Statistical Learning for Data Science

MSDS534

Lecture 05: Kernel Methods

Department of Statistics & Biostatistics
Rutgers University

Acknowledgement

- Some of the figures in this presentation are taken from *An Introduction to Statistical Learning, with applications in R* (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.
- Some of the figures in this presentation are taken from *Elements of Statistical Learning* (Springer, 2009) with permission from the authors: T. Hastie, R. Tibshirani and J. Friedman.
- Some of the figures in this presentation are taken from *Pattern Recognition and Machine Learning* (Springer, 2006) with permission from the author: Christopher M. Bishop.
- Some of the Lab codes in this presentation are taken from *An Introduction to Statistical Learning, with applications in R* (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

Reading Assignments

- ESL: § 14.5.1, § 14.5.3, § 14.5.4
- ISL: § 10.2, § 10.4, § 10.5
- PRML: § 12.3

Outline

Principal Component Analysis

- Suppose we have N training points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^p$. Denote by $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)'$ the $N \times p$ data matrix.
- Find a direction along which the data has the largest variation.

$$\max_{\|\boldsymbol{v}\|=1} \{\text{sample variance of } \boldsymbol{X}\boldsymbol{v}\}.$$

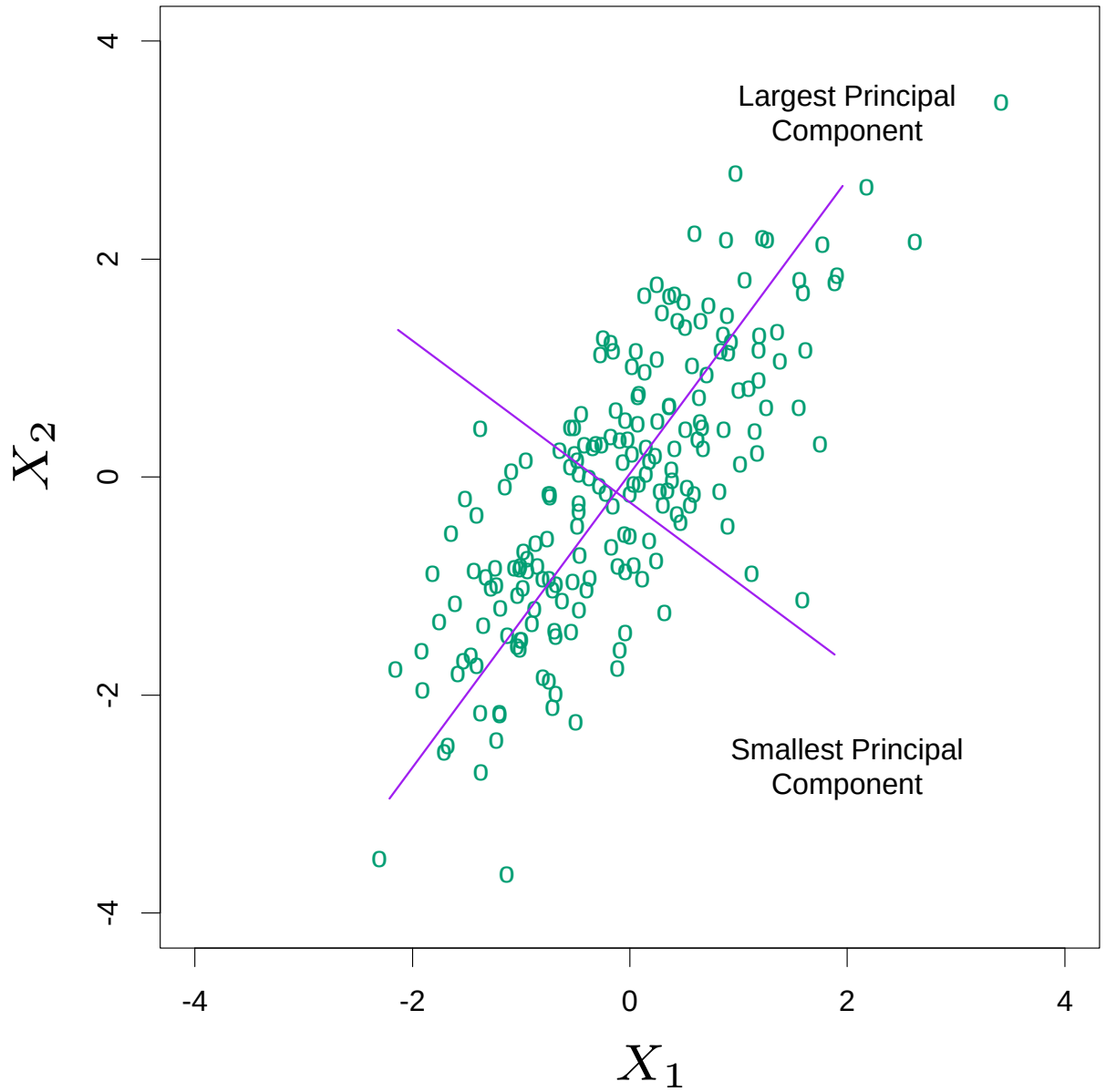
- Find a line that is "closest" to the data.
  - A line in $\mathbb{R}^p$ can be represented as $f(\eta) = \boldsymbol{\mu} + \eta\boldsymbol{v}$, where $\boldsymbol{\mu}$ and $\boldsymbol{v}$ are p-dimensional parameter vectors.
  - For a point $\boldsymbol{x}_i$, we find η_i such that

$$\eta_i = \arg\min_{\eta} \|\boldsymbol{x}_i - (\boldsymbol{\mu} + \eta\boldsymbol{v})\|^2.$$

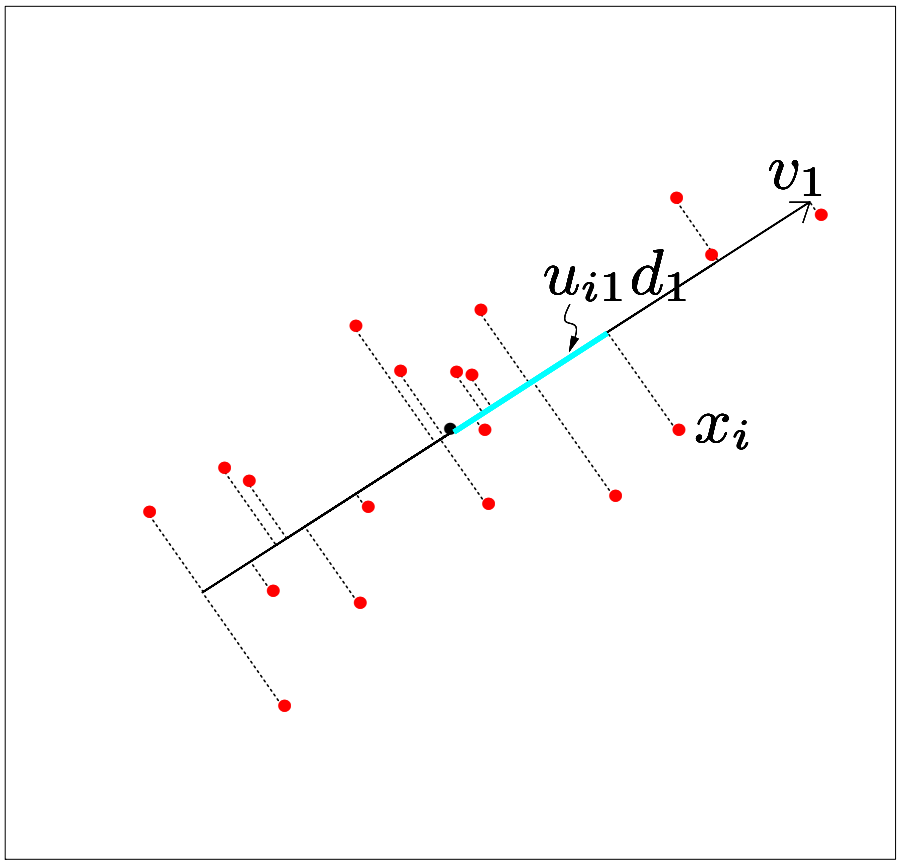
  - Next, fine the "closest" line:

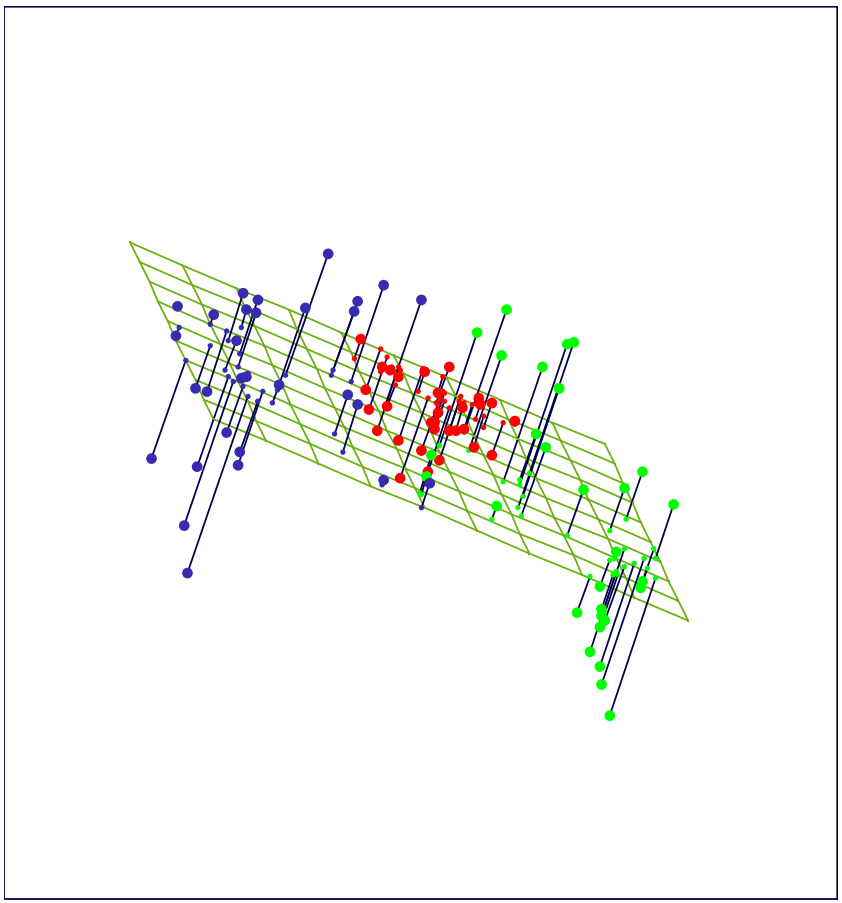
$$\min_{\boldsymbol{\mu}, \|\boldsymbol{v}\|=1} \sum_{i=1}^{N} \|\boldsymbol{x}_i - (\boldsymbol{\mu} + \eta_i\boldsymbol{v})\|^2.$$

Principal Component Direction



Best Linear Approximation





Principal Component Analysis

Orthogonal Matrix

Let $\boldsymbol{A}$ be a $n \times m$ matrix with $n \geq m$. Denote its m columns by $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m$. We say $\boldsymbol{A}$ is orthogonal if its columns are orthonormal, i.e.

$$\boldsymbol{a}_i' \boldsymbol{a}_j = \begin{cases} 1 & \text{if } i = j; \\ 0 & \text{if } i \neq j. \end{cases}$$

- Having find $\hat{\boldsymbol{v}}_1$, find next one, subject to the constraint that it is orthogonal to $\hat{\boldsymbol{v}}_1$.
- Find the best low dimensional linear approximations to the data. Consider the rank-q linear model for representing the p-dimensional data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$.

$$f(\boldsymbol{\eta}) = \boldsymbol{\mu} + \boldsymbol{V}_q \boldsymbol{\eta},$$

where $\boldsymbol{\mu} \in \mathbb{R}^p$ is a location vector, $\boldsymbol{V}_q$ is a $p \times q$ orthogonal matrix, and $\boldsymbol{\eta} \in \mathbb{R}^q$ is a vector of parameters. Fitting such a model to the data by least squares amounts to minimizing the reconstruction error

$$\min_{\boldsymbol{\mu}, \{\boldsymbol{\eta}_i\}, \boldsymbol{V}_q} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{\mu} - \boldsymbol{V}_q \boldsymbol{\eta}_i\|^2.$$

PCA: Preprocessing

1. Compute $\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i$, and then subtract the sample mean from each observation

$$\boldsymbol{x}_i := \boldsymbol{x}_i - \bar{\boldsymbol{x}}.$$

2. Optional. Preferred when features are on different scales. Normalize each feature.

$$\hat{s}_j = \sqrt{\sum_{i=1}^{N} \boldsymbol{x}_{ij}^2} \quad \text{then} \quad \boldsymbol{x}_{ij} := \frac{\boldsymbol{x}_{ij}}{s_j} \quad \text{for all } 1 \le i \le N,\ 1 \le j \le p.$$

From now on we always assume Step 1 has been done. Problems become

- Direction of maximum sample variance

$$\max_{\|\boldsymbol{v}_1\|=1} \boldsymbol{v}_1' \boldsymbol{X}' \boldsymbol{X} \boldsymbol{v}_1.$$

- Best linear approximation

$$\min_{\{\eta_i\},\boldsymbol{v}_1} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{v}_1 \eta_i\|^2.$$

Principal Directions & Best Linear Approximation

- Principal directions. Having found $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_j$, find $\boldsymbol{v}_j$ by

$$\max_{\boldsymbol{v}} \quad \boldsymbol{v}' \boldsymbol{X}' \boldsymbol{X} \boldsymbol{v}$$
$$\text{subject to:} \quad \boldsymbol{v} \perp \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{j-1}.$$

- Best linear approximation. Find the optimal orthogonal matrix $\boldsymbol{V}_q \in \mathbb{R}^{p \times q}$,

$$\min_{\{\boldsymbol{\eta}_i\}, \boldsymbol{V}_q} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{V}_q \boldsymbol{\eta}_i\|^2.$$

  - Fix $\boldsymbol{V}_q$, we must have
$$\hat{\boldsymbol{\eta}}_i = \boldsymbol{V}_q' \boldsymbol{x}_i.$$
  - The problem is reduced to
$$\min_{\boldsymbol{V}_q} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{V}_q \boldsymbol{V}_q' \boldsymbol{x}_i\|^2.$$

Singular Value Decomposition

The singular value decomposition (SVD) of the $N \times p$ (assume $N \geq p$) matrix $\boldsymbol{X}$ has the form

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}'.$$

- $\boldsymbol{U}$ and $\boldsymbol{V}$ are $N \times p$ and $p \times p$ orthogonal matrices.
- The columns of $\boldsymbol{V}$, denoted by $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$, span the row space of $\boldsymbol{X}$.
- The columns of $\boldsymbol{U}$, $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p$, span the column space of $\boldsymbol{X}$.
- $\boldsymbol{D}$ is a $p \times p$ diagonal matrix, with diagonal entries $d_1 \geq d_2 \geq \cdots \geq d_p \geq 0$, which are called singular values of $\boldsymbol{X}$.
- $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$ are called right singular vectors. They are also eigenvectors of the matrix $\boldsymbol{X}'\boldsymbol{X}$, corresponding to the eigenvalues $d_1^2 \geq d_2^2 \geq \cdots \geq d_p^2$.
- $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p$ are called left singular vectors. They are also eigenvectors of the matrix $\boldsymbol{X}\boldsymbol{X}'$, corresponding to the eigenvalues $d_1^2 \geq d_2^2 \geq \cdots \geq d_p^2$. The rest eigenvalues of $\boldsymbol{X}\boldsymbol{X}'$ are all zero.
- It can be rewritten as $\boldsymbol{X} = d_1\boldsymbol{u}_1\boldsymbol{v}_1' + \ldots + d_p\boldsymbol{u}_p\boldsymbol{v}_p'$.
- Such a decomposition is unique (up to a sign change) if all d_j are distinct.

PCA: Solution

- Compute the singular value decomposition (SVD) of $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)'$

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}'.$$

.

- The principal directions are the right singular vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$.
- For the best linear approximation problem, for each $1 \leq q \leq p$, the optimal $\boldsymbol{V}_q = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_q)$.

$\boldsymbol{v}_m$	m-th principal direction
$\boldsymbol{z}_m = \boldsymbol{X}\boldsymbol{v}_m = d_m\boldsymbol{u}_m$	m-th principal component
$\boldsymbol{v}_m$	loadings of the m-th principal component

*Sparse PCA

- Joliffe et al (2003). SCoTLASS.

$$\max v' \boldsymbol{X}' \boldsymbol{X} v \quad \text{subject to } \|v\| = 1, \sum_{j=1}^{p} |v_j| \leq t.$$

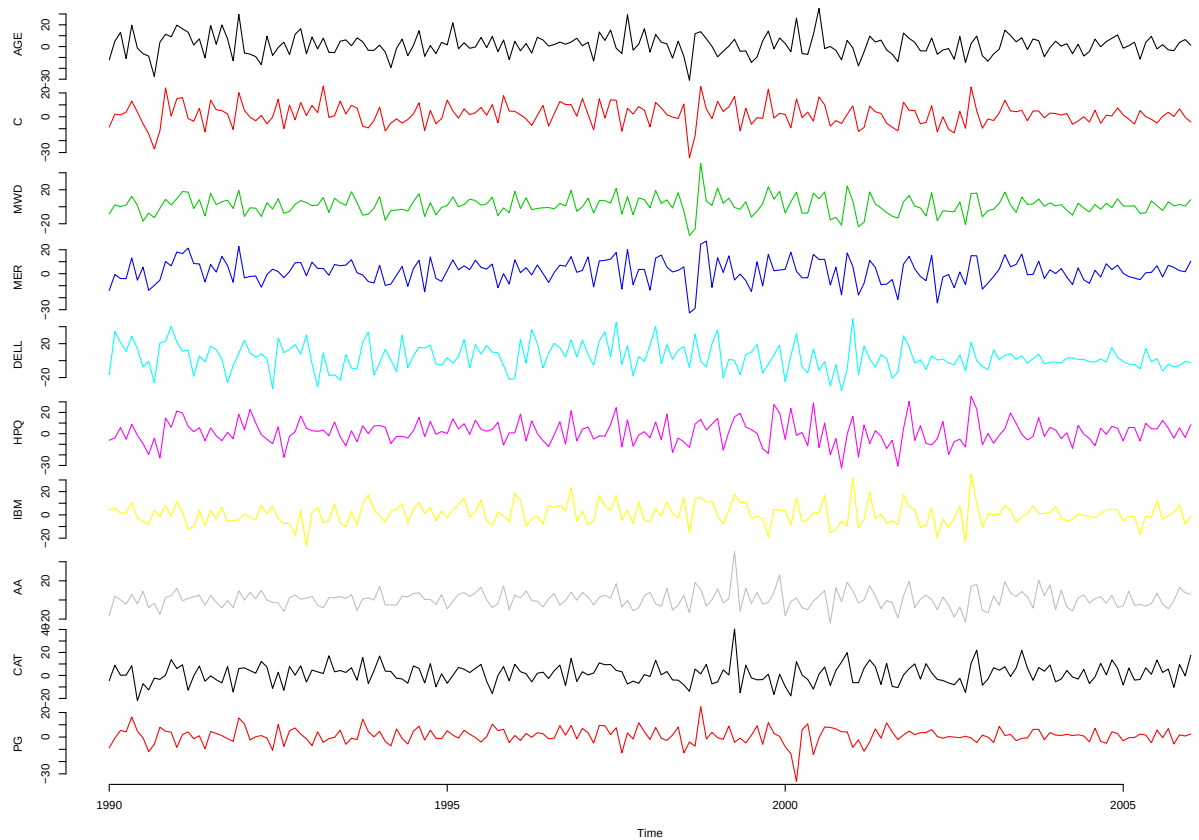
  - Not convex means the computations are difficult.
- Zou et al (2006). Sparse PCA.

$$\min_{\theta, \boldsymbol{v}_1} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \theta \boldsymbol{v}_1' \boldsymbol{x}_i\|^2 + \lambda \|\boldsymbol{v}_1\|_2^2 + \lambda_1 \|\boldsymbol{v}_1\|_1 \quad \text{subject to } \|\theta\|_2 = 1.$$

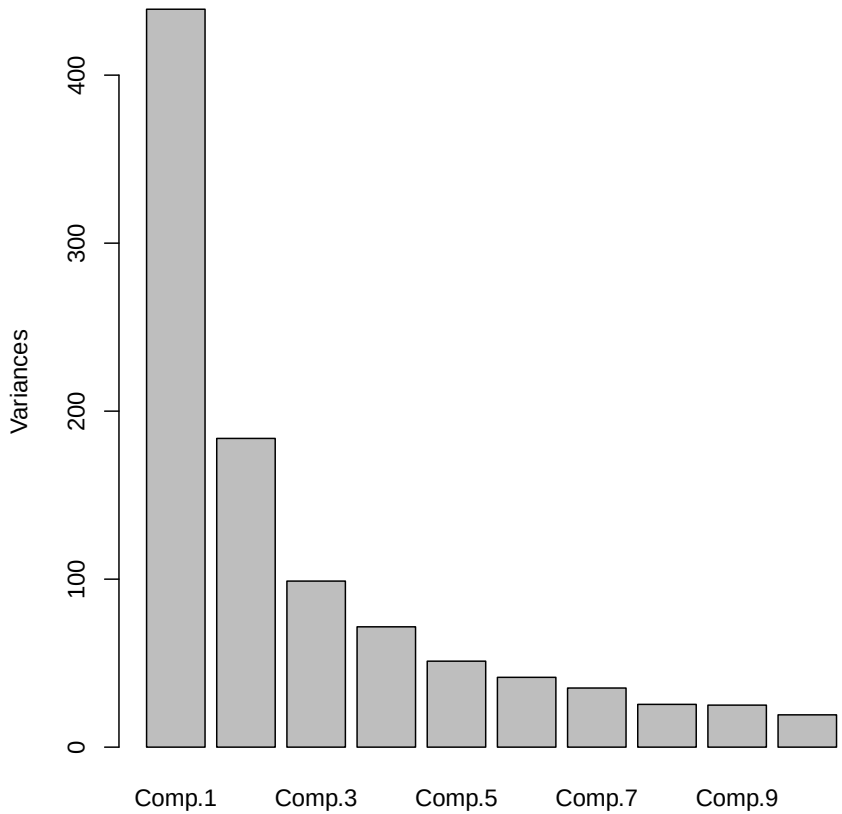
  - Fix θ, minimization over $\boldsymbol{v}_1$ is equivalent to elastic-net problems, which can be solved by LARS.
  - Fix $\boldsymbol{v}_1$, minimization over θ is done by a SVD calculation.

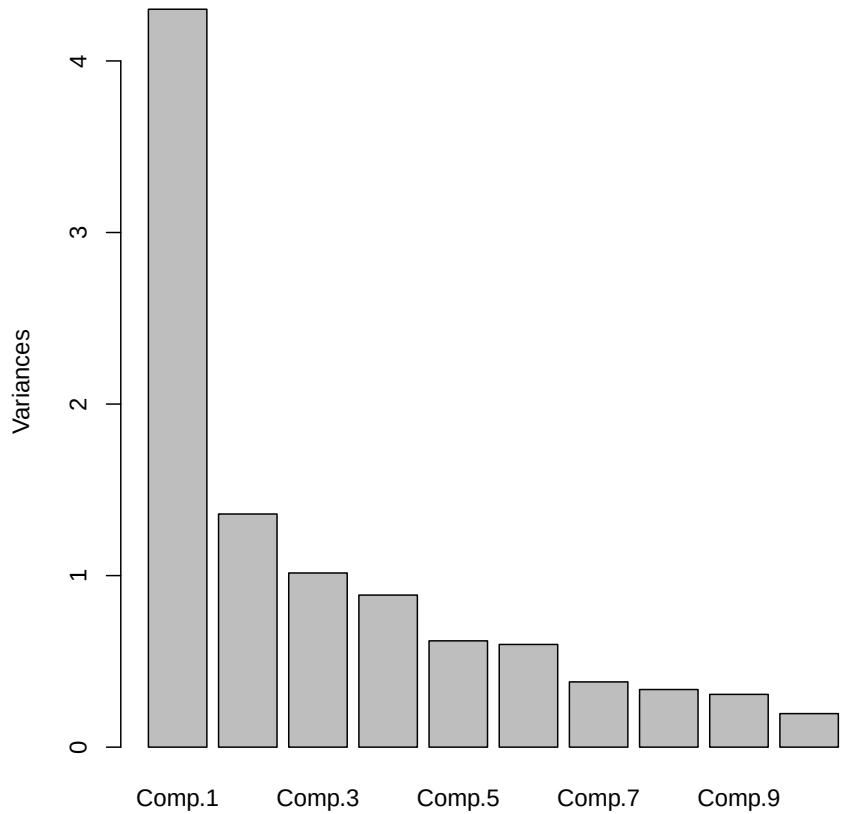
Example

- Monthly excess returns (including dividends) of ten stocks: A.G. Edwards, Citigroup, Morgan Stanley, Merrill Lynch, Dell, HP, IBM, Alcoa, Caterpillar, and P&G.
- Use the monthly series of 3-month Treasury bill rates of the secondary market as the risk-free interest rate to obtain simple excess returns.
- The sample span is from January 1990 to December 2006.



Asset Excess Returns





Outline

PCA Revisited

- Assume the data matrix $\boldsymbol{X}$ is $N \times p$, we now allow $p \geq N$, which is usually called a "high-dimensional" problem.
- Set $r = \min\{p, N\}$.
- The SVD of $\boldsymbol{X}$ has the form $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}'$.:
  - $\boldsymbol{U}$ is $N \times r$, orthogonal.
  - $\boldsymbol{V}$ is $p \times r$, orthogonal.
  - $\boldsymbol{D}$ is $r \times r$, diagonal, with nonnegative diagonal entries.
- The principal components $\boldsymbol{z}_1 = d_1\boldsymbol{u}_1, \ldots, \boldsymbol{z}_r = d_r\boldsymbol{u}_r$ can be obtained without calculating the principal directions $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$.
  - Find the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r$ and eigenvectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r$ of the $N \times N$ matrix $\boldsymbol{X}\boldsymbol{X}'$.
  - The m-th principal component is given by $\boldsymbol{z}_m = \sqrt{\lambda_m}\,\boldsymbol{u}_m$, for $1 \leq m \leq r$.
- The (i, j)-th entry of $\boldsymbol{X}\boldsymbol{X}'$ is $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$.
- Suppose a new object $\boldsymbol{x}_0$ arrives, its principal components can also be calculated without knowing $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$.

$$z_{0m} := \boldsymbol{x}_0'\boldsymbol{v}_m = \boldsymbol{x}_0'\boldsymbol{X}'\boldsymbol{u}_m/d_m = \frac{1}{\sqrt{\lambda_m}} \sum_{i=1}^{N} u_{im}\langle \boldsymbol{x}_0, \boldsymbol{x}_i \rangle.$$

Derived Inputs and Kernel PCA

- Lift to a higher dimensional space using M-dimensional derived inputs $h(\boldsymbol{x}_i)$. Let $h(\boldsymbol{X}) = [h(\boldsymbol{x}_1), \ldots, h(\boldsymbol{x}_N)]'$ be the $N \times M$ matrix with derived inputs.
- The principal components obtained using the derived inputs can be obtained from eigenvalues and eigenvectors of $h(\boldsymbol{X})[h(\boldsymbol{X})]'$.
- The (i, j)-th entry of $h(\boldsymbol{X})[h(\boldsymbol{X})]'$ is $\langle h(\boldsymbol{x}_i), h(\boldsymbol{x}_j)\rangle$.
- To calculate the principal components of a new object $\boldsymbol{x}_0$, one only needs to calculate $\langle h(\boldsymbol{x}_0), h(\boldsymbol{x}_i)\rangle$.
- To calculate the principal components, all needed are the inner products between (derived) input vectors.
- Go one step further: replacing $\langle h(\boldsymbol{x}_i), h(\boldsymbol{x}_j)\rangle$ by $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, where $K(\cdot, \cdot)$ is a suitable kernel function.

Kernel PCA and Centering

- Let $\boldsymbol{K}$ be the $N \times N$ matrix whose (i,j)-th entry is $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.
- Note that $h(\boldsymbol{x}_i)$ may not be centered, i.e. $\sum_{i=1}^{N} h(\boldsymbol{x}_i) \neq \boldsymbol{0}$.
- For the original PCA, easy to center the original inputs.
- For kernel PCA, how to center the infinite-dimensional derived inputs?
- Let $\boldsymbol{1} = \boldsymbol{1}_N$ be the N-dimensional vector with all entries equal to 1.

$$\left[\boldsymbol{I} - \frac{1}{N}\boldsymbol{1}\boldsymbol{1}'\right]\boldsymbol{x}^j = \boldsymbol{x}^j - \bar{x}_j\boldsymbol{1}.$$

- Let $\boldsymbol{J} = \boldsymbol{1}\boldsymbol{1}'/N$. Original PCA depends on the matrix

$$[(\boldsymbol{I} - \boldsymbol{J})\boldsymbol{X}]\,[(\boldsymbol{I} - \boldsymbol{J})\boldsymbol{X}]' = (\boldsymbol{I} - \boldsymbol{J})(\boldsymbol{X}\boldsymbol{X}')(\boldsymbol{I} - \boldsymbol{J}).$$

- Similarly, in kernel PCA, use the matrix

$$\tilde{\boldsymbol{K}} = (\boldsymbol{I} - \boldsymbol{J})\boldsymbol{K}(\boldsymbol{I} - \boldsymbol{J}).$$

Kernel PCA Solution

- Suppose the rank of $\tilde{\boldsymbol{K}}$ is r.
- Let $\lambda_1 \geq \cdots \geq \lambda_r$ and $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r$ be the eigenvalues and eigenvectors of $\tilde{\boldsymbol{K}}$.
- The m-th principal component is $\boldsymbol{z}_m = \sqrt{\lambda_m}\, \boldsymbol{u}_m$.
- Suppose a new object $\boldsymbol{x}_0$ arrives, its principal components are calculated as

$$z_{0m} = \frac{1}{\sqrt{\lambda_m}} \sum_{i=1}^{N} u_{im} \tilde{K}(\boldsymbol{x}_0, \boldsymbol{x}_i),$$

where

$$\tilde{K}(\boldsymbol{x}_0, \boldsymbol{x}_i) = \left[K(\boldsymbol{x}_0, \boldsymbol{x}_i) - \frac{1}{N} \sum_{j=1}^{N} K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \frac{1}{N} \sum_{j=1}^{N} K(\boldsymbol{x}_0, \boldsymbol{x}_j) + \frac{1}{N^2} \mathbf{1}' \boldsymbol{K} \mathbf{1} \right].$$

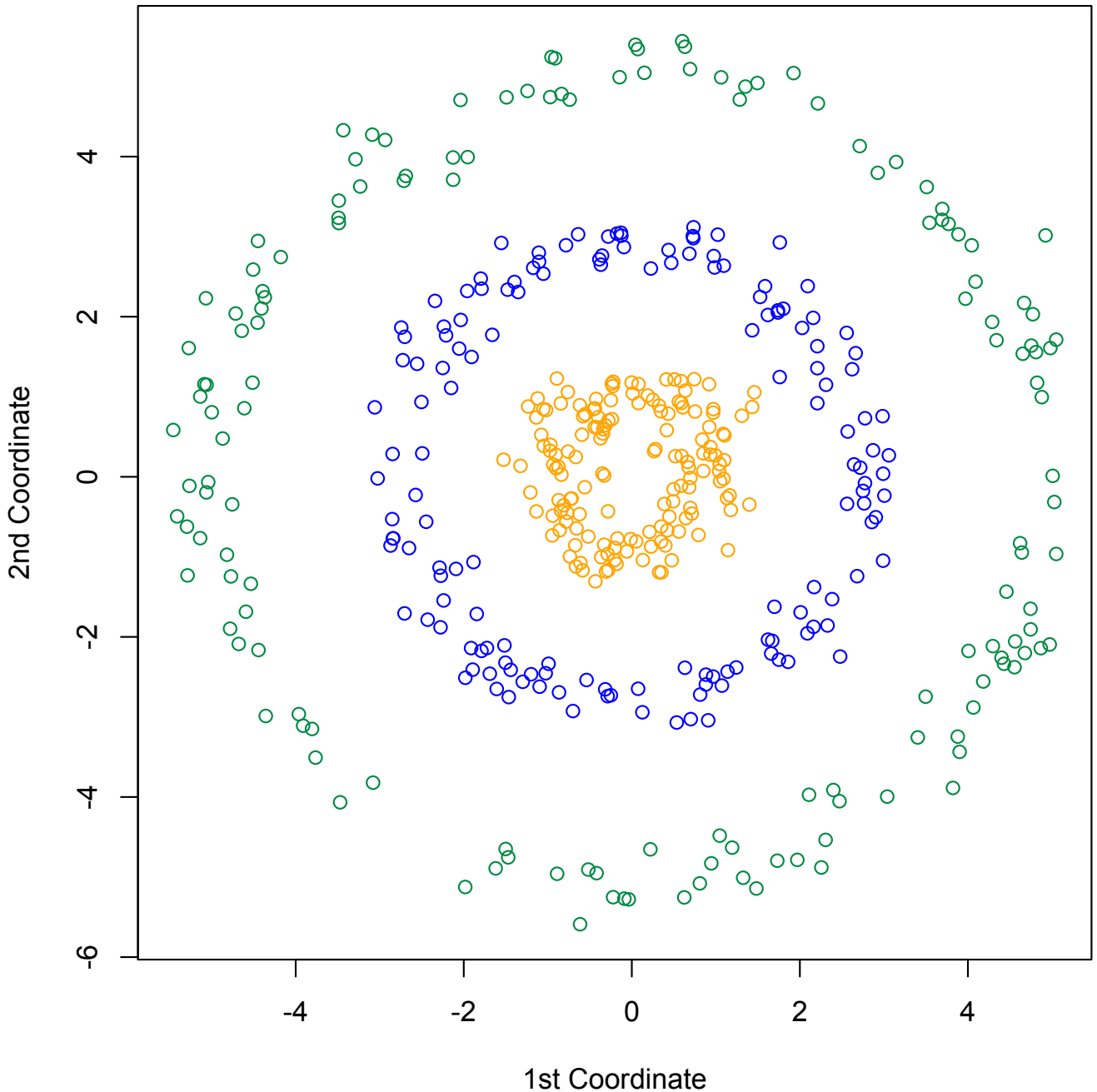
- The principal components of a new object in matrix form is (arguably) simpler. It is left as a homework problem.

Example

- Three clusters. The points are distributed uniformly on the circle, with radius 1, 2.8 and 5 in the three groups.
- Each coordinate of a point is added by a Gaussian noise with standard deviation 0.25.

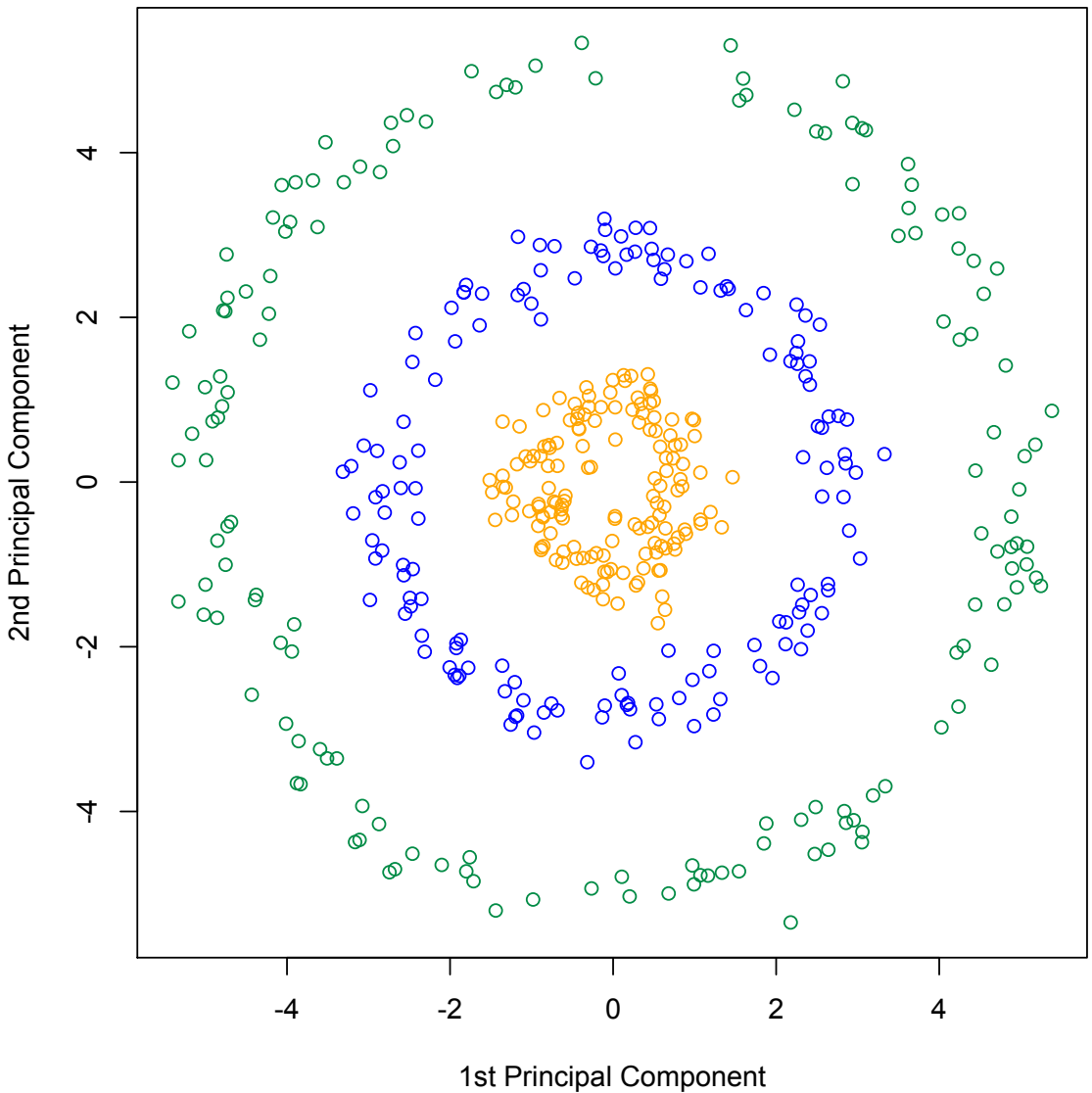
```
n=150
p=2

set.seed(123)
X=matrix(rnorm(p*n*3),nrow=n*3)
X.norm=apply(X^2,MAR=1,FUN=sum)
X.norm=sqrt(X.norm)
X=X/X.norm
label=rep(c(1,2,3),each=n)
radius=rep(c(1,2.8,5),each=n)
color=rep(c("orange","blue","springgreen4"),each=n)
X=X*radius
X=X+.25*matrix(rnorm(p*n*3),nrow=n*3)
plot(X[,1],X[,2],col=color,xlab="1st Coordinate",ylab="2nd Coordinate")
```

Example: Data



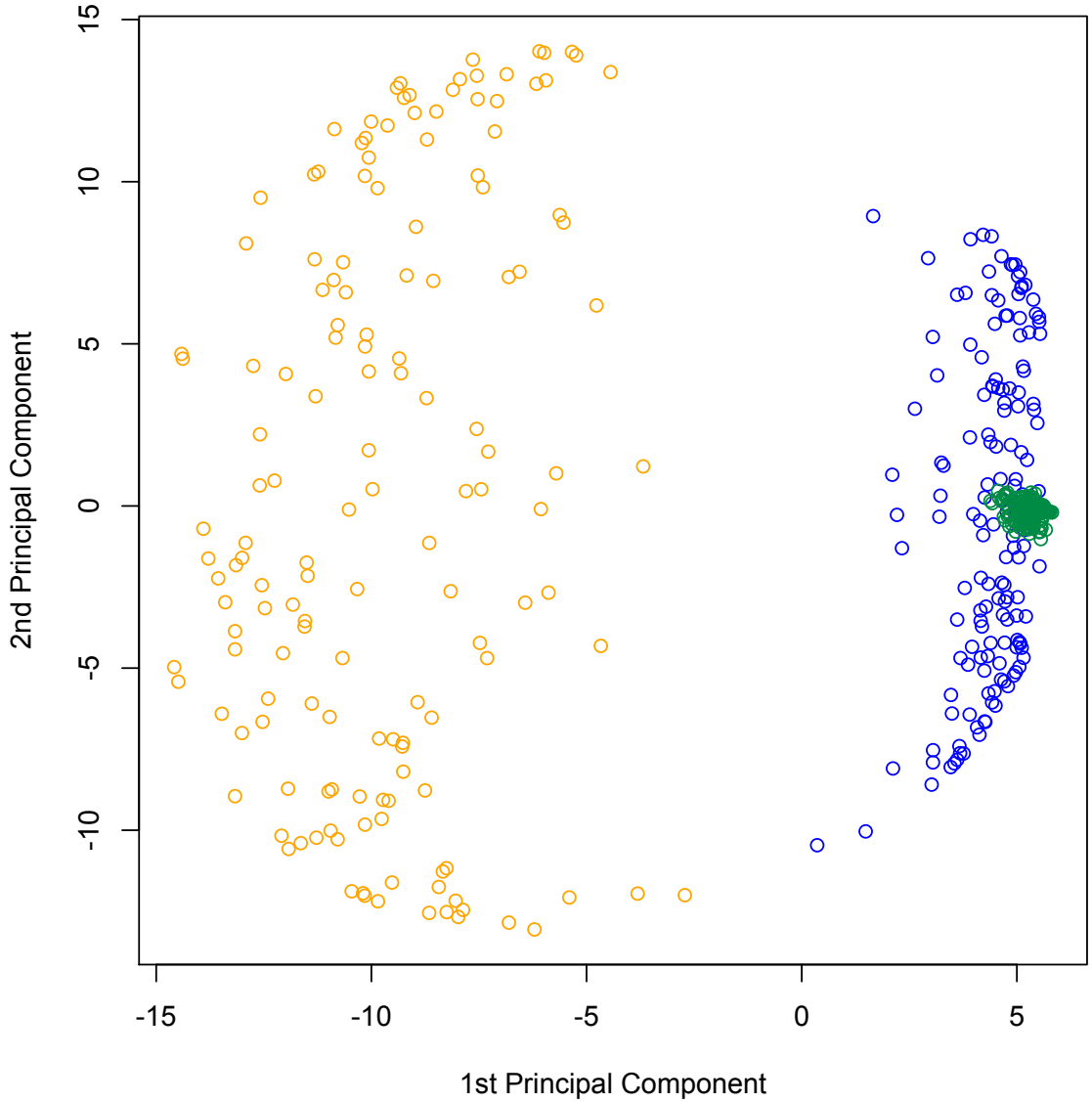
Example: PCA

```
X.pca=svd(X)$u
X.d=svd(X)$d
plot(X.pca[,1]*X.d[1],X.pca[,2]*X.d[2],col=color,
       xlab="1st Principal Component",ylab="2nd Principal Component")
```



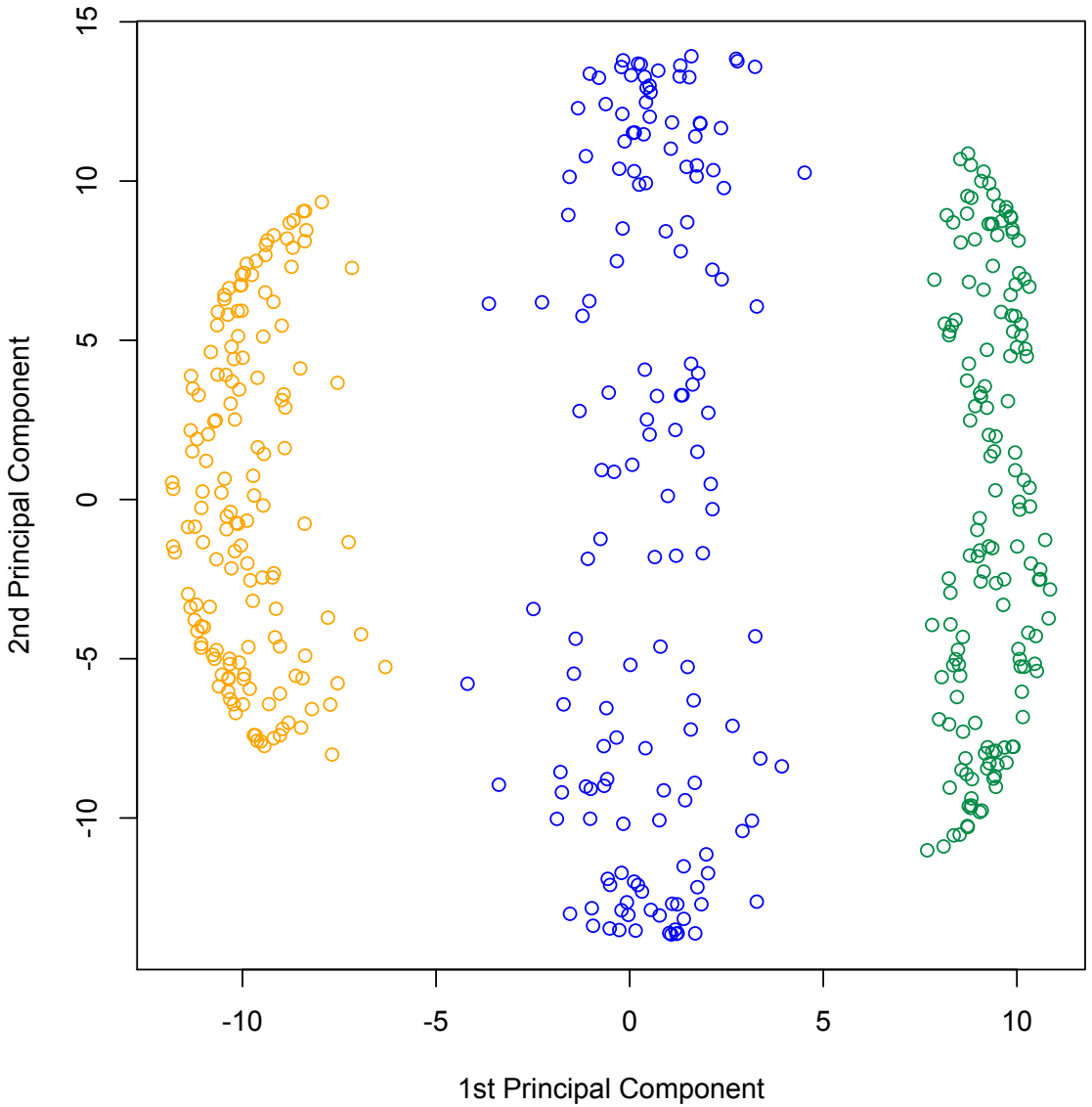
Example: Kernel PCA 1

```
X.kpc1=kpca(X,kernel="rbfdot",kpar=list(sigma=.5),features=2)
plot(rotated(X.kpc1),col=color,
     xlab="1st Principal Component",ylab="2nd Principal Component")
```



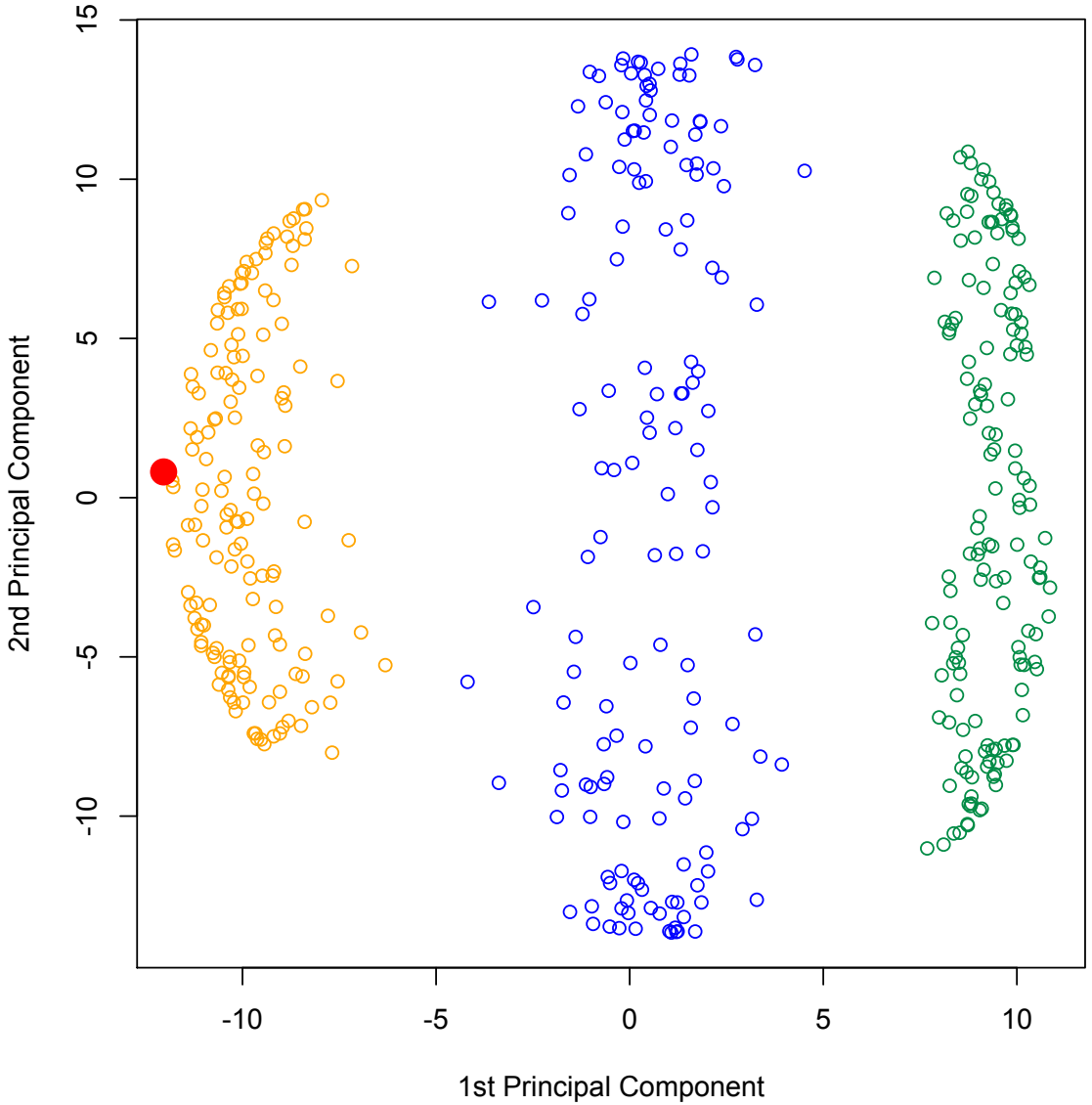
Example: Kernel PCA 2

```
X.kpc2=kpca(X,kernel="rbfdot",kpar=list(sigma=.1),features=2)
plot(rotated(X.kpc2),col=color,
     xlab="1st Principal Component",ylab="2nd Principal Component")
```



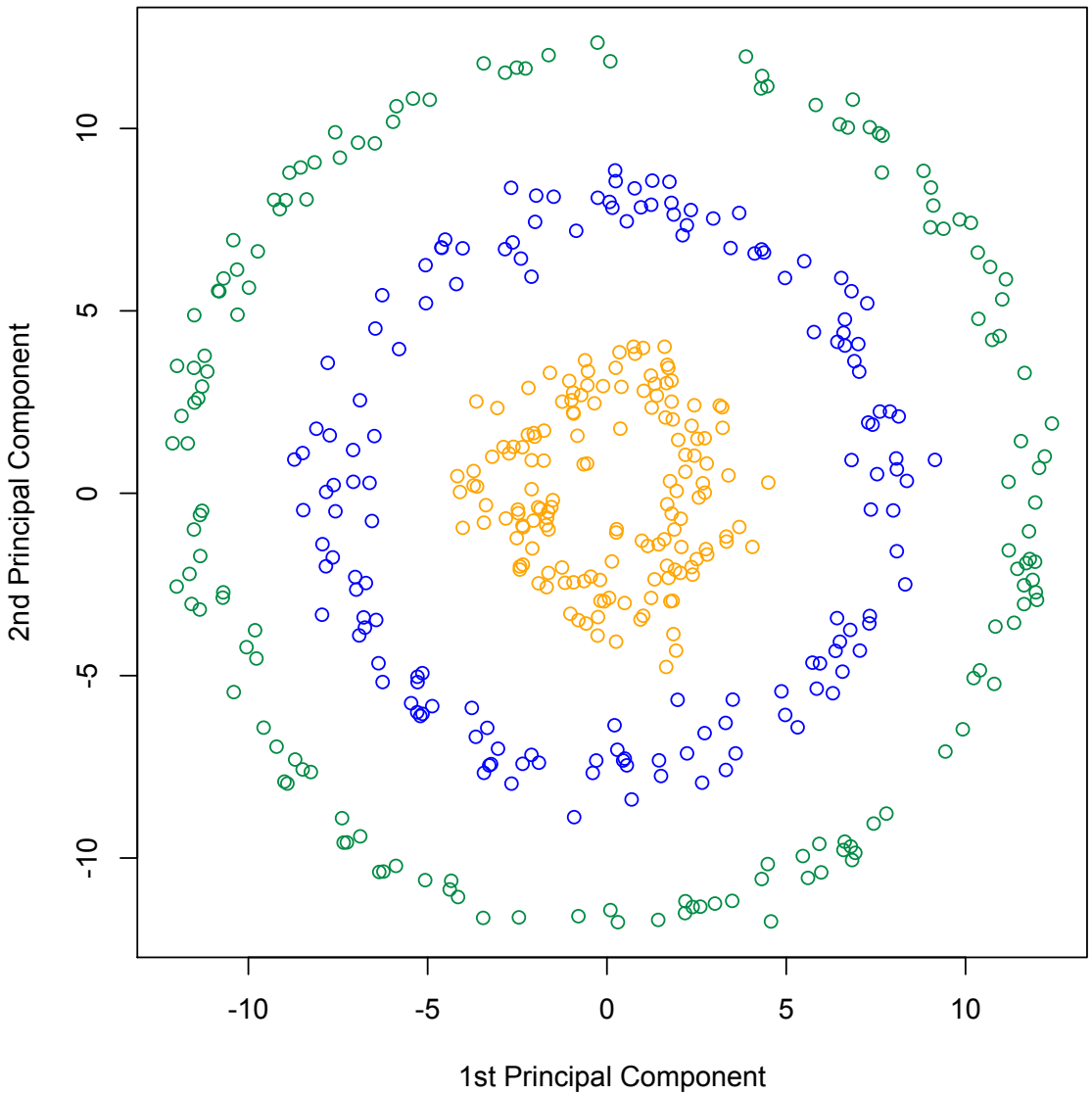
Example: Kernel PCA 2 with a New Point

```
newx = predict(X.kpc2,t(c(0,0)))
points(newx,pch=19,cex=2,col="red")
```



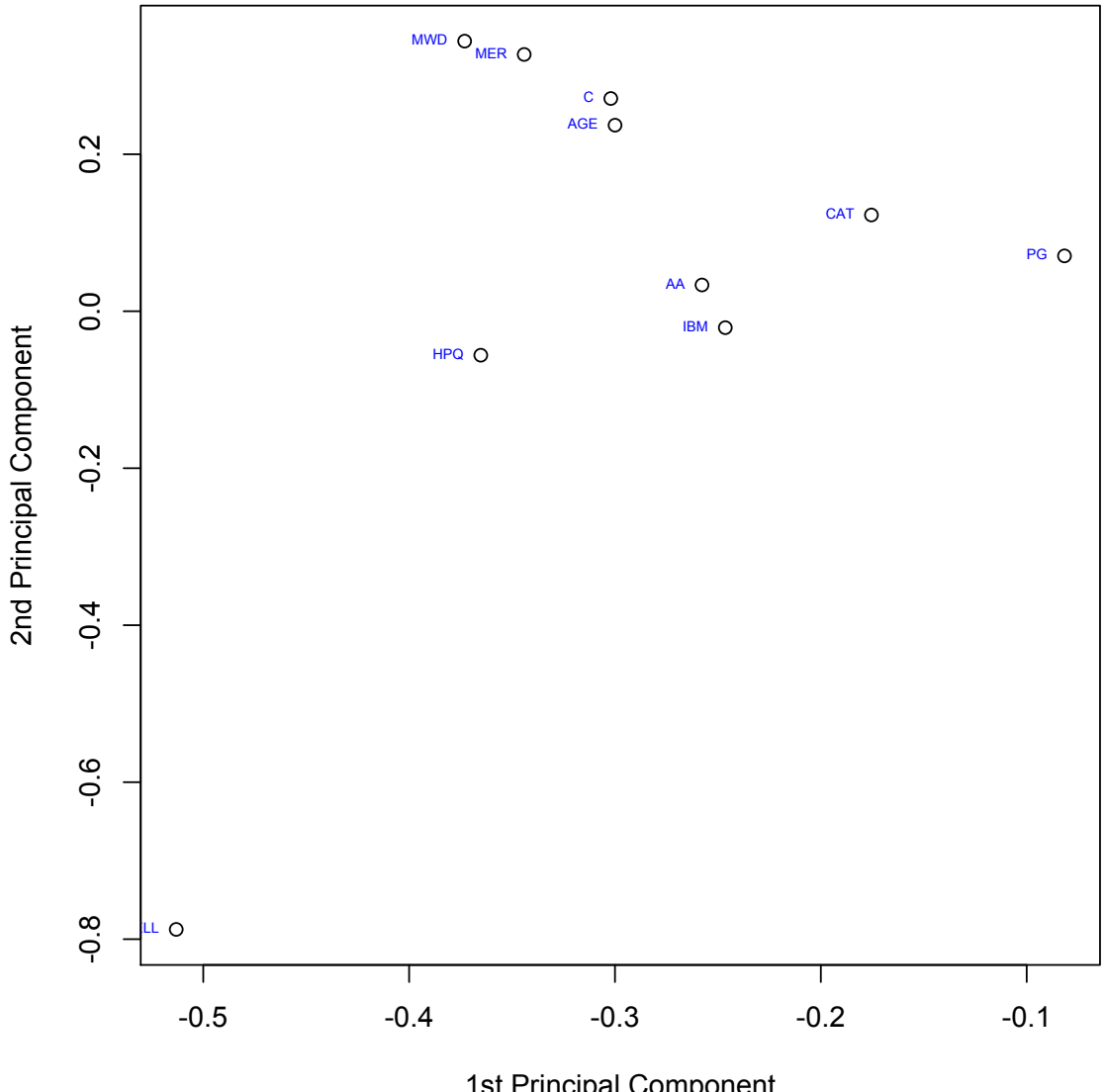
Example: Kernel PCA 3

```
X.kpc3=kpca(X,kernel="rbfdot",kpar=list(sigma=.01),features=2)
plot(rotated(X.kpc3),col=color,
     xlab="1st Principal Component",ylab="2nd Principal Component")
```



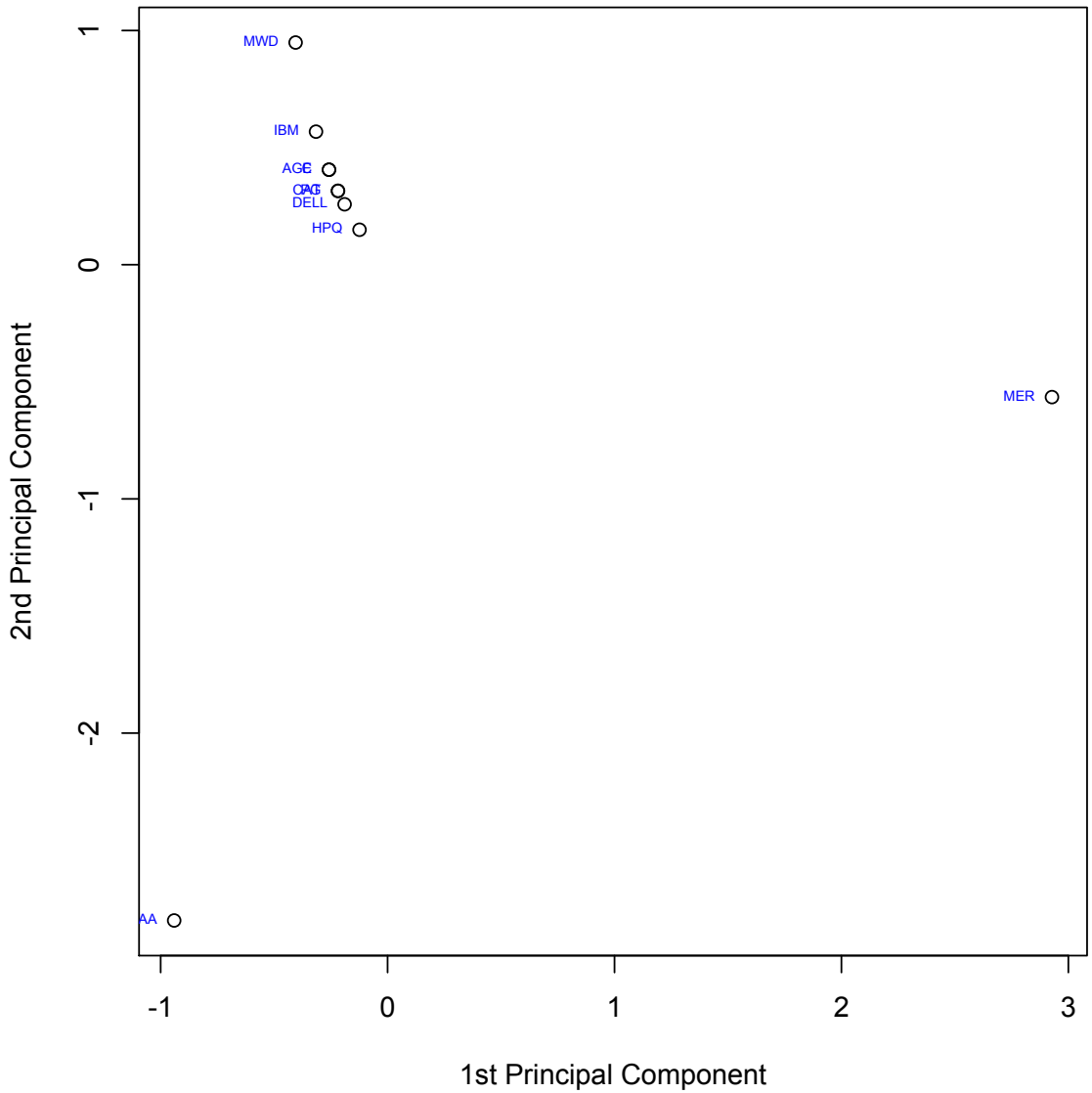
Excess Return: PCA

```
er10=read.table("04_er10.txt",header=T)
er10.mat=t(as.matrix(er10))
er10.pca=svd(er10.mat)
plot(er10.pca$u[,1],er10.pca$u[,2],xlab="1st Principal Component",ylab="2nd Principal Compo
text(er10.pca$u[,1],er10.pca$u[,2], rownames(er10.mat), cex=0.5, pos=2, col="blue")
```



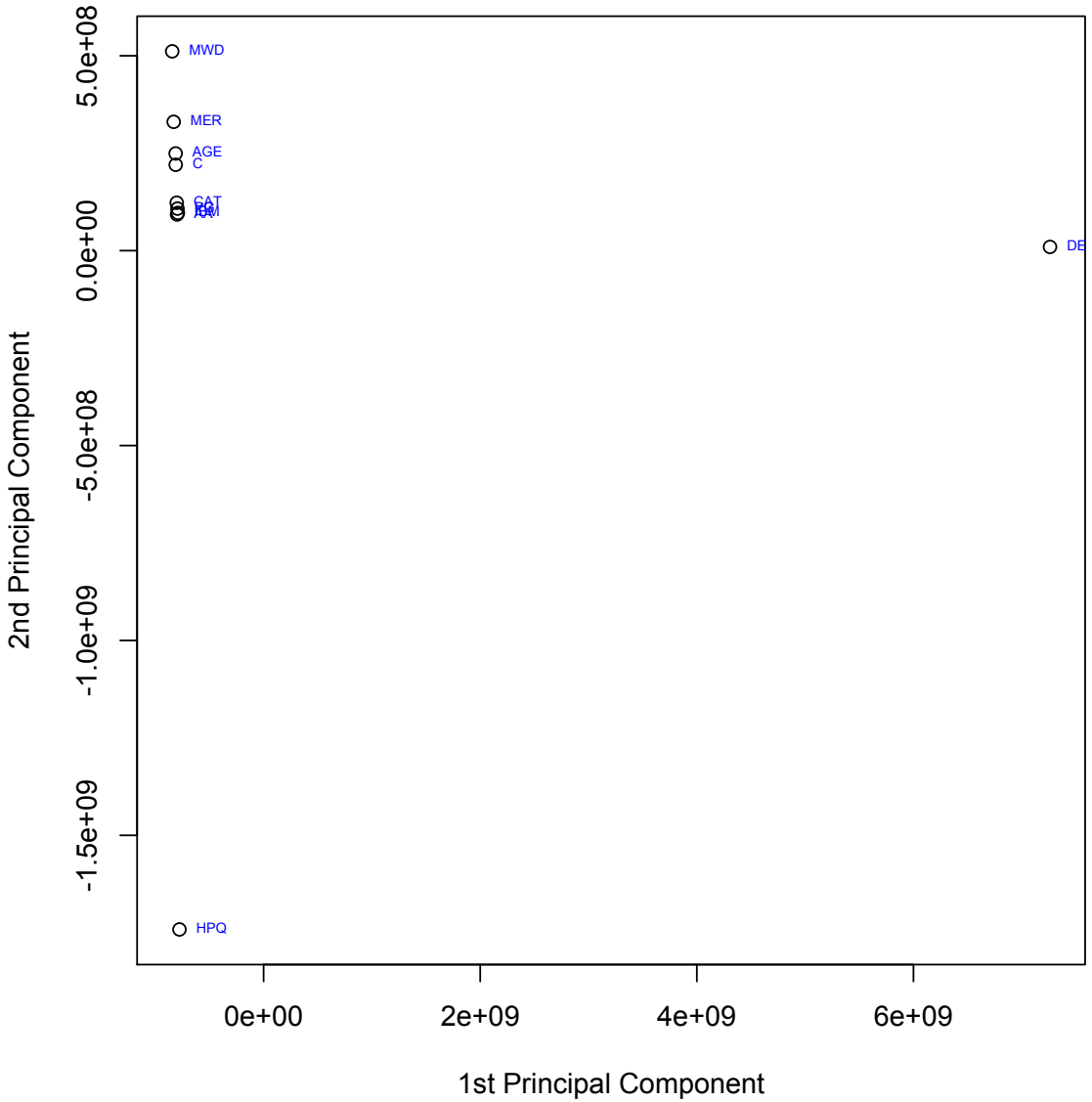
Excess Return: Kernel PCA 1

```
kpc1=kpca(er10.mat,kernel="rbfdot",kpar=list(sigma=.4),features=2)
## quite stable over different sigma
plot(rotated(kpc1),xlab="1st Principal Component",ylab="2nd Principal Component")
text(rotated(kpc1), rownames(er10.mat), cex=0.5, pos=2, col="blue")
```



Excess Return: Kernel PCA 2

```
kpc2=kpca(er10.mat,kernel="polydot",kpar=list(degree=4,scale=1,offset=1),features=2)
## quite stable over different sigma
plot(rotated(kpc2),xlab="1st Principal Component",ylab="2nd Principal Component")
text(rotated(kpc2), rownames(er10.mat), cex=0.5, pos=4, col="blue")
```



Outline

Similarity, Kernel, Similarity Graph

- Clustering: partition the observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ into a few clusters.
- Traditional K-means does not work well when the clusters are non-convex.
- Spectral clustering is designed to better handle such a situation.
- Suppose there is a $N \times N$ matrix $\boldsymbol{S}$ of pairwise similarities between all observation pairs.
  - $s_{ij} := \boldsymbol{S}[i, j]$ is the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.
  - E.g. $s_{ij} = 1/(d_{ij} + 1)$, where $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$.
- Kernel method uses $s_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, where $K(\cdot, \cdot)$ is a kernel function.
  - Radial basis kernel. $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\{-\alpha\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\}$.
- The observations can be represented in an undirected similarity graph $G = \langle V, E \rangle$.
  - Each observation is a vertex: $\boldsymbol{x}_i$ corresponds to the vertex v_i.
  - The edge connecting $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ has the weight s_{ij}.
  - The graph G and the similarity matrix $\boldsymbol{S}$ are equivalent.

Mutual K-Nearest-Neighbor Graph

- Suppose there is a $N \times N$ matrix of pairwise similarities s_{ij} between all observation pairs.
- Define the Mutual K-nearest-neighbor graph $\boldsymbol{W}$ as follows.
  - $\boldsymbol{W}$ is a $N \times N$ symmetric matrix. Denote its (i, j)-th entry by $w_{ij} := \boldsymbol{W}[i, j]$.
  - $w_{ii} = 0$.
  - For each pair (i, j), if $\boldsymbol{x}_j$ is among the k-nearest neighbor of $\boldsymbol{x}_i$, or vice versa, then define $w_{ij} = w_{ji} := s_{ij}$.
  - Otherwise, define $w_{ij} = w_{ji} = 0$.
  - The $N \times N$ matrix $\boldsymbol{W}$ can be understood as a graph with N vertexes, where the edge between the i-th and j-th vertex is given by w_{ij}.
- Define $g_i = \sum_{j=1}^{N} w_{ij}$, and let $\boldsymbol{G}$ be the diagonal matrix with diagonal elements $\{g_1, \ldots, g_N\}$.
- The Unnormalized graph Laplacian is defined by $\boldsymbol{L} = \boldsymbol{G} - \boldsymbol{W}$.
  - Can also use the normalized version: $\tilde{\boldsymbol{L}} := \boldsymbol{I} - \boldsymbol{G}^{-1}\boldsymbol{W}$.

Spectral Clustering

- Spectral clustering finds the matrix $\boldsymbol{Z}_{N\times m}$, whose j-th column is the eigenvector corresponding to the $(N-j)$-th largest eigenvalue of $\boldsymbol{L}$.
- In other words, the j-th column of $\boldsymbol{Z}$ corresponds to the $(j+1)$-th smallest eigenvalue of $\boldsymbol{L}$.
- Now treat the i-th row as the new 'input' of the i-th point, and apply the standard K-means algorithm.
- How does it work? Let $\boldsymbol{f} \in \mathbb{R}^N$ be a unit vector,

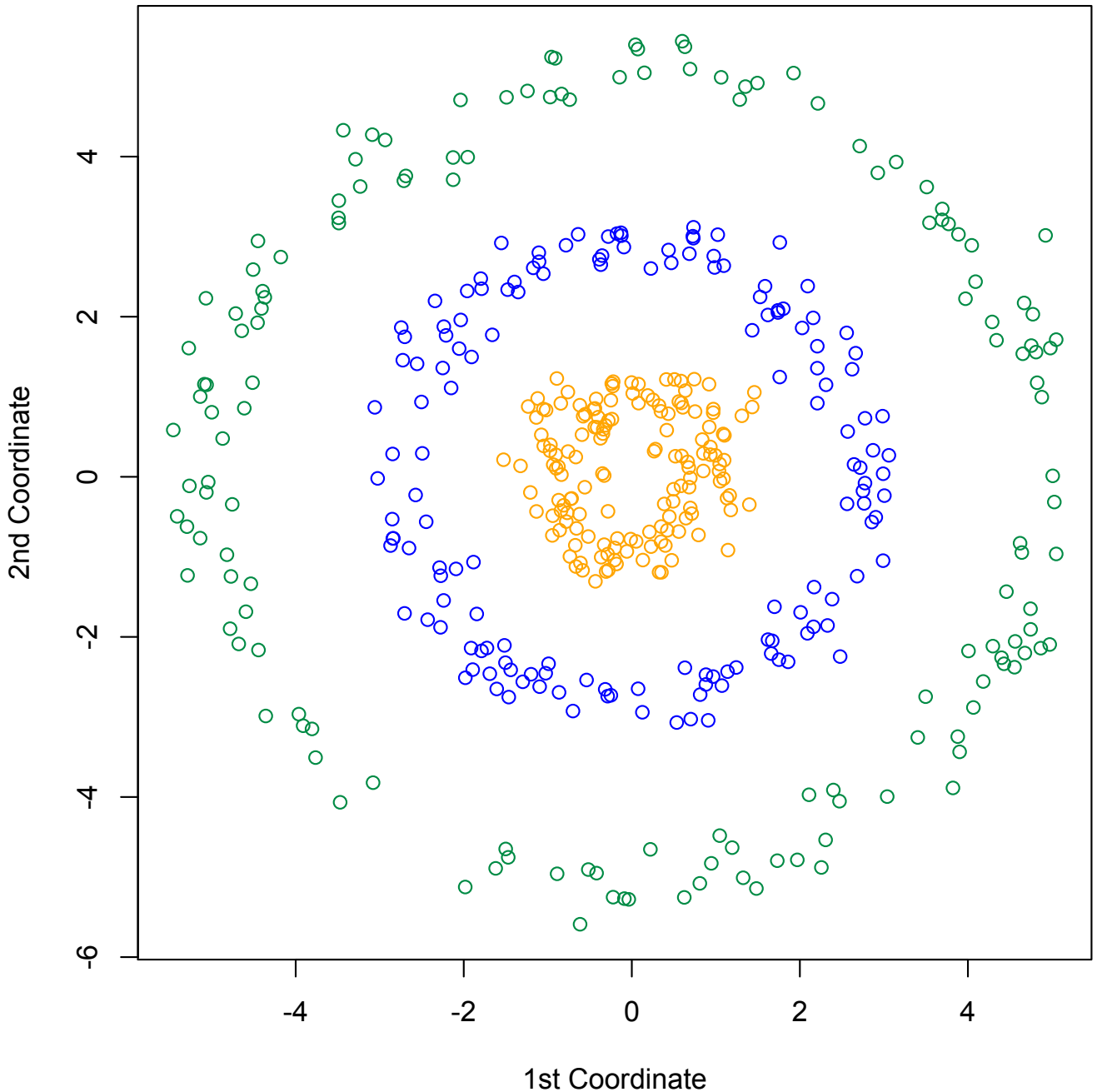
$$\boldsymbol{f}'\boldsymbol{L}\boldsymbol{f} = \sum_{i=1}^{N} g_i f_i^2 - \sum_{i,j=1}^{N} f_i f_j w_{ij} = \frac{1}{2}\sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2.$$

  - A pair (i,j) with a larger similarity w_{ij} will push the coordinates f_i and f_j closer.
- Caution! There is always an eigenvalue 0, with the constant eigenvector.
- If the graph $\boldsymbol{W}$ is not connected, there will be more 0 eigenvalues.

Relationship with Kernel PCA

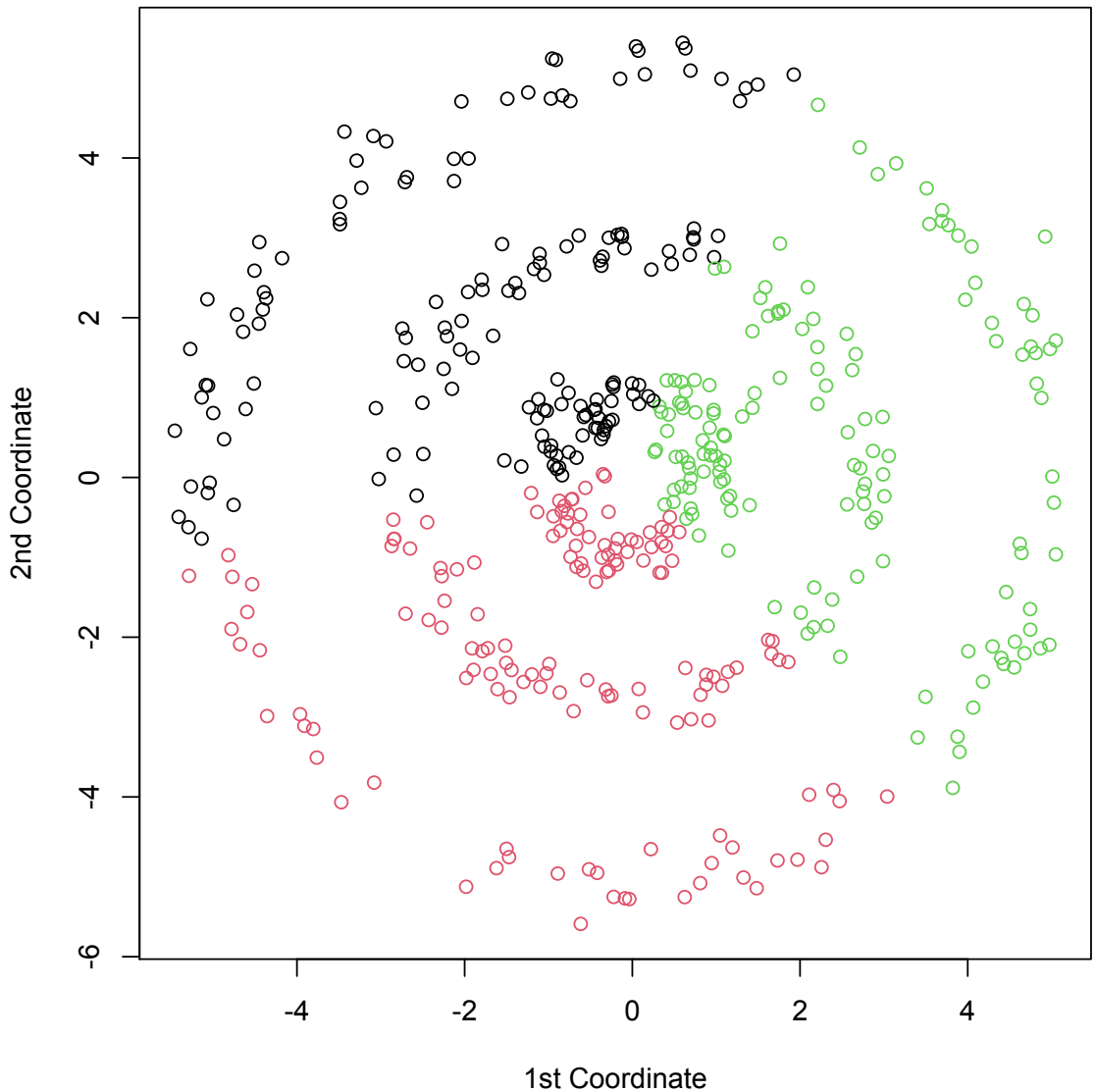
- The Kernel matrix $\boldsymbol{K}$ is the same as the similarity matrix $\boldsymbol{S}$.
- The graph $\boldsymbol{W}$ is a localized version of $\boldsymbol{S}$.
- Kernel PCA uses the centered version $\tilde{\boldsymbol{K}}$ of $\boldsymbol{K}$.
- Kernel PCA finds eigenvector corresponding to the largest eigenvalues of $\tilde{\boldsymbol{K}}$, which is equivalent to finding the eigenvectors corresponding to the smallest eigenvalues of $\boldsymbol{I} - \tilde{\boldsymbol{K}}$.
- Spectral clustering finds the eigenvectors corresponding to the smallest eigenvalues of $\boldsymbol{G} - \boldsymbol{W}$.

Example: Data



Example: K-Means

```
X.kmean=kmeans(X,centers=3)
plot(X,col=X.kmean$cluster,xlab="1st Coordinate",ylab="2nd Coordinate")
```

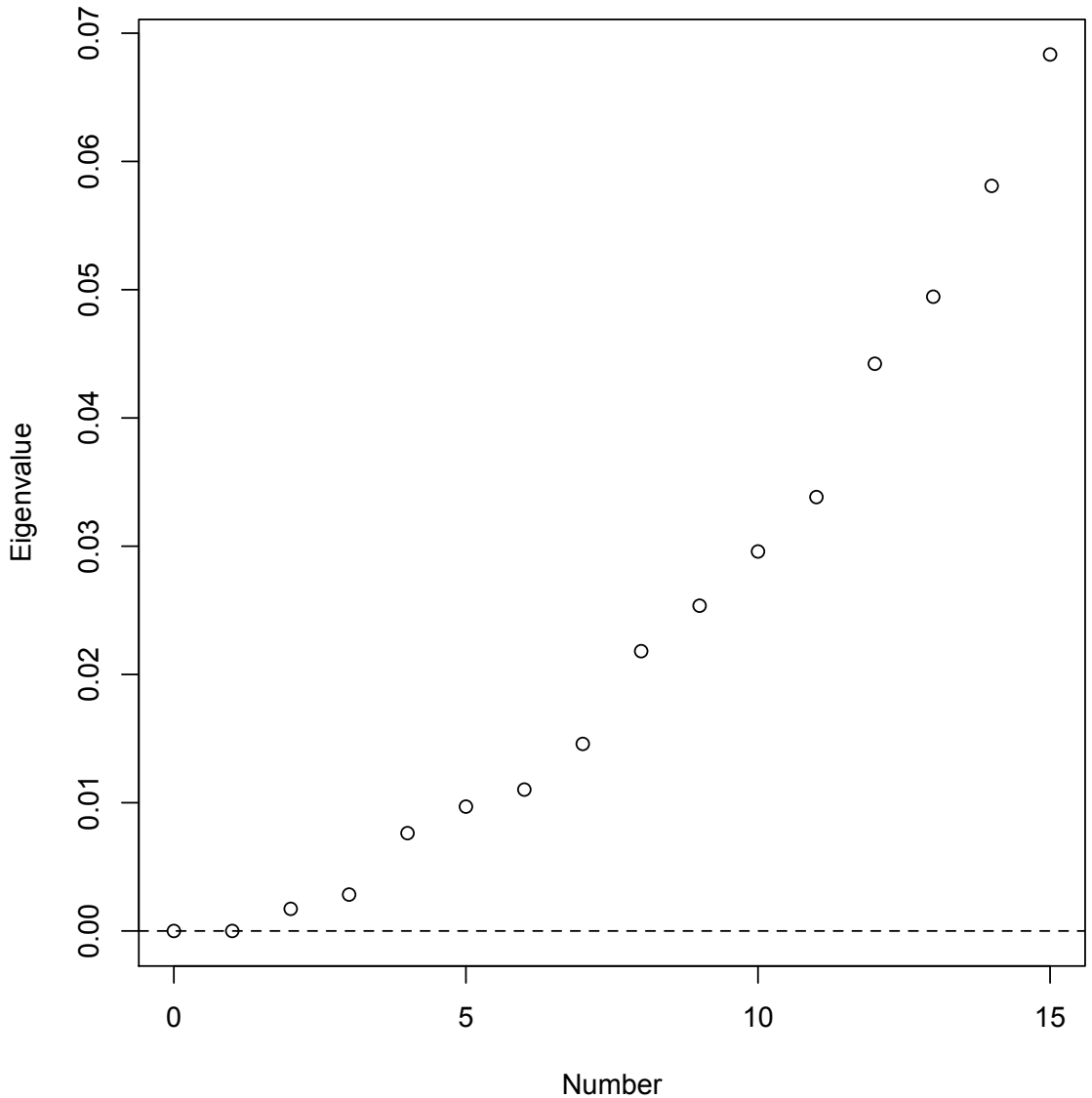


Example: Spectral Clustering

```
########### Spectral clustering
knn.sym=function(X,k=5){
    n=dim(X)[1]
    X.dist=as.matrix(dist(X))
    NN.sym=array(0,c(n,n))
    for (i in 1:n){
        xorder=order(X.dist[i,])
        NN.sym[i,xorder[1:(k+1)]]=1
    }
    NN.sym=NN.sym+t(NN.sym)
    NN.sym[NN.sym==2]=1
    return(NN.sym)
}

spec.clust=function(X,k=5,alpha=1){
    n=dim(X)[1]
    X.dist=as.matrix(dist(X))
    X.sim=exp(-alpha*X.dist^2) ###### Radial basis kernel with parameter alpha
    NN=knn.sym(X,k)
    W=X.sim*NN
    g=apply(W,MAR=1,FUN=sum)
    G=diag(g)
    L=G-W
    L.spd=eigen(L)
    return(L.spd)
}
```

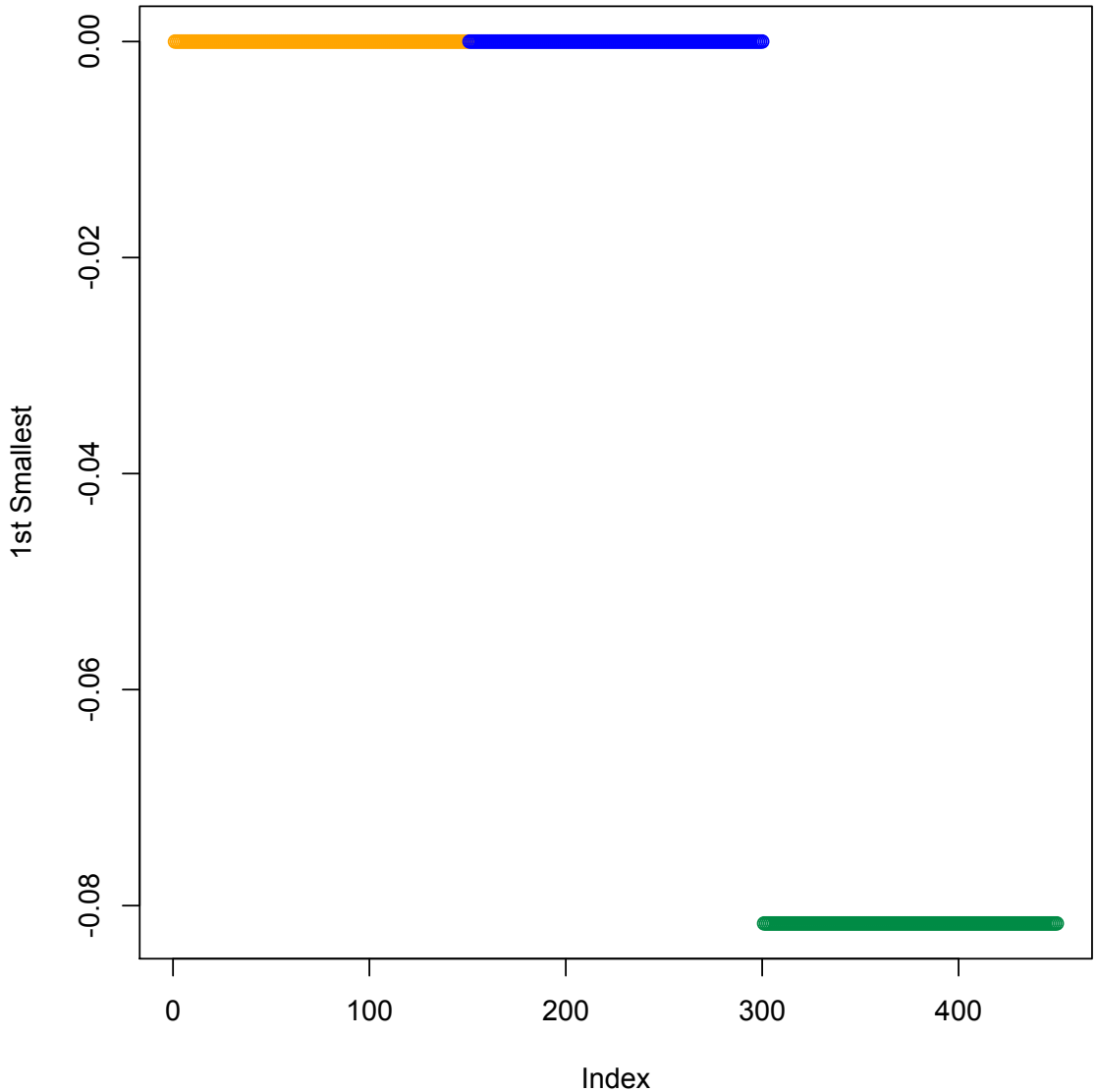
Example: Eigenvalues

```
X.spc1=spec.clust(X,k=5,alpha=.1)
n=dim(X)[1]
plot(0:15,X.spc1$values[n-(0:15)],xlab="Number",ylab="Eigenvalue")
abline(h=0,lty=2)
```



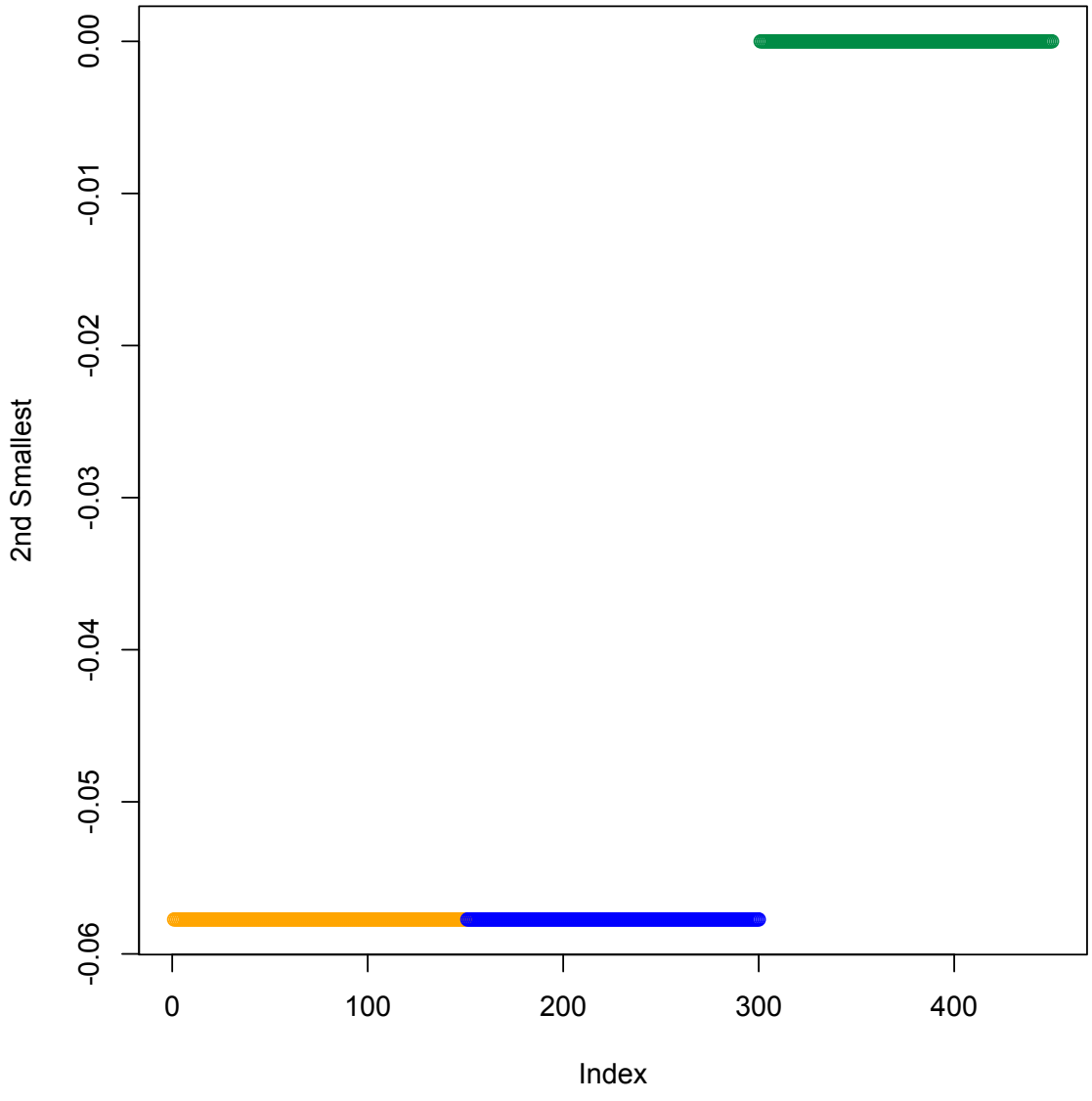
Example: 1st Smallest Eigenvector

```
plot(1:n,X.spc1$vectors[,n],xlab="Index",ylab="1st Smallest",col=color)
```



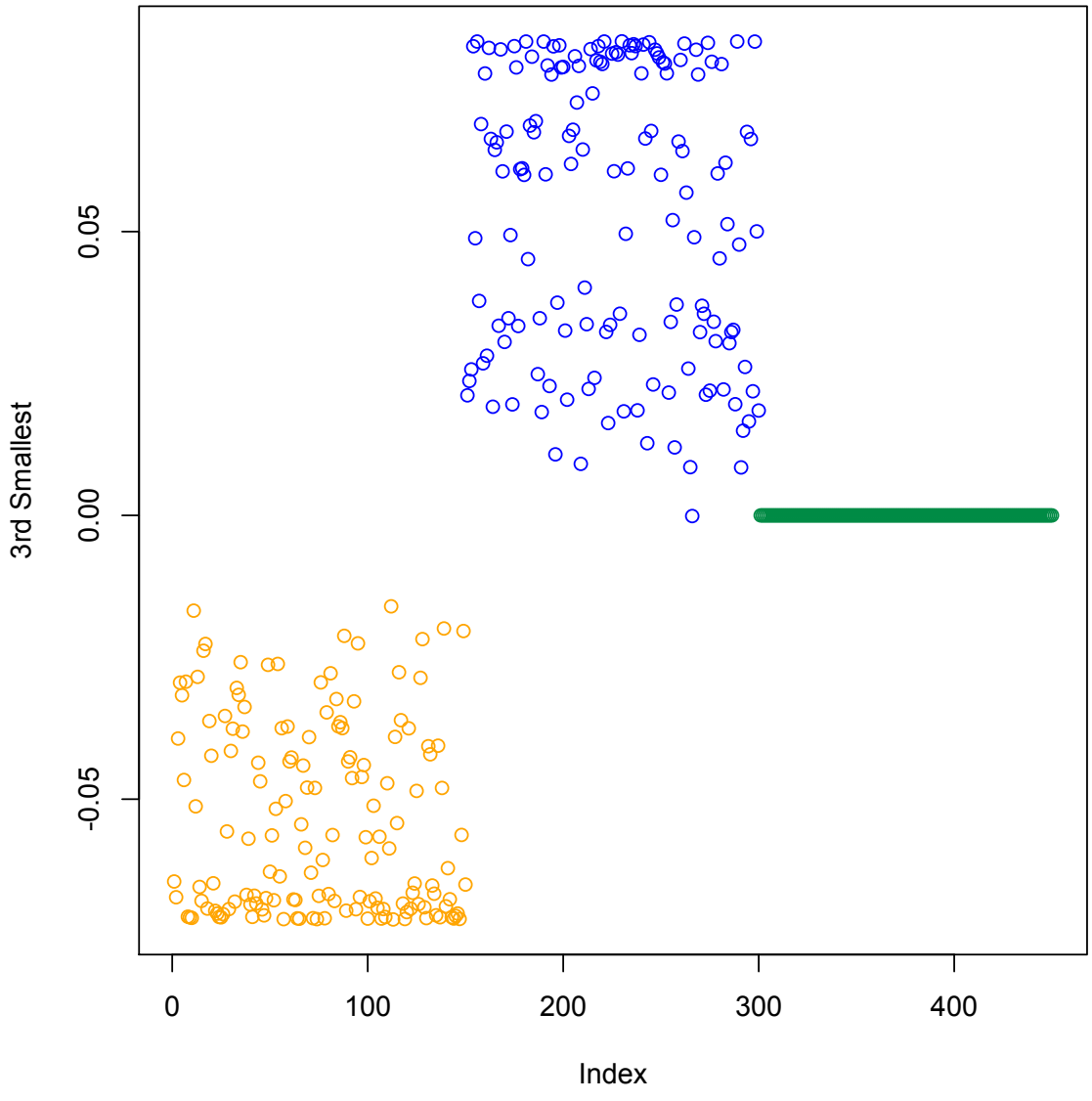
Example: 2nd Smallest Eigenvector

```
plot(1:n,X.spc1$vectors[,n-1],xlab="Index",ylab="2nd Smallest",col=color)
```



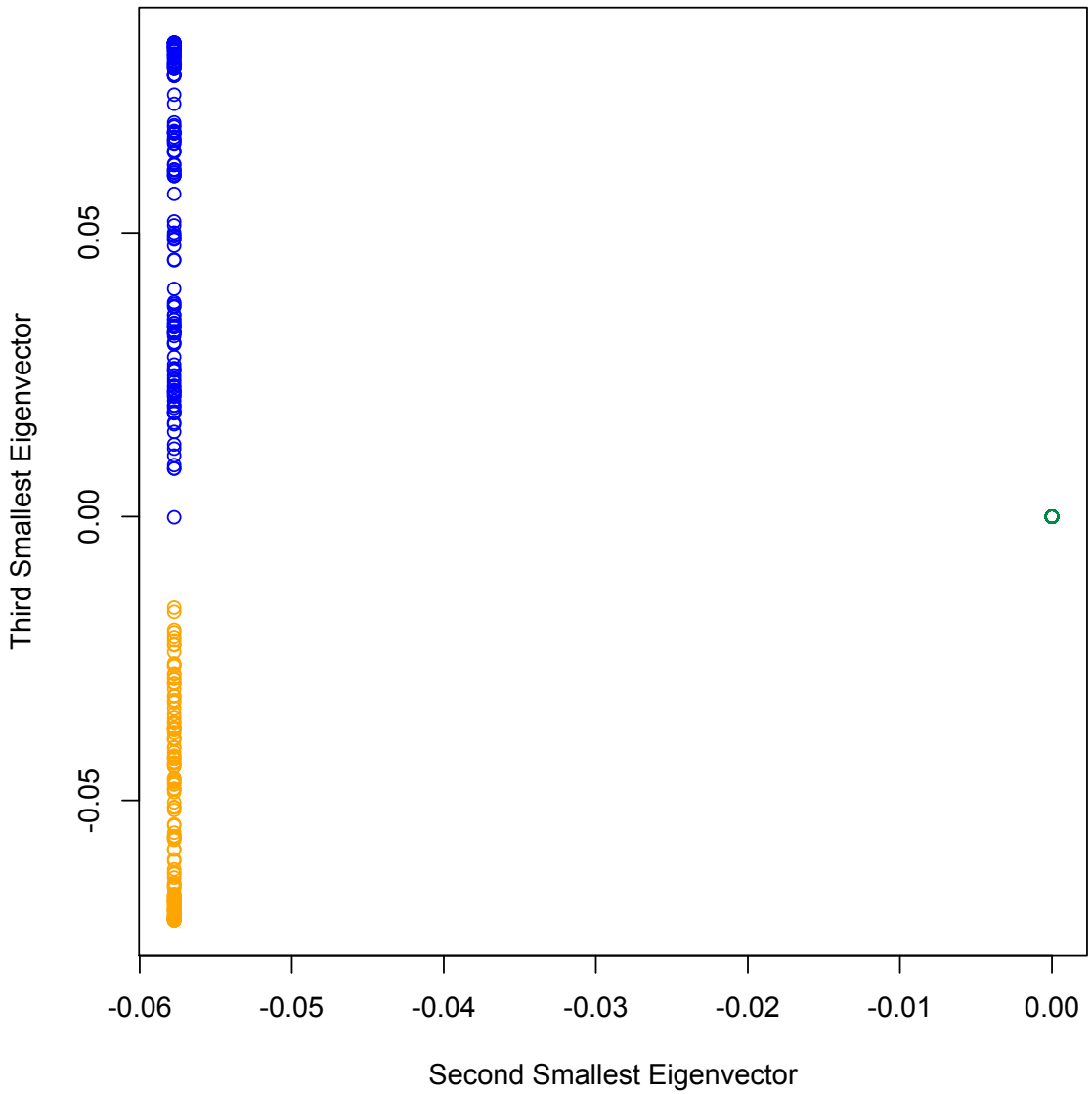
Example: 3rd Smallest Eigenvector

```
plot(1:n,X.spc1$vectors[,n-2],xlab="Index",ylab="3rd Smallest",col=color)
```



Example: 3rd vs 2nd

```
plot(X.spc1$vectors[,n-1],X.spc1$vectors[,n-2],
     xlab="Second Smallest Eigenvector",ylab="Third Smallest Eigenvector",col=c
```



Example: K-Means Using Z

```
X.new=cbind(X.spc1$vectors[,n-1],X.spc1$vectors[,n-2])
X.new.kmean=kmeans(X.new,centers=3)
plot(X,col=X.new.kmean$cluster,xlab="1st Coordinate",ylab="2nd Coordinate")
```