Construction of a Delaunay complex, Vietoris-Rips complex, and Persistence Diagrams. Plotting representative cycles.

Topological Data Analysis with R - Lab 1

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

Instructions:

This is an R Markdown document. I will assume that you are using RStudio.

When you execute code within the notebook, the results appear beneath the code. To execute a chunk of code click the *Run* button within the chunk or place your cursor inside it and press *Cmd+Shift+Enter*.

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file). The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

Try to understand all of the commands. Use the R Help (bottom right) to learn about the commands being used. All text after the '#' symbol is ignored.

The following three commands only need to be run once to install the required R packages on your system. Remove the # symbol at the start of each line, run the commands, and then add the "#" symbol to the start of each line.

```
#install.packages("TDA") # R Topological Data Analysis package
#install.packages("deldir") # R Delaunay and Voronoi package
#install.packages("kernlab") # R Support Vector Machine package
```

```
# The following commands load the packages' functions into memory.
library(TDA)
library(deldir)
```

In the next chunk we define a function. The input to the function is on the right hand side of the first line. By default, the function will return the output of the last command. Executing the function loads it into memory, ready to be used.

Once you've executed the function, look for it in the Environment in the top right panel. If you want to start over, you can clear the environment using the broom icon.

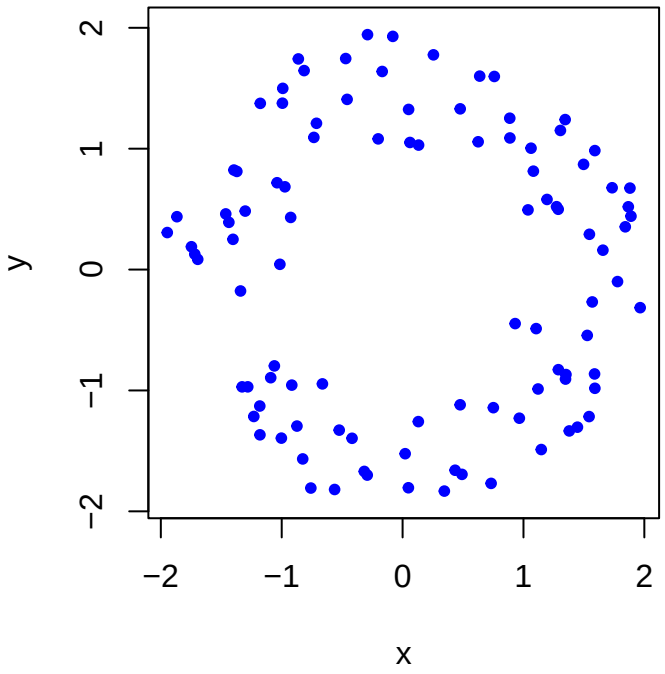
```
sample.annulus <- function(num.pts, inner.radius, outer.radius){
  theta <- runif(num.pts) * 2 * pi
  radius <- sqrt(runif(num.pts, inner.radius^2, outer.radius^2))
  x <- radius * cos(theta)
  y <- radius * sin(theta)
  cbind(x,y)
}
```

The next line runs the function we just defined with specified input and assigns the output to a variable.

Once you've assigned a value to this variable, look for it in the Environment. You can click on the table icon to inspect the data.

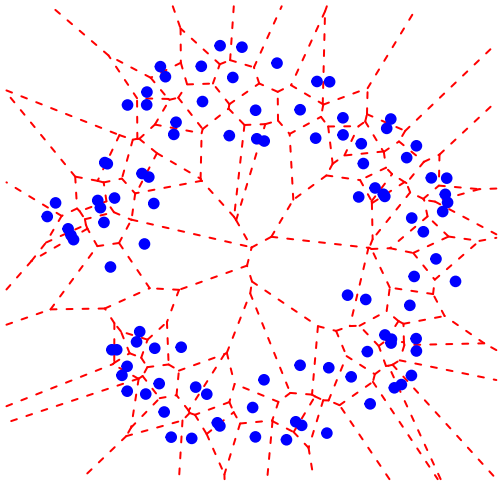
Next we plot the data. There are many plotting options. You can learn more about them using the help using the Help window on the bottom right panel, searching for 'plot' and clicking on 'base::plot'.

```
par(pty="s") # force the plotting region to be square
X <- sample.annulus(num.pts = 100,inner.radius = 1,outer.radius = 2)
plot(X, pch=20, col='blue', asp=1)
```

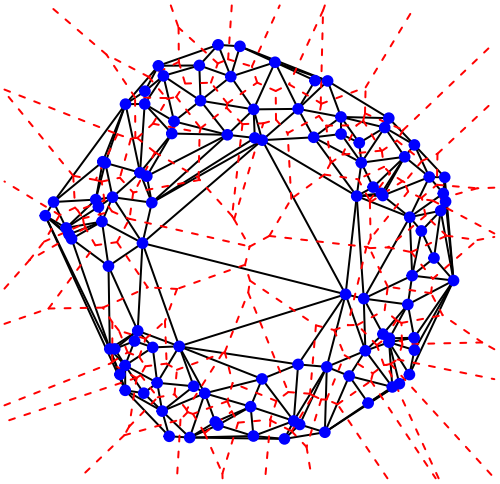


The remainder of the lab follows a similar pattern. When you run a block of code, you should read each line to understand how the function or script is defined.

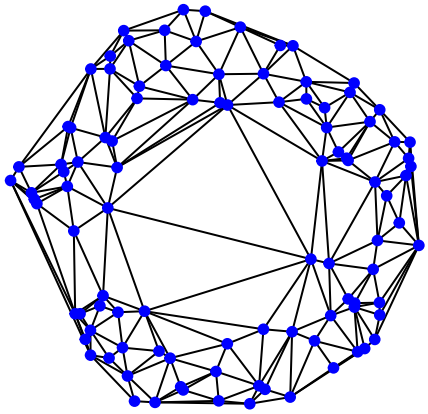
```
DelVor <- deldir(X[,1], X[,2], suppressMsge = TRUE)

# plot the Voronoi tesselation:
plot(DelVor, pch=20, cmpnt_col=c('black','red','blue'), wlines= ('tess'))
```



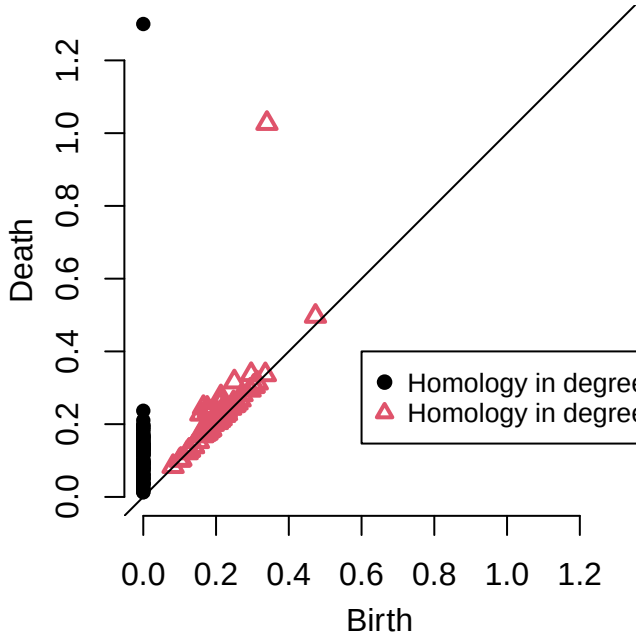
```
# plot the Voronoi cells and their dual (the Delaunay complex):
plot(DelVor, pch=20, cmpnt_col=c('black','red','blue'), wlines= ('both'))
```



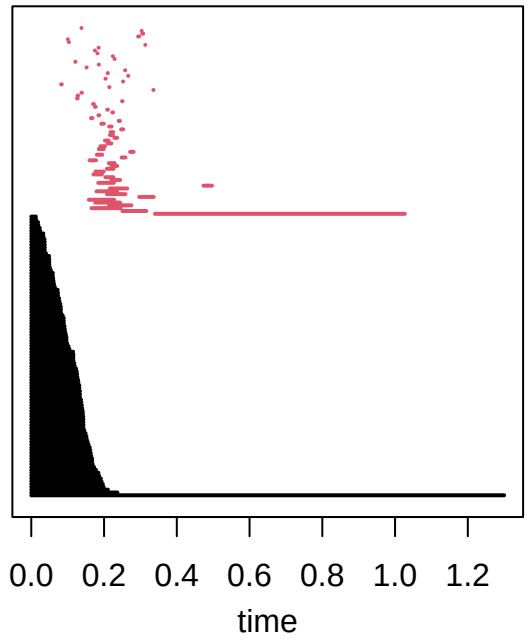
```
# plot just the Delaunay complex:
plot(DelVor, pch=20, cmpnt_col=c('black','red','blue'), wlines= ('triang'))
```



```
par(pty="s") # force the plotting region to be square
PH.output <- alphaComplexDiag(X)

# Get persistence diagram from output of persistent homology computation
PD <- PH.output[["diagram"]]

# birth and death values have been squared for some reason, so take the square root
PD[,2] <- sqrt(PD[,2])
PD[,3] <- sqrt(PD[,3])

# plot the persistence diagram
plot(PD, asp=1, diagLim = c(0,1.3))
legend(0.6, 0.4, c('Homology in degree 0','Homology in degree 1'),
       col = c(1,2), pch = c(19,2), cex = .8, pt.lwd = 2)
```

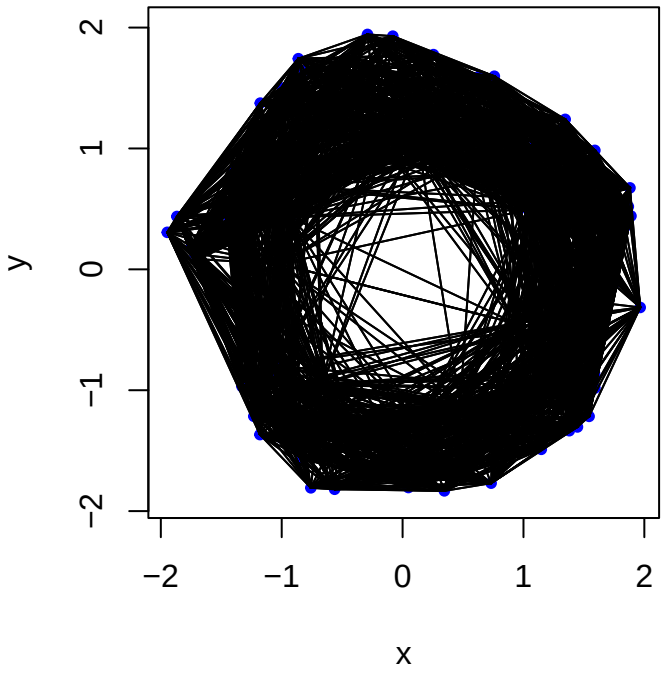



```
# plot the bar code
plot(PD, diagLim = c(0,1.3), barcode=TRUE)
```

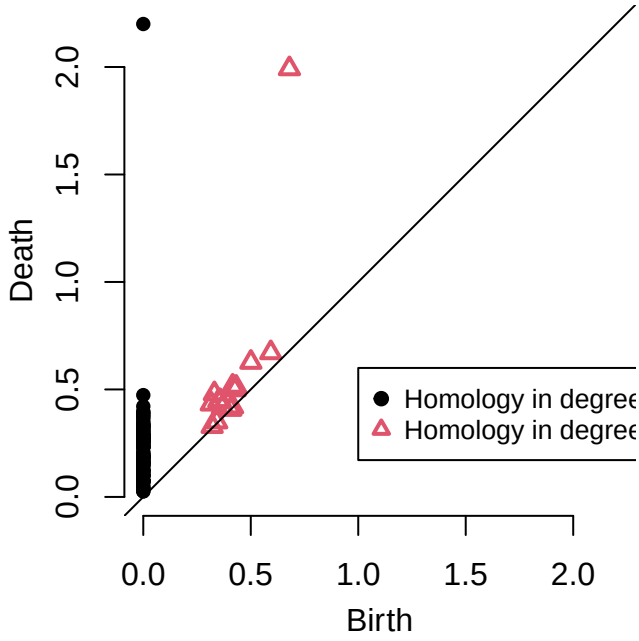



```
par(pty="s") # force the plotting region to be square
eucl.dist <- function(u, v) sqrt(sum((u - v) ^ 2))
max.filtration <- 2.2

# Plot the Vietoris-Rips complex with distance parameter equal to max.filtration
plot(X, pch=20, col='blue', asp=1)
num.pts <- dim(X)[1]
for(i in 1:num.pts)
  for(j in 1:num.pts)
    if (eucl.dist(X[i,],X[j,]) < max.filtration)
```

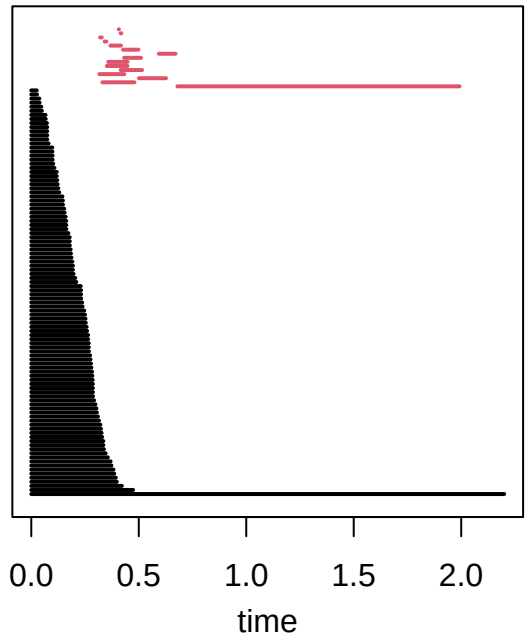
```
      lines(rbind(X[i,],X[j,]))
```




```
par(pty="s") # force the plotting region to be square

# Compute the persistent homology
PH.output <- ripsDiag(X, maxdimension = 1, maxscale = max.filtration)

# Get persistence diagram from output of persistent homology computation
PD <- PH.output[["diagram"]]

# plot the persistence diagram
plot(PD, asp=1, diagLim = c(0, max.filtration))
legend(1, 0.6, c('Homology in degree 0','Homology in degree 1'), col = c(1,2), pch = c(19,2), cex = .8,
```

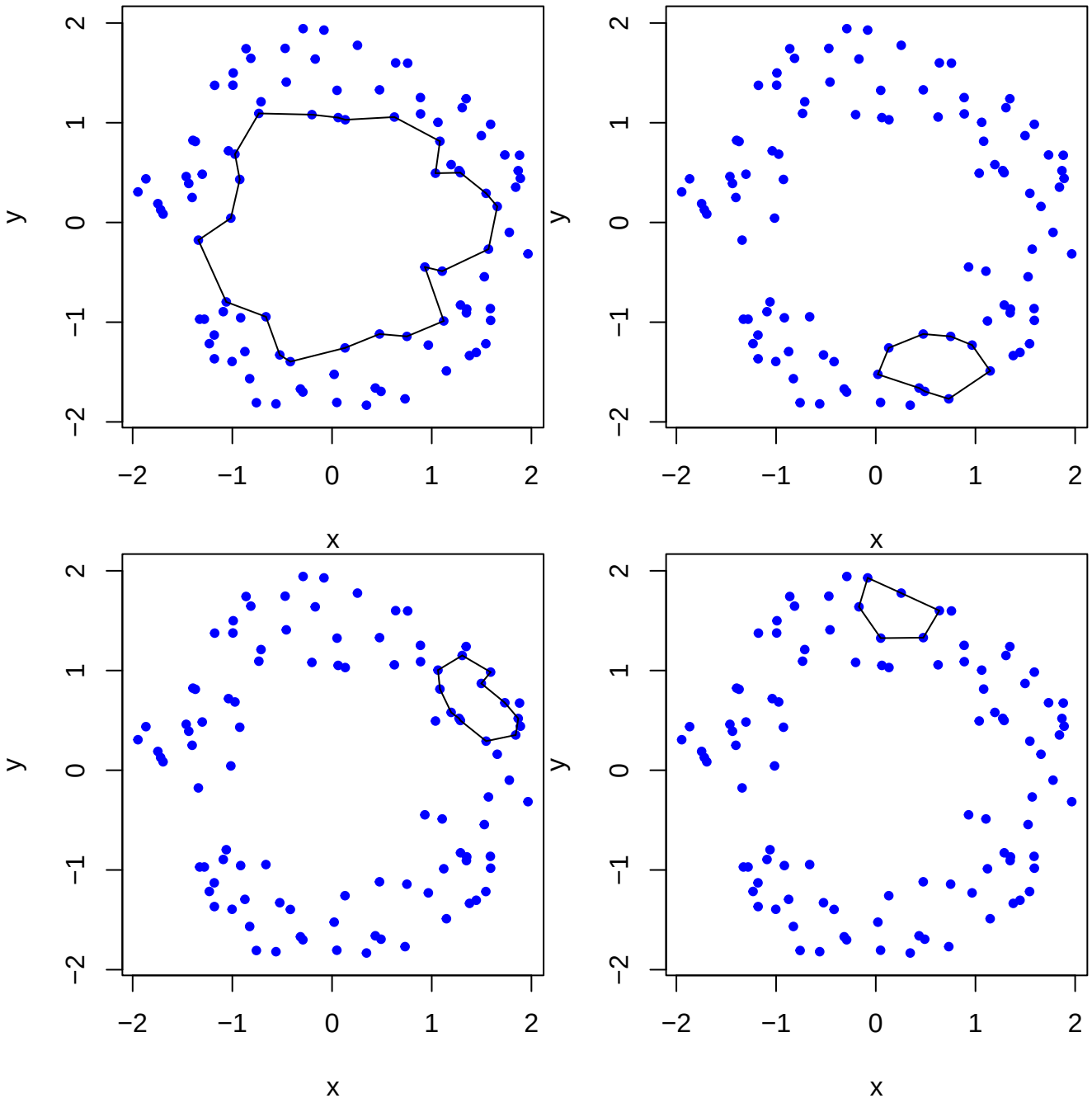



```
# plot the bar code
plot(PD, diagLim = c(0, max.filtration), barcode=TRUE)
```



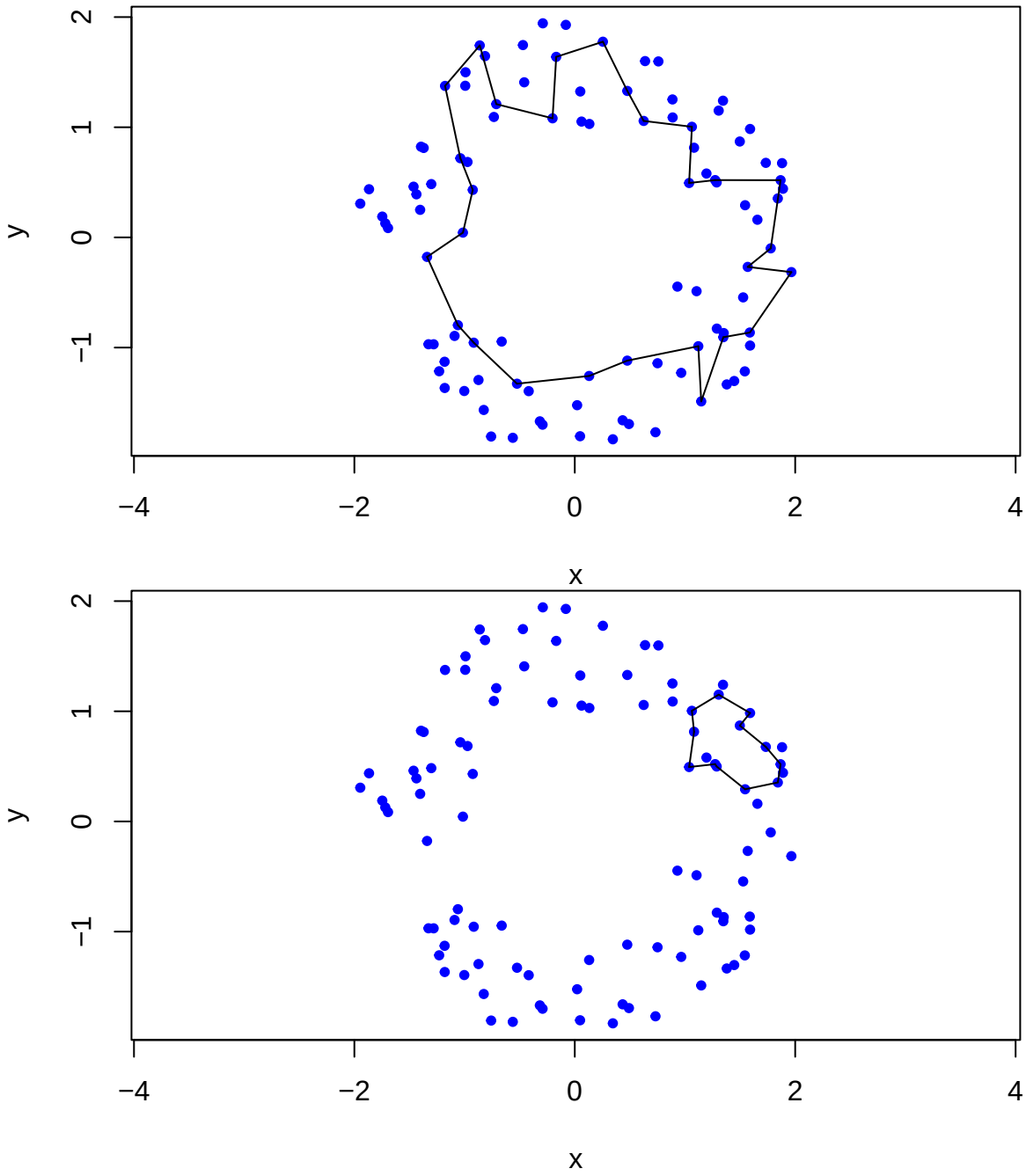

```
par(pty="s") # force the plotting region to be square
PH.output <- alphaComplexDiag(X, maxdimension = 1, library = c("GUDHI", "Dionysus"), location = TRUE)
PD <- PH.output[["diagram"]]
ones <- which(PD[, 1] == 1)
persistence <- PD[ones,3] - PD[ones,2]
cycles <- PH.output[["cycleLocation"]][ones[order(persistence, decreasing=TRUE)]]
for (i in 1:(min(4,length(cycles)))){
  plot(X, pch=20, col='blue', asp=1)
  for (j in 1:dim(cycles[[i]])[1])
    lines(cycles[[i]][j,,])
```

```
}
```

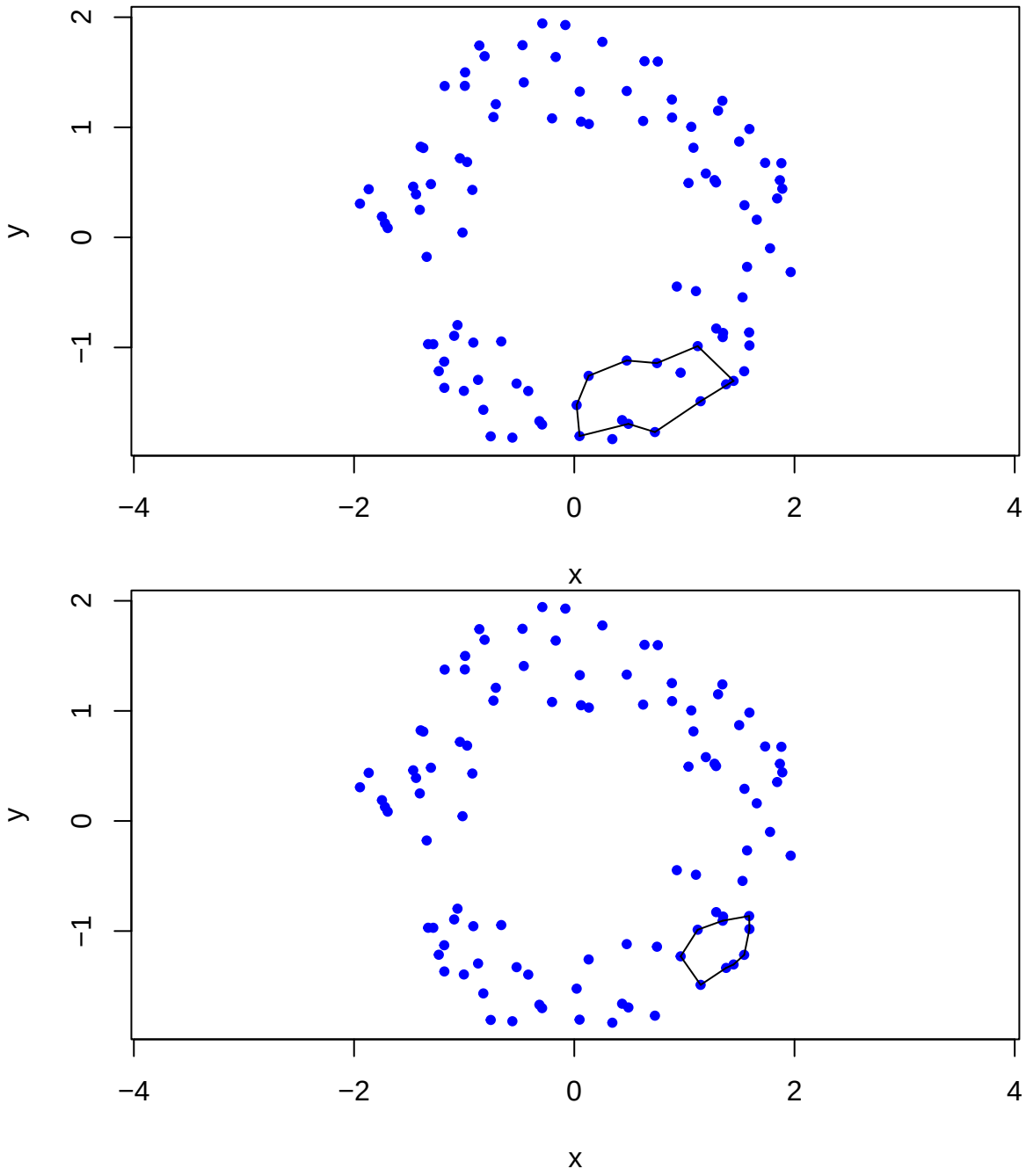



```
PH.output <- ripsDiag(X, maxdimension = 1, maxscale = max.filtration, library = c("GUDHI", "Dionysus"),
PD <- PH.output[["diagram"]]
ones <- which(PD[, 1] == 1)
persistence <- PD[ones,3] - PD[ones,2]
cycles <- PH.output[["cycleLocation"]][ones[order(persistence, decreasing=TRUE)]]
for (i in 1:(min(4,length(cycles)))){
  plot(X, pch=20, col='blue', asp=1)
  for (j in 1:dim(cycles[[i]])[1])
    lines(cycles[[i]][j,,])
```

```
}
```


y
2
1
0
−1
−4
−2
0
2
4
x

y
2
1
0
−1
−4
−2
0
2
4
x



Exercises

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

1. Figure out what all of the pieces of function sample.annulus do.
2. How does the homology class with the largest representative cycle differ from all of the other homology classes?
3. Add Gaussian noise to each coordinate of each of the sampled points. Hint: Look up the function 'rnorm' in the Help browser and recall the 'matrix' command from the 'Introduction to R' tutorial.
4. Compare how the shape of the sampled points changes and how the persistence diagram changes for standard deviations 0, 0.1, 0.2,0.3, 0.4, and 0.5.
5. Sample points from the "flat" torus in R^4, whose first two coordinates and last two coordinates are

independently obtained by sampling from a circle. Compute the persistence diagram of the Vietoris-Rips filtration.

Additional remark(s)

If you've made changes to your file then save it.