Cluster Analysis

Sam Bowyer

2023-02-17

Task 1

Hierarchical Clustering

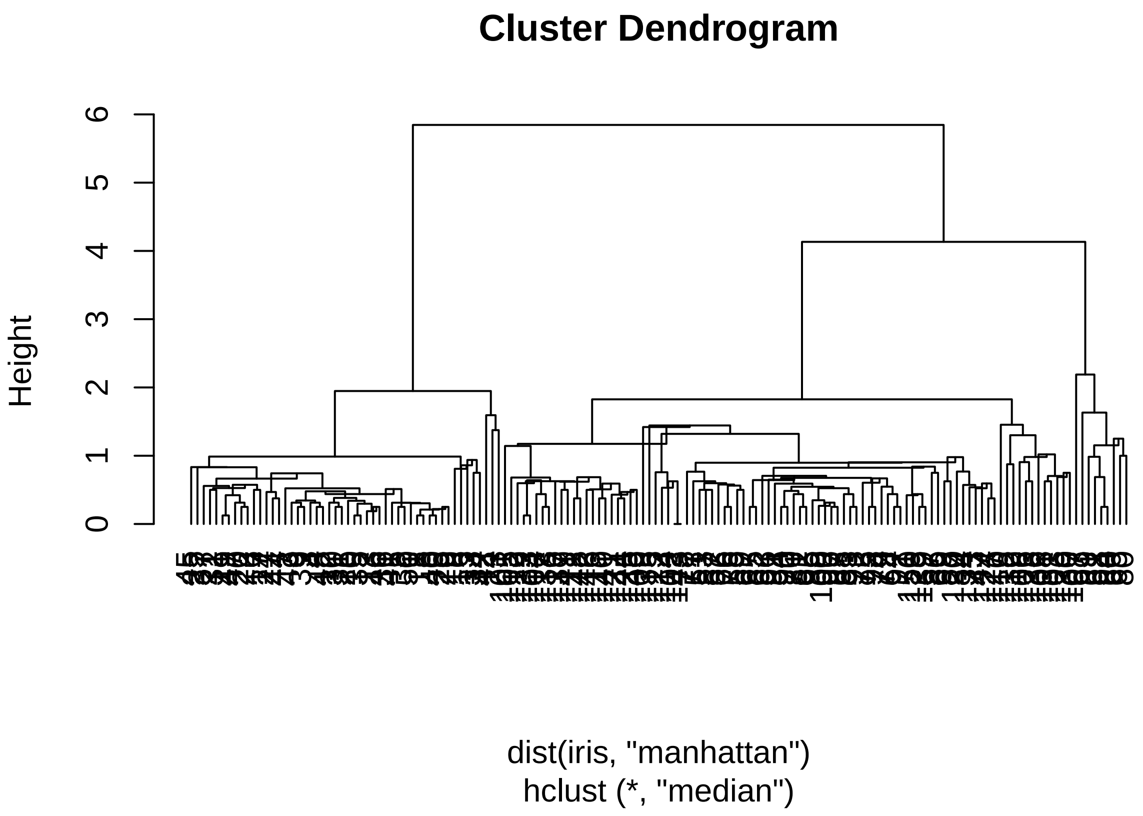
```
data(iris)
X = iris[,1:4]
head(X)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
## 2          4.9         3.0          1.4         0.2
## 3          4.7         3.2          1.3         0.2
## 4          4.6         3.1          1.5         0.2
## 5          5.0         3.6          1.4         0.2
## 6          5.4         3.9          1.7         0.4
```

```
unique(iris[,5]) # 3 types of flower, so 3 clusters would be nice
```

```
## [1] setosa     versicolor virginica
## Levels: setosa versicolor virginica
```

```
hc = hclust(dist(iris, "manhattan"), "med") # dist does manhattan distance, "med" specifies agglomerati
plot(hc, hang=-1)  # hang is optional here, a -ve value makes the labels hang down from 0 (looks a bit
```



Looks like $K = 3$ is most stable (Find a better way to word that), so cut the tree to obtain the three clusters.

```r
clusters = cutree(hc, 3)
clusters
```

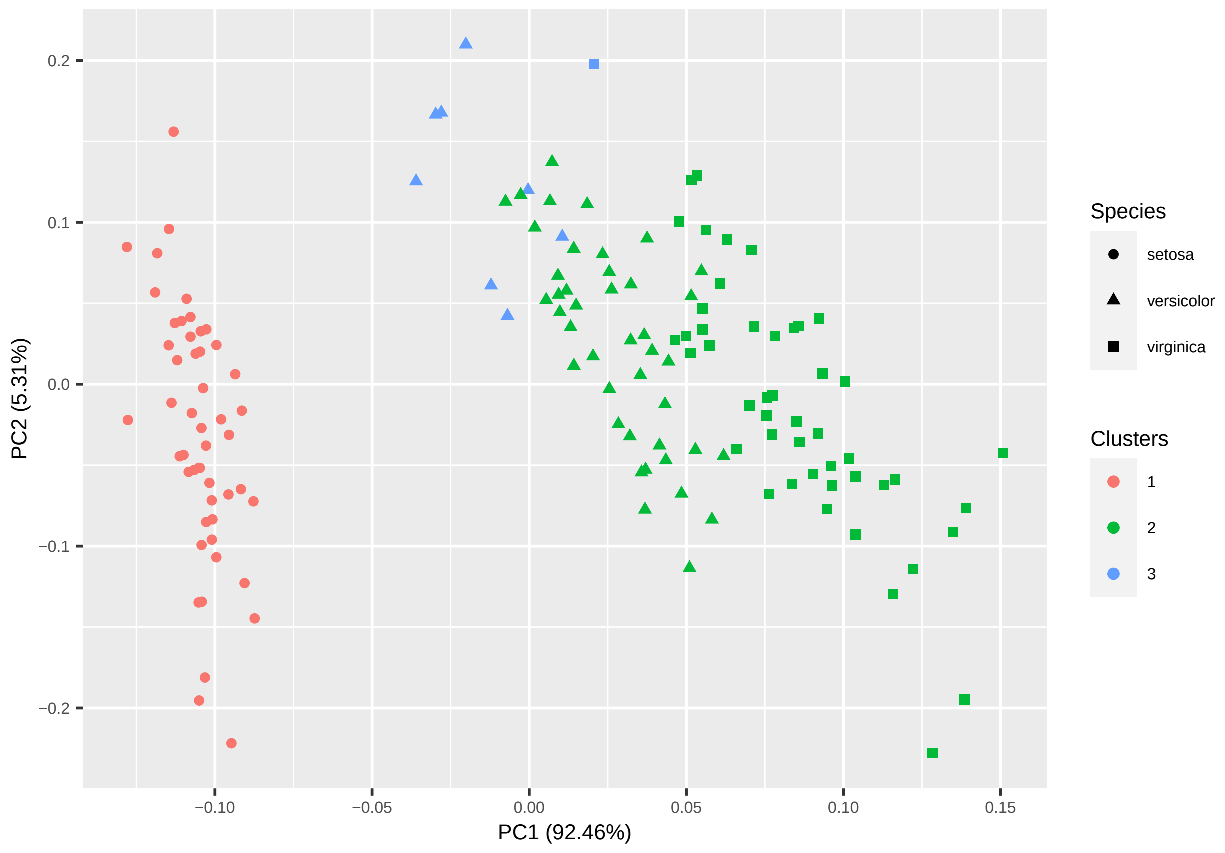
```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 2 3 3 2 3 2 3 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 3 2 2 2 2 2 2 2 3 2 2 2 2 2
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [149] 2 2
```

Now perform PCA to get 2D reduction of `iris` and colour by cluster.

```r
irisClusters = iris
irisClusters$Clusters = as.factor(clusters)

pca = prcomp(X, center=TRUE, retx=TRUE)

library(ggfortify)
```

```
## Loading required package: ggplot2
```

```r
autoplot(pca, data=irisClusters, colour="Clusters", shape="Species")  +
  theme(text=element_text(size=8))
```



Nearly right but quite a few are wrong (stress that this isn't an entirely valid metric—we're not really doing classification here)

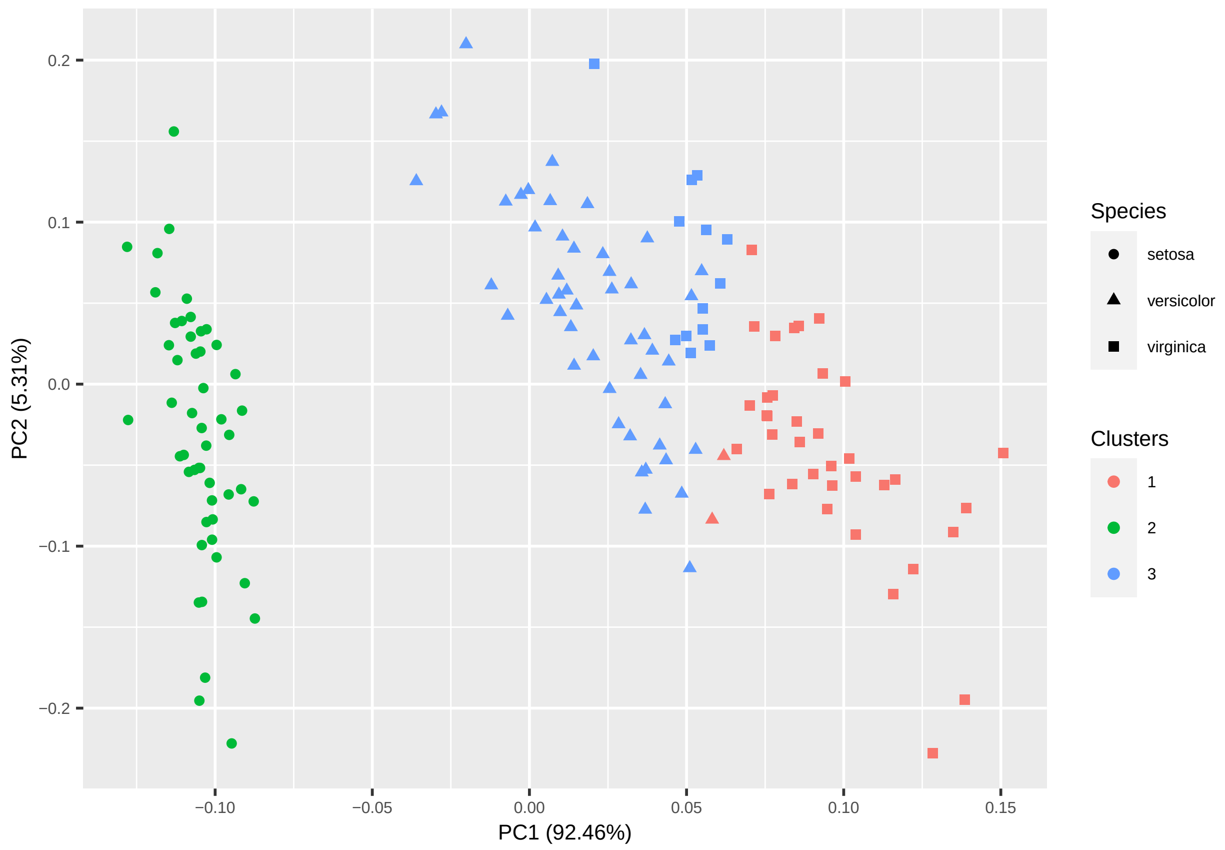
```
# Assuming cluster 1 is setosa, 2 is virginica and 3 is versicolor
sum((irisClusters$Cluster == 1 & irisClusters$Species == "setosa") |
    (irisClusters$Cluster == 2 & irisClusters$Species == "virginica") |
    (irisClusters$Cluster == 3 & irisClusters$Species == "versicolor")) / nrow(iris)
```

```
## [1] 0.7133333
```

K-Means Clustering

again use manhattan distance [bc. . . ?] again use K=3, we know there are 3 species

```
cl = kmeans(X, 3, algorithm="Lloyd")
irisClusters$Clusters = as.factor(cl$cluster)
autoplot(pca, data=irisClusters, colour="Clusters", shape="Species")  +
  theme(text=element_text(size=8))
```



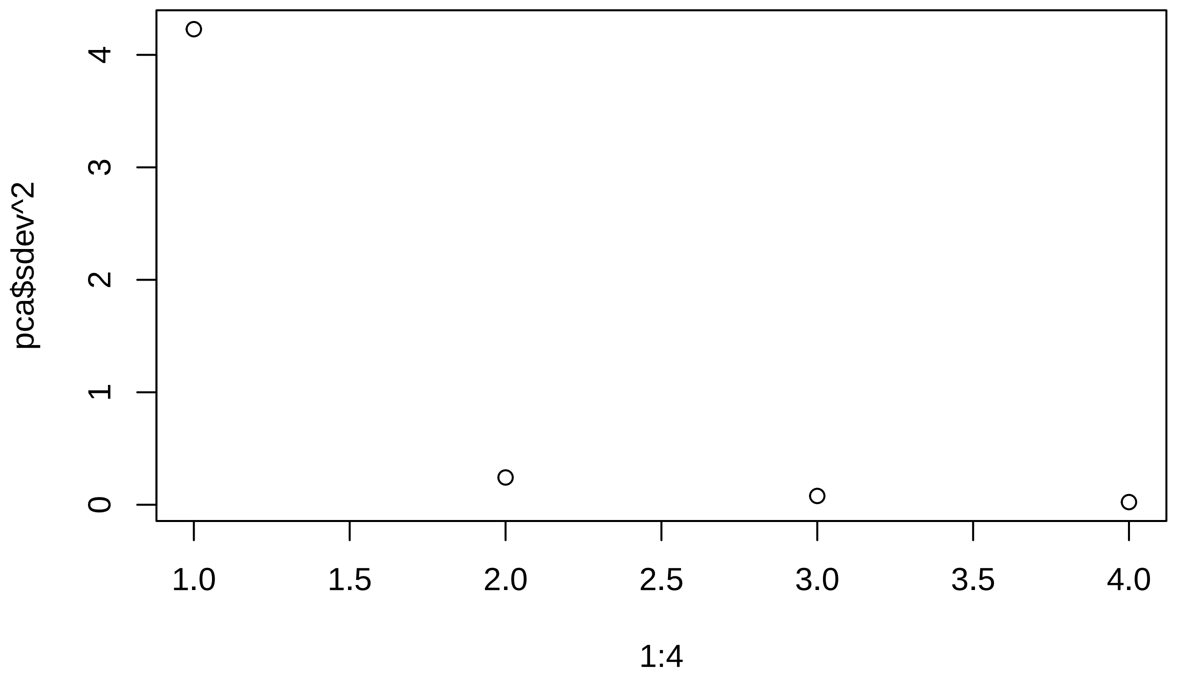
better than hierarchical

```
# Assuming cluster 1 is virginica, 2 is setosa and 3 is versicolor
sum((irisClusters$Cluster == 1 & irisClusters$Species == "virginica") |
    (irisClusters$Cluster == 2 & irisClusters$Species == "setosa") |
    (irisClusters$Cluster == 3 & irisClusters$Species == "versicolor")) / nrow(iris)
```

```
## [1] 0.8933333
```

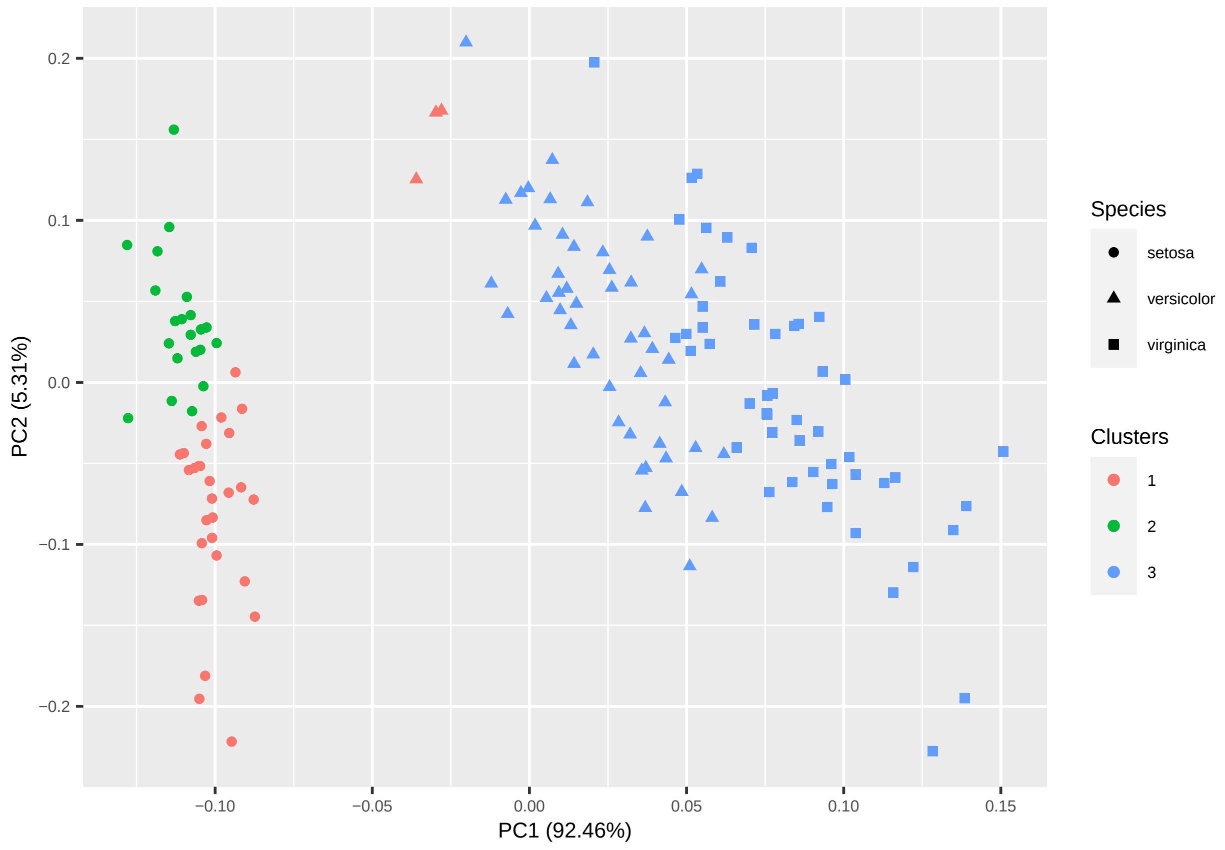
Low-Dimension K-Means Clustering Looking at a scree plot, choose $q = 2 < p = 4$. (more interesting comparison to be made w/ full-dim k-means just above if we can plot it in 2D (?))

```
plot(1:4, pca$sdev^2)
```



Then do K-means clustering again

```
cl = kmeans(pca$x[,1:2], 3, algorithm="Lloyd")
irisClusters$Clusters = as.factor(cl$cluster)
autoplot(pca, data=irisClusters, colour="Clusters", shape="Species")  +
  theme(text=element_text(size=8))
```

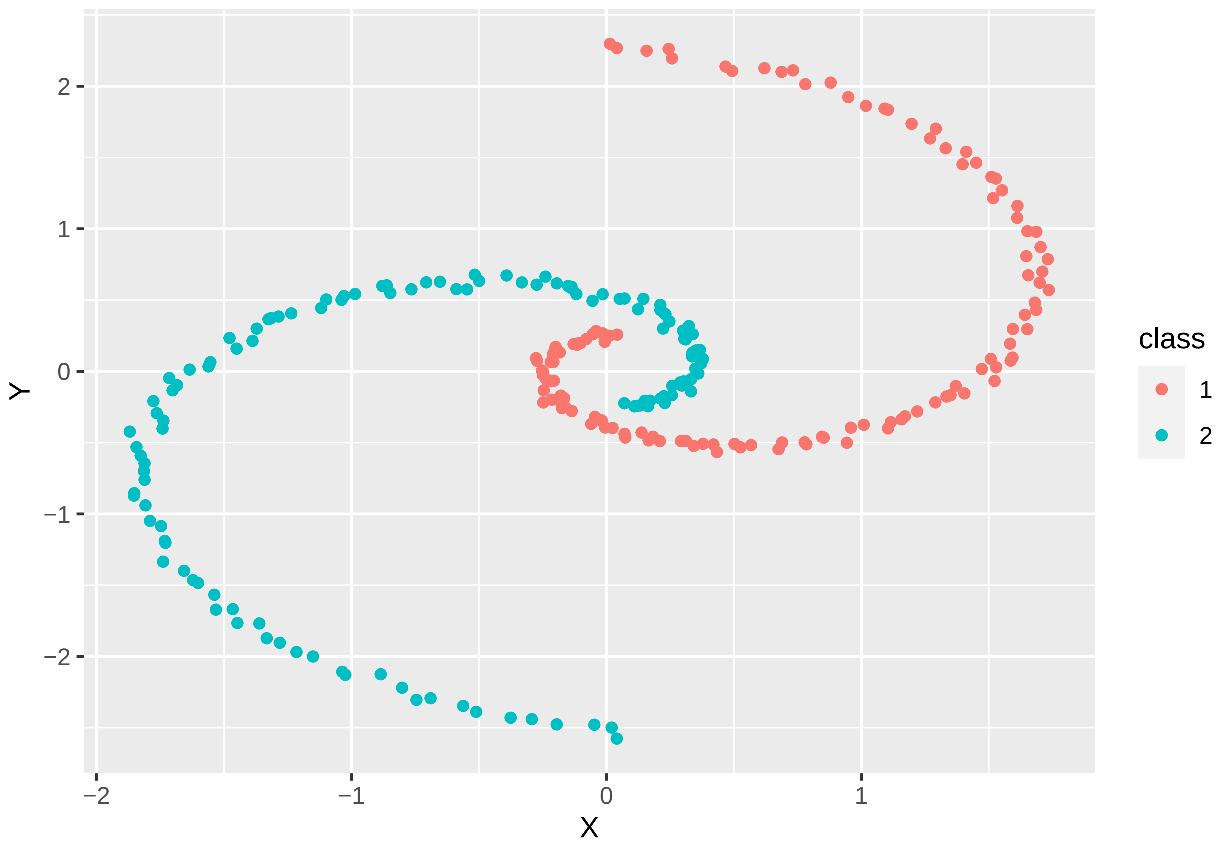


Way worse than both other results, get's confused bc the versicolor and virginica are so similar it puts them together into one cluster

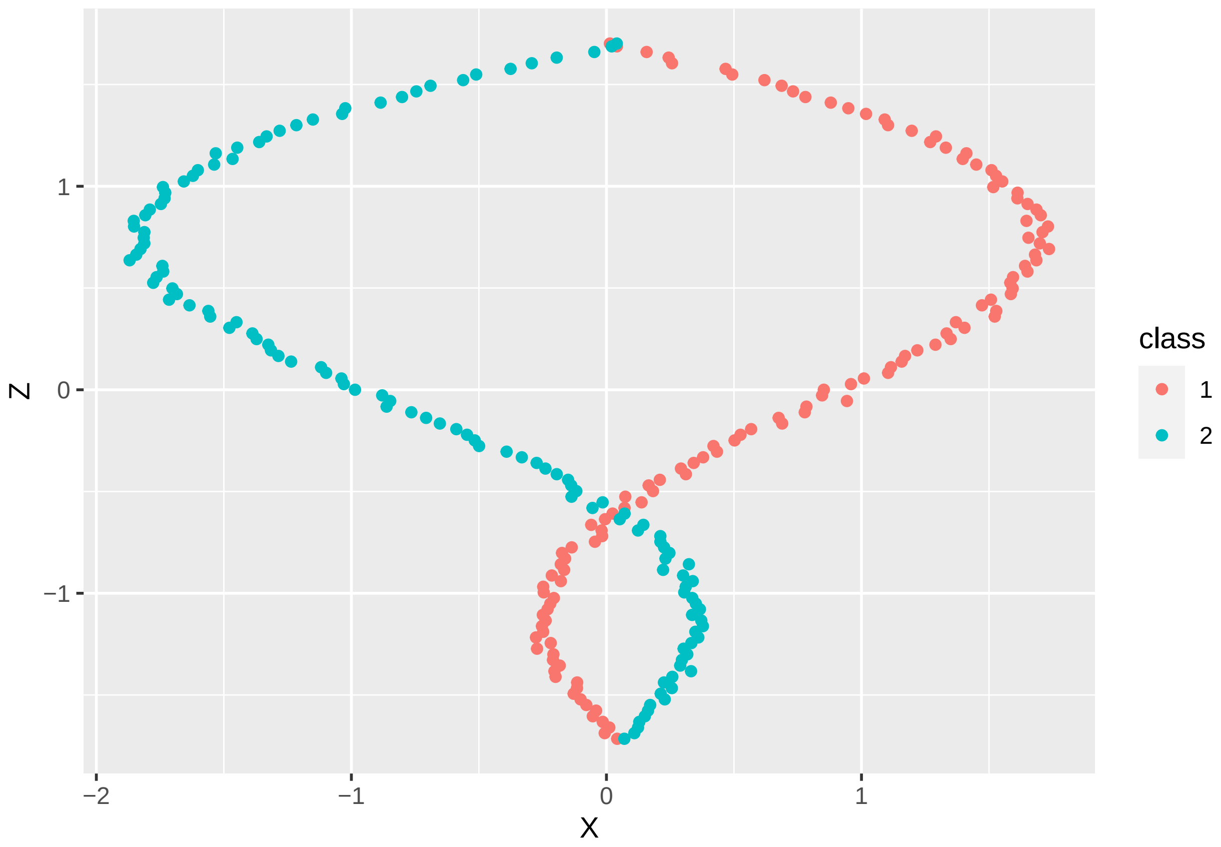
Task 2

Use spiral dataset from last week we'll be working with three-dimensional data in the shape of concentric spirals in the x-y plane moving along the z-axis, generated below.

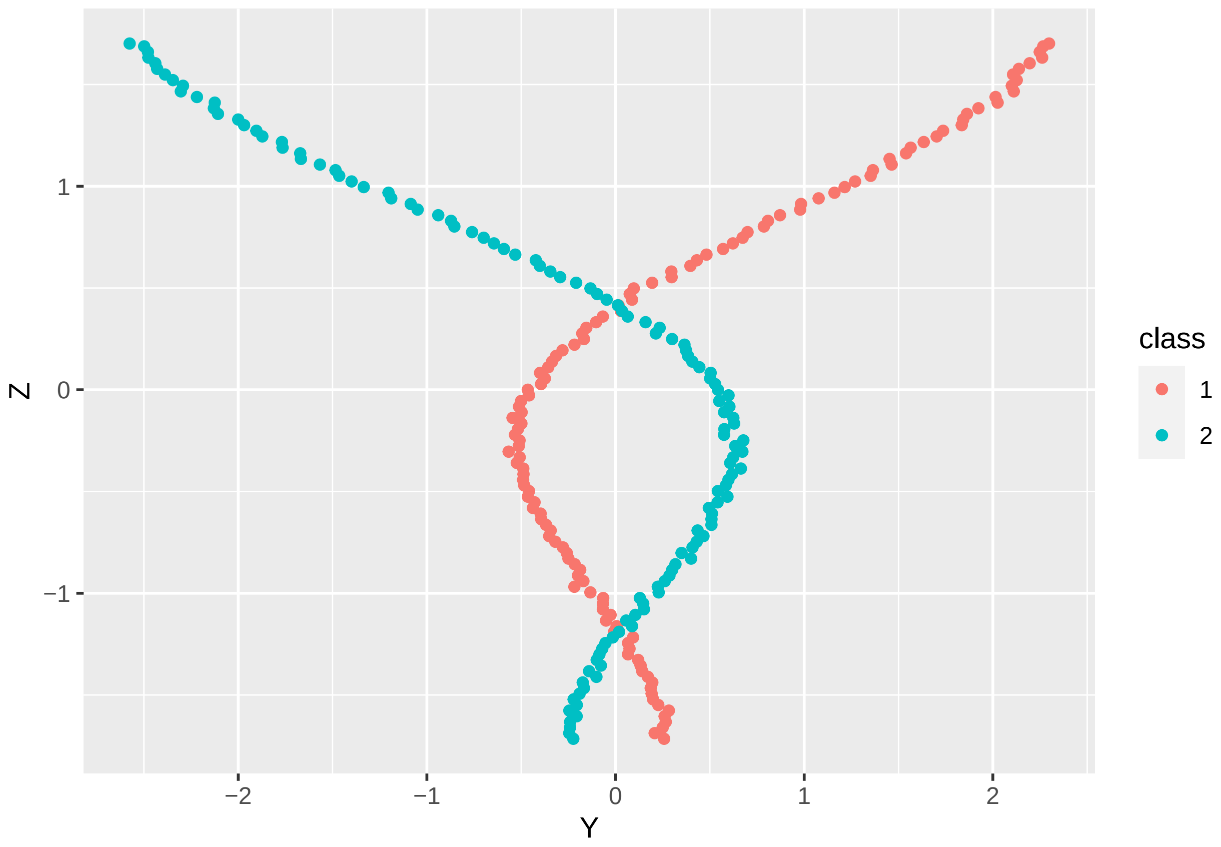
```r
n = 250  # Total number of data points

Xs = matrix(rep(0,n*3), nrow=n)

for (i in 1:n/2){
  for (class in 0:1){
    coords = c(cos(i*3*pi/(n)), sin(i*3*pi/(n))) * (class*2 -1) * ((i*(8+class))/6)
    coords = coords + rnorm(2, 0, c(2,2))   # Add some noise
    coords = coords + c(0, -n/12) * (class*2 -1)  # shift the two spirals so that they don't join up at
    Xs[class*n/2 + i,] = c(coords, i)  # add the z=i dimension to moves spirals along z axis
  }
}
Xs = scale(Xs)

# Put data into a dataframe
data = as.data.frame(cbind(Xs, c(rep(0,n/2), rep(1,n/2))))
colnames(data) = c("X", "Y", "Z", "class")
data[,"class"] = as.factor(data[,"class"] + 1)

# Plot
library(ggplot2)
ggplot(data = data, aes(X, Y, color = class)) +
  geom_point()
```



```
ggplot(data = data, aes(X, Z, color = class)) +
  geom_point()
```



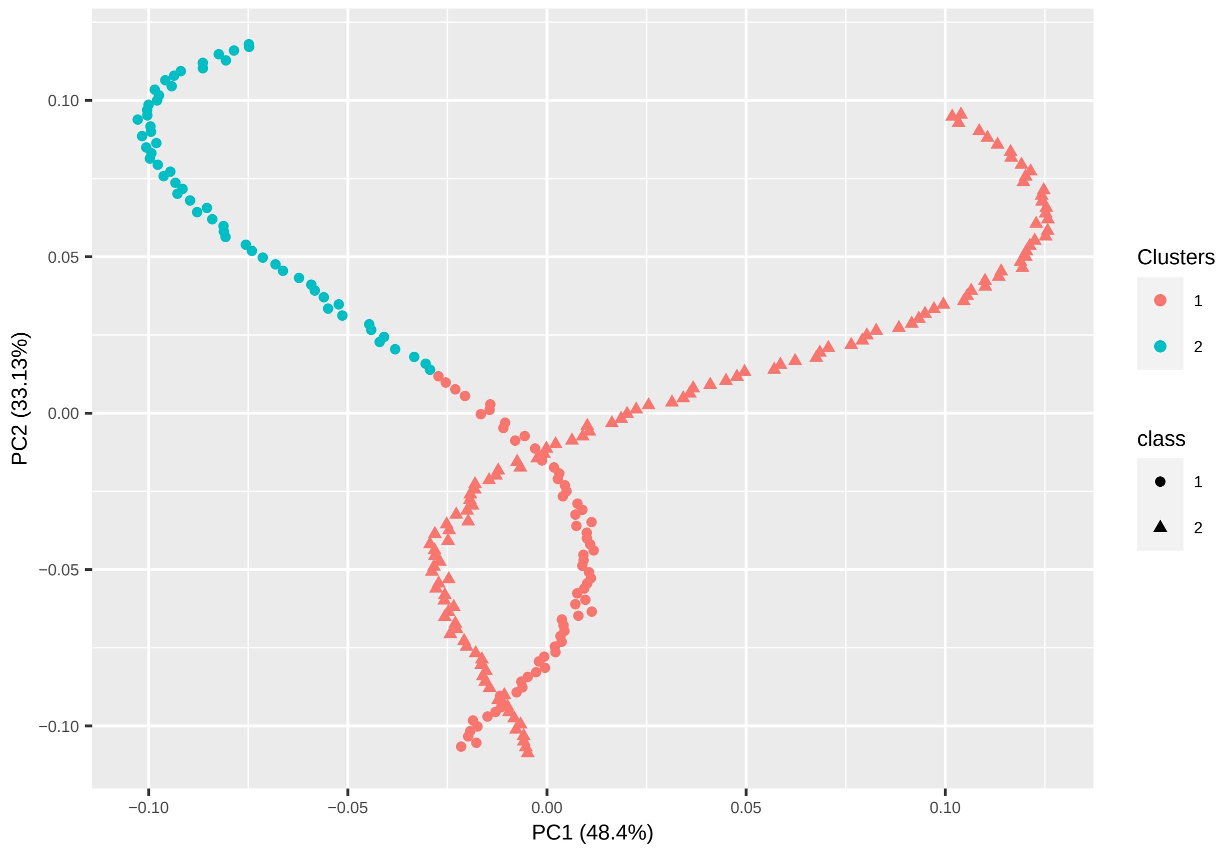
```
ggplot(data = data, aes(Y, Z, color = class)) +
  geom_point()
```



K-Means Clustering

use $K = 2$ b/c we know there are two classes in the data

```
pca = prcomp(Xs, center=TRUE, retx=TRUE)

cl = kmeans(Xs, 2, algorithm="Lloyd")
dataClusters = data
dataClusters$Clusters = as.factor(cl$cluster)
autoplot(pca, data=dataClusters, colour="Clusters", shape="class")  +
  theme(text=element_text(size=8))
```



Didn't really work

```
# Assuming cluster 1 is class 2, cluster 2 is class 1 (most generous interpretation)
sum((dataClusters$Cluster == 1 & dataClusters$class == 2) |
    (dataClusters$Cluster == 2 & dataClusters$class == 1)) / n
```

```
## [1] 0.732
```

Spectral Clustering

We'll use Manhattan distance $d_{il} = ||x_i^0 - x_l^0||$ with similarity $s_{il} = \exp(-\sigma d_{il}^2)$, i.e. use the Gaussian kernel $k(x, x') = \exp(-\sigma||x - x'||^2)$ for $\sigma = 0.5, 2.315329, 5$ where that middle one comes from median trick.

```
median(dist(Xs, method="euclidean"))
```

```
## [1] 2.282372
```

Using the same distance but with similarity $s_{il} = \exp(-\sigma d_{il})$ gives us the Laplacian kernel: $k(x, x') = \exp(-\sigma||x - x'||^2)$ which we'll also test with $\sigma = 0.5, 3.592143, 5$ where that middle one comes from median trick for manhattan distance rather than Euclidean.

```
median(dist(Xs, method="manhattan"))
```

```
## [1] 3.388194
```

use eigengap heurisitic to choose M clusters via `estimate_k` from `Spectrum` (which requires a similarity/Gram matrix—we use `kernelMatrix` from `kernlab`)

```
library(kernlab)
```

```
##
```

```
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

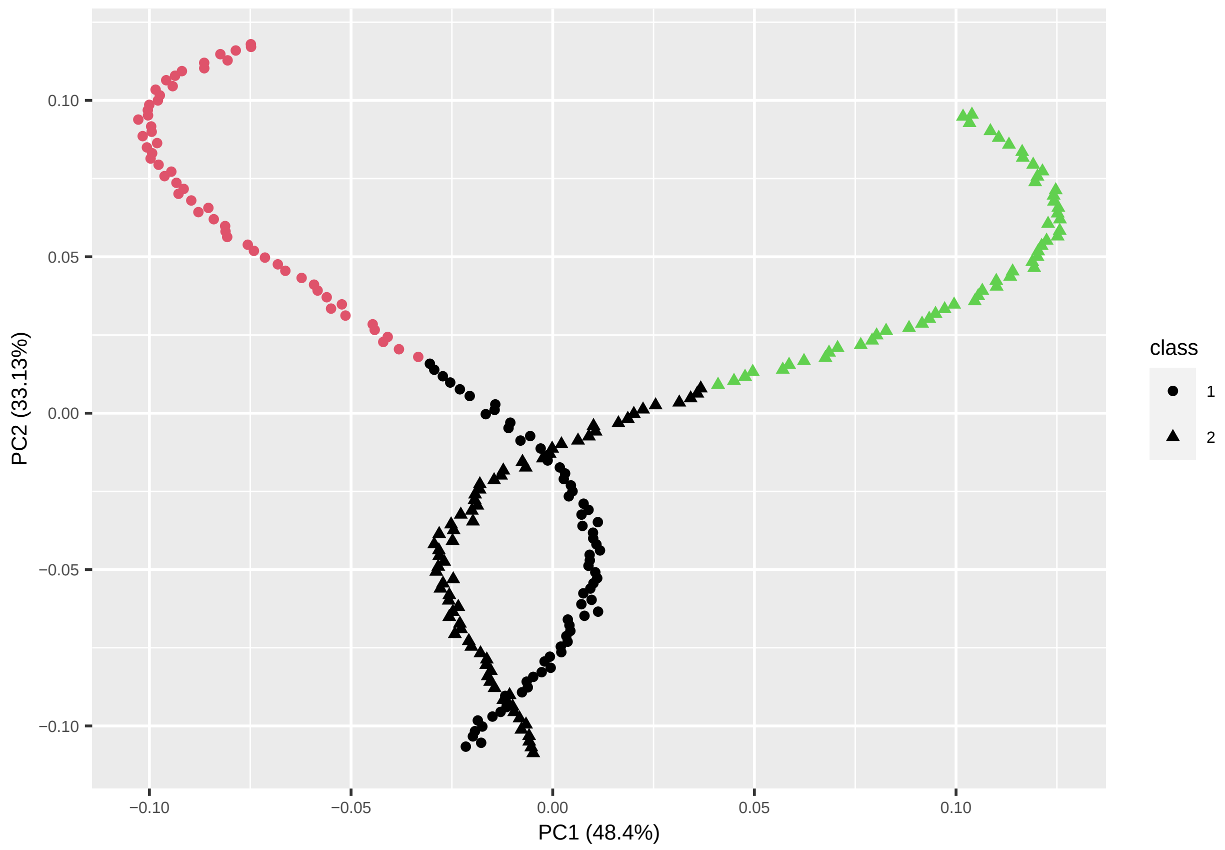
```
library(Spectrum)

eigenGap <- function(eigVals){
  diffs = rep(0, length(eigVals)-2)
  for (i in 1:(length(eigVals)-2)){
    diffs[i] = eigVals[i+1] - eigVals[i]
  }
  return(which.max(diffs) + 1)
}

rbf_sigmas = c(0.5, 2.315329, 5)
lap_sigmas = c(0.5, 3.592143, 5)

rbf_Ms = rep(0,3)
lap_Ms = rep(0,3)

for (i in 1:3){
  rbfkern = rbfdot(rbf_sigmas[i])
  lapkern = laplacedot(lap_sigmas[i])
  rbf_Ms[i] = eigenGap(estimate_k(kernelMatrix(rbfkern, Xs), showplots=FALSE)$z)
  lap_Ms[i] = eigenGap(estimate_k(kernelMatrix(lapkern, Xs), showplots=FALSE)$z)
}
```

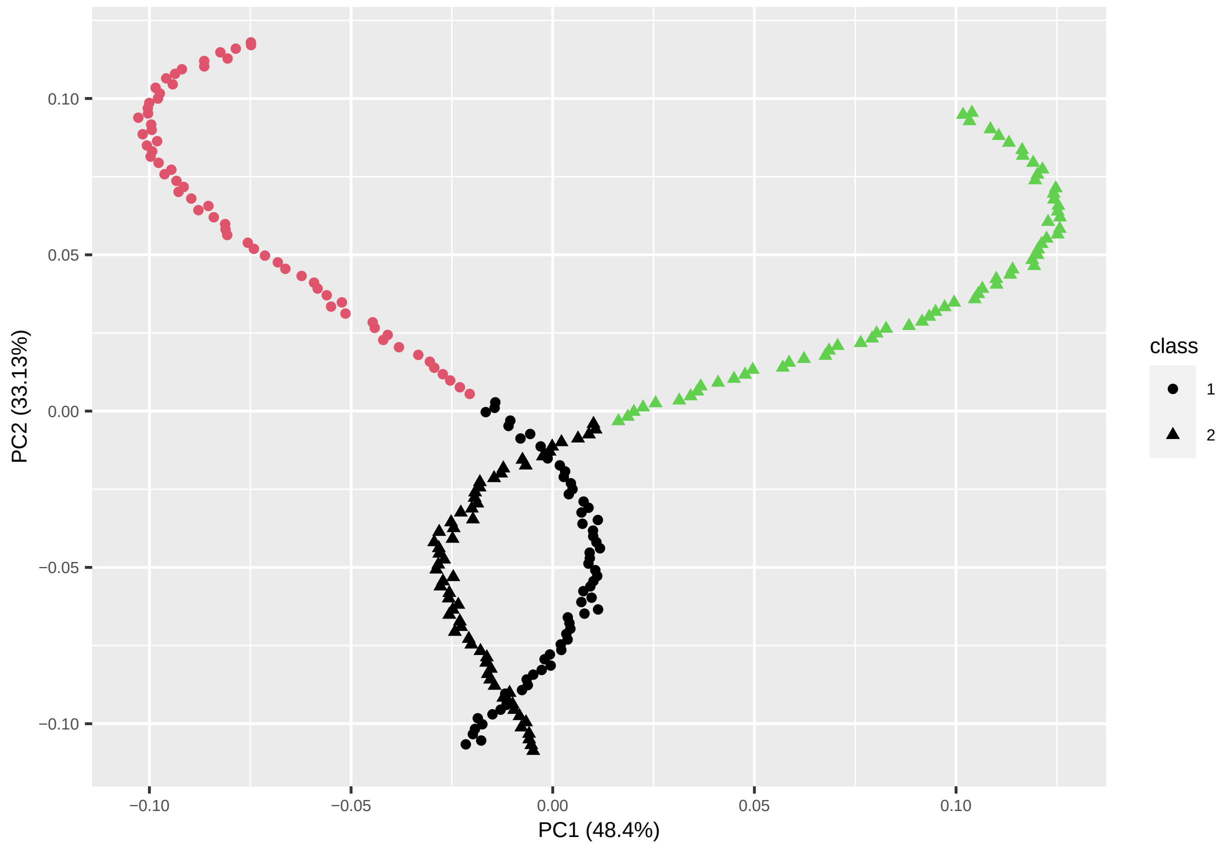
```
## egap optimal K: 3

## egap optimal K: 3

## egap optimal K: 5
## egap optimal K: 5

## egap optimal K: 9
## egap optimal K: 9
```

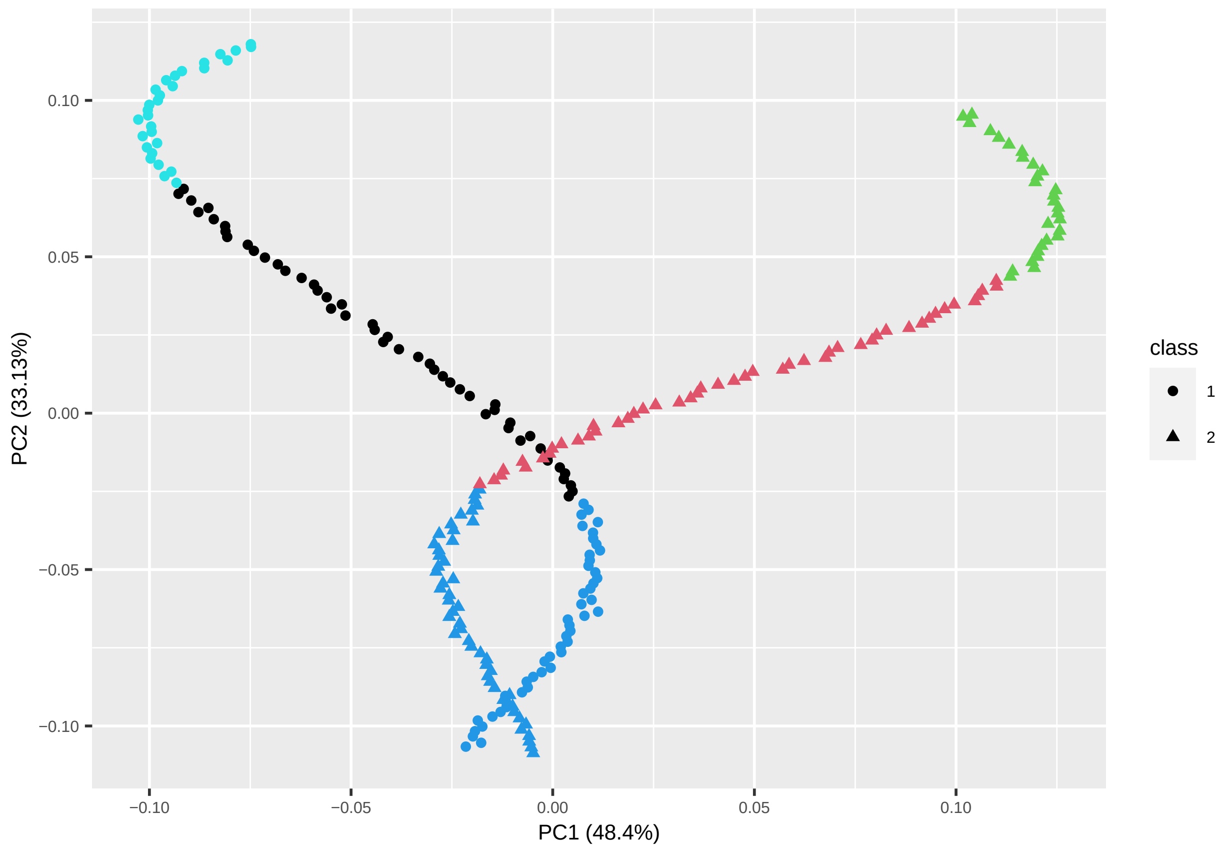
```
rbf_Ms = c(3,3,5)
lap_Ms = c(5,9,9)
```

look at the rbf results

```
for (i in 1:3){
  sc = specc(Xs, centers=rbf_Ms[i], kernel="rbfdot", kpar=list(sigma=rbf_sigmas[i]))

  g = autoplot(pca, data=dataClusters, colour=sc, shape="class")  +
    theme(text=element_text(size=8))
  print(g)
}
```

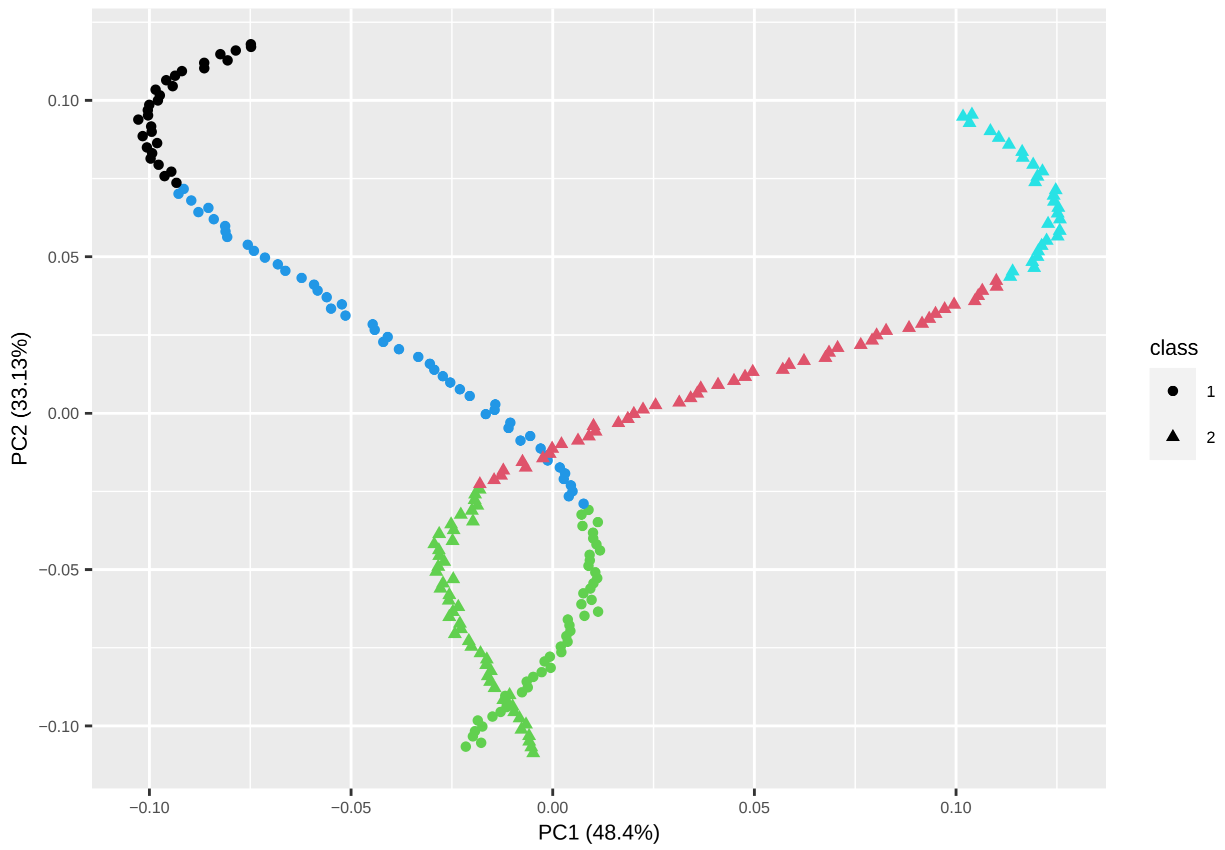

PC2 (33.13%)
0.10
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
0.10
PC1 (48.4%)
class
1
2


PC2 (33.13%)
0.10
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
0.10
PC1 (48.4%)
class
1
2

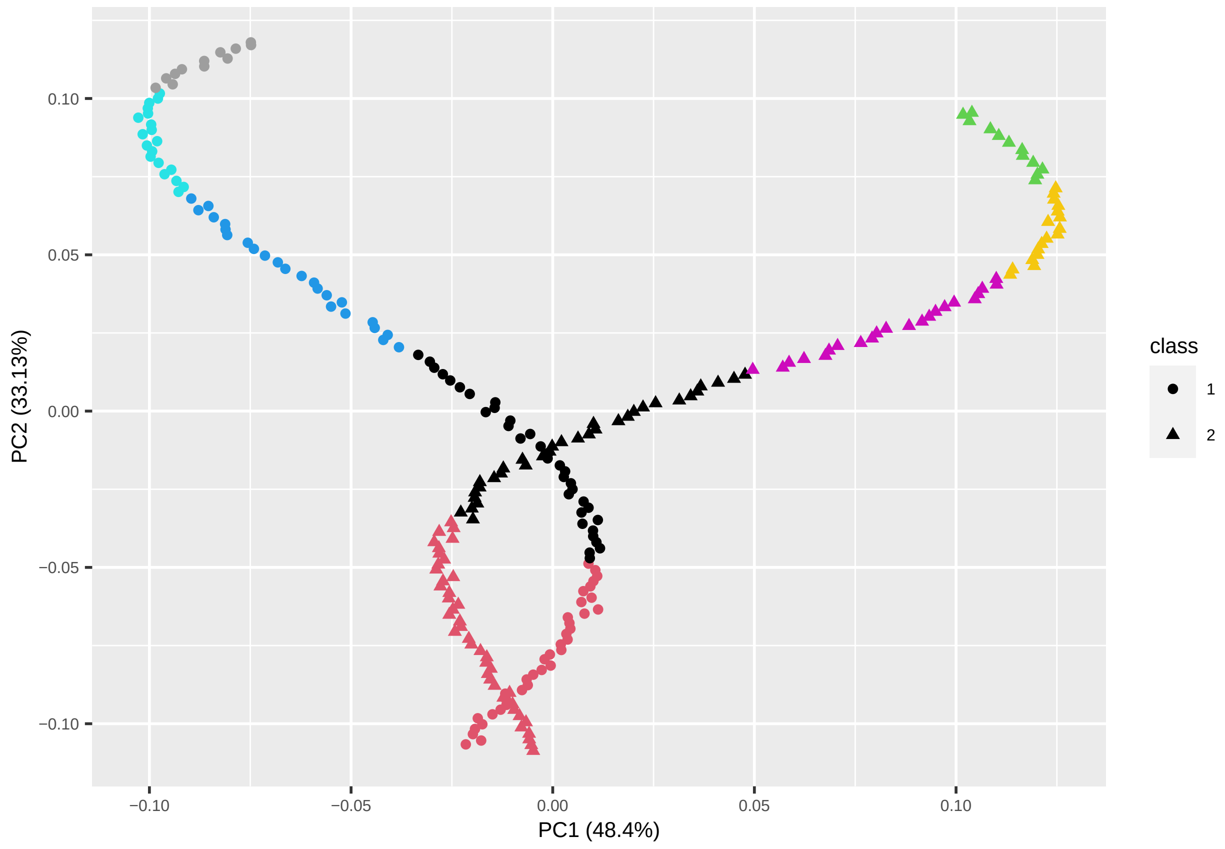


ooh—M=3 seems good—now look at the laplace results

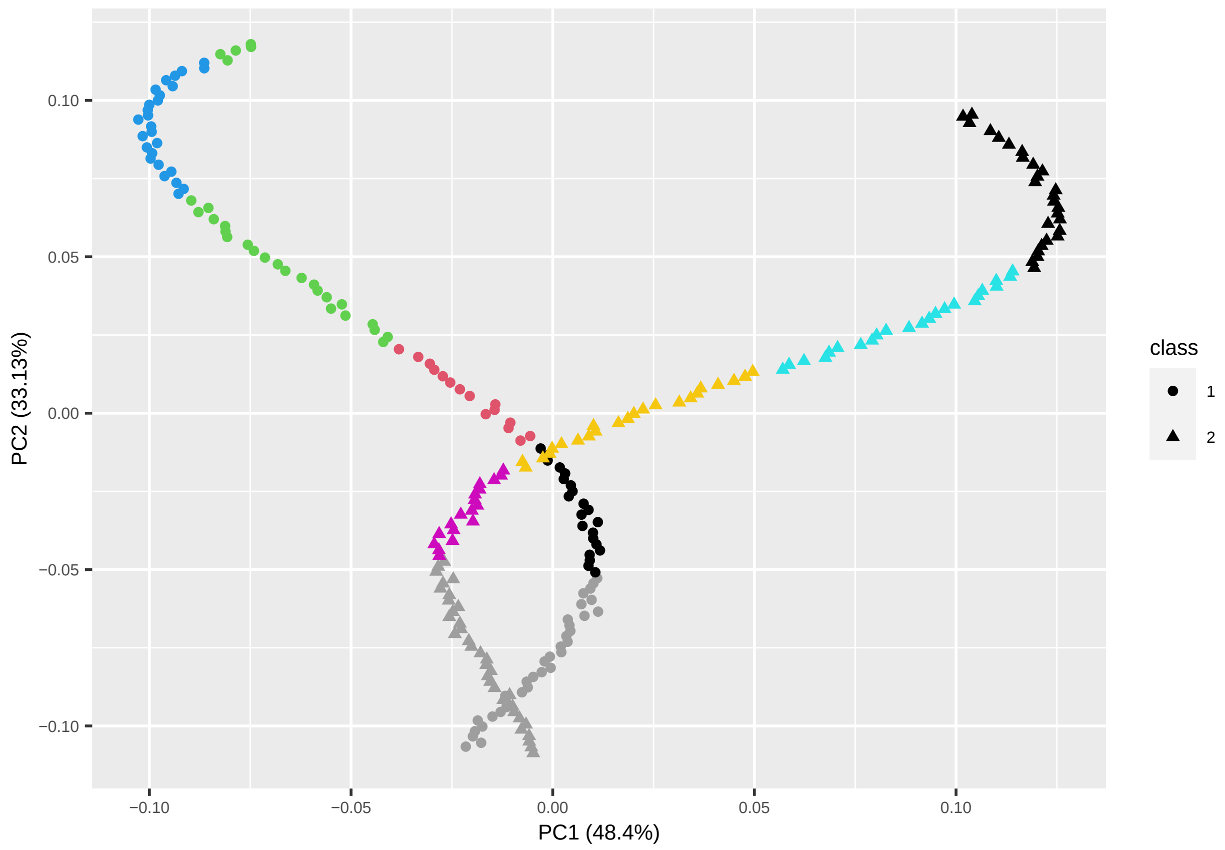
```
for (i in 1:3){
  sc = specc(Xs, centers=lap_Ms[i], kernel="laplacedot", kpar=list(sigma=lap_sigmas[i]))

  g = autoplot(pca, data=dataClusters, colour=sc, shape="class")  +
    theme(text=element_text(size=8))
  print(g)
}
```



PC2 (33.13%)
0.10
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
0.10
PC1 (48.4%)
class
1
2



PC2 (33.13%)
0.10
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
0.10
PC1 (48.4%)
class
1
2



quite good, not separating in the way we'd want for a classification task w/ 2 classes, (particularly b/c we're using higher Ms now for laplace), but the $\sigma = 0.5$ actually seems quite nice and the clusters make sense intuitively

Task 3

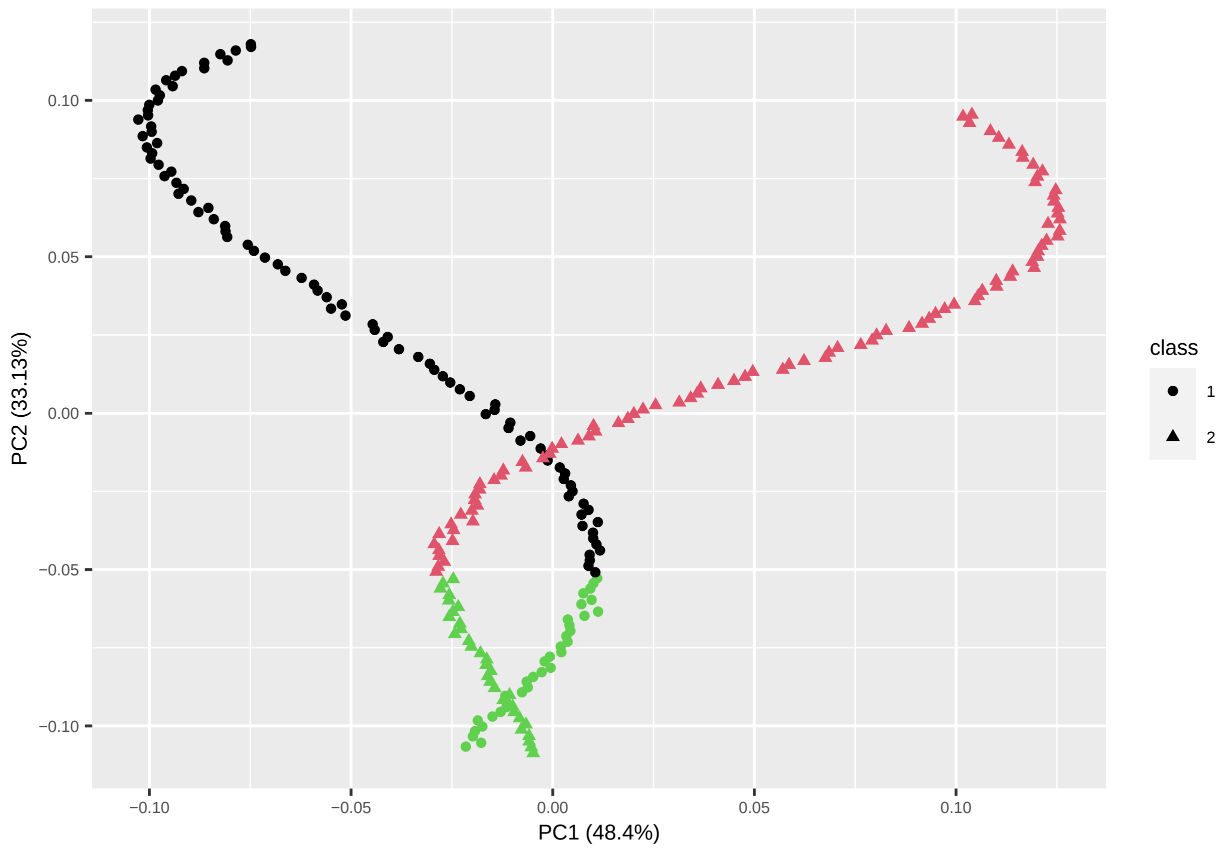
pt 2

use `kkmeans` from `klic` compare w/ best resutls from previous section: - $K = 3$ w/ rbf median trick - $K = 5$ w/ Laplace $\sigma = 0.5$ - $K = 2$ w/ both rbf median trick and Laplace $\sigma = 0.5$ (best kernels so far, will allow us to compare results against regular K-means in Task 2)

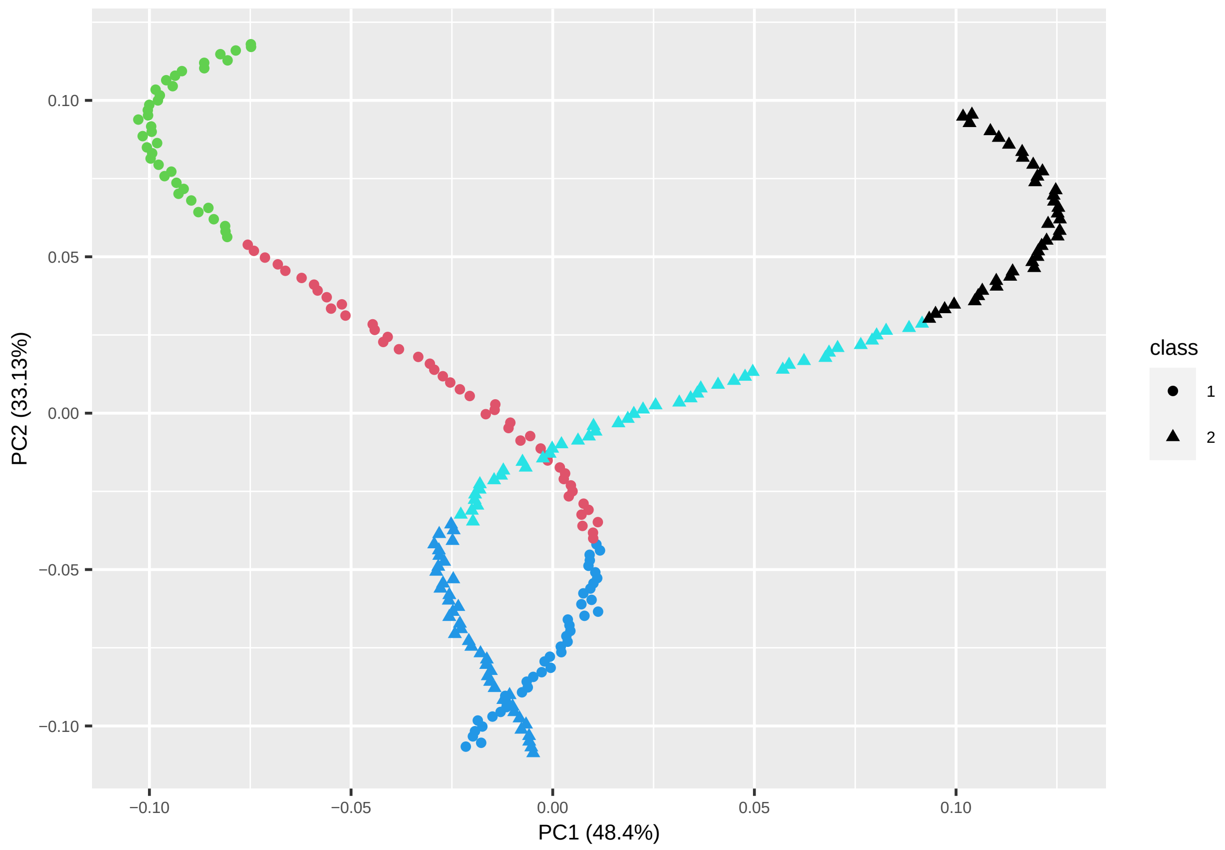
```
library(klic)
```

```
##
## Attaching package: 'klic'

## The following object is masked from 'package:kernlab':
##
##     kkmeans
```

```
rbfkern = rbfdot(rbf_sigmas[2])
lapkern = laplacedot(lap_sigmas[1])

rbfKernMatrix = kernelMatrix(rbfkern, Xs)
lapKernMatrix = kernelMatrix(lapkern, Xs)

clusters = kkmeans(rbfKernMatrix, list("cluster_count"=3))$clustering
autoplot(pca, data=dataClusters, colour=clusters, shape="class")  +
    theme(text=element_text(size=8))
```



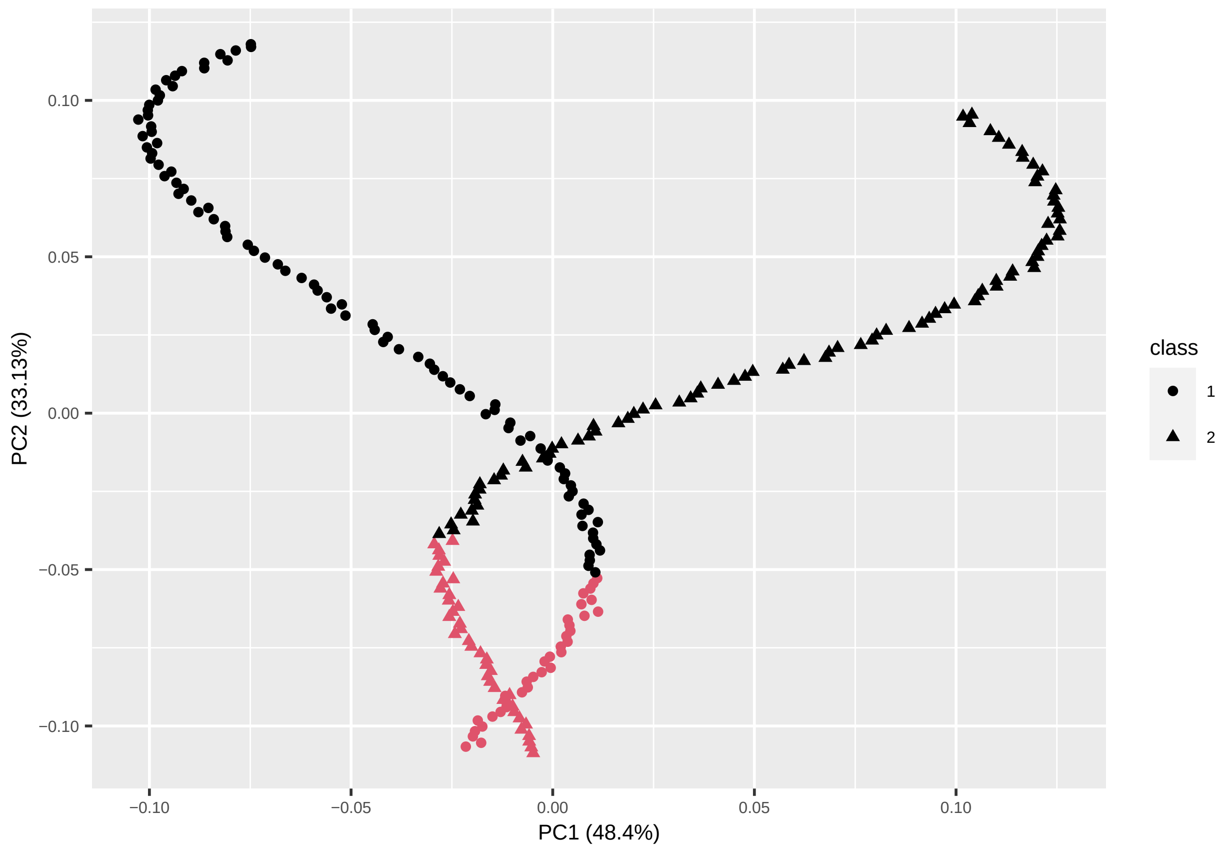
```
clusters = kkmeans(lapKernMatrix, list("cluster_count"=5))$clustering
autoplot(pca, data=dataClusters, colour=clusters, shape="class")  +
    theme(text=element_text(size=8))
```



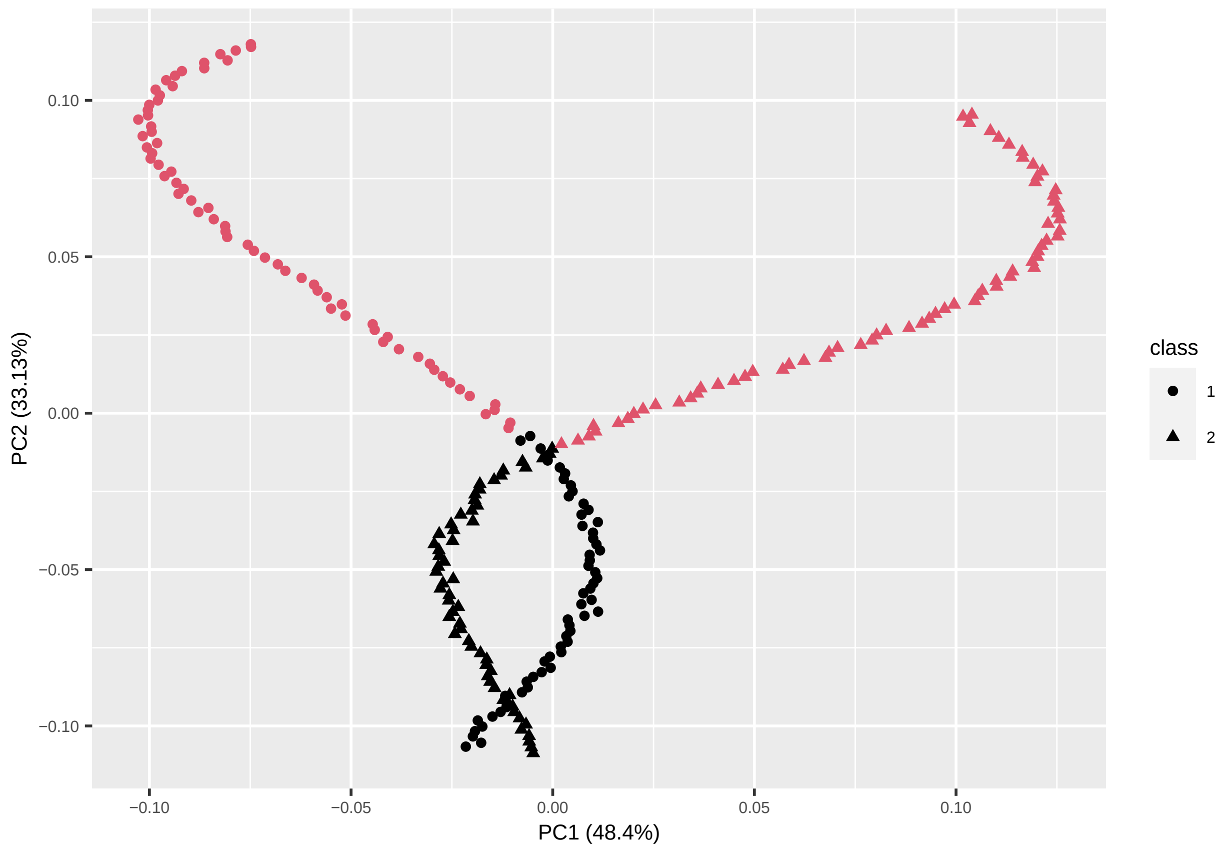
the above: first is okay, second is kinda crap

how about for k=2?

```r
clusters = kkmeans(rbfKernMatrix, list("cluster_count"=2))$clustering
autoplot(pca, data=dataClusters, colour=clusters, shape="class")  +
    theme(text=element_text(size=8))
```



```
clusters = kkmeans(lapKernMatrix, list("cluster_count"=2))$clustering
autoplot(pca, data=dataClusters, colour=clusters, shape="class")  +
    theme(text=element_text(size=8))
```



not correct in terms of the dataset's classes but makes more sense than regular k-means