Death vectors and persistence landscapes, principal components analysis (PCA), loading vectors, and the explained variance.

Introduction to Topological Data Analysis with R - Lab 2

Peter Bubenik (September 2020), with modifications by Johnathan Bush and Iryna Hartsock (July 2022)

Functions

```
euclidean.distance <- function(u, v) sqrt(sum((u - v) ^ 2))
```

```
# Sample points from an annulus
sample.annulus <- function(num.pts, inner.radius, outer.radius){
  theta <- runif(num.pts) * 2 * pi
  radius <- sqrt(runif(num.pts, inner.radius^2, outer.radius^2))
  x <- radius * cos(theta)
  y <- radius * sin(theta)
  cbind(x,y)
}
```

```
# Take difference of vectors of potentially different lengths
difference.vectors <-function(vector.1,vector.2){
  length.1 <- length(vector.1)
  length.2 <- length(vector.2)
  difference.vector = numeric(max(length.1,length.2))
  difference.vector = as(difference.vector, "sparseVector")
  difference.vector[1:length.1] = difference.vector[1:length.1] + vector.1
  difference.vector[1:length.2] = difference.vector[1:length.2] - vector.2
}
```

```
death.vector <- function(PD){
  zeroes <- which(PD[, 1] == 0)
  PD.0 <- PD[zeroes,2:3]
  dv <- vector()
  if ((min(PD.0[,"Birth"]) == 0) && (max(PD.0[,"Birth"]) == 0))
    dv <- sort(PD.0[,2], decreasing=TRUE)
  return(dv)
}
```

```
# Matrix of death vectors from a list of persistence diagrams
death.vector.matrix <- function(PD.list){
  num.pts <- length(which(PD.list[[1]][,1] == 0))
  DVM <- matrix(0L, nrow = length(PD.list), ncol = num.pts - 1)
  for (c in 1 : length(PD.list))
    DVM[c,] <- death.vector(PD.list[[c]])[-1]
  return(DVM)
}
```

```r
# Matrix of persistence landscape row vectors from list of persistence landscapes
landscape.matrix.from.list <- function(PL.list){
  n <- length(PL.list)
  m <- ncol(PL.list[[1]])
  max.depth <- integer(n)
  for (i in 1:n)
    max.depth[i] <- nrow(PL.list[[i]])
  K <- max(max.depth)
  PL.matrix <- Matrix(0, nrow = n, ncol = m*K, sparse = TRUE)
  for (i in 1:n)
    for (j in 1:max.depth[i])
      PL.matrix[i,(1+(j-1)*m):(j*m)] <- PL.list[[i]][j,]
  return(PL.matrix)
}
```

```r
# Convert a vector to a persistence landscape
landscape.from.vector <- function(PL.vector, t.vals){
  m <- length(t.vals)
  K <- length(PL.vector)/m
  PL <- Matrix(0, nrow = K, ncol=m, sparse = TRUE)
  for (i in 1:K){
    PL[i,1:m] <- PL.vector[(1+(i-1)*m):(i*m)]
  }
  return(PL)
}
```

```r
# Plot Persistence Landscape
plot.landscape <- function(PL,t.vals){
  plot(t.vals,PL[1,],type="l",ylab="Persistence",xlab="Parameter values",col=1,ylim=c(min(PL),max(PL)))
  for(i in 2:dim(PL)[1])
    lines(t.vals,PL[i,],type="l",col=i)
}
```

```r
# Remove dimension zero points from a persistence diagram
remove.dimzero.from.pd <- function(PD){
  zeroes <- which(PD[, 1] == 0)
  newPD <- PD[-zeroes,]
  return(newPD)
}
```

```r
# Discrete Persistence Landscape
discrete.PL <- function(PD,t.vals){
  get.tent <- function(interval,t.vals){
    y <- numeric(length(t.vals))
    # Compute slope +1 part of tent
    going_up <- which(t.vals > interval[1] & t.vals <= mean(interval))
    y[going_up] <- (t.vals[going_up] - interval[1])
    # Compute slope -1 part of tent
    going_down <- which(t.vals > mean(interval) & t.vals < interval[2])
    y[going_down] <- (interval[2] - t.vals[going_down])
    return(y)
  }

  # discrete landscape using sparse matrix
  PL <- Matrix(0, nrow = dim(PD)[1], ncol = length(t.vals), sparse = TRUE)
```

```
  for(i in 1:dim(PD)[1])
    PL[i, ] <- get.tent(PD[i, ],t.vals)
  for(j in 1:length(t.vals)){
    PL[ ,j] <- sort(PL[ ,j], decreasing = TRUE)
  }
  actual.depth <- sum( rowSums(PL) != 0)
  PL <- PL[1:actual.depth,]
  return(PL)
}
```

```
plot.persistence.landscape <- function(PL.single.degree,t.vals){
  plot(t.vals,PL.single.degree[1,],type="l",ylab="Persistence",
       xlab="Parameter values",col=1)
  for(i in 2:dim(PL.single.degree)[1])
    lines(t.vals,PL.single.degree[i,],type="l",col=i)
}
```

```
# Plot the Voronoi cells and dual and Delaunay complex
plot.delaunay <- function(X){
  DelVor <- deldir(X[,1], X[,2], suppressMsge = TRUE)

  # Voronoi cells:
  plot(DelVor, pch=20, cmpnt_col=c("black","red","blue"), wlines= ("tess"))

  # Voronoi cells and their dual (the Delaunay complex):
  plot(DelVor, pch=20, cmpnt_col=c("black","red","blue"), wlines= ("both"))

  # Delaunay complex:
  plot(DelVor, pch=20, cmpnt_col=c("black","red","blue"), wlines= ("triang"))
}
```

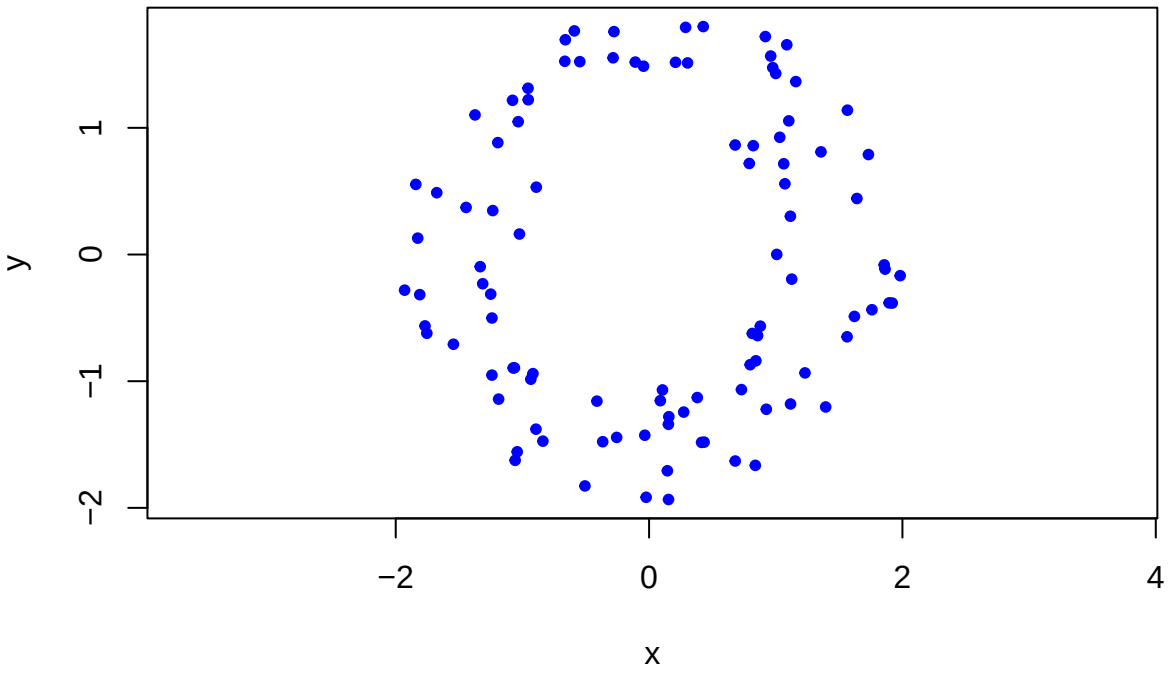
```
# Permutation test for two matrices consisting of row vectors
permutation.test <- function(M1 ,M2, num.repeats = 10000){
  # append zeros if necessary so that the matrices have the same number of columns
  num.columns <- max(ncol(M1),ncol(M2))
  M1 <- cbind(M1, Matrix(0,nrow=nrow(M1),ncol=num.columns-ncol(M1)))
  M2 <- cbind(M2, Matrix(0,nrow=nrow(M2),ncol=num.columns-ncol(M2)))
  t.obs <- euclidean.distance(colMeans(M1),colMeans(M2))
  k <- dim(M1)[1]
  M <- rbind(M1,M2)
  n <- dim(M)[1]
  count <- 0
  for (i in 1:num.repeats){
    permutation <- sample(1:n)
    t <- euclidean.distance(colMeans(M[permutation[1:k],]),colMeans(M[permutation[(k+1):n],]))
    if (t >= t.obs)
      count <- count + 1
  }
  return(count/num.repeats)
}
```

Parameters

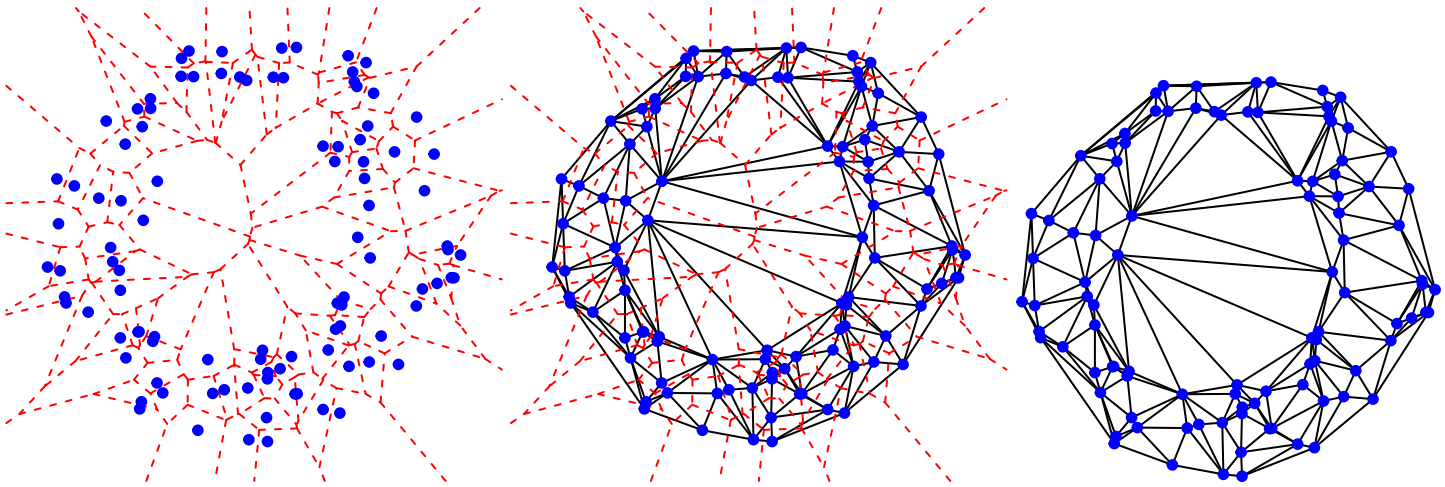
```
num.pts <- 100
inner.radius <- 1
outer.radius <- 2
min.t <- 0
max.t <- 1.2
t.steps <- 200
t.vals <- seq(min.t,max.t,(max.t-min.t)/t.steps)
```

Persistence landscapes

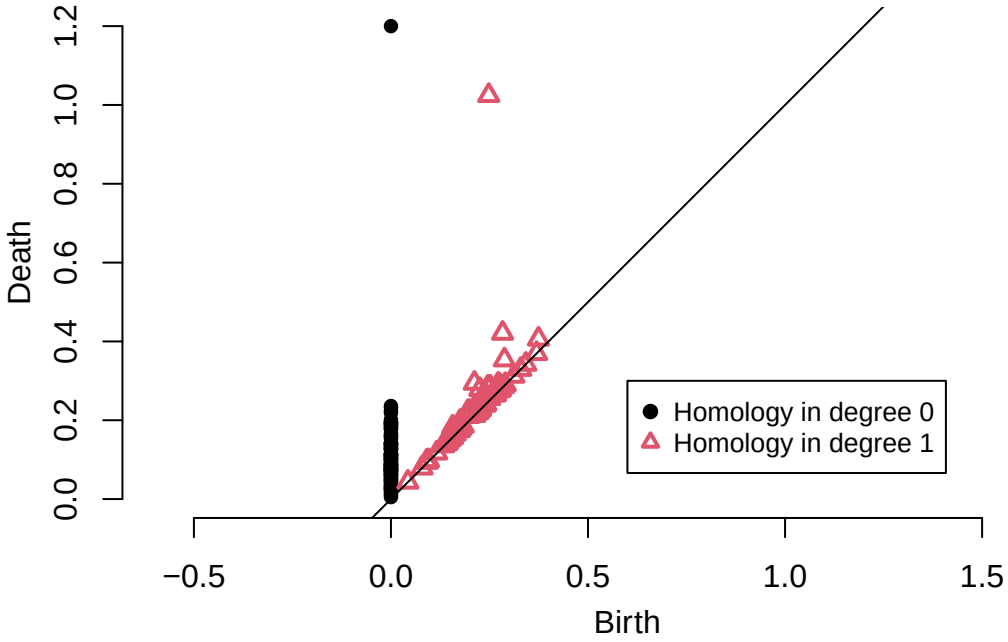
```
X <- sample.annulus(num.pts,inner.radius,outer.radius)
plot(X, pch=20, col="blue", asp=1)
```



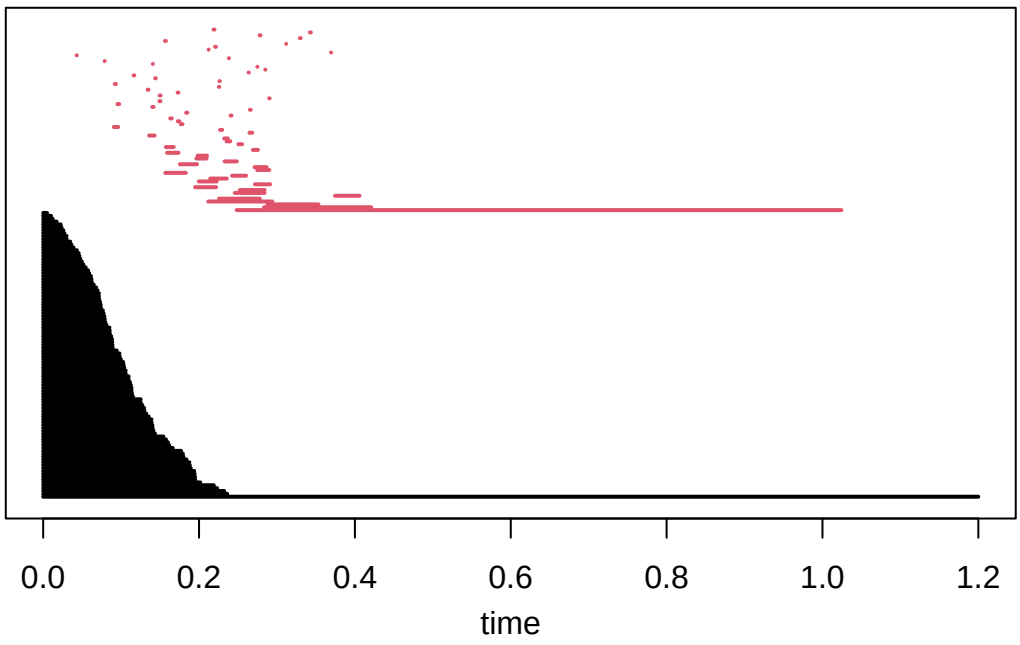

```
# Delaunay Complex and its Persistence Diagram

# plot the Voronoi cells and Delaunay complex (3 plots)
plot.delaunay(X)
```



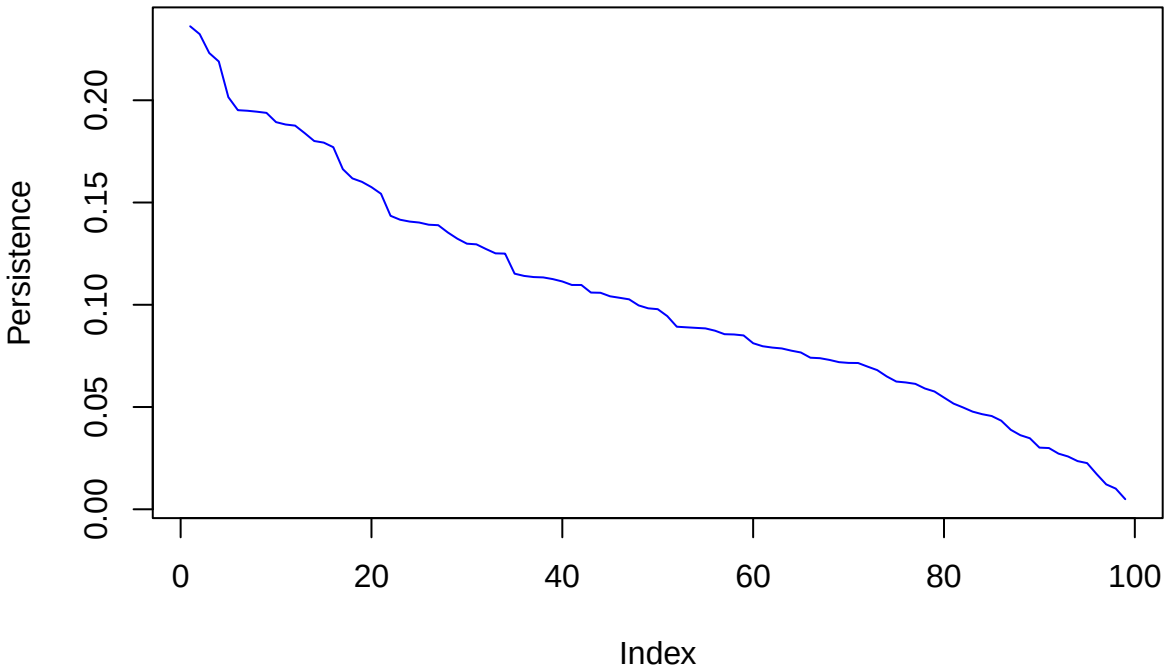
```
# compute persistent homology
PH.output <- alphaComplexDiag(X)
PD <- PH.output[["diagram"]]

# filtration values have been squared for some reason - fix by taking square root
PD[,2:3] <- sqrt(PD[,2:3])

# plot the persistence diagram
plot(PD, asp=1, diagLim = c(0,max.t))
legend(0.5*max.t, 0.25*max.t, c("Homology in degree 0","Homology in degree 1"),
       col = c(1,2), pch = c(19,2), cex = .8, pt.lwd = 2)
```




```
# plot the bar code
plot(PD, diagLim = c(0,max.t), barcode=TRUE)
```

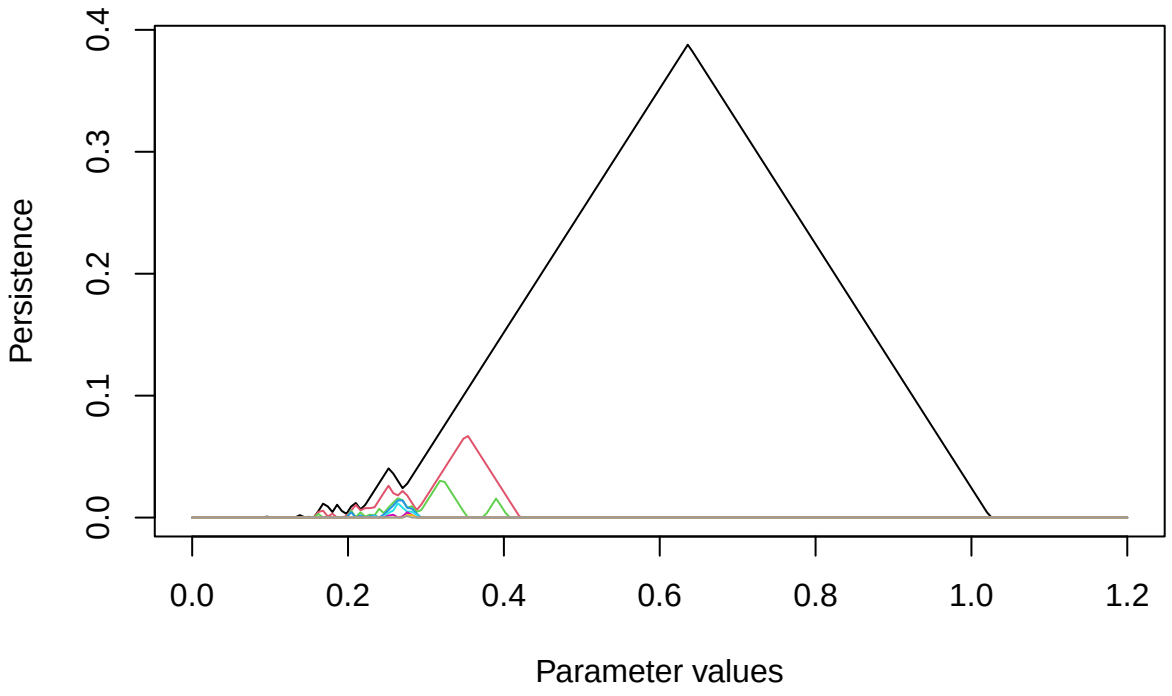



```
DV <- death.vector(PD)
DVr <- DV[-1]
plot(DVr, type="l", col="blue", ylab="Persistence")
```




```
# Remove dimension 0 from PD
PD.1 <- remove.dimzero.from.pd(PD)
```

```
# Persistence Landscapes in dimension 1
PD.1 <- remove.dimzero.from.pd(PD)
PL.1 <- discrete.PL(PD.1[,2:3],t.vals)
plot.persistence.landscape(PL.1,t.vals)
```



Exercises (Part I):

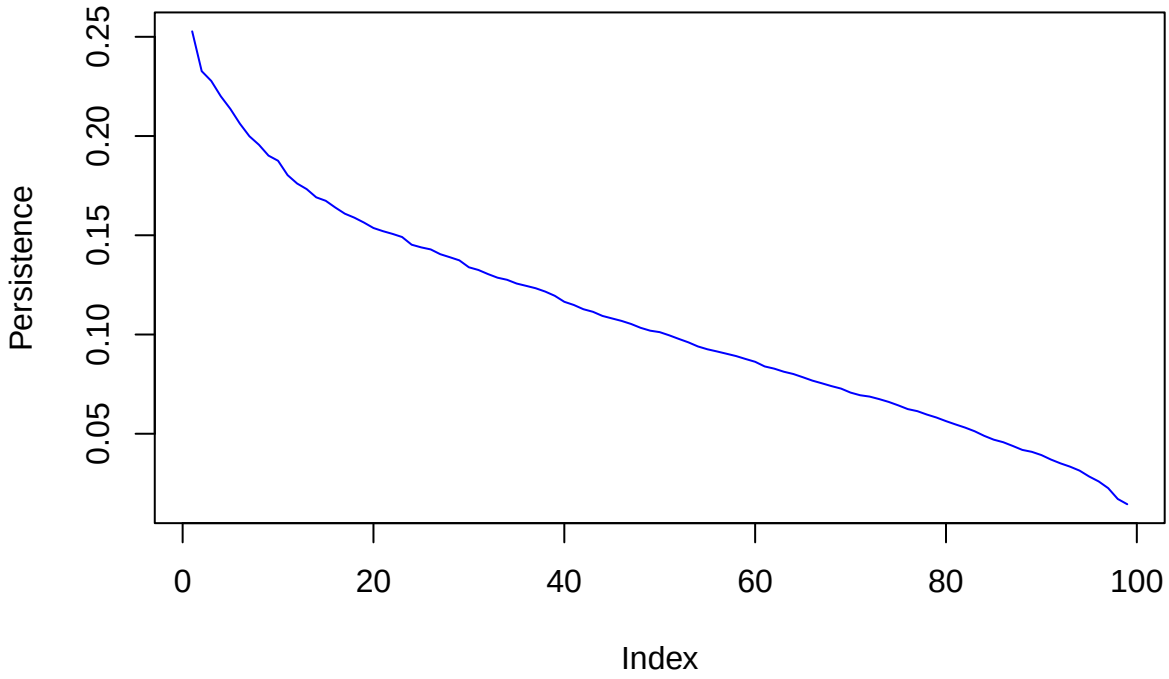
In collaboration with some of the other participants, do the following.

1. Explain/understand all of the parts of the function death.vector.
2. Repeat the computation above for samples generated from annuli with different inner/outer radii.

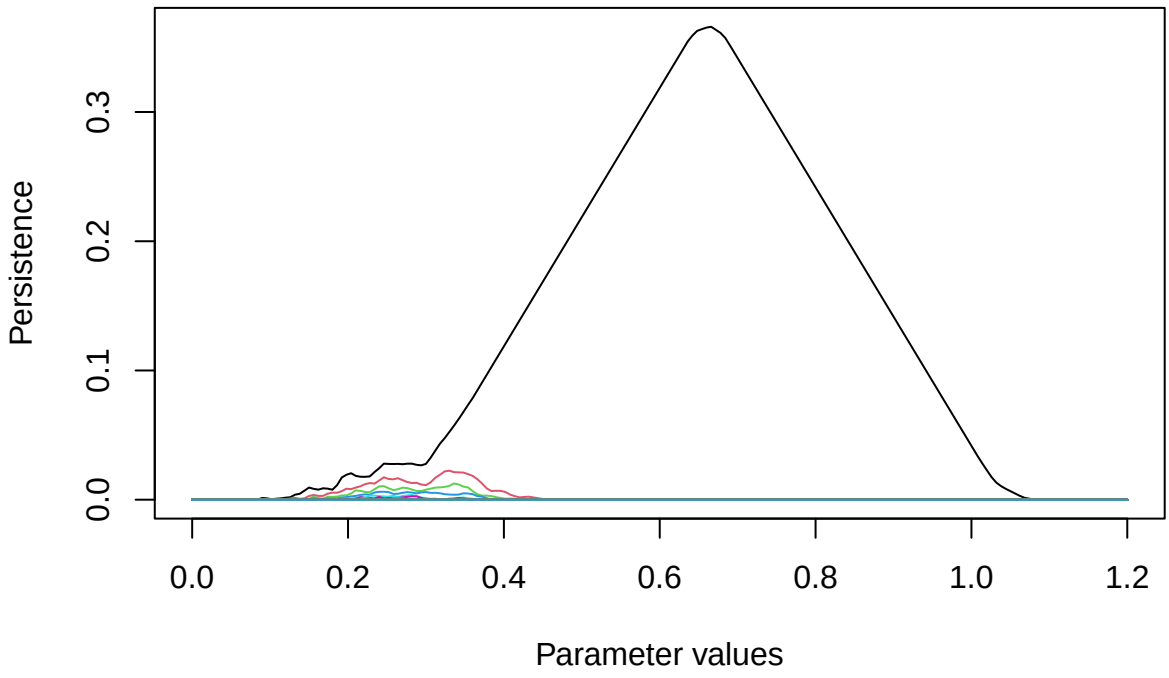
Death vectors, p-values

```
#Reset parameters for sample
num.pts <- 100
inner.radius <- 1
outer.radius <- 2
```

```
#Compute the death vectors and persistence landscapes for 10 samples and average them

num.repeats <- 10

PD.list <- vector("list",num.repeats)
for (c in 1 : num.repeats){
  X <- sample.annulus(num.pts,inner.radius,outer.radius)
  PH.output <- alphaComplexDiag(X)
  PD.list[[c]] <- PH.output[["diagram"]]
  PD.list[[c]][,2:3] <- sqrt(PD.list[[c]][,2:3])
}

DV.matrix <- death.vector.matrix(PD.list)
average.DV <- colMeans(DV.matrix)
plot(average.DV, type="l", col="blue", ylab = "Persistence")
```

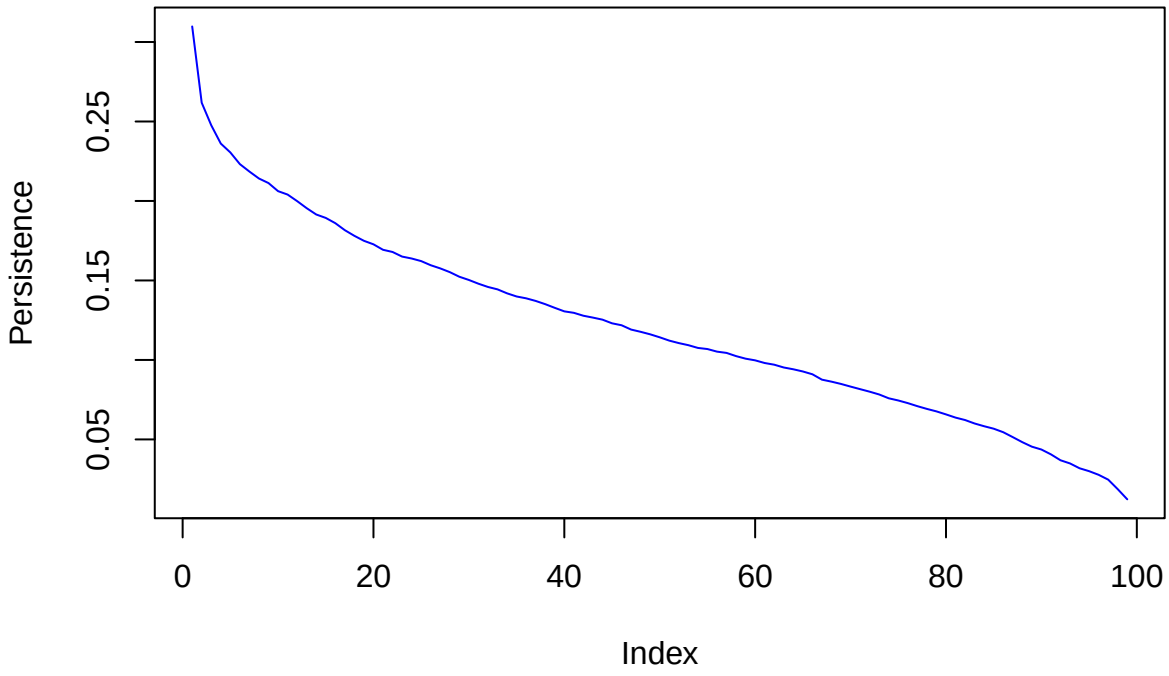



```
PL.list <- vector("list",num.repeats)
for (c in 1 : num.repeats)
  PL.list[[c]] <- t(landscape(PD.list[[c]],dimension=1,KK=1:100,tseq=t.vals))
PL.matrix <- landscape.matrix.from.list(PL.list)
average.PL.vector <- colMeans(PL.matrix, sparseResult = TRUE)
average.PL <- landscape.from.vector(average.PL.vector,t.vals)
plot.landscape(average.PL,t.vals)
```

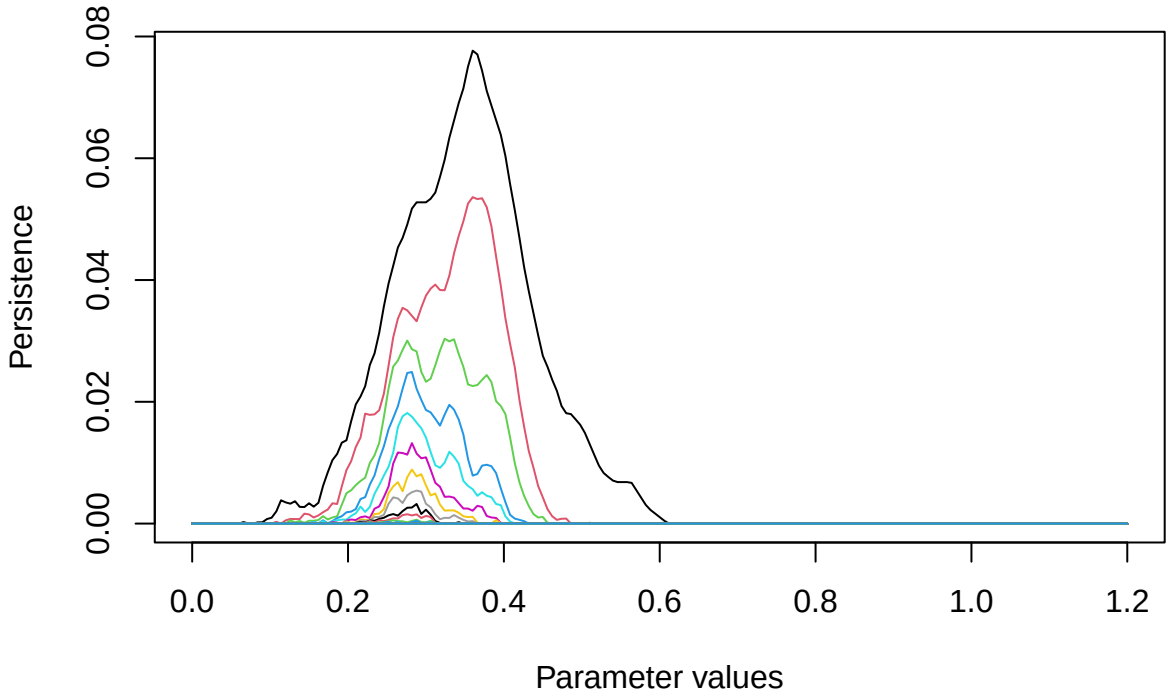



```
#Repeat the computations above with inner.radius = 0.25.

inner.radius <- 0.25
```

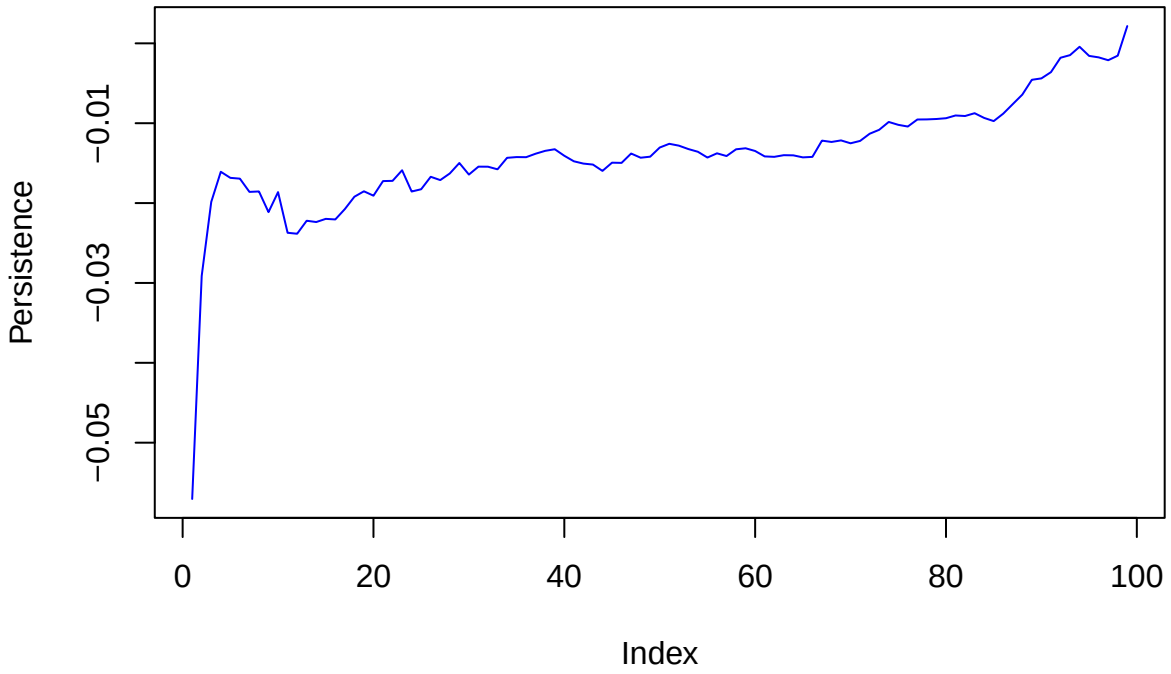
```
PD.list.2 <- vector("list",num.repeats)
for (c in 1 : num.repeats){
  X <- sample.annulus(num.pts,inner.radius,outer.radius)
  PH.output <- alphaComplexDiag(X)
  PD.list.2[[c]] <- PH.output[["diagram"]]
  PD.list.2[[c]][,2:3] <- sqrt(PD.list.2[[c]][,2:3])
}

DV.matrix.2 <- death.vector.matrix(PD.list.2)
average.DV.2 <- colMeans(DV.matrix.2)
plot(average.DV.2, type="l", col="blue", ylab = "Persistence")
```



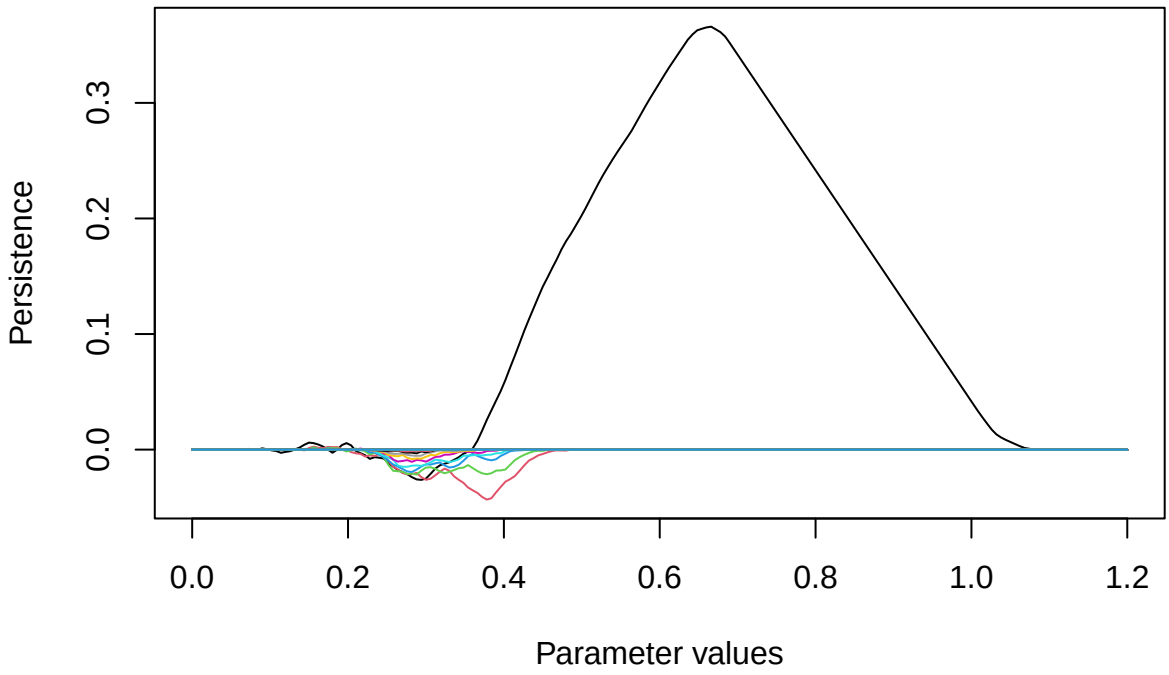

```
PL.list.2 <- vector("list",num.repeats)
for (c in 1 : num.repeats)
  PL.list.2[[c]] <- t(landscape(PD.list.2[[c]],dimension=1,KK=1:100,tseq=t.vals))
PL.matrix.2 <- landscape.matrix.from.list(PL.list.2)
average.PL.vector.2 <- colMeans(PL.matrix.2, sparseResult = TRUE)
average.PL.2 <- landscape.from.vector(average.PL.vector.2,t.vals)
plot.landscape(average.PL.2,t.vals)
```




```
# The difference in death vectors and persistence landscapes
difference.average.DV <- average.DV - average.DV.2
plot(difference.average.DV, type="l", col="blue", ylab="Persistence")
```




```
difference.average.PL.vector <- difference.vectors(average.PL.vector,average.PL.vector.2)
difference.average.PL <- landscape.from.vector(difference.average.PL.vector,t.vals)
plot.landscape(difference.average.PL,t.vals)
```




```
# p values for differences in the average landscapes
permutation.test(DV.matrix,DV.matrix.2)
```

```
## [1] 1e-04
```

```
permutation.test(PL.matrix,PL.matrix.2,num.repeats=1000)
```

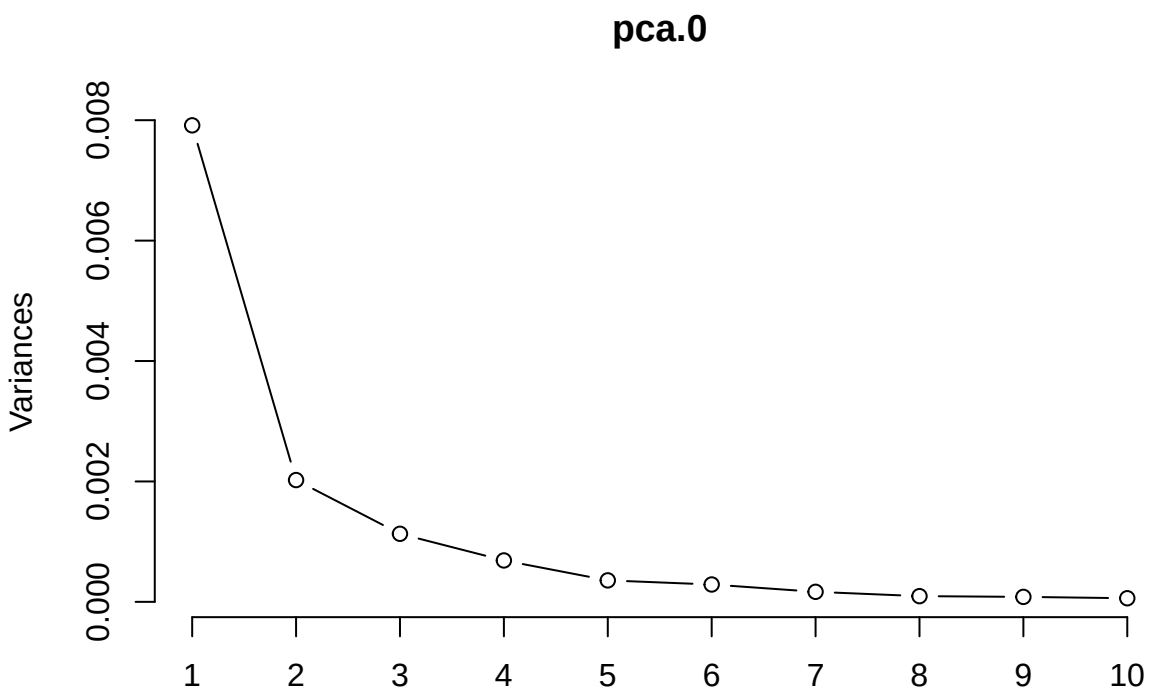
```
## [1] 0
```

Exercises (Part II):

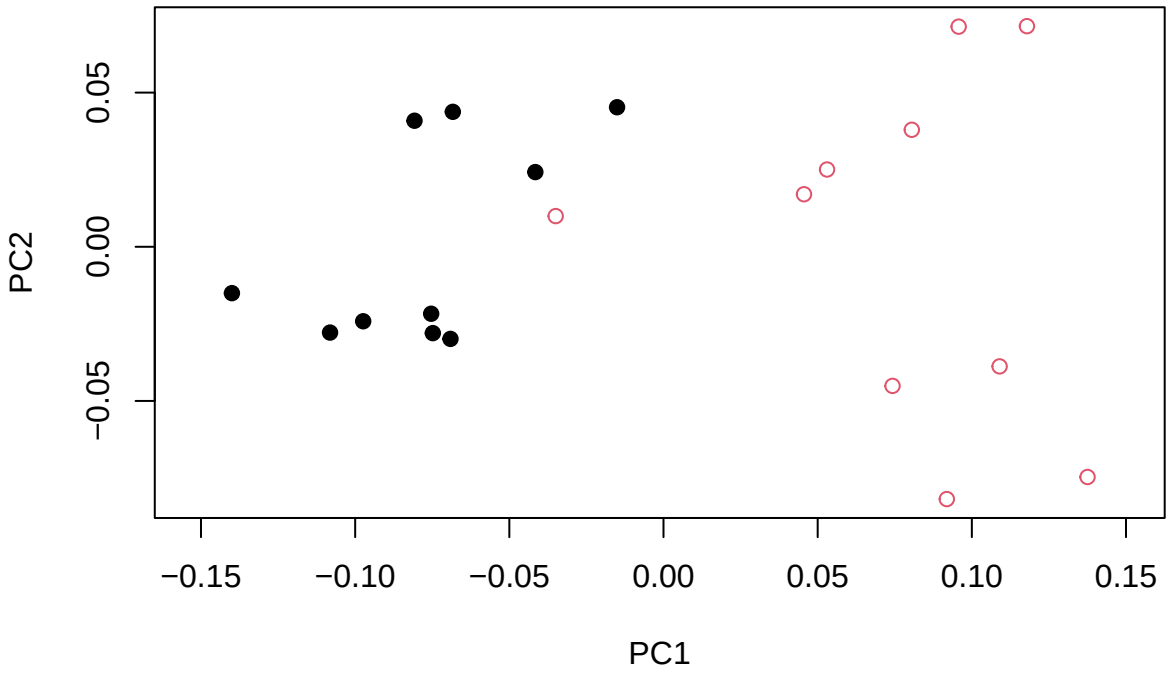
3. Interpret the plot of the difference of the average persistence landscape for inner.radius = 0.5 and inner.radius = 0.25.
4. Interpret the p-value from the permutation tests for the norm of the difference of the feature vectors for the two classes of sampled points.
5. Find a value of inner radius for the second sample class such that the difference of the average death vectors is not significant but the the difference of the average persistence landscapes is.
6. Approximately what inner.radius is at the threshold for which there is a significant difference of the average persistence landscapes?
7. Name one or more ways in which you could change parameters other than the inner.radius and outer.radius to improve the ability of the average persistence landscape to detect the difference between the samples from the two annuli.

PCA

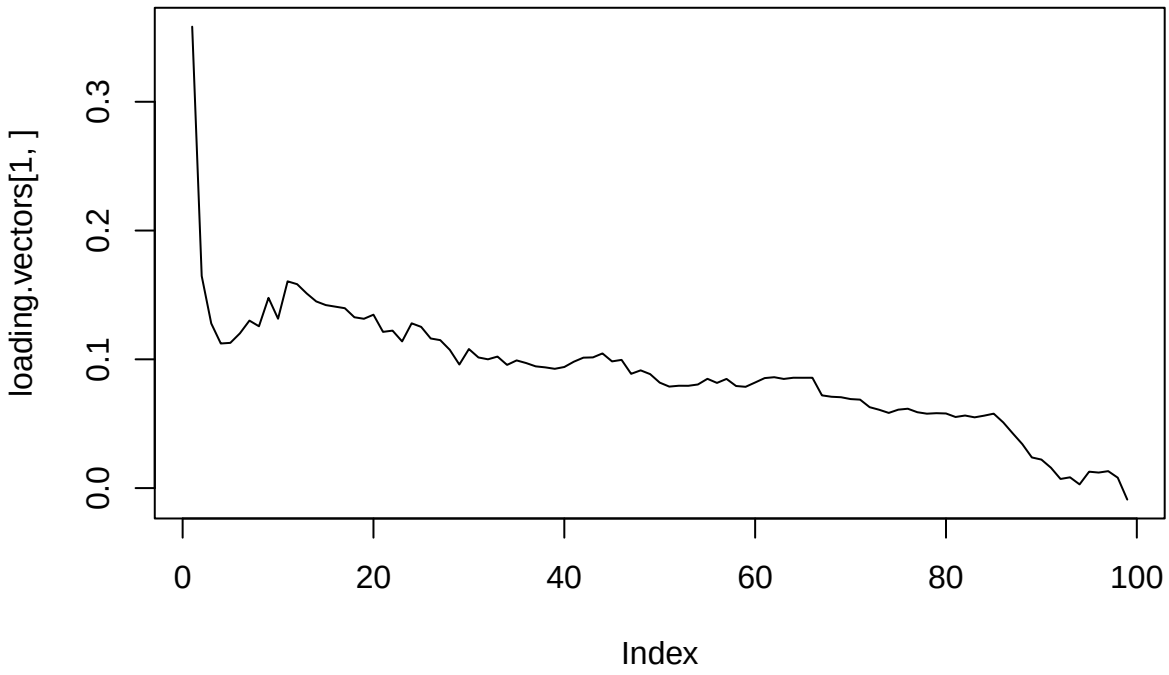
```
#Plot variance, PCA projections, loading vectors, and landscapes
data.labels <- c(rep(1,nrow(PL.matrix)), rep(2,nrow(PL.matrix.2)))
DV.vectors <- rbind(DV.matrix,DV.matrix.2)
pca.0 <- prcomp(DV.vectors,rank=10)
plot(pca.0,type="l")
```



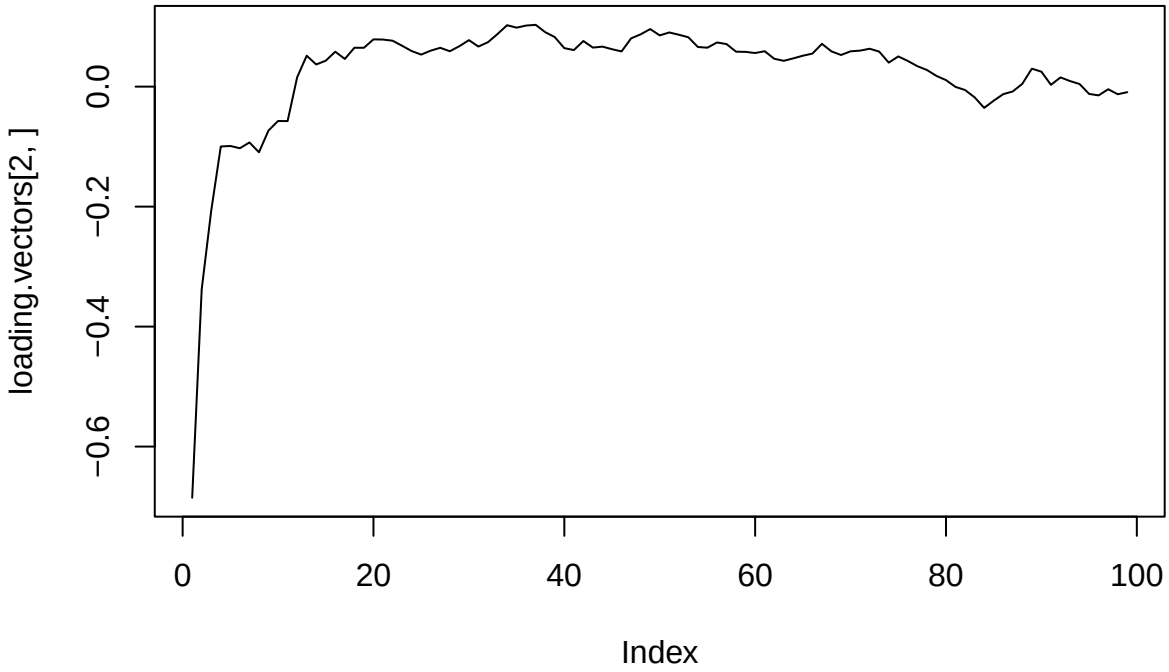

```
plot(pca.0$x[,1:2], col=data.labels, pch=17+(2*data.labels), asp=1)
```



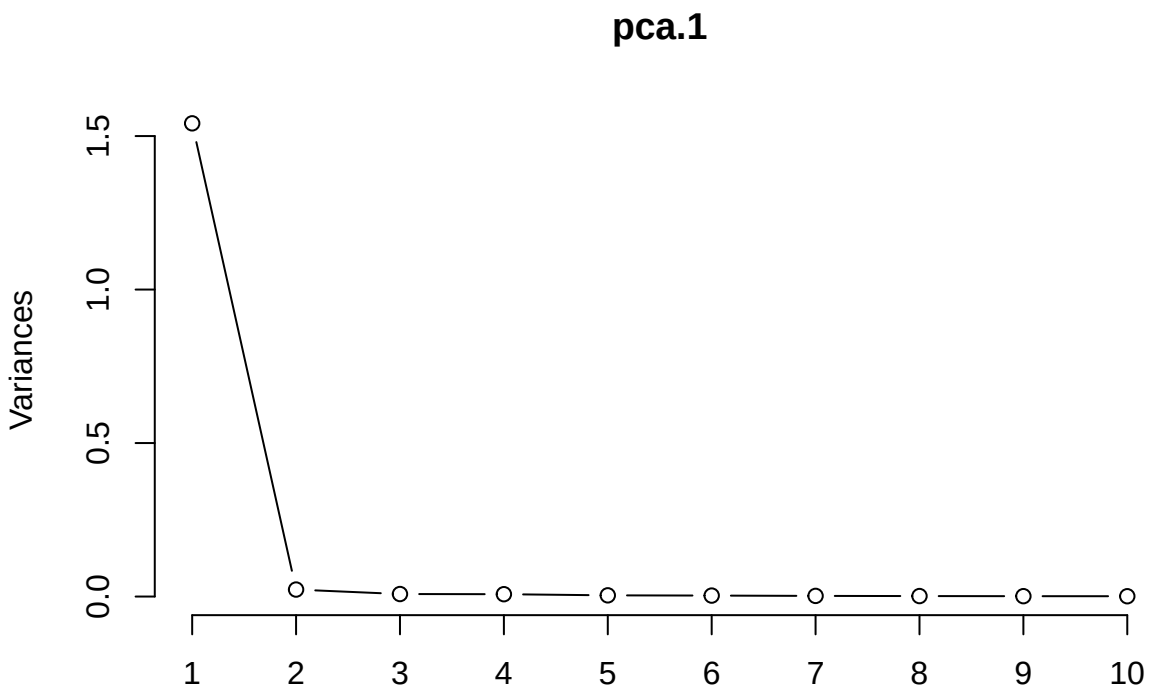

```
loading.vectors <- t(pca.0$rotation)
plot(loading.vectors[1,],type="l")
```



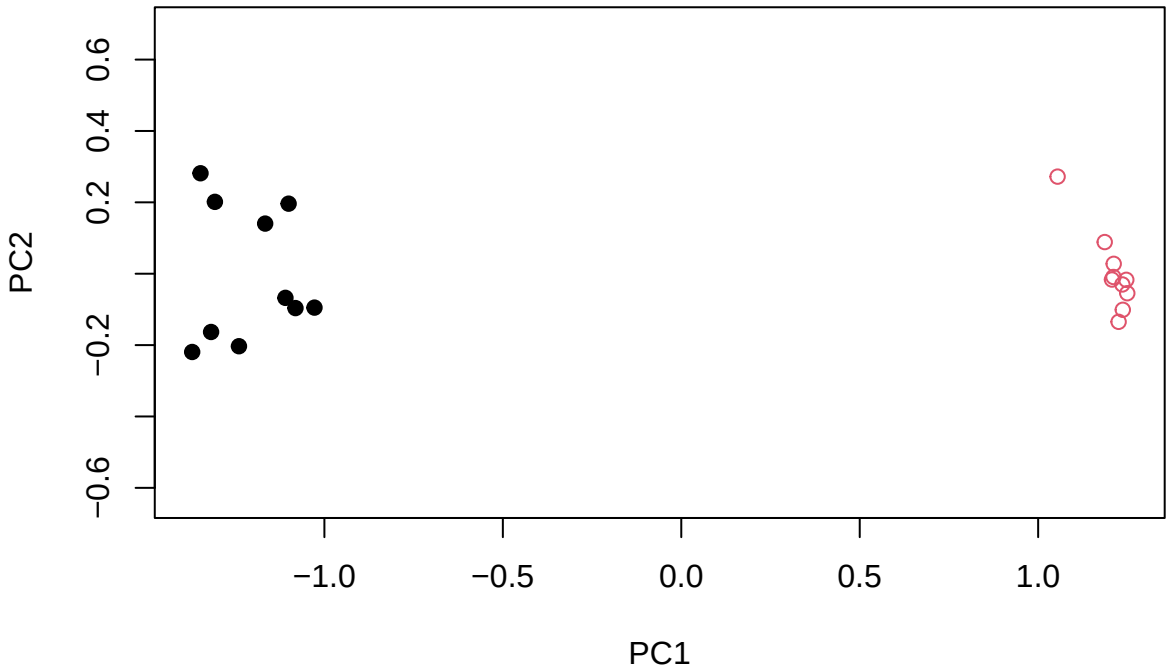

```
plot(loading.vectors[2,],type="l")
```



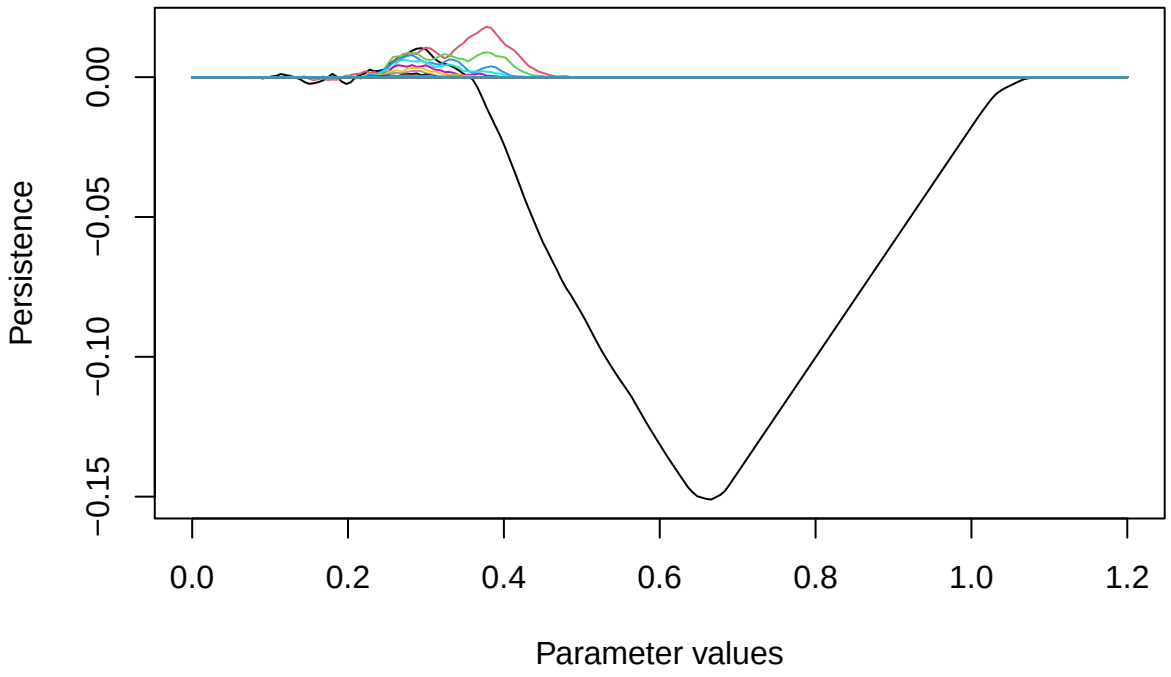

```
PL.vectors <- rbind(PL.matrix,PL.matrix.2)
pca.1 <- prcomp(PL.vectors,rank=10)
plot(pca.1,type="l")
```



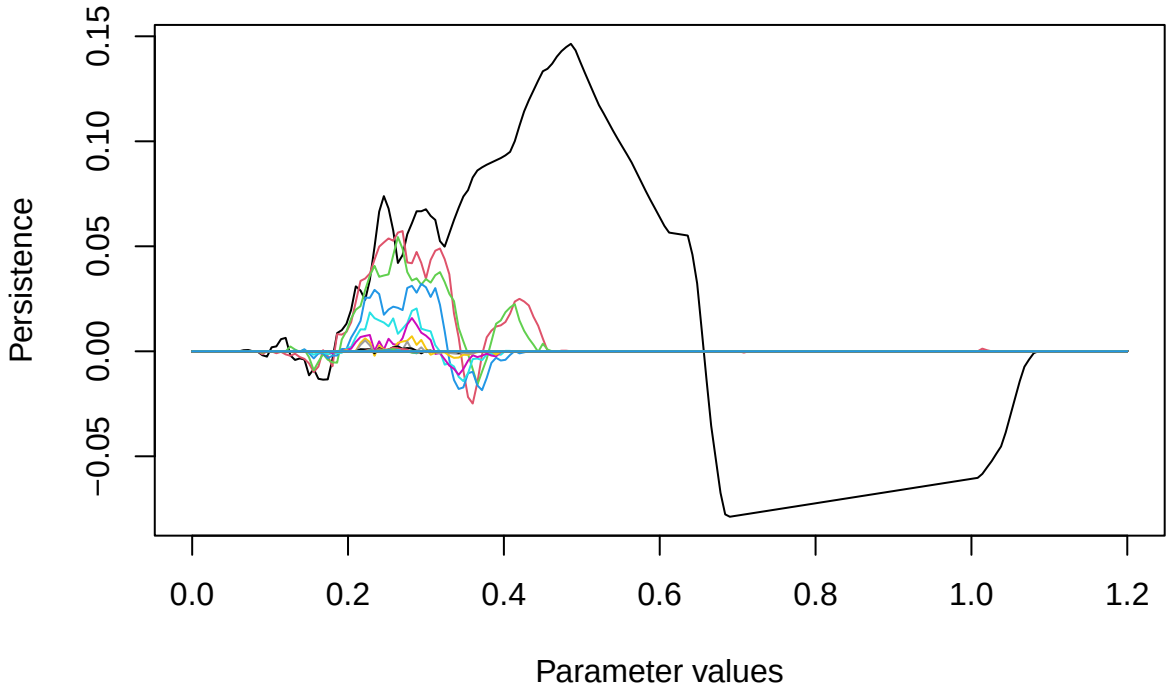

```
plot(pca.1$x[,1:2], col=data.labels, pch=17+(2*data.labels), asp=1)
```




```
loading.vectors <- t(pca.1$rotation)
plot.landscape(landscape.from.vector(loading.vectors[1,],t.vals),t.vals)
```




```
plot.landscape(landscape.from.vector(loading.vectors[2,],t.vals),t.vals)
```



Exercises (Part III):

8. Compare and contrast the Explained Variance Plot for PCA for persistent homology in degree 0 (death vectors) and persistence homology in degree 1 (persistence landscapes).
9. Compare and contrast the 2D PCA plots for persistent homology in degree 0 and degree 1.
10. Interpret the first loading vector for PCA on the Persistence Landscapes in degree 1.
11. Compare the first loading vector for PCA with the difference of the averages. Should you expect these to be simliar?
12. Change the inner radius of the second set of samples to be similar to the first and repeat.