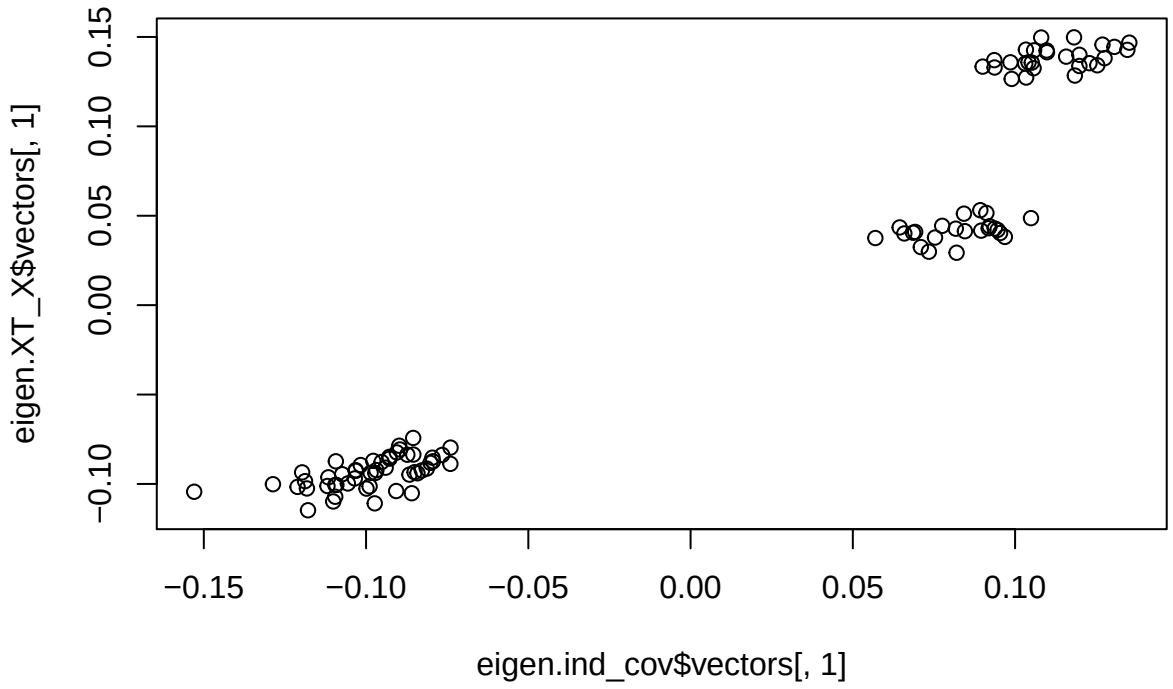

```
plot(eigen.ind_cor$vectors[,1],eigen.ind_cor$vectors[,2],col=color(z))
```



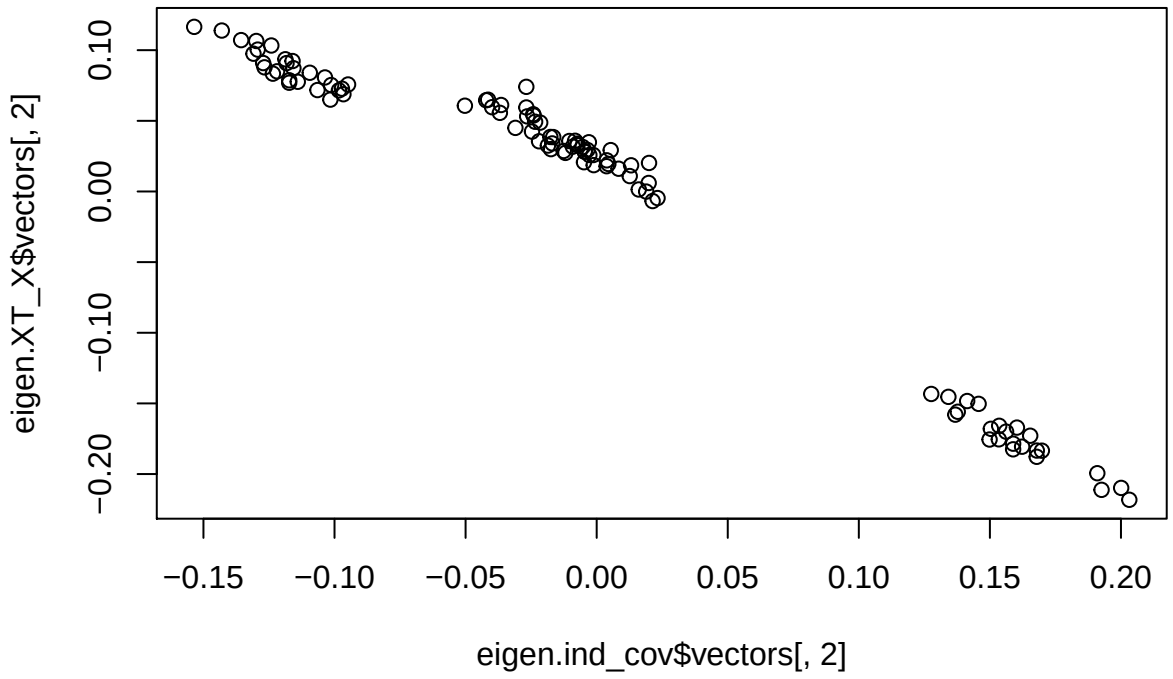

```
plot(eigen.ind_cor_x$vectors[,1],eigen.ind_cor_x$vectors[,2],col=color(z))
```



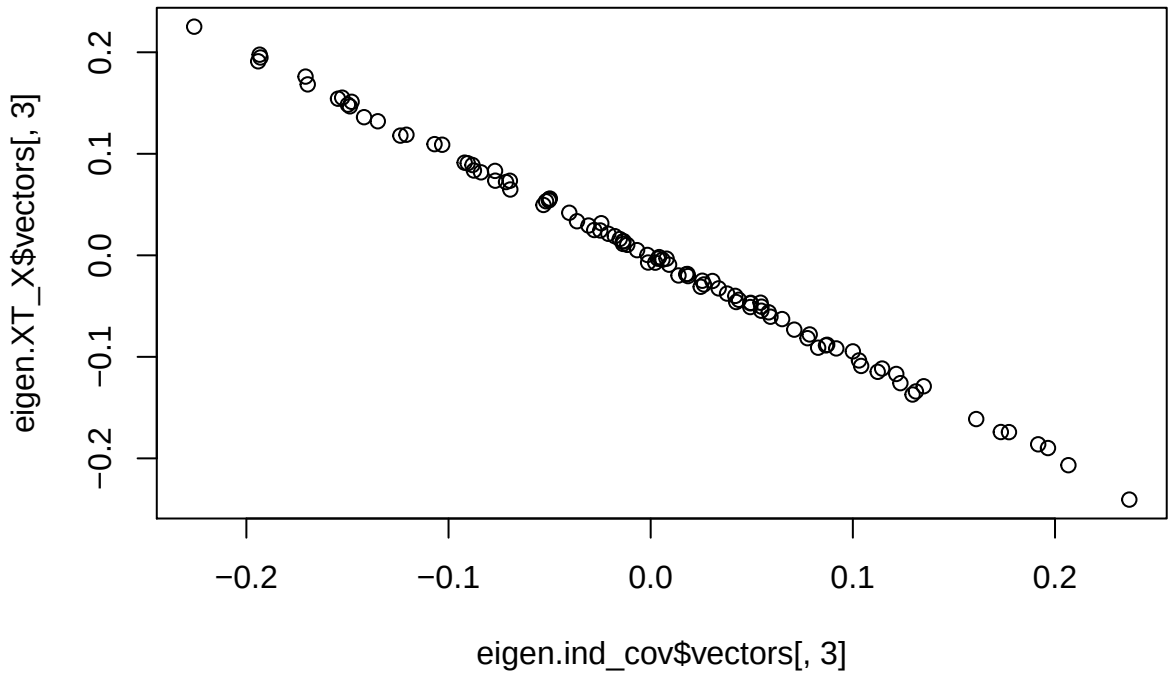

```
plot(eigen.ind_cov$vectors[,1],eigen.XT_X$vectors[,1])
```



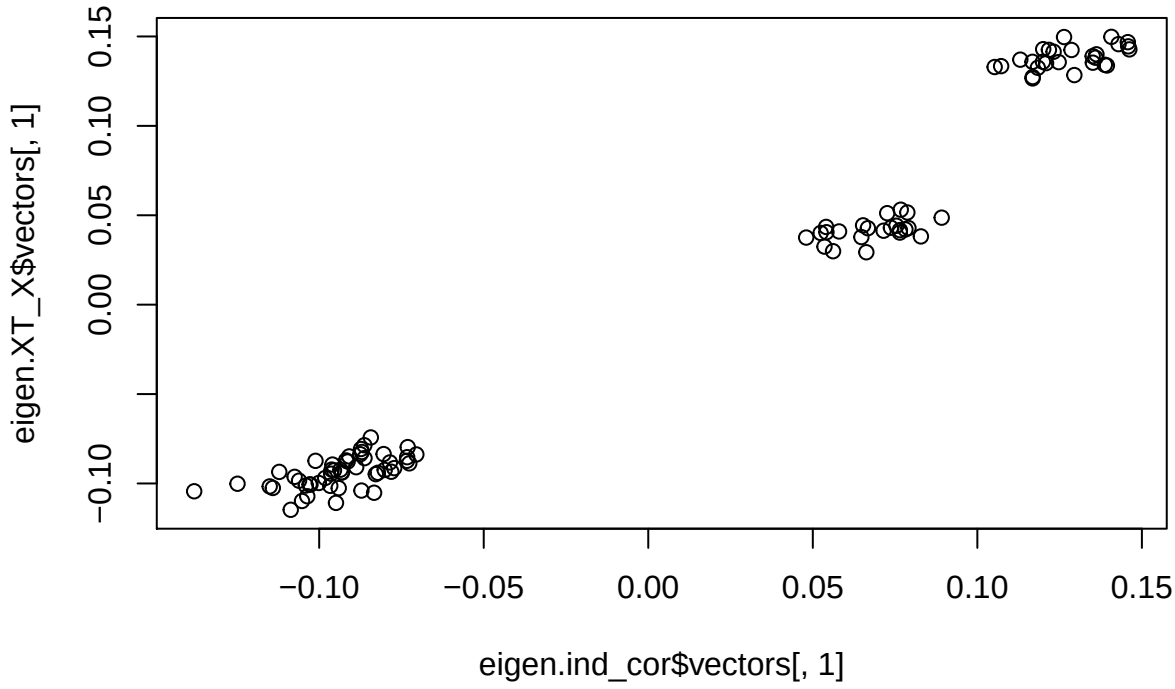

```
plot(eigen.ind_cov$vectors[,2],eigen.XT_X$vectors[,2])
```



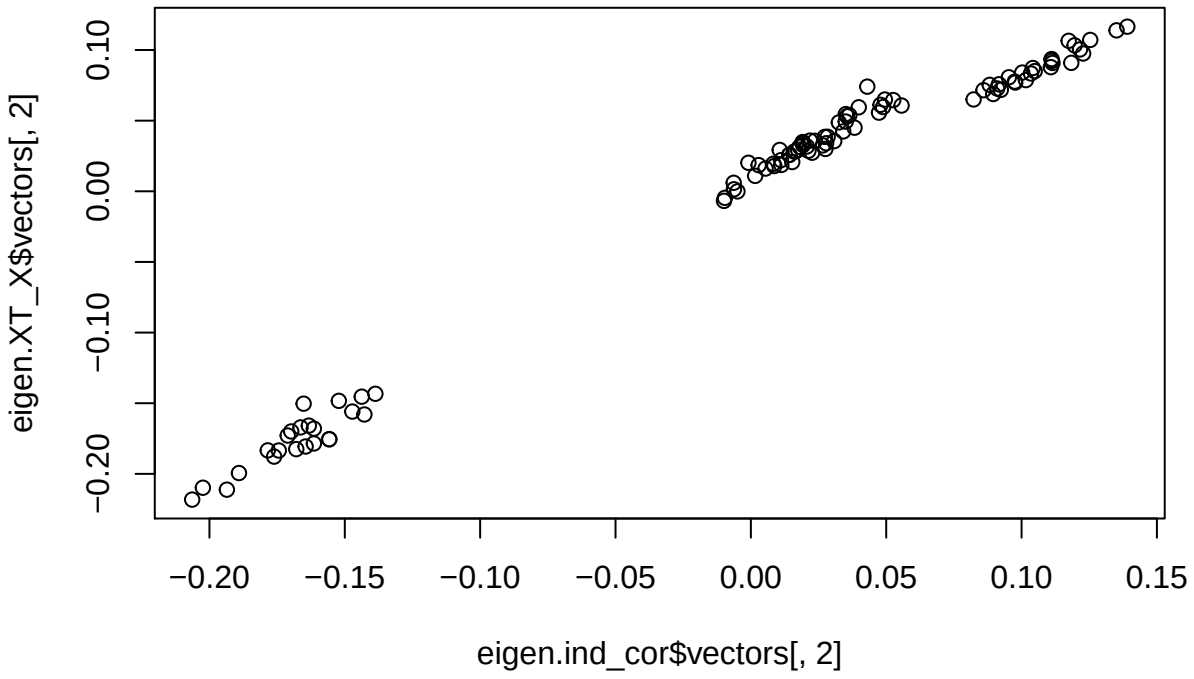

```
plot(eigen.ind_cov$vectors[,3],eigen.XT_X$vectors[,3])
```



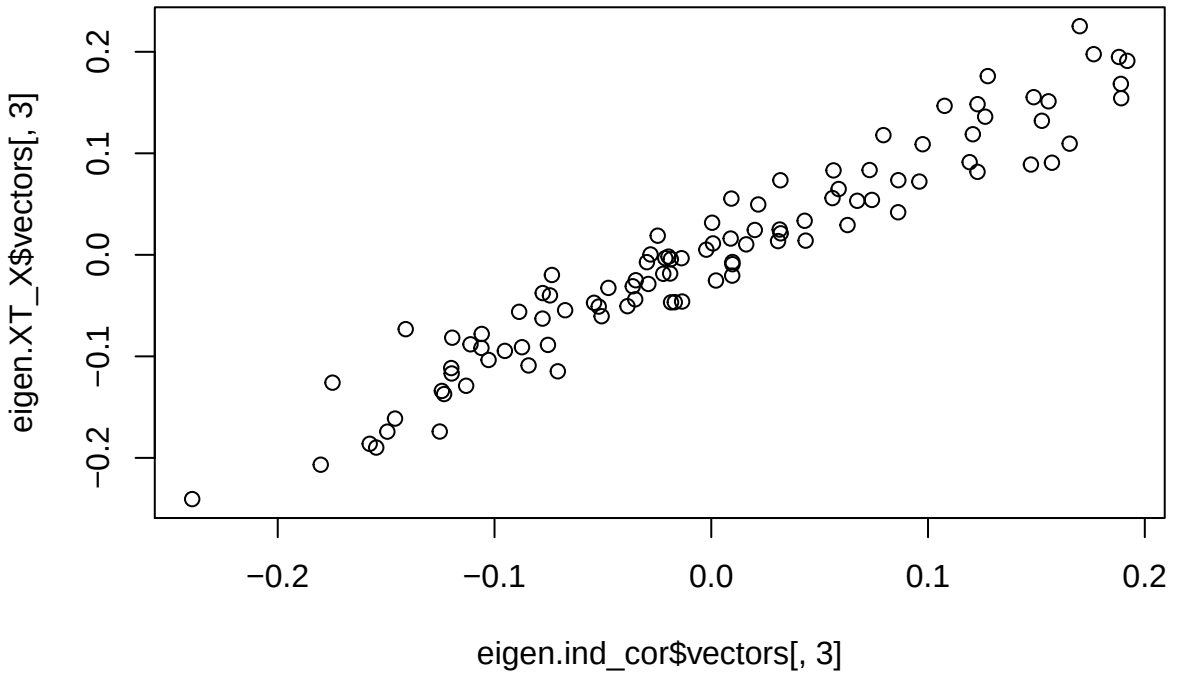

```
plot(eigen.ind_cor$vectors[,1],eigen.XT_X$vectors[,1])
```



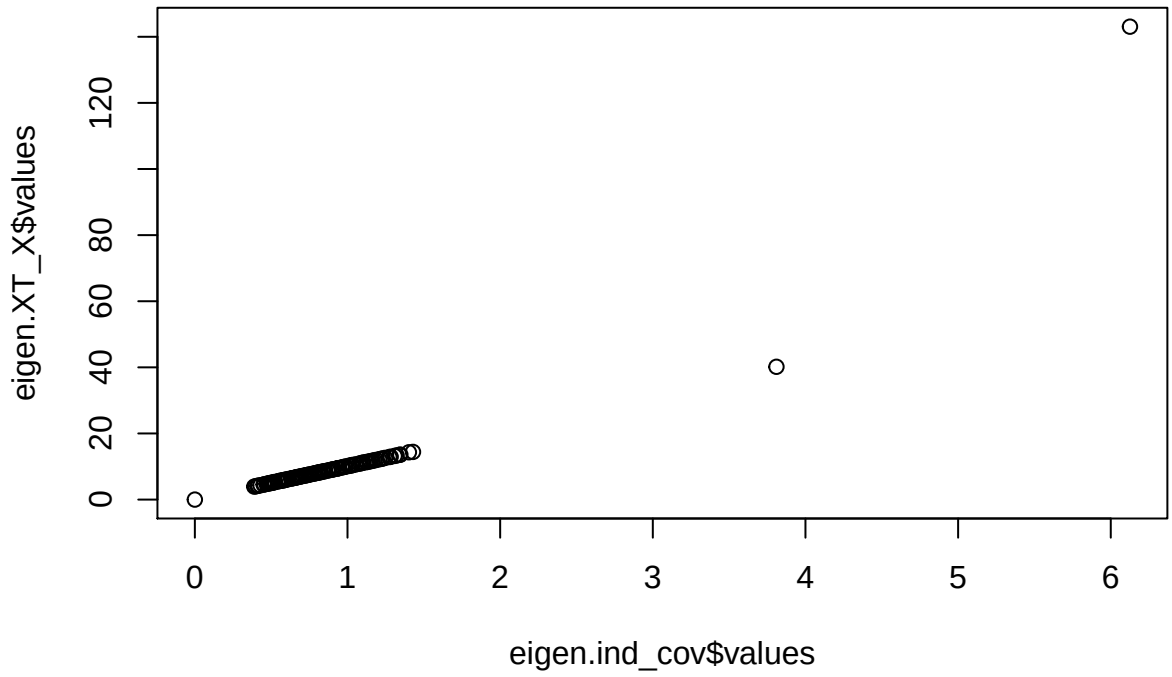

```
plot(eigen.ind_cor$vectors[,2],eigen.XT_X$vectors[,2])
```



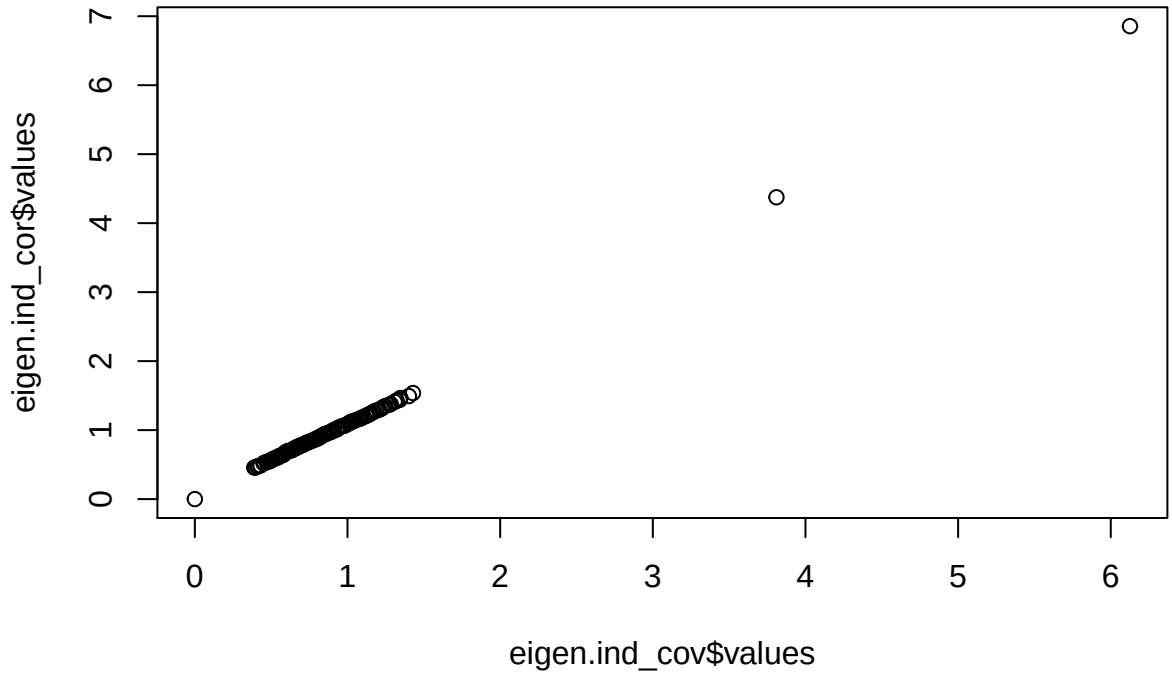

```
plot(eigen.ind_cor$vectors[,3],eigen.XT_X$vectors[,3])
```



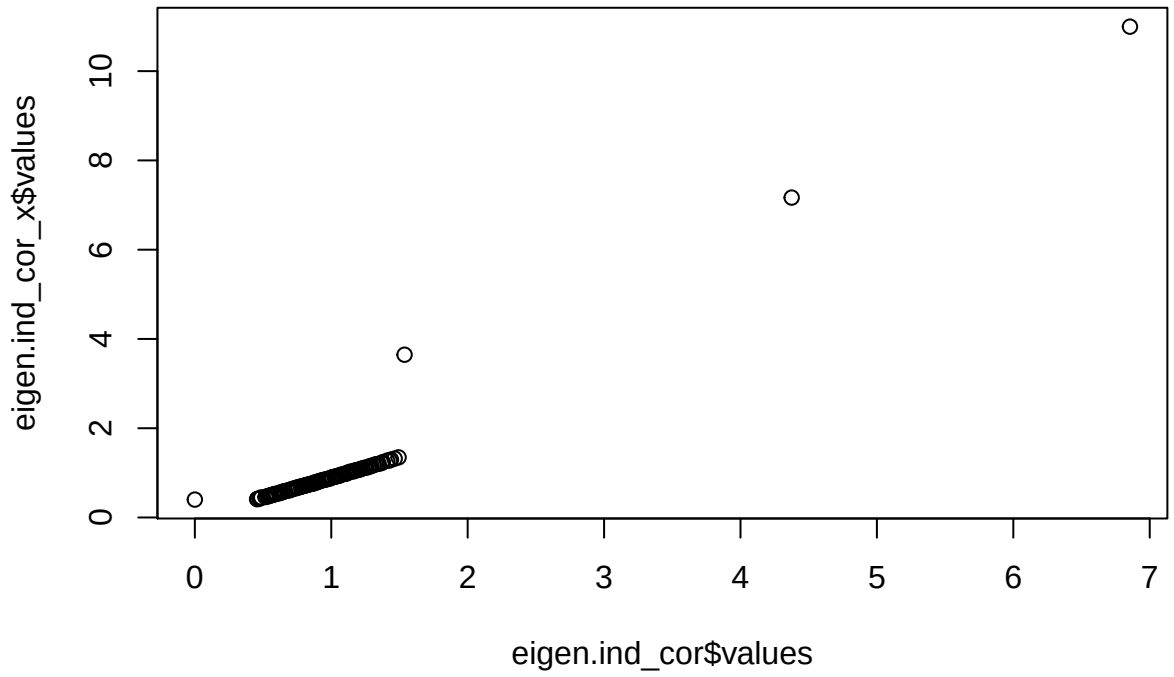

```
plot(eigen.ind_cov$values,eigen.XT_X$values)
```



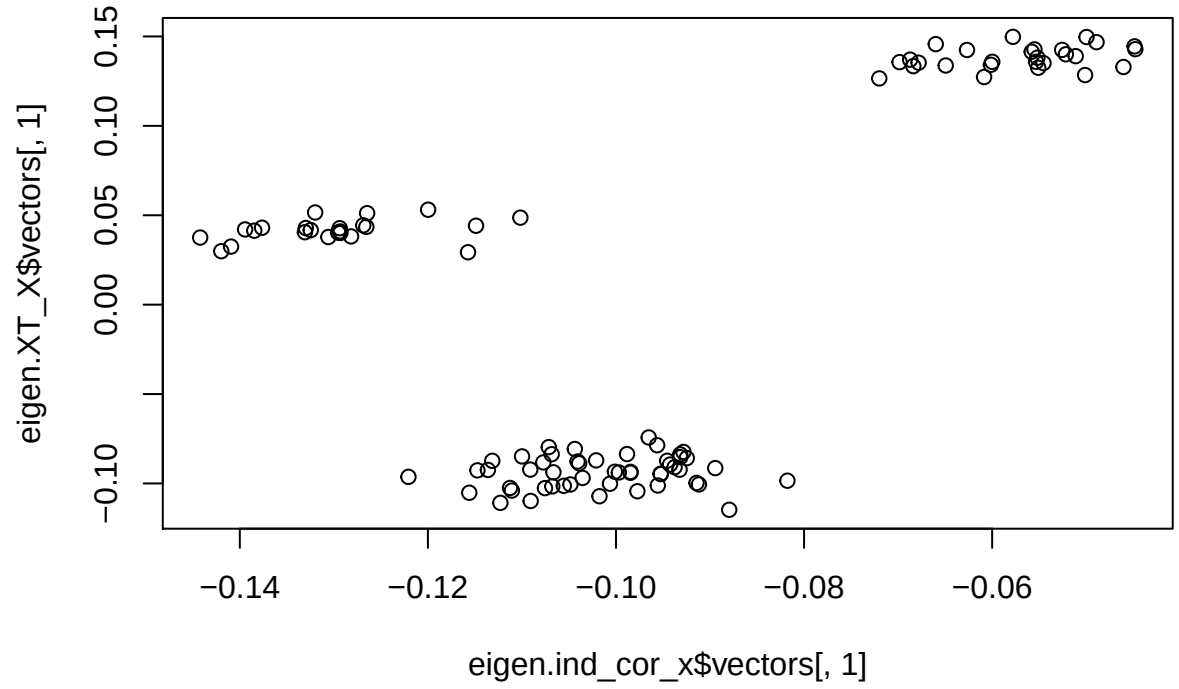

```
plot(eigen.ind_cov$values,eigen.ind_cor$values)
```



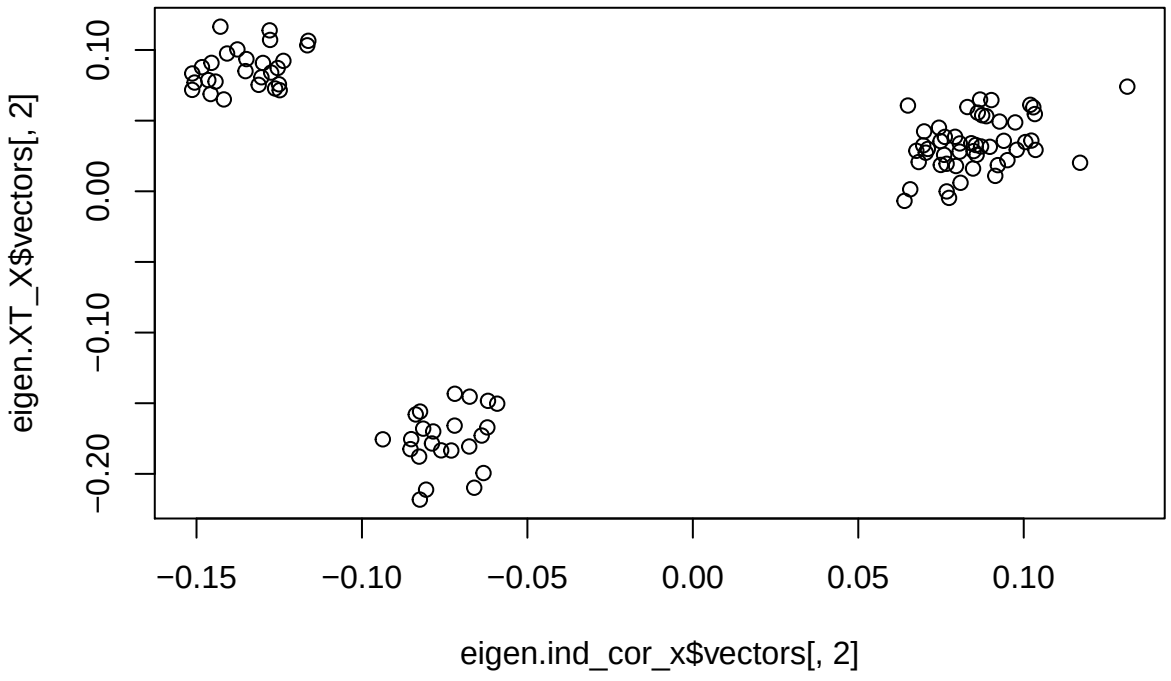

```
plot(eigen.ind_cor$values,eigen.ind_cor_x$values)
```




```
plot(eigen.ind_cor_x$vectors[,1],eigen.XT_X$vectors[,1])
```




```
plot(eigen.ind_cor_x$vectors[,2],eigen.XT_X$vectors[,2])
```

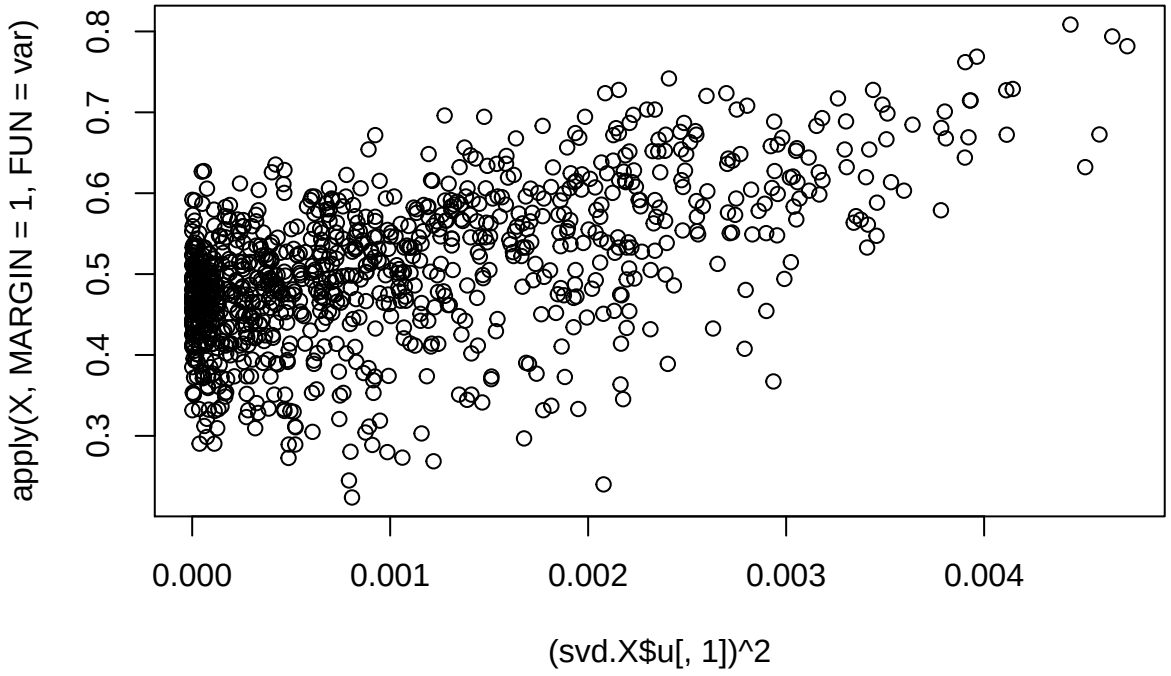


The eigenvalues from svd match those from the eigenvalues from the matrix Xt_X

Now let us see how

Higher varying genes have higher weights assigned to them

```
plot((svd.X$u[,1])^2,apply(X,MARGIN=1,FUN=var))
```

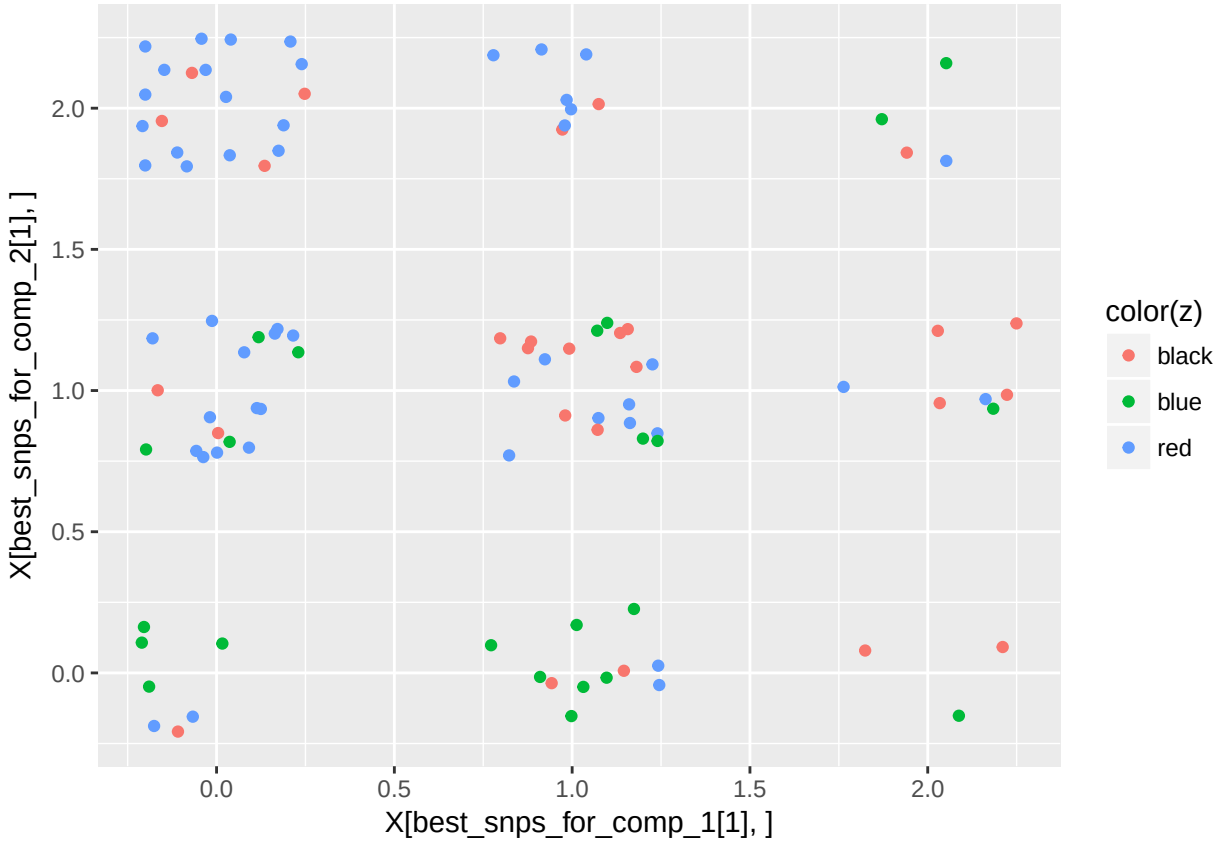



```
cor((svd.X$u[,1])^2,apply(X,MARGIN=1,FUN=var))
```

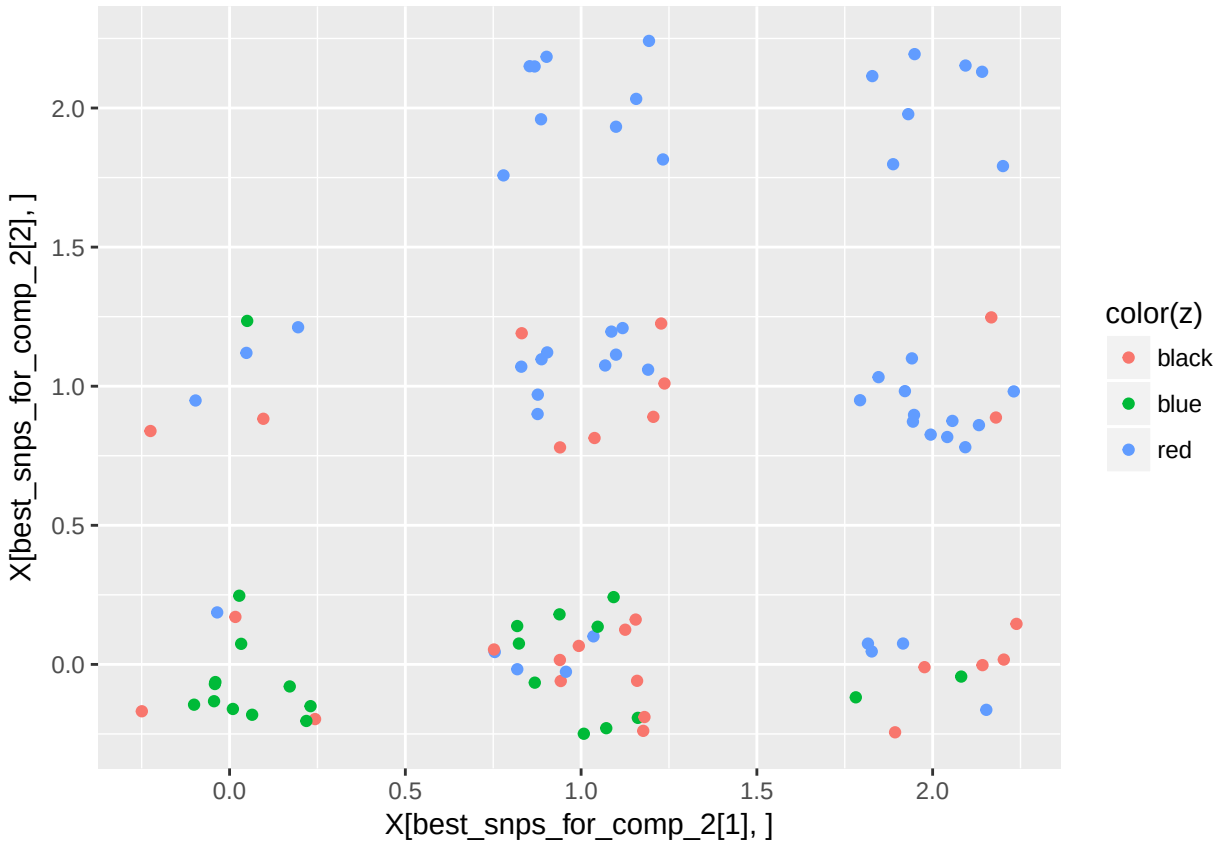
```
## [1] 0.571295
```

```
best_snps_for_comp_1 = sort(svd.X$u[,1],decreasing=T,index.return=T)$ix
best_snps_for_comp_2 = sort(svd.X$u[,2],decreasing=T,index.return=T)$ix
```

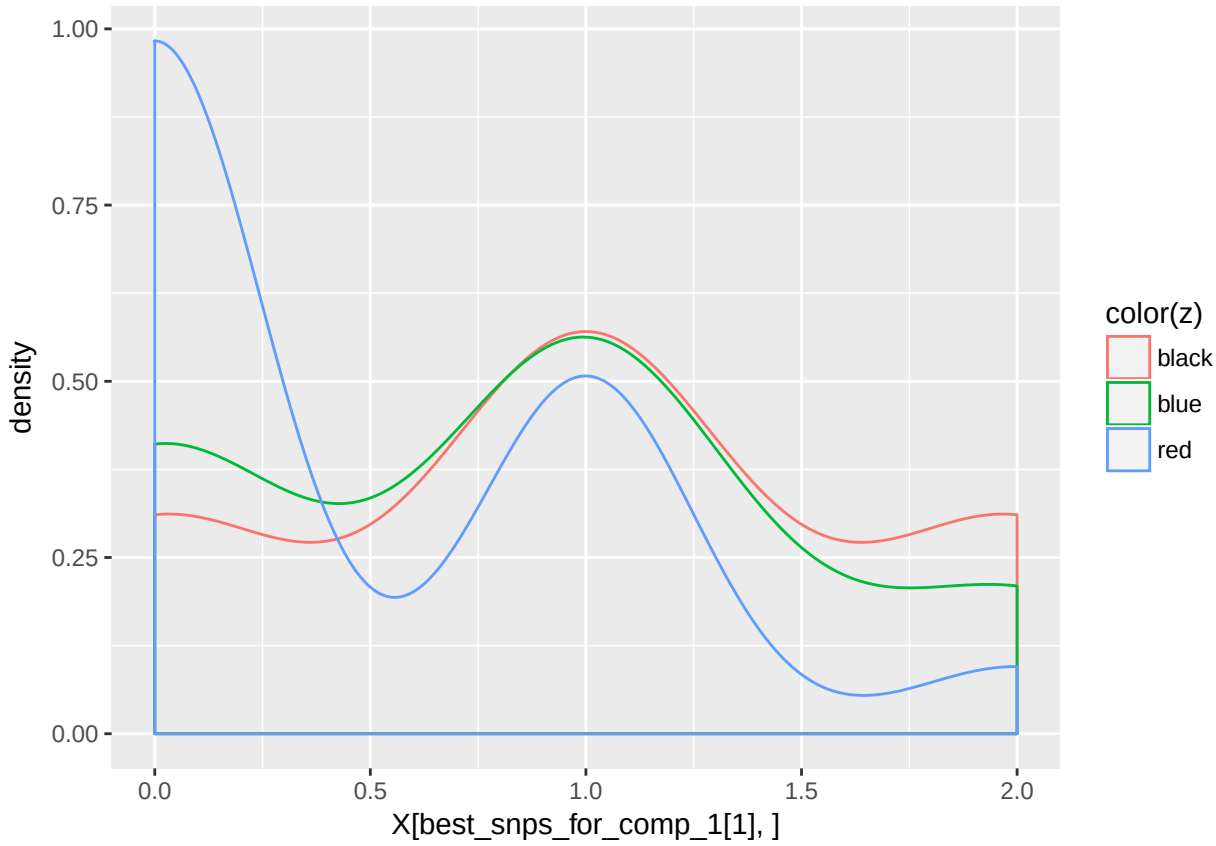
```
genes_ranked_by_var = sort(apply(X,MARGIN=1,FUN=var),index.return=T,decreasing=T)
ggplot(data.frame()) + geom_jitter(aes(x=X[best_snps_for_comp_1[1],],y=X[best_snps_for_comp_2[1],],colo
```



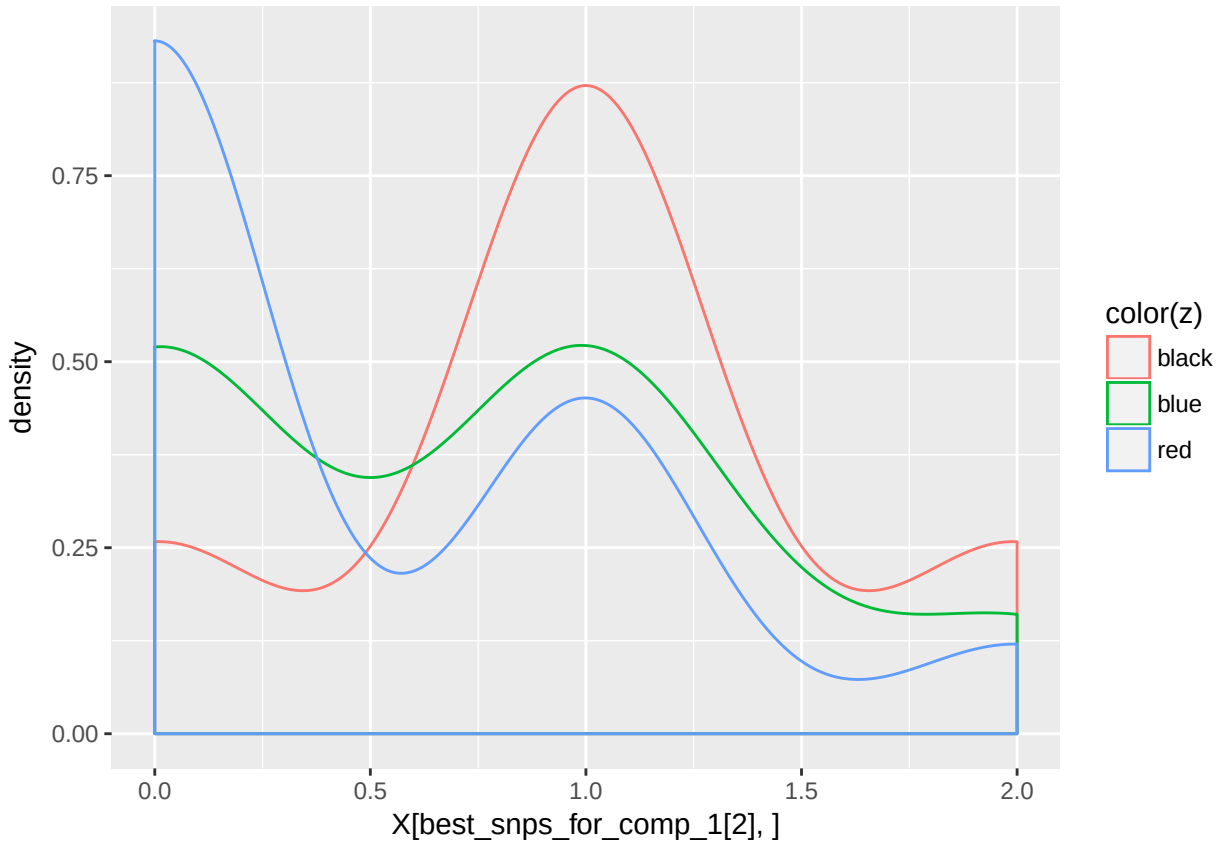

```
ggplot(data.frame()) + geom_jitter(aes(x=X[best_snps_for_comp_2[1],],y=X[best_snps_for_comp_2[2],],colo
```



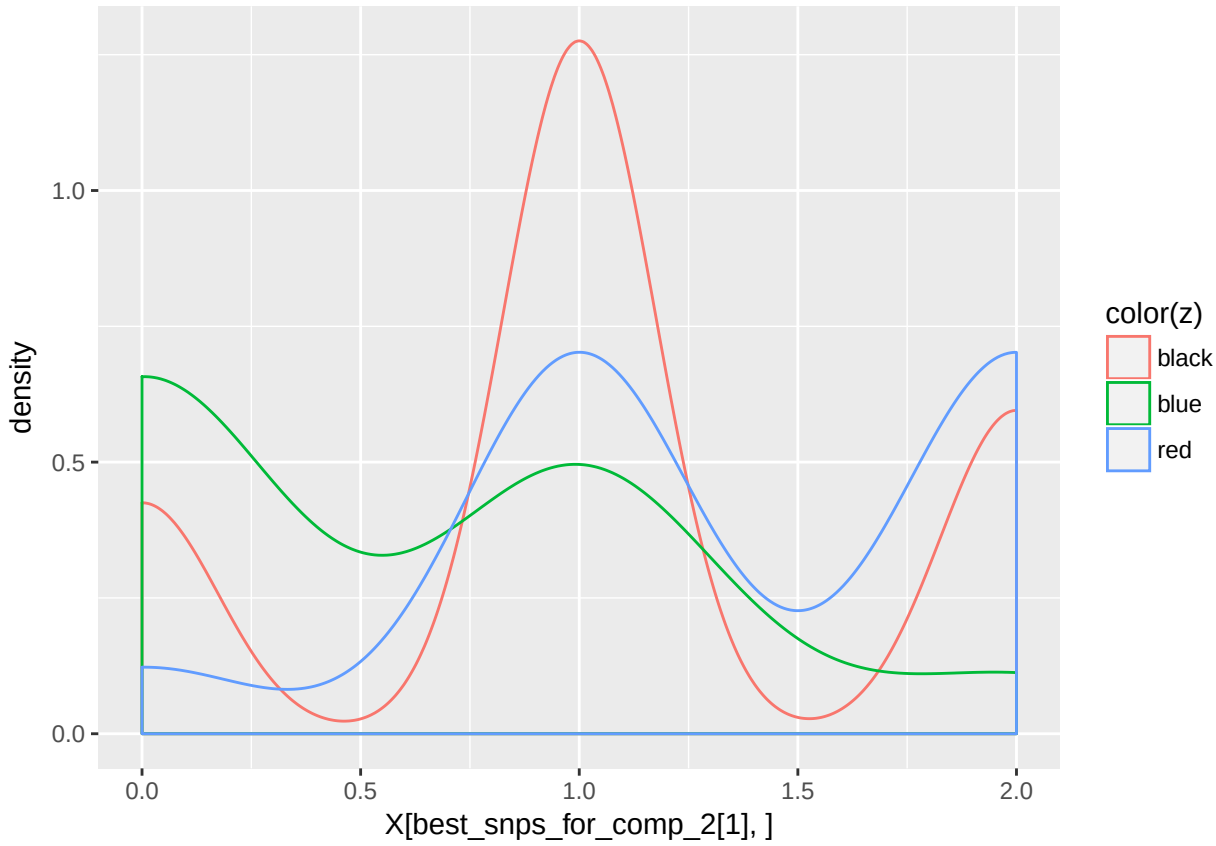

```
ggplot(data.frame()) + geom_density(aes(x=X[best_snps_for_comp_1[1],],color=color(z)))
```



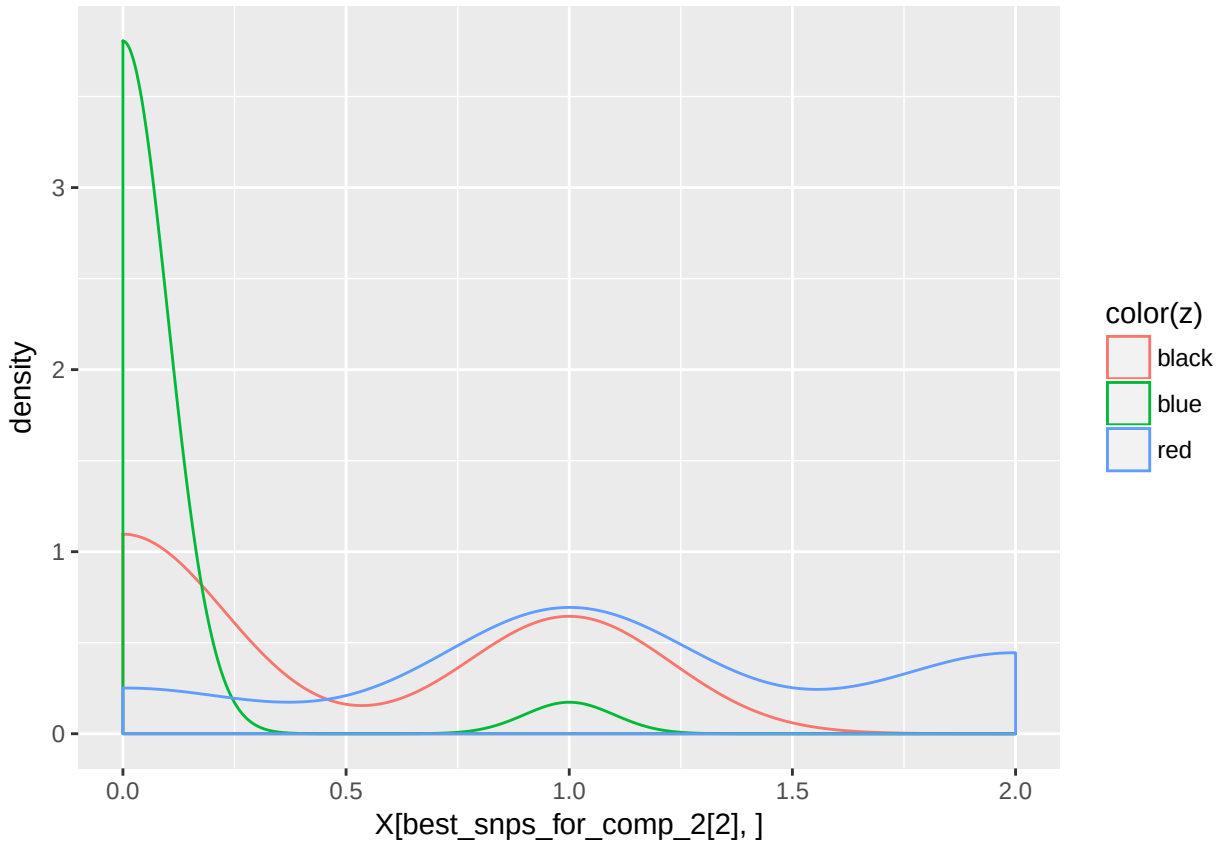

```
ggplot(data.frame()) + geom_density(aes(x=X[best_snps_for_comp_1[2],],color=color(z)))
```




```
ggplot(data.frame()) + geom_density(aes(x=X[best_snps_for_comp_2[1],],color=color(z)))
```




```
ggplot(data.frame()) + geom_density(aes(x=X[best_snps_for_comp_2[2],],color=color(z)))
```



How does centering affect the calculation of eigenvectors?

```
X_row_center = X - apply(X,MARGIN=1,FUN=mean)
X_col_center = X - apply(X,MARGIN=2,FUN=mean)

X_row_center_svd = svd(X_row_center)
X_col_center_svd = svd(X_col_center)
```

$$X = U\Sigma V^T \tag{25}$$

$$U^T X = \Sigma V^T \rightarrow X^T U = V\Sigma^T \rightarrow X^T U = V\Sigma \tag{26}$$

In this case, the columns of U are the eigenvectors of XX^T. $XX^T = U\Sigma^2 U^T$ which is proportional to the covariance matrix between SNPs. In this case, $V\Sigma$ contains the principal component scores of, which are the subject's projected into the space of U.

```
X_XT_row_center = X_row_center %*% t(X_row_center)
X_XT_row_center_svd = svd(X_XT_row_center)

X_XT_row_center_svd$v[1:10,1:10]
```

```
##               [,1]        [,2]          [,3]         [,4]          [,5]
##  [1,] -0.024552462 0.040847923 -2.748811e-02  0.007864760  0.0016212429
##  [2,] -0.040601112 0.015687032 -2.920196e-02 -0.028521382 -0.0265176655
```

```
##  [3,] -0.005762329 0.004481244 -2.611652e-03  0.003942769 -0.0008672528
##  [4,] -0.040694268 0.027956312  7.208302e-03  0.045554497  0.0299430961
##  [5,] -0.028653583 0.023497608 -9.850736e-05 -0.001104559  0.0098100391
##  [6,] -0.038643533 0.017796125  8.221389e-02 -0.022640147 -0.0219128865
##  [7,]  0.023030365 0.009826453  4.716979e-02  0.002208541 -0.0017139945
##  [8,] -0.021632572 0.021135319  4.815065e-02  0.019155827  0.0303154857
##  [9,] -0.030377934 0.033096921 -2.129595e-03  0.005945772  0.0192773559
## [10,] -0.058296023 0.051962697  1.587981e-02 -0.027742547  0.0262128961
##               [,6]         [,7]         [,8]          [,9]        [,10]
##  [1,] -0.028738701  0.023061016 -0.009476320  0.0384973693 -0.052846355
##  [2,]  0.007917629 -0.009814894 -0.048483657 -0.0330896438 -0.025750462
##  [3,]  0.007826702 -0.042887280 -0.046770795 -0.0204862242  0.019233038
##  [4,] -0.044475168 -0.008227474 -0.006329493 -0.0432737661  0.008796124
##  [5,] -0.012788627 -0.007568257 -0.073937946 -0.0002543004 -0.039888152
##  [6,]  0.037781330  0.017624916  0.011183191 -0.0027328649 -0.001694902
##  [7,] -0.044378582  0.032656656 -0.042756811  0.0696068360 -0.063178132
##  [8,]  0.024755310  0.033789363 -0.018933504  0.0388303153 -0.014668363
##  [9,]  0.001223291 -0.002424191 -0.039462872 -0.0280020851 -0.030339777
## [10,]  0.015796799 -0.008823876  0.041484170 -0.0179052807  0.017183029
```

```
X_XT_row_center_svd$u[1:10,1:10]
```

```
##               [,1]        [,2]          [,3]         [,4]          [,5]
##  [1,] -0.024552462 0.040847923 -2.748811e-02  0.007864760  0.0016212429
##  [2,] -0.040601112 0.015687032 -2.920196e-02 -0.028521382 -0.0265176655
##  [3,] -0.005762329 0.004481244 -2.611652e-03  0.003942769 -0.0008672528
##  [4,] -0.040694268 0.027956312  7.208302e-03  0.045554497  0.0299430961
##  [5,] -0.028653583 0.023497608 -9.850736e-05 -0.001104559  0.0098100391
##  [6,] -0.038643533 0.017796125  8.221389e-02 -0.022640147 -0.0219128865
##  [7,]  0.023030365 0.009826453  4.716979e-02  0.002208541 -0.0017139945
##  [8,] -0.021632572 0.021135319  4.815065e-02  0.019155827  0.0303154857
##  [9,] -0.030377934 0.033096921 -2.129595e-03  0.005945772  0.0192773559
## [10,] -0.058296023 0.051962697  1.587981e-02 -0.027742547  0.0262128961
##               [,6]         [,7]         [,8]          [,9]        [,10]
##  [1,] -0.028738701  0.023061016 -0.009476320  0.0384973693 -0.052846355
##  [2,]  0.007917629 -0.009814894 -0.048483657 -0.0330896438 -0.025750462
##  [3,]  0.007826702 -0.042887280 -0.046770795 -0.0204862242  0.019233038
##  [4,] -0.044475168 -0.008227474 -0.006329493 -0.0432737661  0.008796124
##  [5,] -0.012788627 -0.007568257 -0.073937946 -0.0002543004 -0.039888152
##  [6,]  0.037781330  0.017624916  0.011183191 -0.0027328649 -0.001694902
##  [7,] -0.044378582  0.032656656 -0.042756811  0.0696068360 -0.063178132
##  [8,]  0.024755310  0.033789363 -0.018933504  0.0388303153 -0.014668363
##  [9,]  0.001223291 -0.002424191 -0.039462872 -0.0280020851 -0.030339777
## [10,]  0.015796799 -0.008823876  0.041484170 -0.0179052807  0.017183029
```

```
X_XT_row_center_svd$u[1:10,1:10]
```

```
##               [,1]        [,2]          [,3]         [,4]          [,5]
##  [1,] -0.024552462 0.040847923 -2.748811e-02  0.007864760  0.0016212429
##  [2,] -0.040601112 0.015687032 -2.920196e-02 -0.028521382 -0.0265176655
##  [3,] -0.005762329 0.004481244 -2.611652e-03  0.003942769 -0.0008672528
##  [4,] -0.040694268 0.027956312  7.208302e-03  0.045554497  0.0299430961
##  [5,] -0.028653583 0.023497608 -9.850736e-05 -0.001104559  0.0098100391
##  [6,] -0.038643533 0.017796125  8.221389e-02 -0.022640147 -0.0219128865
##  [7,]  0.023030365 0.009826453  4.716979e-02  0.002208541 -0.0017139945
```

```
##  [8,] -0.021632572 0.021135319  4.815065e-02  0.019155827  0.0303154857
##  [9,] -0.030377934 0.033096921 -2.129595e-03  0.005945772  0.0192773559
## [10,] -0.058296023 0.051962697  1.587981e-02 -0.027742547  0.0262128961
##               [,6]         [,7]         [,8]          [,9]        [,10]
##  [1,] -0.028738701  0.023061016 -0.009476320  0.0384973693 -0.052846355
##  [2,]  0.007917629 -0.009814894 -0.048483657 -0.0330896438 -0.025750462
##  [3,]  0.007826702 -0.042887280 -0.046770795 -0.0204862242  0.019233038
##  [4,] -0.044475168 -0.008227474 -0.006329493 -0.0432737661  0.008796124
##  [5,] -0.012788627 -0.007568257 -0.073937946 -0.0002543004 -0.039888152
##  [6,]  0.037781330  0.017624916  0.011183191 -0.0027328649 -0.001694902
##  [7,] -0.044378582  0.032656656 -0.042756811  0.0696068360 -0.063178132
##  [8,]  0.024755310  0.033789363 -0.018933504  0.0388303153 -0.014668363
##  [9,]  0.001223291 -0.002424191 -0.039462872 -0.0280020851 -0.030339777
## [10,]  0.015796799 -0.008823876  0.041484170 -0.0179052807  0.017183029
```

```
X_row_center_svd$u[1:10,1:10]
```

```
##               [,1]         [,2]          [,3]         [,4]          [,5]
##  [1,] -0.024552462 -0.040847923 -2.748811e-02 -0.007864760  0.0016212429
##  [2,] -0.040601112 -0.015687032 -2.920196e-02  0.028521382 -0.0265176655
##  [3,] -0.005762329 -0.004481244 -2.611652e-03 -0.003942769 -0.0008672528
##  [4,] -0.040694268 -0.027956312  7.208302e-03 -0.045554497  0.0299430961
##  [5,] -0.028653583 -0.023497608 -9.850736e-05  0.001104559  0.0098100391
##  [6,] -0.038643533 -0.017796125  8.221389e-02  0.022640147 -0.0219128865
##  [7,]  0.023030365 -0.009826453  4.716979e-02 -0.002208541 -0.0017139945
##  [8,] -0.021632572 -0.021135319  4.815065e-02 -0.019155827  0.0303154857
##  [9,] -0.030377934 -0.033096921 -2.129595e-03 -0.005945772  0.0192773559
## [10,] -0.058296023 -0.051962697  1.587981e-02  0.027742547  0.0262128961
##               [,6]         [,7]         [,8]          [,9]        [,10]
##  [1,] -0.028738701  0.023061016 -0.009476320 -0.0384973693 -0.052846355
##  [2,]  0.007917629 -0.009814894 -0.048483657  0.0330896438 -0.025750462
##  [3,]  0.007826702 -0.042887280 -0.046770795  0.0204862242  0.019233038
##  [4,] -0.044475168 -0.008227474 -0.006329493  0.0432737661  0.008796124
##  [5,] -0.012788627 -0.007568257 -0.073937946  0.0002543004 -0.039888152
##  [6,]  0.037781330  0.017624916  0.011183191  0.0027328649 -0.001694902
##  [7,] -0.044378582  0.032656656 -0.042756811 -0.0696068360 -0.063178132
##  [8,]  0.024755310  0.033789363 -0.018933504 -0.0388303153 -0.014668363
##  [9,]  0.001223291 -0.002424191 -0.039462872  0.0280020851 -0.030339777
## [10,]  0.015796799 -0.008823876  0.041484170  0.0179052807  0.017183029
```

From the above, we can see that the eigenvectors match.

Now we can see that the projection and the PC scores match.

```
X_row_center_proj = t(X_row_center) %*% X_row_center_svd$u
X_row_center_proj_2 = X_row_center_svd$v %*% diag(X_row_center_svd$d)

X_row_center_proj[1:10,1:10]
```

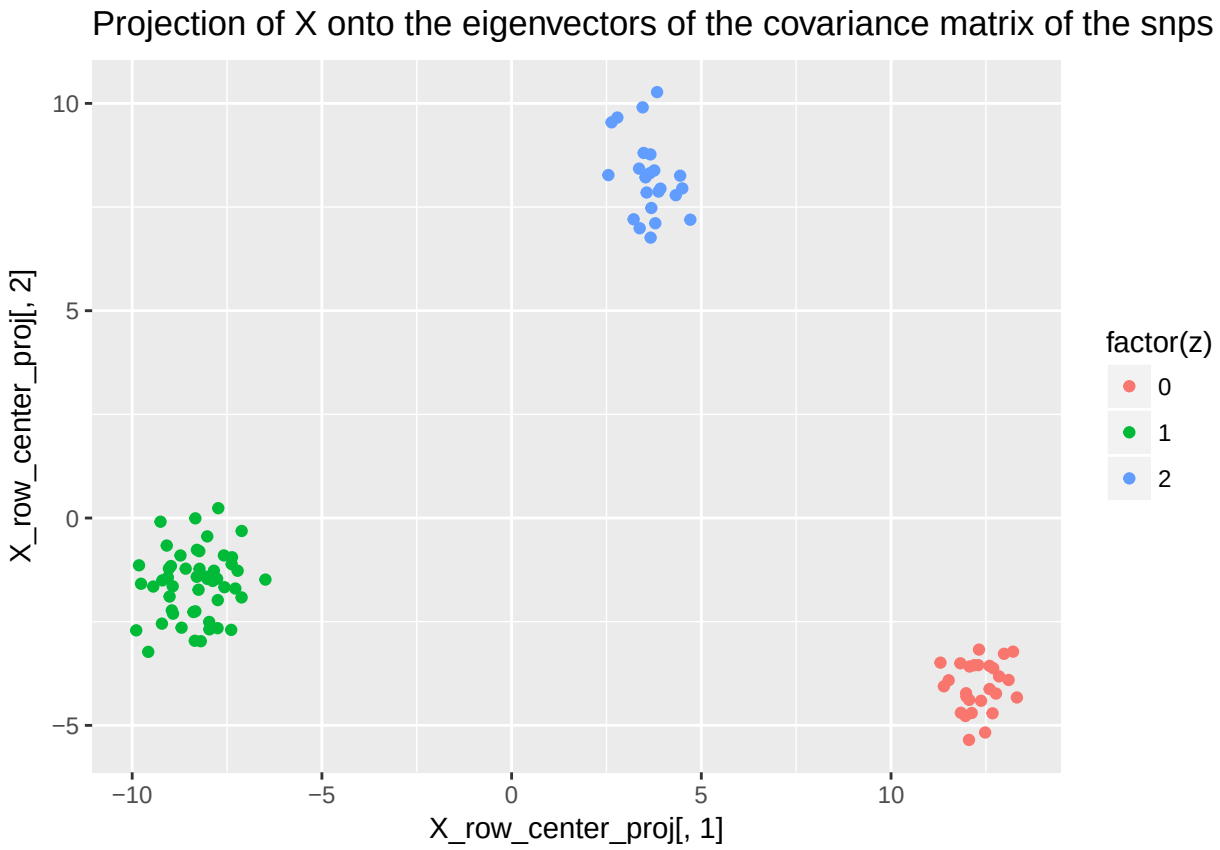
```
##            [,1]         [,2]       [,3]        [,4]       [,5]       [,6]
##  [1,] -7.846217 -1.270036647 -0.4023730  4.05329074 -2.1594595  0.2185521
##  [2,] -9.253630 -0.088642220  4.8822750 -2.10998704 -2.0246352 -0.7273914
##  [3,] -8.335302 -0.007499656  6.1156208  3.06484735  0.7871385 -0.8222928
##  [4,]  4.710031  7.195297398  2.3213200 -3.87678466  1.4131607 -1.8308907
##  [5,] -8.251529 -1.729747801 -4.7065584  5.58701338 -7.0808167  3.7507543
##  [6,] -9.032248 -1.218483708 -3.0761510 -0.18458451 -0.4602096  0.1856219
```

```
##  [7,] 13.097973 -3.905154061 -0.1008850 -1.16862075  0.6886747 -4.4421432
##  [8,]  3.480817  8.805329228 -0.4106961  1.12238133 -0.1703554 -0.7618663
##  [9,] -7.281794 -1.700229663  5.5862597  1.78920715  2.0946701  2.0169328
## [10,]  3.558591  7.850760073 -5.9201669  0.09357405  4.0581932  1.5308145
##             [,7]        [,8]       [,9]      [,10]
##  [1,] -1.4126597  2.93974778 -1.3584131  0.1740651
##  [2,] -3.7785689  1.18108468 -3.7195850  5.5125395
##  [3,]  2.3193656  4.73780878  1.4598392  4.0110079
##  [4,] -3.5433424  1.84129780  0.2175583 -0.2514225
##  [5,]  4.7370649 -0.69525045  0.7074928 -2.1442346
##  [6,] -0.4297280 -2.49892946 -5.0769386  1.4027130
##  [7,] -0.6539535 -0.06508031  0.1404776  2.3876104
##  [8,] -2.5610147 -3.82898888 -0.5070615  1.8379780
##  [9,]  5.1643308 -1.71055980 -0.3605189 -0.5290079
## [10,] -0.9438742  1.18362640 -1.3363702  1.9143755
```

```
X_row_center_proj_2[1:10,1:10]
```

```
##            [,1]         [,2]       [,3]        [,4]       [,5]       [,6]
##  [1,] -7.846217 -1.270036647 -0.4023730  4.05329074 -2.1594595  0.2185521
##  [2,] -9.253630 -0.088642220  4.8822750 -2.10998704 -2.0246352 -0.7273914
##  [3,] -8.335302 -0.007499656  6.1156208  3.06484735  0.7871385 -0.8222928
##  [4,]  4.710031  7.195297398  2.3213200 -3.87678466  1.4131607 -1.8308907
##  [5,] -8.251529 -1.729747801 -4.7065584  5.58701338 -7.0808167  3.7507543
##  [6,] -9.032248 -1.218483708 -3.0761510 -0.18458451 -0.4602096  0.1856219
##  [7,] 13.097973 -3.905154061 -0.1008850 -1.16862075  0.6886747 -4.4421432
##  [8,]  3.480817  8.805329228 -0.4106961  1.12238133 -0.1703554 -0.7618663
##  [9,] -7.281794 -1.700229663  5.5862597  1.78920715  2.0946701  2.0169328
## [10,]  3.558591  7.850760073 -5.9201669  0.09357405  4.0581932  1.5308145
##             [,7]        [,8]       [,9]      [,10]
##  [1,] -1.4126597  2.93974778 -1.3584131  0.1740651
##  [2,] -3.7785689  1.18108468 -3.7195850  5.5125395
##  [3,]  2.3193656  4.73780878  1.4598392  4.0110079
##  [4,] -3.5433424  1.84129780  0.2175583 -0.2514225
##  [5,]  4.7370649 -0.69525045  0.7074928 -2.1442346
##  [6,] -0.4297280 -2.49892946 -5.0769386  1.4027130
##  [7,] -0.6539535 -0.06508031  0.1404776  2.3876104
##  [8,] -2.5610147 -3.82898888 -0.5070615  1.8379780
##  [9,]  5.1643308 -1.71055980 -0.3605189 -0.5290079
## [10,] -0.9438742  1.18362640 -1.3363702  1.9143755
```

```
ggplot(data.frame()) + geom_point(aes(x=X_row_center_proj[,1],X_row_center_proj[,2],color=factor(z))) +
```



From this we can see that the columns of U specify the weights of each SNP for each eigenvector.

$$X = U\Sigma V^T \tag{27}$$

$$XV = U\Sigma \tag{28}$$

In this case, the columns of V are the eigenvectors of X^TX. In this case, we are projecting the SNPs onto the eigenvectors of the covariance matrix of the individuals. So $U\Sigma$ contain the coordinates of each gene in the space of the principal components of V.

```
X_col_center_proj = X_col_center %*% X_col_center_svd$v
X_col_center_proj_2 = X_col_center_svd$u %*% diag(X_col_center_svd$d)

X_col_center_proj[1:10,1:10]
```

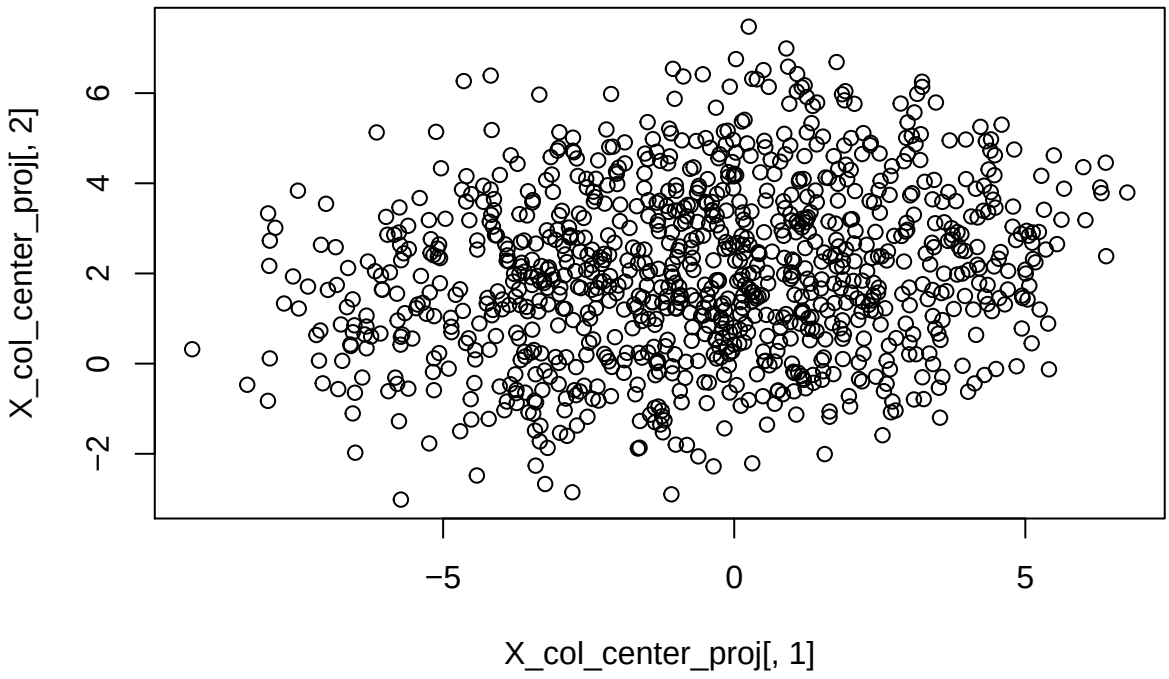
```
##            [,1]      [,2]       [,3]        [,4]        [,5]        [,6]
##  [1,] -3.343749  1.394115  2.2813704 -0.70262647 -0.20951295  0.04467965
##  [2,]  1.111623  4.041672  0.4357096 -0.74318636  0.78022178 -0.68915440
##  [3,]  2.591798  1.227678 -0.1925397 -0.10770676 -0.06659558 -0.02973618
##  [4,]  2.000544  4.295445  0.8909273  0.13842334 -1.19961161  0.77040164
##  [5,] -1.536053  2.237363  1.2439872  0.04092401 -0.01367543  0.26447175
##  [6,] -2.814691  2.815006  1.0809892  2.24408191  0.46096776 -0.54303073
##  [7,] -2.780757 -2.854006  0.9296179  1.27773187 -0.13941224 -0.03859929
##  [8,]  3.733970  3.004380  0.3668232  1.22520202 -0.52753260  0.78628579
##  [9,]  3.680396  3.806311  0.8735526 -0.11024624 -0.10587512  0.48506806
## [10,] -1.084963  5.108967  2.3613447  0.46939971  0.69333649  0.68385364
```

```
##              [,7]        [,8]       [,9]      [,10]
##  [1,] -0.75259413  0.61746945 -0.2603271 -0.8900692
##  [2,]  0.21797927 -0.26384837 -1.2254035  0.8630743
##  [3,]  0.20823429 -1.11709791 -1.1731212  0.4670236
##  [4,] -1.16366435 -0.18356923 -0.1632185  1.1332843
##  [5,] -0.36208723 -0.17373241 -1.8950357  0.1240210
##  [6,]  0.95340949  0.44742552  0.2753461  0.1172292
##  [7,] -1.16238980  0.87683837 -1.1161977 -1.6456035
##  [8,]  0.64462361  0.85715158 -0.4858172 -0.9812181
##  [9,]  0.05369428 -0.07534497 -0.9908566  0.6960802
## [10,]  0.38907536 -0.22808566  1.0552735  0.4469870
```

```
X_col_center_proj_2[1:10,1:10]
```

```
##            [,1]      [,2]       [,3]        [,4]        [,5]        [,6]
##  [1,] -3.343749  1.394115  2.2813704 -0.70262647 -0.20951295  0.04467965
##  [2,]  1.111623  4.041672  0.4357096 -0.74318636  0.78022178 -0.68915440
##  [3,]  2.591798  1.227678 -0.1925397 -0.10770676 -0.06659558 -0.02973618
##  [4,]  2.000544  4.295445  0.8909273  0.13842334 -1.19961161  0.77040164
##  [5,] -1.536053  2.237363  1.2439872  0.04092401 -0.01367543  0.26447175
##  [6,] -2.814691  2.815006  1.0809892  2.24408191  0.46096776 -0.54303073
##  [7,] -2.780757 -2.854006  0.9296179  1.27773187 -0.13941224 -0.03859929
##  [8,]  3.733970  3.004380  0.3668232  1.22520202 -0.52753260  0.78628579
##  [9,]  3.680396  3.806311  0.8735526 -0.11024624 -0.10587512  0.48506806
## [10,] -1.084963  5.108967  2.3613447  0.46939971  0.69333649  0.68385364
##              [,7]        [,8]       [,9]      [,10]
##  [1,] -0.75259413  0.61746945 -0.2603271 -0.8900692
##  [2,]  0.21797927 -0.26384837 -1.2254035  0.8630743
##  [3,]  0.20823429 -1.11709791 -1.1731212  0.4670236
##  [4,] -1.16366435 -0.18356923 -0.1632185  1.1332843
##  [5,] -0.36208723 -0.17373241 -1.8950357  0.1240210
##  [6,]  0.95340949  0.44742552  0.2753461  0.1172292
##  [7,] -1.16238980  0.87683837 -1.1161977 -1.6456035
##  [8,]  0.64462361  0.85715158 -0.4858172 -0.9812181
##  [9,]  0.05369428 -0.07534497 -0.9908566  0.6960802
## [10,]  0.38907536 -0.22808566  1.0552735  0.4469870
```

```
plot(X_col_center_proj[,1],X_col_center_proj[,2])
```



Conclusion on how to do PCA when number of samples are less than the number of features.

Let X be a matrix of m rows of features and n columns of samples. Center and/or standardize each colum, then compute the SVD of $X = U\Sigma V^T$. U are the eigenvectors of XX^T (proportional to the m by m covariance matrix of features). Each column of U is an eigenvector of the covariance matrix. $U^T X = \Sigma V^T \rightarrow X^T U = V\Sigma$. $U_{i,j}$ is the weight of feature i in eigenvector j. Column j of $X^T U$ or $V\Sigma$ are the scores of all individuals along the j principal component.

So ultimately it comes down to are we interested in seeing the relationship between features or between individuals.

How do different covariance matrices affect the calculation of different Mahalanobis distances?

```
cov.mat = matrix(
  c(1,   -0.5,
    -0.5, 1),ncol=2)

eigen.decomp = eigen(cov.mat)

vals = c(-2,1)
sqrt(vals %*% solve(cov.mat) %*% vals)
```

```
##      [,1]
## [1,]    2
```

```
#distance from origin in white transformed space
vals.proj.scaled = vals %*% eigen.decomp$vectors / sqrt(eigen.decomp$values)
sqrt(sum(vals.proj.scaled^2))
```

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

```
angle(c(1,0),t(vals.proj.scaled))
```

```
##      [,1]
## [1,]   30
```

```
sum(vals.proj.scaled)
```

```
## [1] 2.732051
```

```
vals = c(2,2)
sqrt(vals %*% solve(cov.mat) %*% vals)
```

```
##      [,1]
## [1,]    4
```

```
vals = c(-1,3)
sqrt(vals %*% solve(cov.mat) %*% vals)
```

```
##         [,1]
## [1,] 3.05505
```

```
#distance from origin in white transformed space
vals.proj.scaled = vals %*% eigen.decomp$vectors / sqrt(eigen.decomp$values)
sqrt(sum(vals.proj.scaled^2))
```

```
## [1] 3.05505
```

```
cov.mat = matrix(
  c(4,   4*0.25,
    4*0.25, 4),ncol=2)

vals = c(4,-4)
sqrt(vals %*% solve(cov.mat) %*% vals)
```

```
##          [,1]
## [1,] 3.265986
```

```
cov.mat = matrix(
  c(1,   0.25,
    0.25, 1),ncol=2)

vals = c(2,-2)
sqrt(vals %*% solve(cov.mat) %*% vals)
```

```
##          [,1]
## [1,] 3.265986
```

```
cov.mat = matrix(
  c(1,   -0.25,
    -0.25, 1),ncol=2)

vals = c(2,-2)
sqrt(vals %*% solve(cov.mat) %*% vals)
```

```
##          [,1]
## [1,] 2.529822
```

```
cov.mat = matrix(
  c(1,   0,
    0, 1),ncol=2)

vals = c(3,3)
sqrt(vals %*% solve(cov.mat) %*% vals)
```

```
##          [,1]
## [1,] 4.242641
```

```
cov.mat = matrix(
  c(1,   0.9,
    0.9, 1),ncol=2)

vals = c(3,3)
sqrt(vals %*% solve(cov.mat) %*% vals)
```

```
##          [,1]
## [1,] 3.077935
```

```
cov.mat = matrix(
  c(1,   0.5, -0.75,
    0.5, 1,    0.5,
   -0.75, 0.5,  1),ncol=3)

vals = c(-3,0,3)
sqrt(vals %*% solve(cov.mat) %*% vals)
```

```
##          [,1]
## [1,] 3.207135
```