Cluster Analysis

Sam Bowyer

2023-02-17

Task 1

For this task we will work with the `iris` dataset, containing information about samples of three species of iris flower.

```
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])[1] # 3 types of flower, so K=3 clusters would be interesting to analyse
```

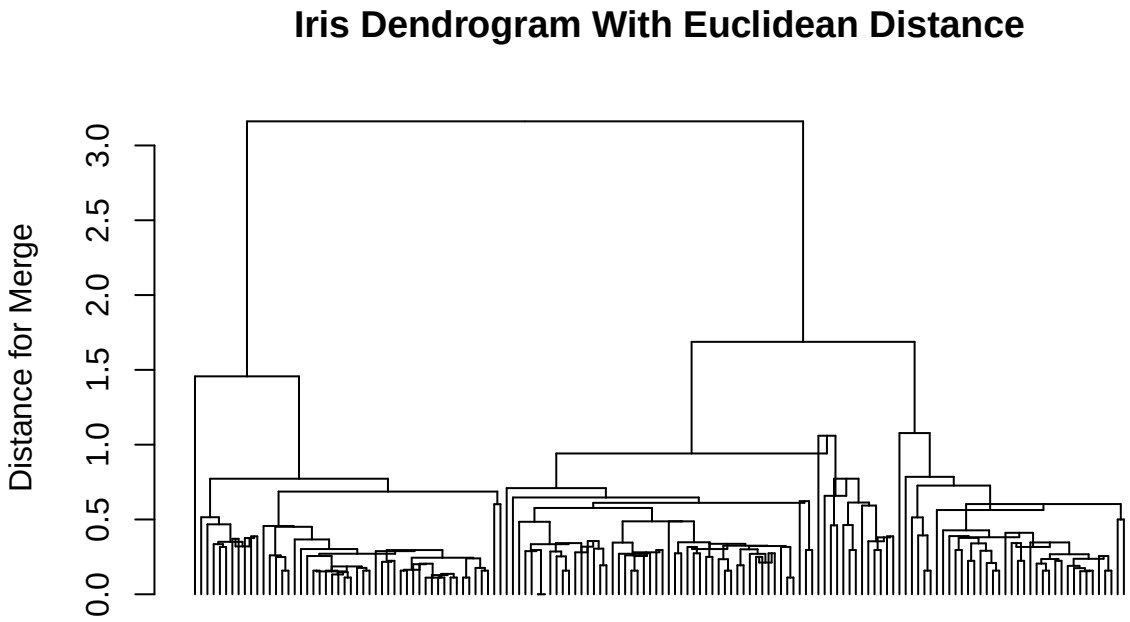
```
## [1] setosa
## Levels: setosa versicolor virginica
```

Agglomerative Clustering

If we use Euclidean distance for agglomerative clustering on this dataset, which seems reasonable since the data is all real valued and continuous, looking at the dendrogram suggests that we should use $K = 2$ clusters, as this is the number of clusters for which there is the biggest gap in distance before the next merge (into a single cluster for the entire dataset) occurs. Note here that we are using the `median` agglomeration method (otherwise knwon as WPGMA—Weighted Pair Group Method with Arithmetic Mean): if clusters C_i and C_j have previously been merged together to create cluster $C_{i\cup j}$, the distance between $C_{i\cup j}$ and another cluster C_k is given by the mean of the average distances between members of C_k and C_i and C_k and C_j, i.e.:

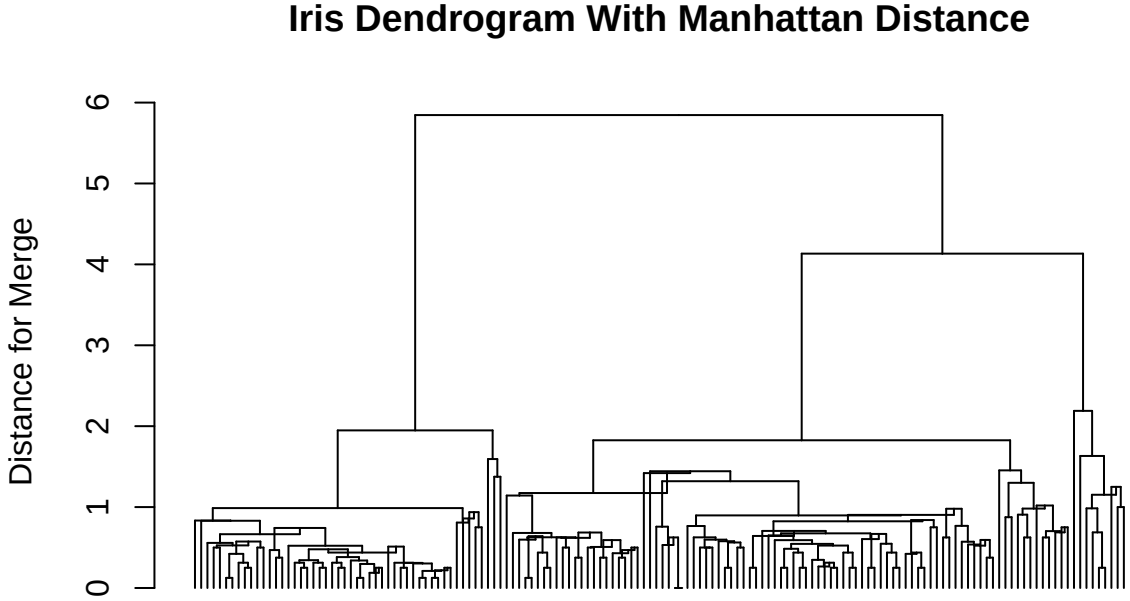
$$d_{(i\cup j),k} = \frac{d_{i,k} + d_{j,k}}{2}.$$

```
hc = hclust(dist(iris, "euclidean"), "median")
hcp = as.dendrogram(hc)
plot(hcp, hang=-1, leaflab="none", main="Iris Dendrogram With Euclidean Distance",
     ylab="Distance for Merge")  # negative hang just makes the labels hang down from 0
```



However, we know that `iris` contains three classes, and whilst we are not performing a classification task in this portfolio, it would be more interesting to analyse the clustering results if we are looking for $K = 3$ clusters—particularly when we move onto K-means clustering in which we have to choose K entirely by ourselves. So, in order to get more a more interesting comparison between agglomerative clustering and K-means clustering on this dataset, we consider a different distance: Manhattan (i.e. L^1) distance, where the distance between two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^p$ is given by $\sum_{q=1}^{p} |\mathbf{u}^{(q)} - \mathbf{v}^{(q)}|$.

```
hc = hclust(dist(iris, "manhattan"), "median")
hcp = as.dendrogram(hc)
plot(hcp, hang=-1, leaflab="none", main="Iris Dendrogram With Manhattan Distance",
     ylab="Distance for Merge")
```



Using the Manhattan distance leads to a dendrogram that suggests we take $K = 3$ clusters, which is what we were hoping for.

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

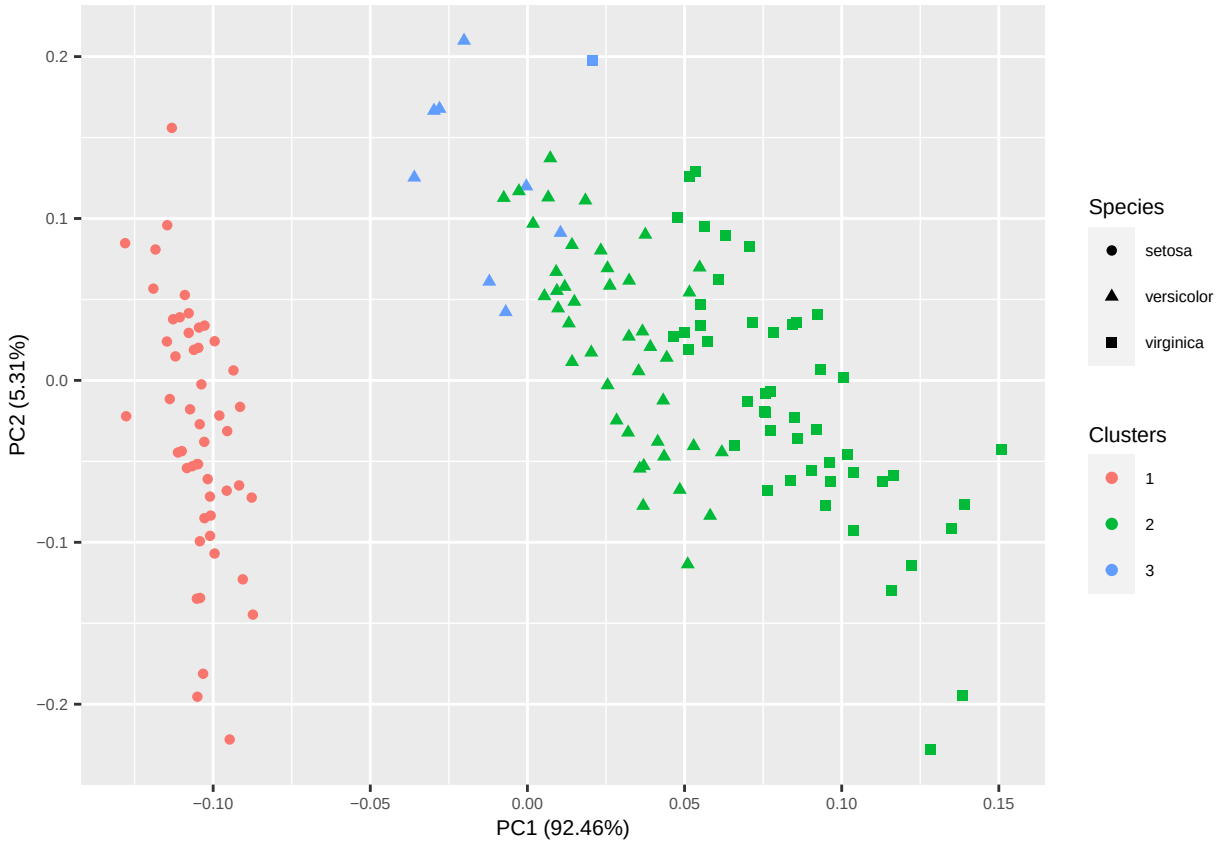
```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 2 3 3 2 3 2 3 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 3 2 2 2 2 2 2 3 2 2 2 2
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2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

We may now visualise the clusters by taking a 2D reduction of the dataset. Note that we are colouring the datapoints by their assigned cluster whilst the datapoint's shape is determined by the species (i.e. its class in the dataset).

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

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

library(ggfortify)
```

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

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



Clearly the algorithm has successfully identified the cluster of setsosa irises (the group on the left with red circles), however, the other two classes are harder to separate (this is particularly obvious when are we looking at a 2D reduction of the data), leading to a very large second (green) cluster and a small and somewhat sparsely distributed third (blue) cluster. Although this is not a classification task, we can see that—taking

the most generous interpretation of the clusters, i.e. that cluster two represents the virginica irises and cluster three the versicolors—the clustering has separated the classes of data with about 71% accuracy.

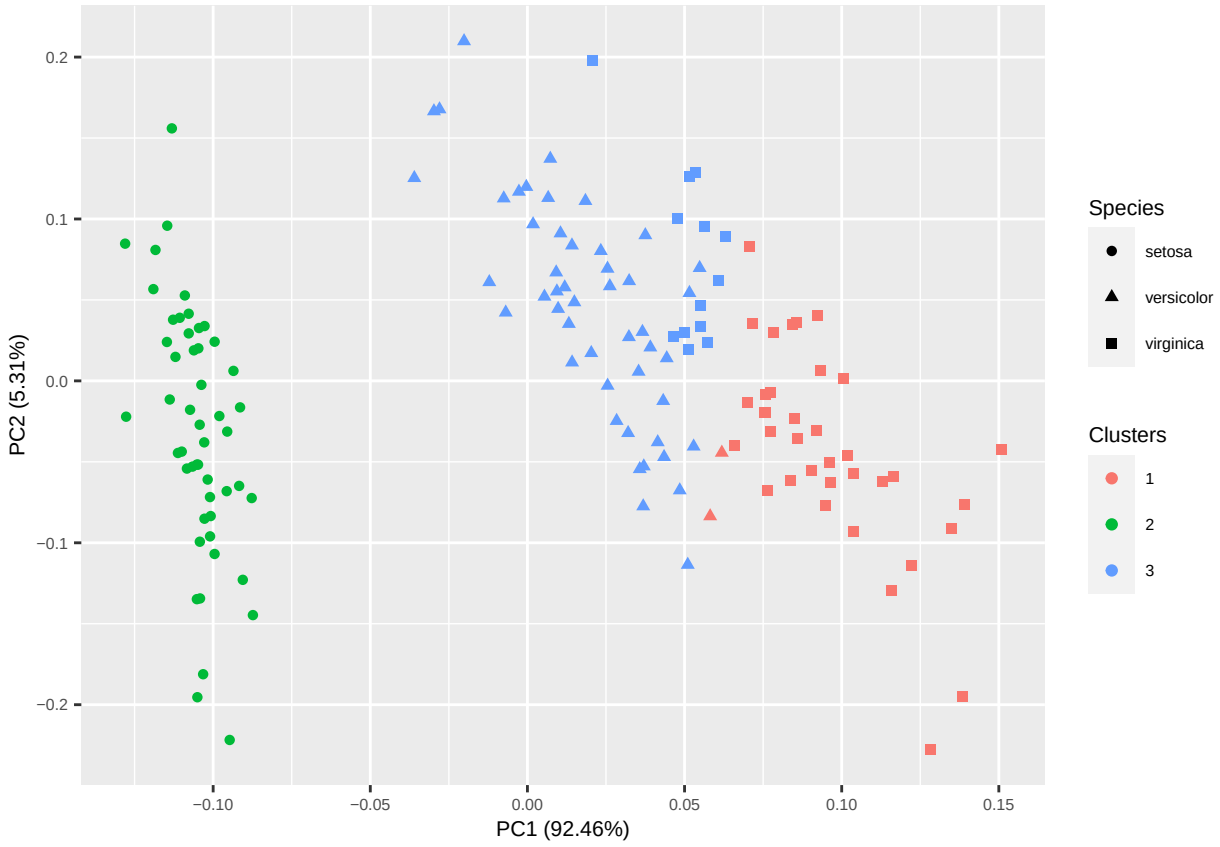
```
# Assuming cluster 1 is setosa, 2 is virginica and 3 is versicolor
sum((irisClusters$Cluster == 1 & irisClusters$Species == "setosa") |
    (irisClusters$Cluster == 2 & irisClusters$Species == "virginica") |
    (irisClusters$Cluster == 3 & irisClusters$Species == "versicolor")) / nrow(iris)
```

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

K-Means Clustering

Carrying on from the agglomerative clustering above, we again look for $K = 3$ clusters, however, this time the algorithm will be using Euclidean distance as it (by design) works with the squared errors from centroids of each cluster.

```
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))
```



The results seem much better than those obtained by agglomerative clustering: the cluster of setosas on the left is maintained whilst we see a better separation of the versicolor and virginica clusters on the right. Overall, this has separated the classes with around 89% accuracy (again, choosing the most generous interpretation of the clusters: setosas in cluster 1, virginicas in cluster 2 and versicolors in cluster 3).

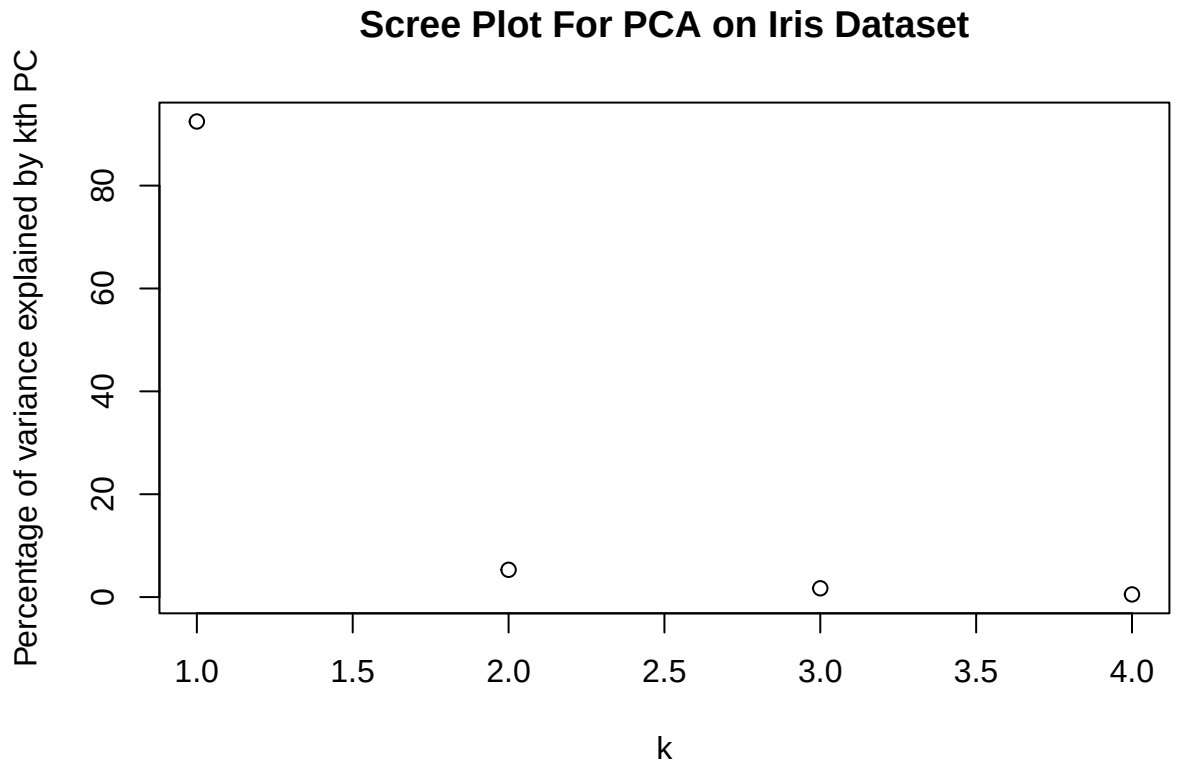
```
# Assuming cluster 1 is virginica, 2 is setosa and 3 is versicolor
sum((irisClusters$Cluster == 1 & irisClusters$Species == "virginica") |
    (irisClusters$Cluster == 2 & irisClusters$Species == "setosa") |
```

```
    (irisClusters$Cluster == 3 & irisClusters$Species == "versicolor")) / nrow(iris)
```

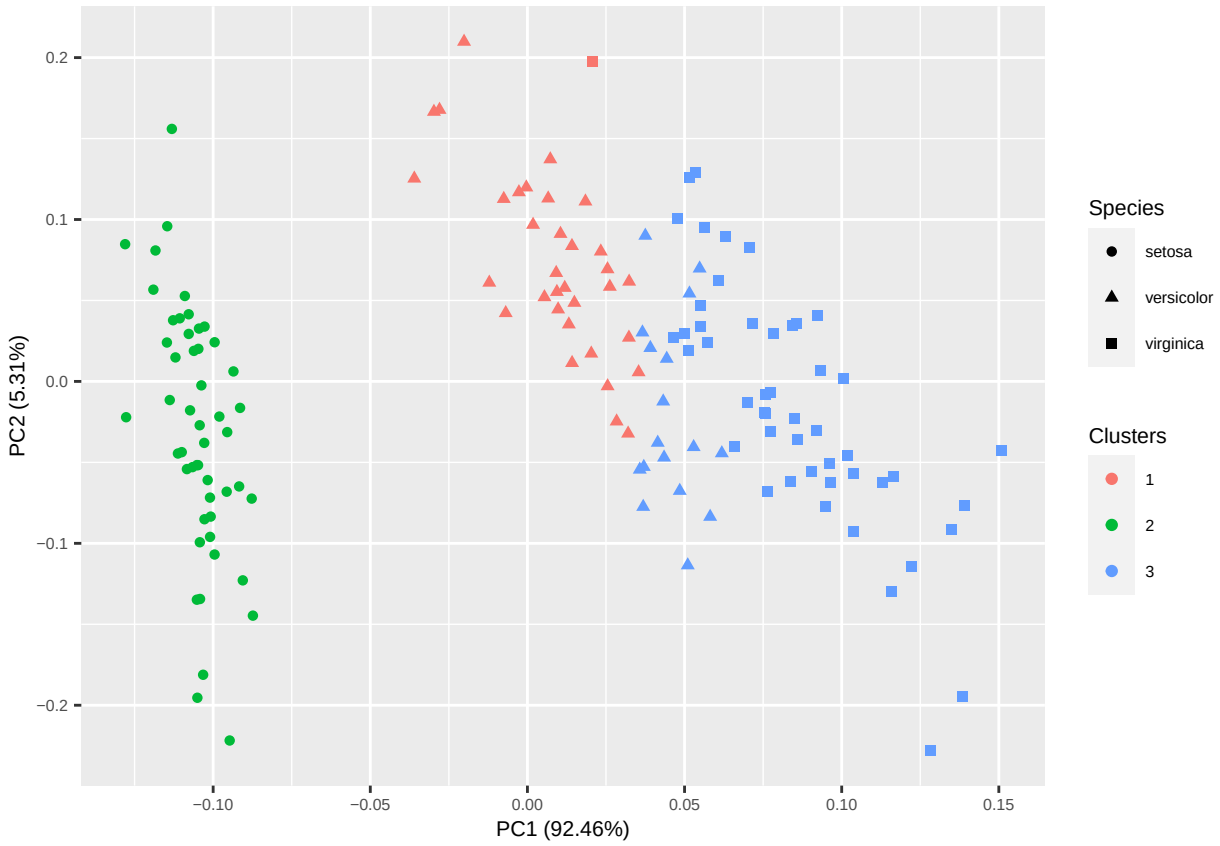
```
## [1] 0.8933333
```

Low-Dimensional K-Means Clustering We can also apply K-means clustering on a q-dimensional PCA reduction of the dataset, with $q < p = 4$. Looking at a scree plot of the variance explained by each principal component, we see that almost all of the variance is explained by the first principal component, so we choose $q = 1 < p = 4$.

```
plot(1:4, 100*pca$sdev^2/sum(pca$sdev^2), main="Scree Plot For PCA on Iris Dataset",
     xlab="k", ylab="Percentage of variance explained by kth PC")
```




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



Note that although we are only using the first PC for the K-means clustering, we are plotting using the first two PCs for continuity with the previous results—but we can see that the clusters do not overlap on the first PC (the x-axis). This produces better results than agglomerative clustering but slightly worse results than full-dimension K-means clustering. The same general pattern of clusters has been found as the previous K-means approach, however, the restriction of only being able to define clusters on the first principal component leads to an accuracy of 55% when considering a cluster-to-class correspondence (assuming cluster 1 is versicolors, cluster 2 is setosas and cluster 3 is virginicas). This is a worse accuracy than that of the agglomerative clustering, however, this is due to limitations in our very loose definition of "accuracy" here—by looking at the plots we can see that this low-dimensional K-means approach actually generates better looking clusters than the agglomerative method.

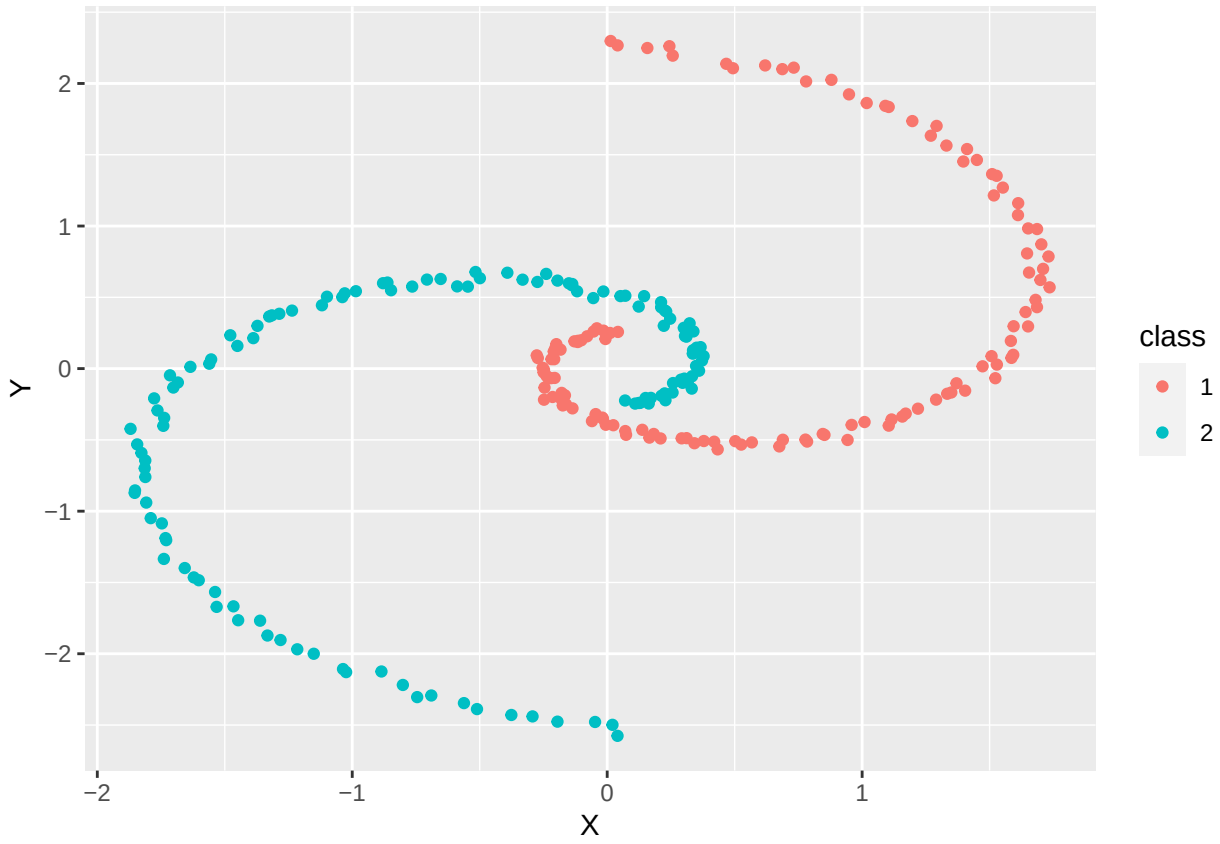
```
# Assuming cluster 1 is versicolor, 2 is setosa and 3 is virginica
sum((irisClusters$Cluster == 1 & irisClusters$Species == "versicolor") |
    (irisClusters$Cluster == 2 & irisClusters$Species == "setosas") |
    (irisClusters$Cluster == 3 & irisClusters$Species == "virginica")) / nrow(iris)
```

```
## [1] 0.5466667
```

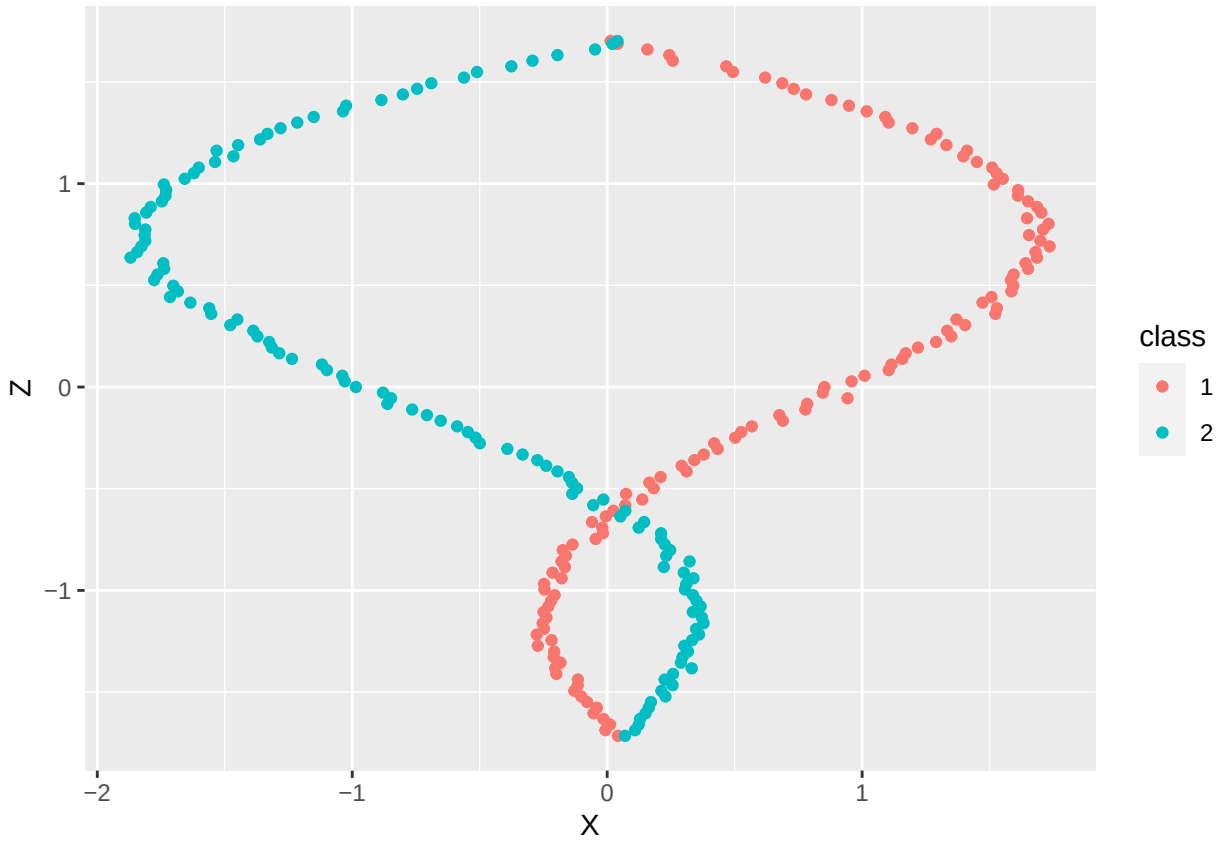
Task 2

For this task 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.

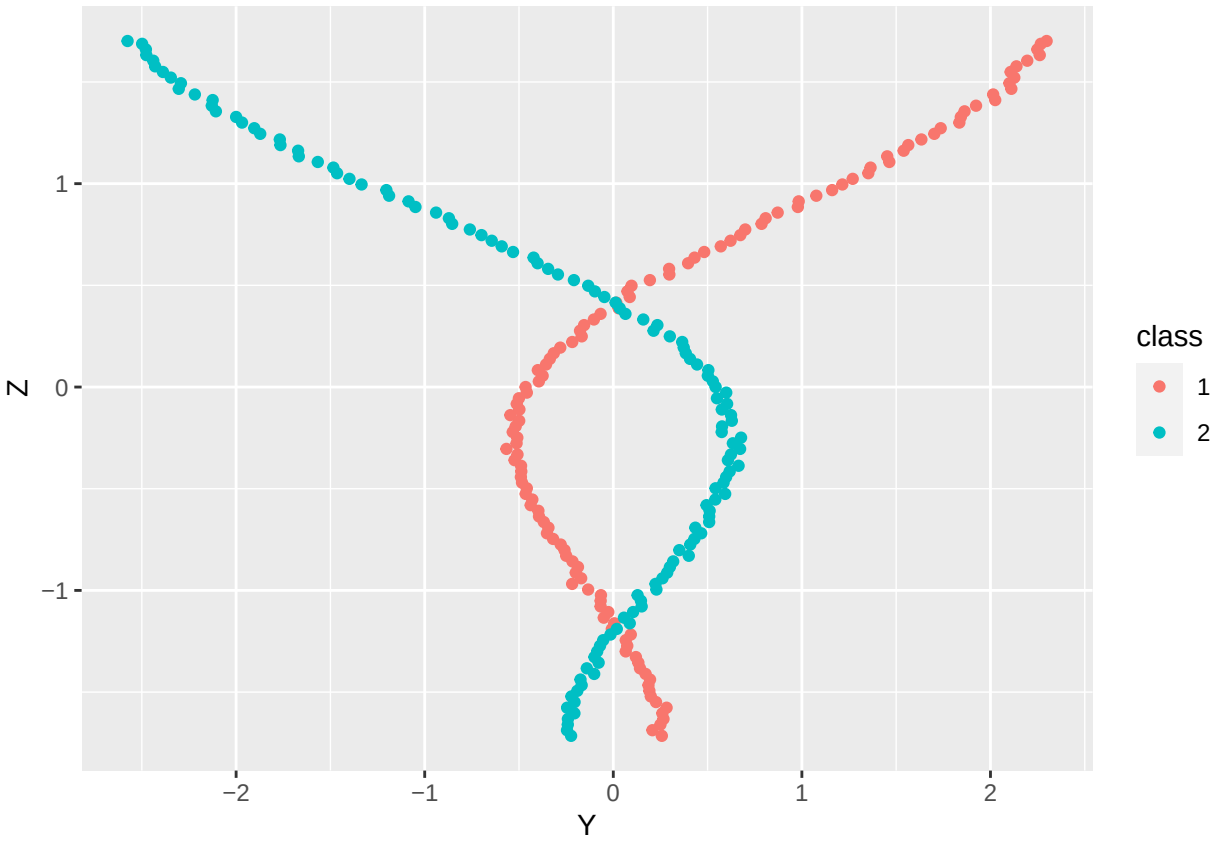
```
n = 250  # Total number of data points

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

    # shift the two spirals so that they don't join up at origin
    coords = coords + c(0, -n/12) * (class*2 -1)

    # add the z=i dimension to moves spirals along z axis
    Xs[class*n/2 + i,] = c(coords, i)
  }
}
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()
```



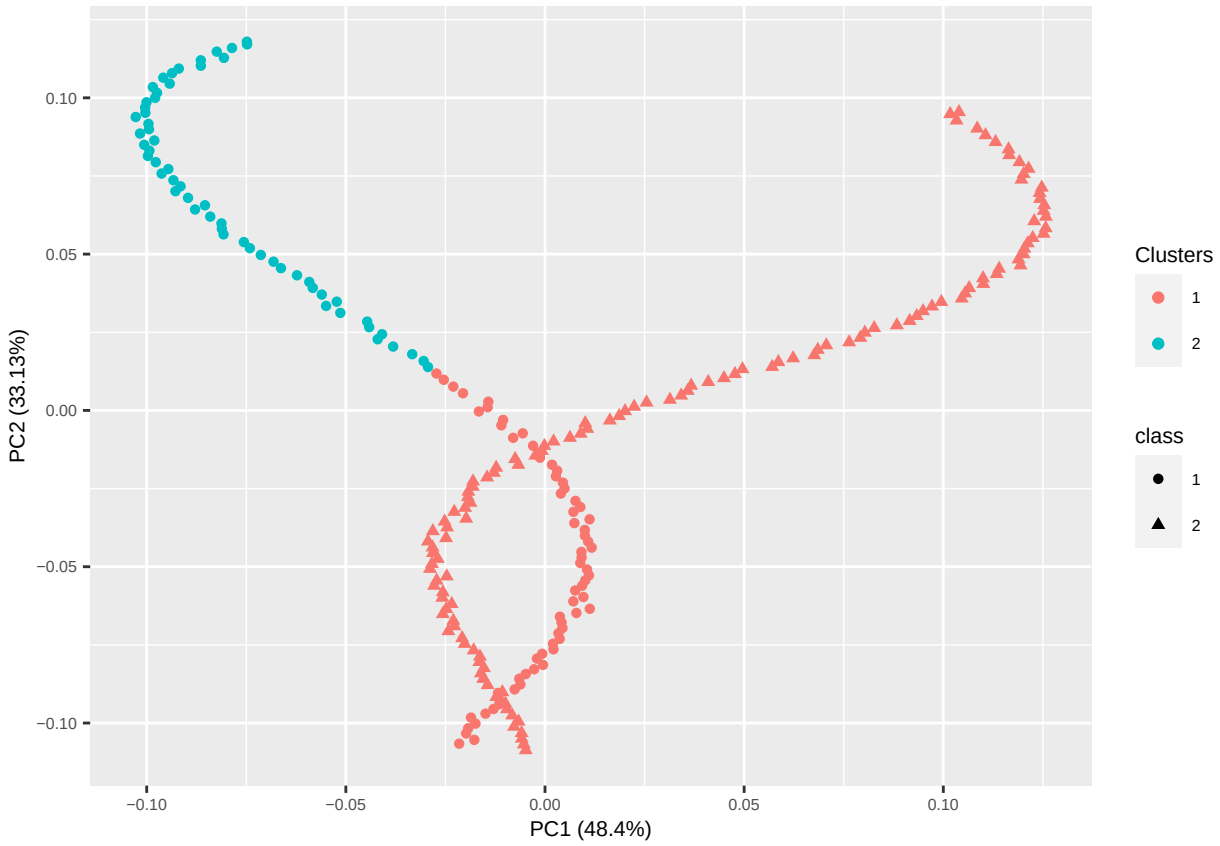

```
ggplot(data = data, aes(Y, Z, color = class)) +
  geom_point()
```



K-Means Clustering

Performing K-means clustering on this dataset with $K = 2$ (since we know that the dataset has two classes—one for each spiral), we see that the produced clusters are far from ideal—they don't manage to separate the classes well since they are contained in non-convex shapes. (Note that we are plotting the clusters on a 2D PCA reduction of the dataset.)

```
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))
```



We can see that this has led to an accuracy of 73% in representing the classes (assumning class 1 corresponds to cluster 2 and vice versa).

```
# 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

For our spectral clustering analysis we'll use Manhattan distance $d_{il} = ||x_i^0 - x_l^0||$, at first with similarity $s_{il} = \exp(-\sigma d_{il}^2)$, i.e. we'll be using the RBF kernel $k(x, x') = \exp(-\sigma||x - x'||^2)$ for $\sigma = 0.05, 2.315329, 20$ where the middle value comes from the median trick (calculating the median pairwise distance of all datapoints).

```
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: $k(x, x') = \exp(-\sigma||x - x'||)$ which we'll test with $\sigma = 0.05, 3.592143, 20$, where again the middle value comes from median trick, but for Manhattan distance rather than Euclidean distance.

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

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

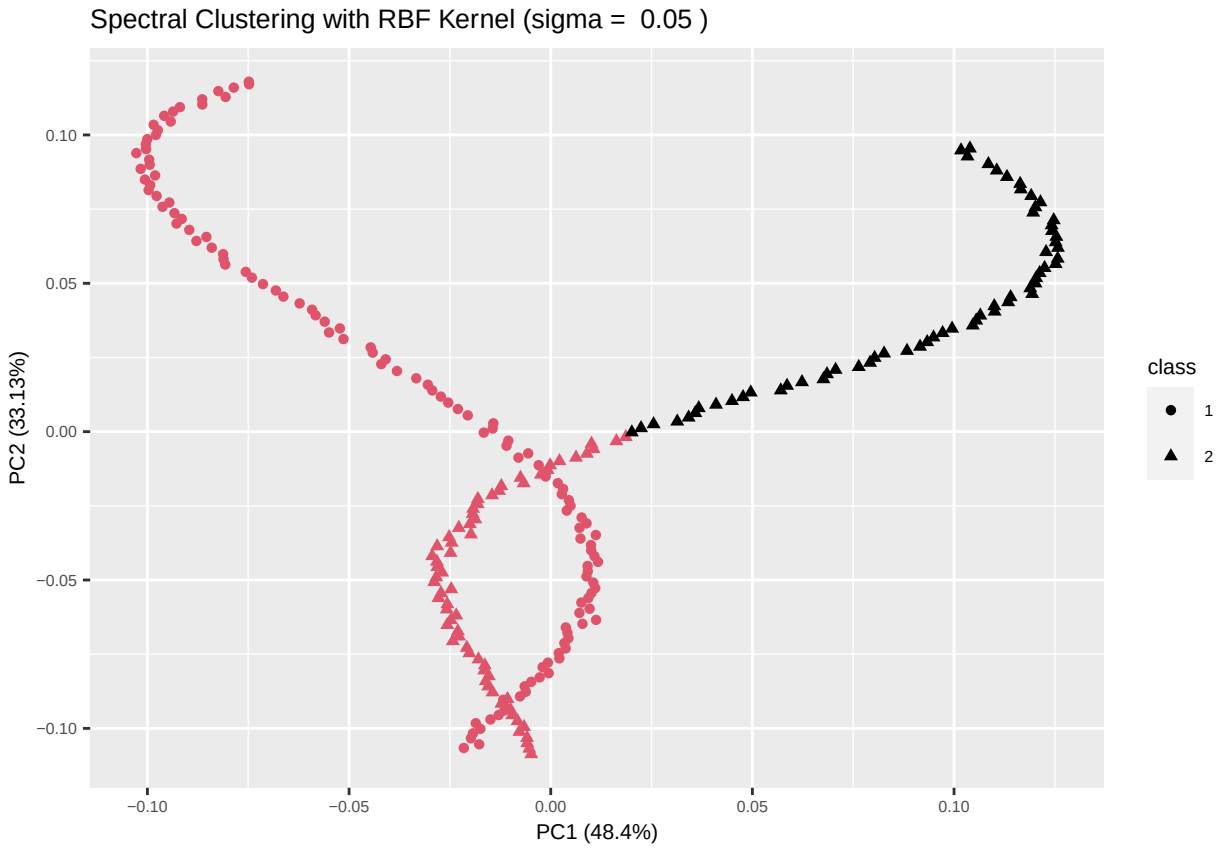
First we'll just look for $K = 2$ clusters in order to get easily comparable results with the K-means we've just performed.

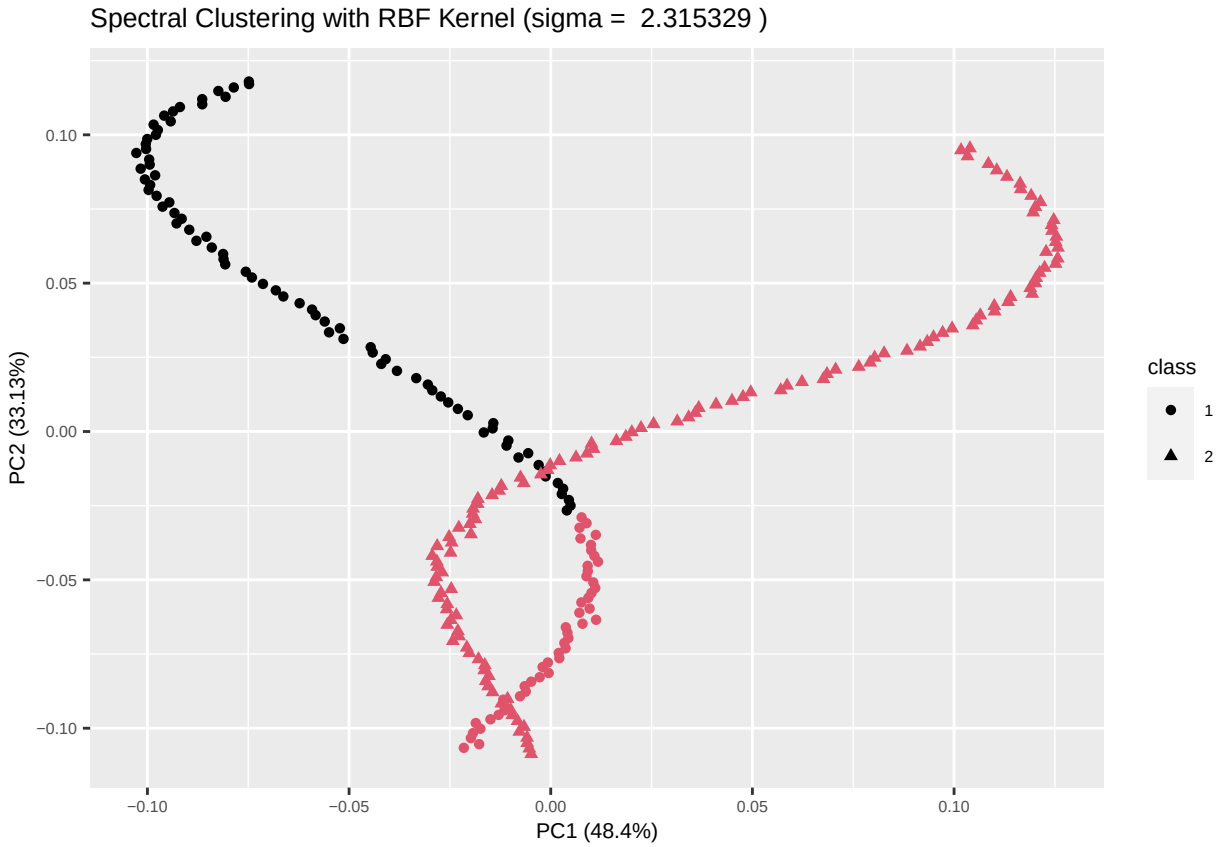
```
library(kernlab)
```

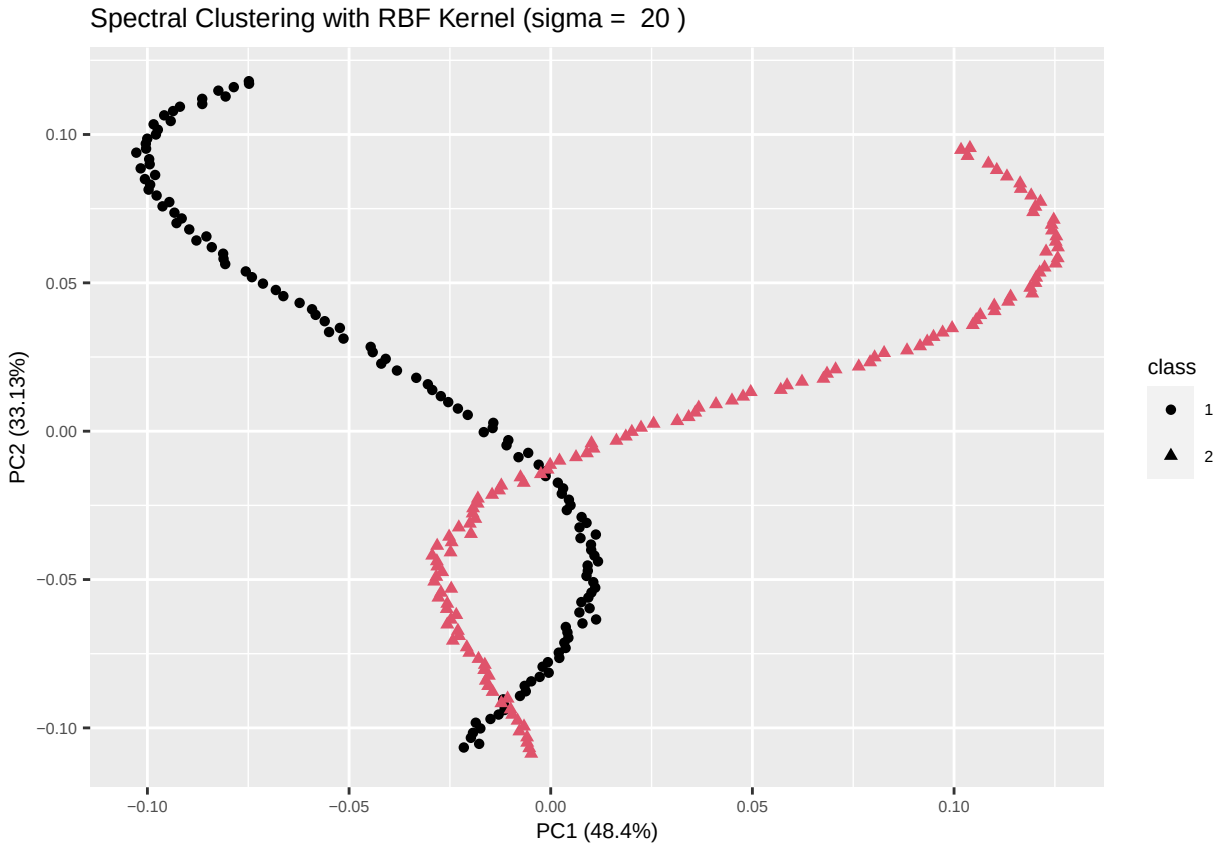
```
##
## Attaching package: 'kernlab'
```

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

```
rbf_sigmas = c(0.05, 2.315329, 20)
lap_sigmas = c(0.05, 3.592143, 20)

for (i in 1:length(rbf_sigmas)){
  sc = specc(Xs, centers=2, kernel="rbfdot", kpar=list(sigma=rbf_sigmas[i]))

  g = autoplot(pca, data=dataClusters, colour=sc, shape="class")  +
    theme(text=element_text(size=8)) +
    ggtitle(paste("Spectral Clustering with RBF Kernel (sigma = ", rbf_sigmas[i], ")"))
  print(g)
}
```

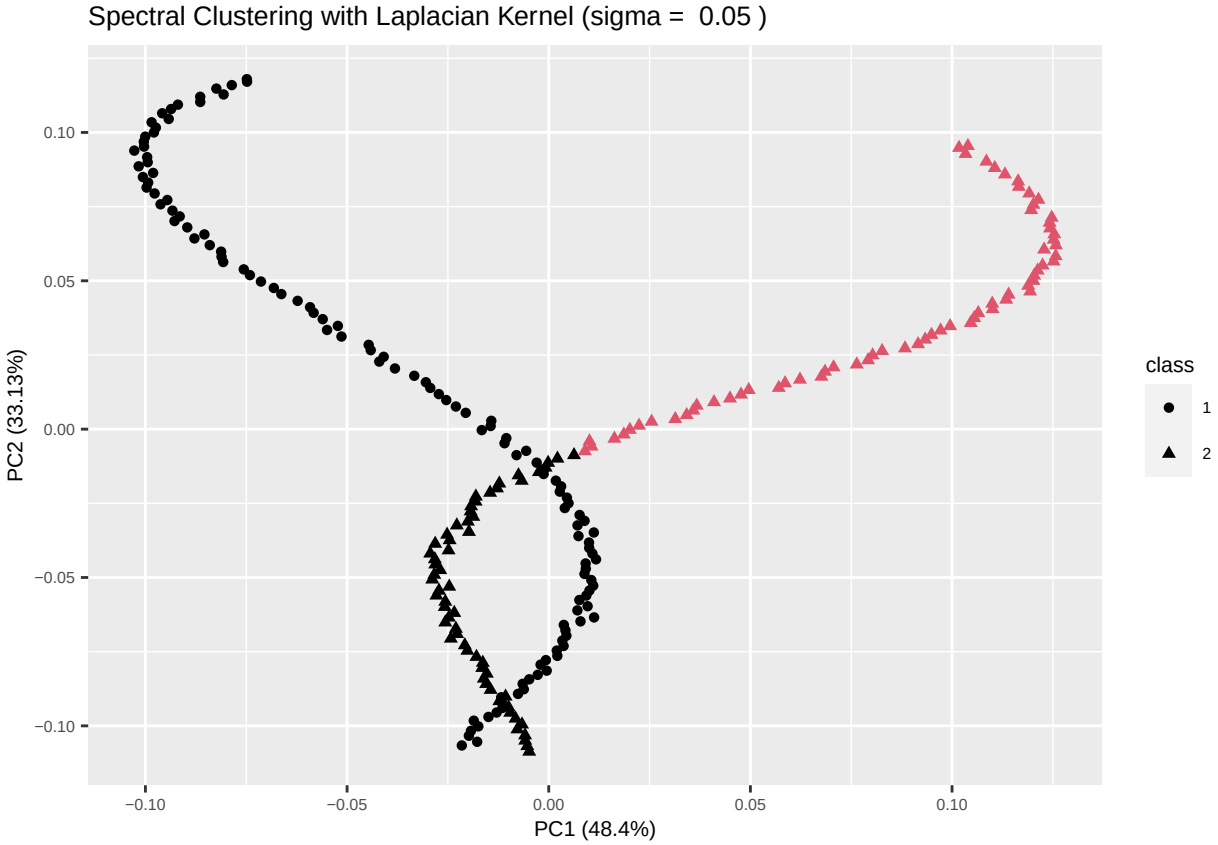


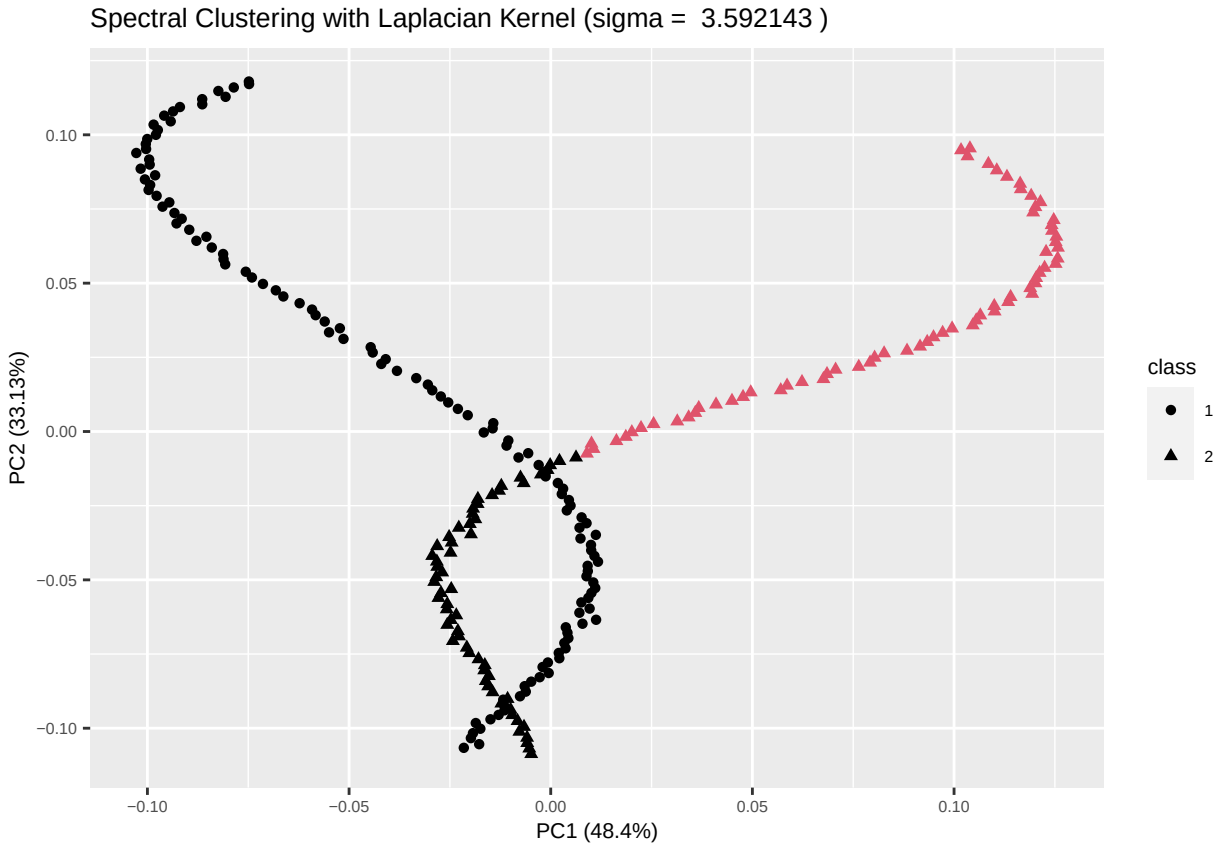

Spectral Clustering with RBF Kernel (sigma = 2.315329)
PC2 (33.13%)
PC1 (48.4%)
0.10
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
0.10
class
1
2

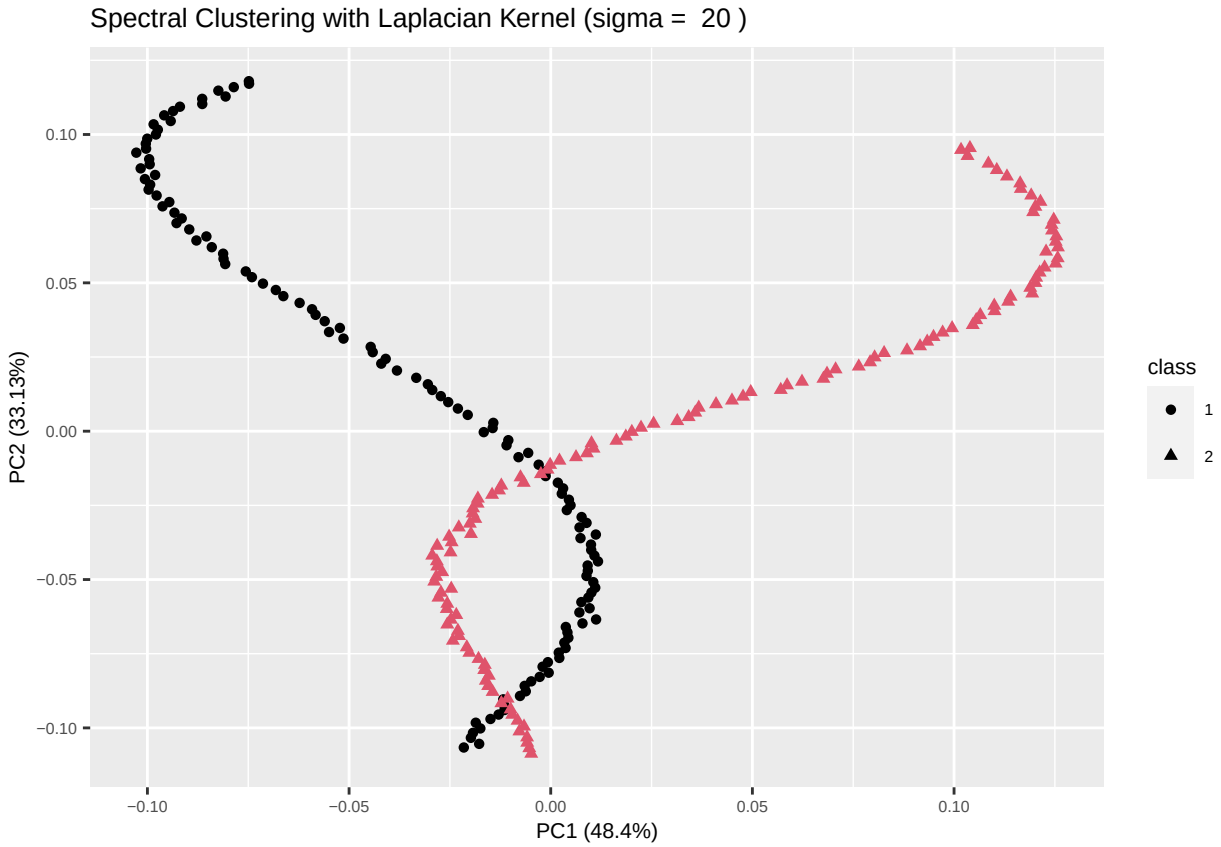

Spectral Clustering with RBF Kernel (sigma = 20)
PC2 (33.13%)
PC1 (48.4%)
0.10
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
0.10
class
1
2

We see that the smaller values of σ give us very similar resutls as the K-means we've just performed, however, the clusters generated with $\sigma = 20$ manage to fully separate the classes in the dataset! When we have a look at the Laplacian kernel we see the same behaviour.

```
for (i in 1:length(lap_sigmas)){
  sc = specc(Xs, centers=2, kernel="laplacedot", kpar=list(sigma=lap_sigmas[i]))

  g = autoplot(pca, data=dataClusters, colour=sc, shape="class")  +
    theme(text=element_text(size=8)) +
    ggtitle(paste("Spectral Clustering with Laplacian Kernel (sigma = ", lap_sigmas[i], ")"))
  print(g)
}
```




Spectral Clustering with Laplacian Kernel (sigma = 3.592143)
PC2 (33.13%)
PC1 (48.4%)
0.10
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
0.10
class
1
2


Spectral Clustering with Laplacian Kernel (sigma = 20)
PC2 (33.13%)
PC1 (48.4%)
0.10
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
0.10
class
1
2

Now we will use the eigengap heuristic to choose K (we are using K for continuity but note that M is used in the notes for the number of clusters in spectral clustering). We do this using the `estimate_k` function from the library `Spectrum` (which requires a similarity matrix—we use `kernelMatrix` from `kernlab`), however, the values of K produced by this function are not always generated using the eigengap heuristic alone, so we will calculate that explicitly ourselves.

```
library(Spectrum)

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

rbf_Ks = rep(0,3)
lap_Ks = rep(0,3)

for (i in 1:3){
  rbfkern = rbfdot(rbf_sigmas[i])
  lapkern = laplacedot(lap_sigmas[i])
  rbf_Ks[i] = eigenGap(estimate_k(kernelMatrix(rbfkern, Xs), showplots=FALSE)$z)
  lap_Ks[i] = eigenGap(estimate_k(kernelMatrix(lapkern, Xs), showplots=FALSE)$z)
  # $z gives us the eigenvalues in the two lines above
}
```

```
## egap optimal K: 3
## egap optimal K: 3
```

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

```
## egap optimal K: 10
## egap optimal K: 10
```

Ignoring the output of the `estimate_k` function (which is more complex than just the eigengap heuristic), we arrive the the following choices of M for the discussed bandwidth parameters σ.

```
rbf_Ks
```

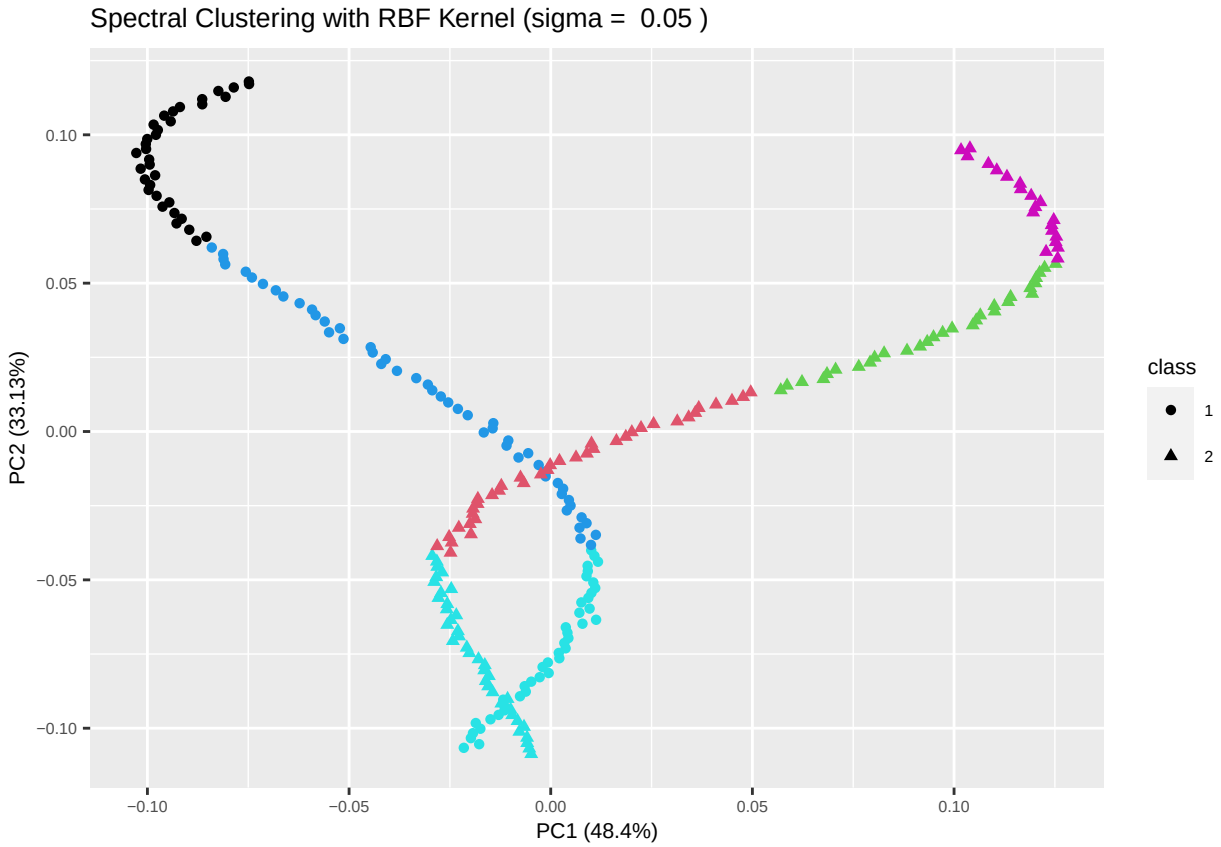
```
## [1] 6 6 3
```

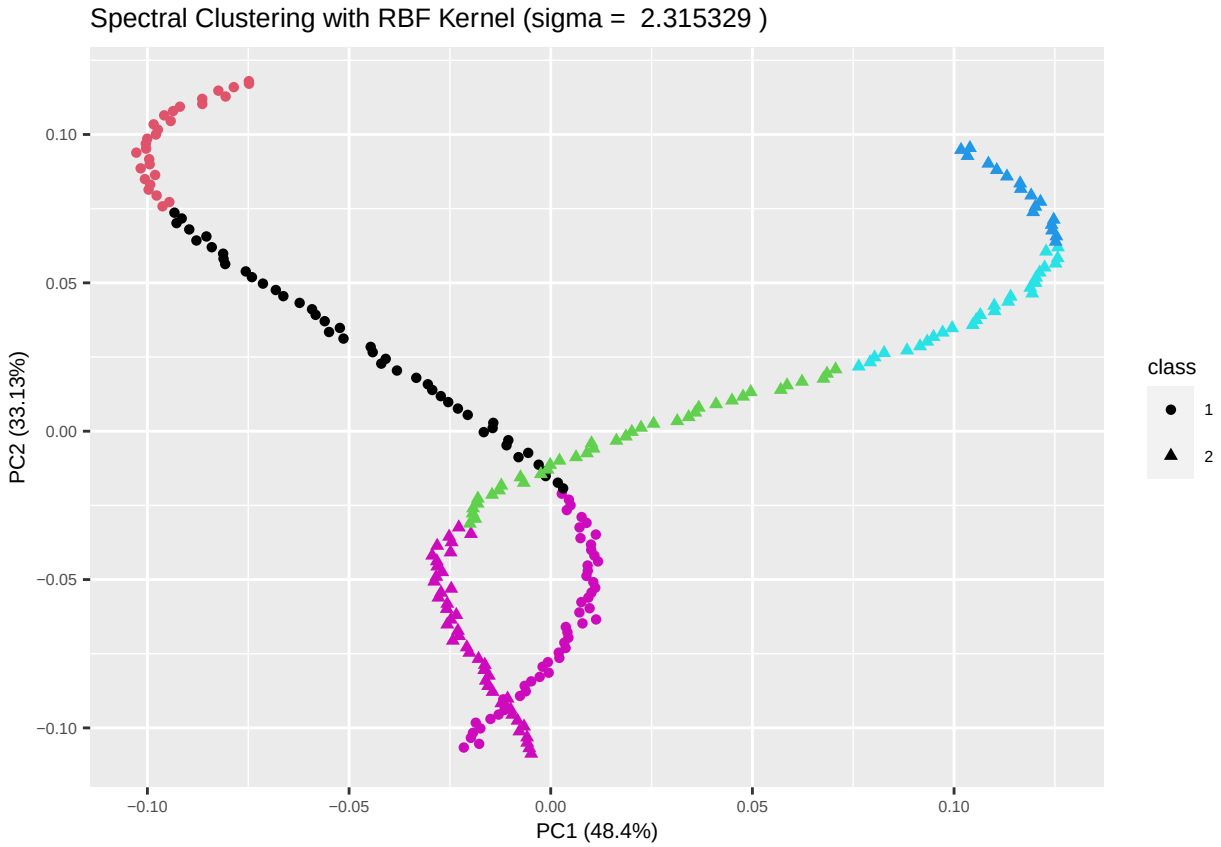
```
lap_Ks
```

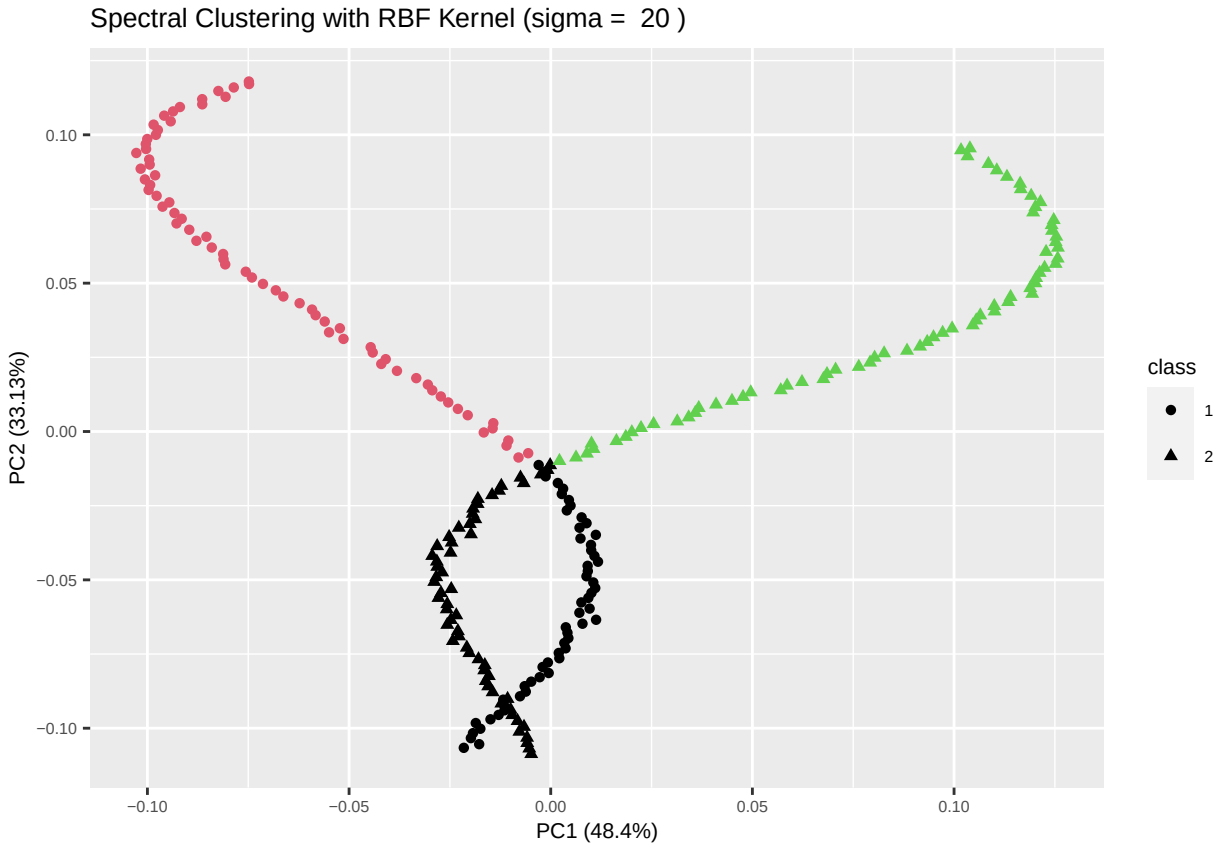
```
## [1] 4 8 7
```

We now plot the corresponding clusterings obtained using the RBF kernel.

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

  g = autoplot(pca, data=dataClusters, colour=sc, shape="class")  +
    theme(text=element_text(size=8)) +
    ggtitle(paste("Spectral Clustering with RBF Kernel (sigma = ", rbf_sigmas[i], ")"))
  print(g)
}
```


Spectral Clustering with RBF Kernel (sigma = 0.05)
PC2 (33.13%)
PC1 (48.4%)
0.10
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
0.10
class
1
2

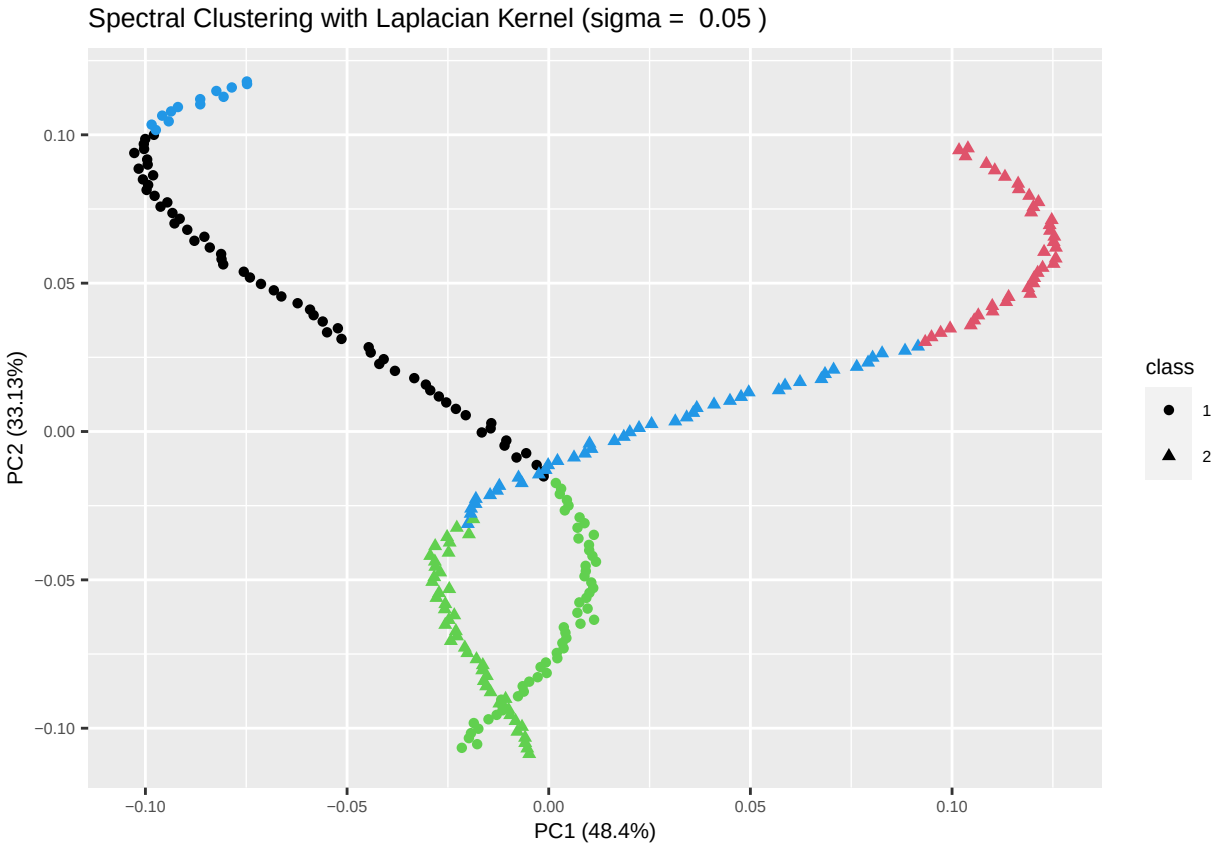

Spectral Clustering with RBF Kernel (sigma = 2.315329)
PC2 (33.13%)
PC1 (48.4%)
0.10
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
0.10
class
1
2

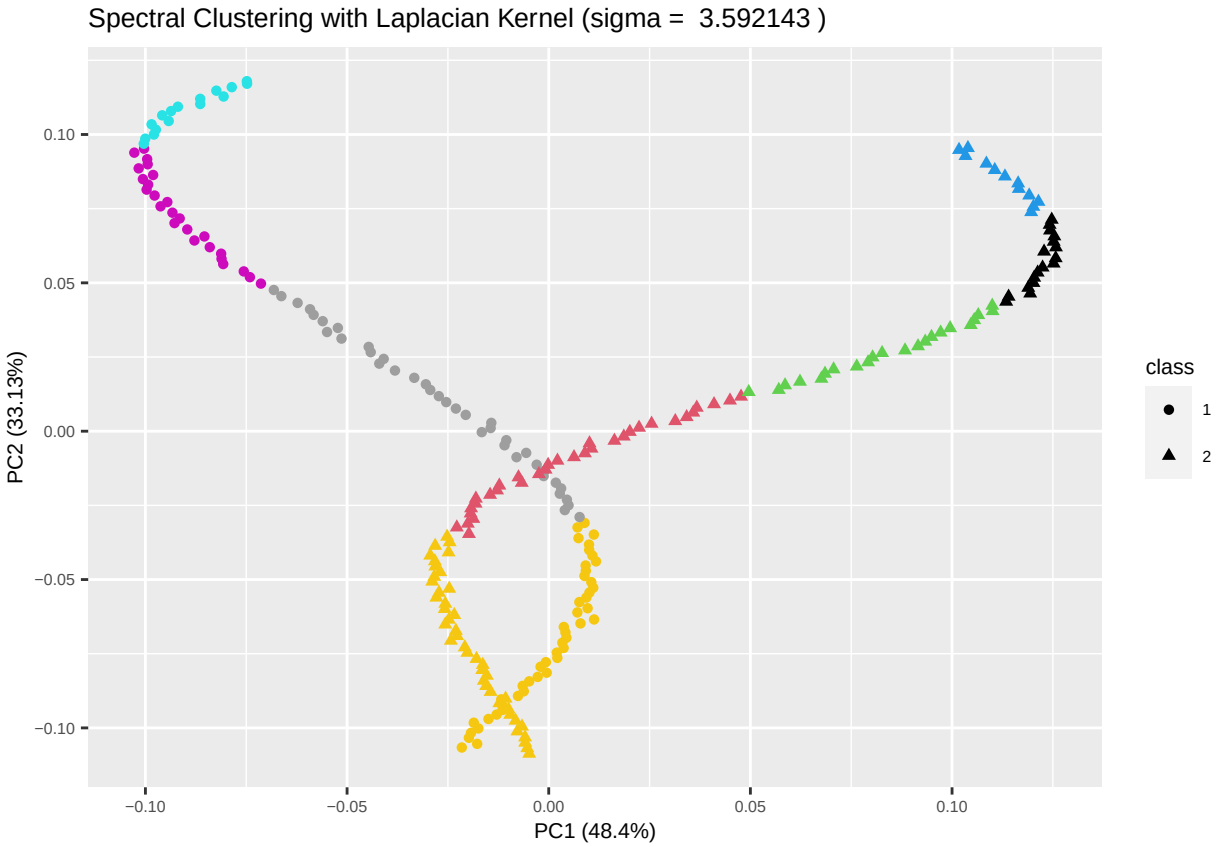


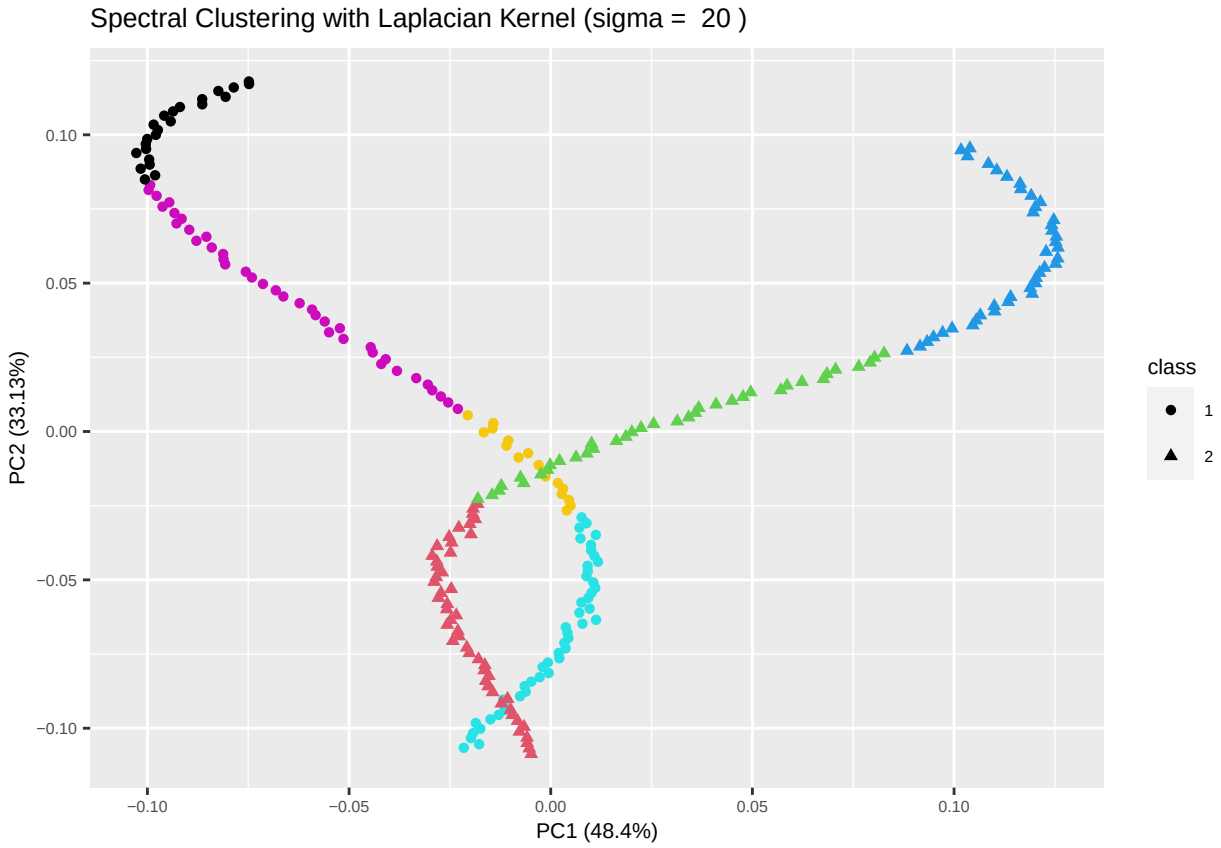
The first two plots don't seem particularly useful, probably due to poor tuning of the kernel. In particular since the dataset is symmetrical we'd at least expect to see symmetrical clusterings, but this only occurs for the $\sigma = 20$ kernel, which finds $K = 3$ reasonable clusters (even if they don't correspond to the classes in the dataset, they do make intuitive sense when we look at the shape of the data).

We see generally worse results when we use the Laplcaian kernel.

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

  g = autoplot(pca, data=dataClusters, colour=sc, shape="class")  +
    theme(text=element_text(size=8)) +
    ggtitle(paste("Spectral Clustering with Laplacian Kernel (sigma = ", lap_sigmas[i], ")"))
  print(g)
}
```


Spectral Clustering with Laplacian Kernel (sigma = 0.05)
PC2 (33.13%)
PC1 (48.4%)
0.10
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
0.10
class
1
2


Spectral Clustering with Laplacian Kernel (sigma = 3.592143)
PC2 (33.13%)
PC1 (48.4%)
0.10
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
0.10
class
1
2



All three of these plots show poor clusterings, much in the same way as the RBF kernel plots with $\sigma = 0.05$ and $\sigma = 2.315329$ do, with even the $\sigma = 20$ clusters under performing (despite the fact that this kernel worked well for $K = 2$). This indicates that the Laplacian kernel is inferior to the RBF kernel for spectral clustering on this dataset when we use it alongside the eigengap heuristic to choose the number of clusters—in fact this heuristic only led to a somewhat reasonable value of K with the RBF kernel for $\sigma = 20$ when it chose $K = 3$ (although perhaps $K = 2$ would've been better still since that is the number of spirals in the dataset).

Task 3

Part 1

With $k(x, x') = \Phi(x)^T \Phi(x')$ it is fairly straightforward to show that performing K-means clustering only requires us to evaluate $k(\cdot, \cdot)$ pointwise. This is because in Lloyd's algorithm we only use the data $\{x_i^0\}_{i=1}^n$ in the following expression (for some $i \in \{1, ..., n\}$ and $k \in \{1, ..., k\}$):

$$||x_i^0 - c_k'||^2$$

for various centers $c_k' = \frac{1}{|C_k'|} \sum_{i \in C_k'} x_i^0$ of different clusters C_k'. If we use the transformed data $\{\Phi(x_i^0)\}_{i=1}^n$ in place of the original data $\{x_i^0\}_{i=1}^n$, we can rewrite this expression as follows,

$$\begin{aligned}
||\Phi(x_i^0) - c_k'||^2 &= (\Phi(x_i^0) - c_k')^T (\Phi(x_i^0) - c_k') \\
&= \Phi(x_i^0)^T \Phi(x_i^0) - 2\Phi(x_i^0)^T c_k' + (c_k')^2 \\
&= k(x_i^0, x_i^0) - 2\Phi(x_i^0)^T \frac{1}{|C_k'|} \sum_{j \in C_k'} \Phi(x_j^0) + \frac{1}{|C_k'|^2} \left(\sum_{j \in C_k'} \Phi(x_j^0) \right)^T \left(\sum_{l \in C_k'} \Phi(x_l^0) \right) \\
&= k(x_i^0, x_i^0) - 2\frac{1}{|C_k'|} \sum_{j \in C_k'} \Phi(x_i^0)^T \Phi(x_j^0) + \frac{1}{|C_k'|^2} \sum_{j,l \in C_k'} \Phi(x_j^0)^T \Phi(x_l^0) \\
&= k(x_i^0, x_i^0) - 2\frac{1}{|C_k'|} \sum_{j \in C_k'} k(x_i^0, x_j^0) + \frac{1}{|C_k'|^2} \sum_{j,l \in C_k'} k(x_j^0, x_l^0).
\end{aligned}$$

In this form, we only need to evaluate $k(\cdot, \cdot)$ pointwise on the original data, meaning we don't have to explicitly calculate $\{\Phi(x_i^0)\}_{i=1}^n$.

Part 2

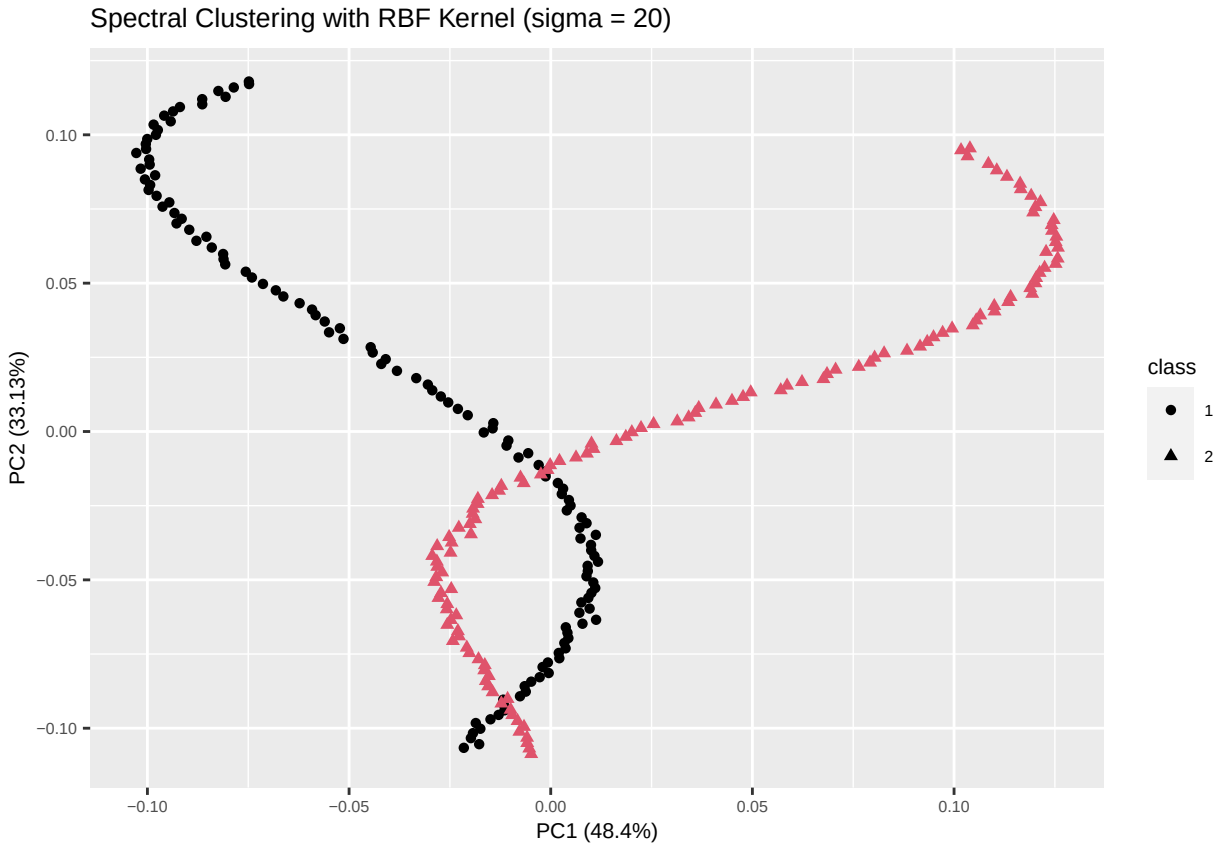
For this part of the task, we'll perform kernel K-means clustering using the kernel that has been performing the best up to this point: the RBF kernel with bandwidth parameter $\sigma = 20$. In order to get useful results that we can compare against non-kernel K-means clustering as well as against spectral clustering, we'll examine kernel K-means on both $K = 2$ and $K = 3$ (even though regular K-means was only performed with $K = 2$).

```
library(klic)
```

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

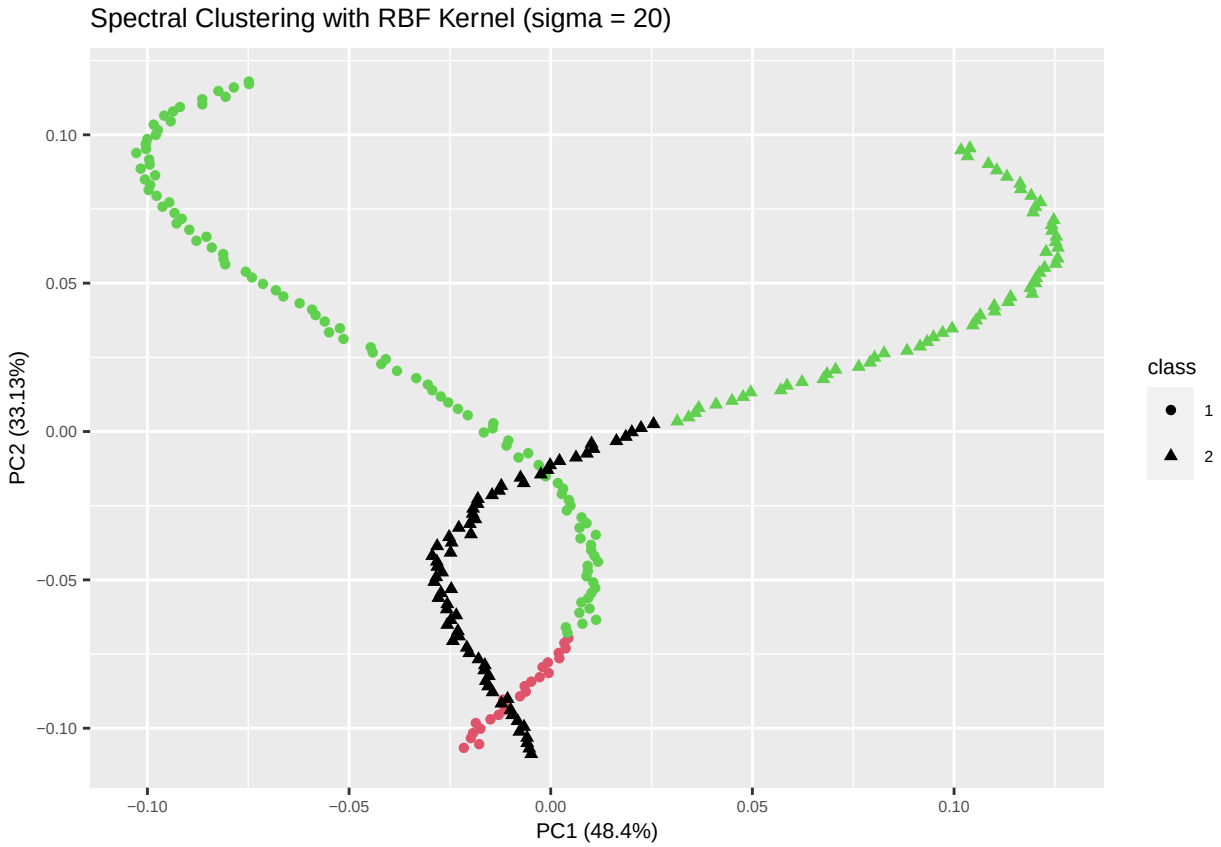
```
## The following object is masked from 'package:kernlab':
##
##     kkmeans
```

```
rbfkern = rbfdot(20)

rbfKernMatrix = kernelMatrix(rbfkern, Xs)

clusters = kkmeans(rbfKernMatrix, list("cluster_count"=2))$clustering
autoplot(pca, data=dataClusters, colour=clusters, shape="class")  +
    theme(text=element_text(size=8)) +
    ggtitle(paste("Spectral Clustering with RBF Kernel (sigma = 20)"))
```



These clusters match up with the classes—this is exactly what we want! Importantly, this is something that regular K-means clustering was unable to do. However, spectral clustering with this same kernel was able to get the ideal clusters too, so in order to further our comparison between spectral clustering and kernel K-means clustering we'll try one final test: using $K = 3$.

```
clusters = kkmeans(rbfKernMatrix, list("cluster_count"=3))$clustering
autoplot(pca, data=dataClusters, colour=clusters, shape="class")  +
    theme(text=element_text(size=8)) +
    ggtitle(paste("Spectral Clustering with RBF Kernel (sigma = 20)"))
```



Since the dataset classes no longer apply here (as we're looking for more clusters than there are classes), we instead have to look for whether the presented clusters seem reasonable, and it appears here that they don't. In particular we see a very small cluster (red) towards the bottom of the plot and the other two clusters fail to separate the different spirals in the dataset, and fail to produce any otherwise useful set of clusters. For this reason it seems that spectral clustering is more suitable for this dataset than both regular K-means clustering and kernel K-means clustering.