Low-dimensional Numerical Examples

Luis Ángel Rodríguez García

2022-09-18

Numerical experiments are important to uncover hidden structures in the data to demonstrate that psc-SNE does a good job on the task of dimension reduction. Then, this document contains information about the simulation of low-dimensional data and defines different cases in each point.

First, we start with $p = 2$ and $r = 1$ where $(\mathbb{S}^p)^r$.

$\mathbb{S}^2$

Let us introduce some different scenarios where each of them has some particular distributions for $\mathbb{S}^2$.

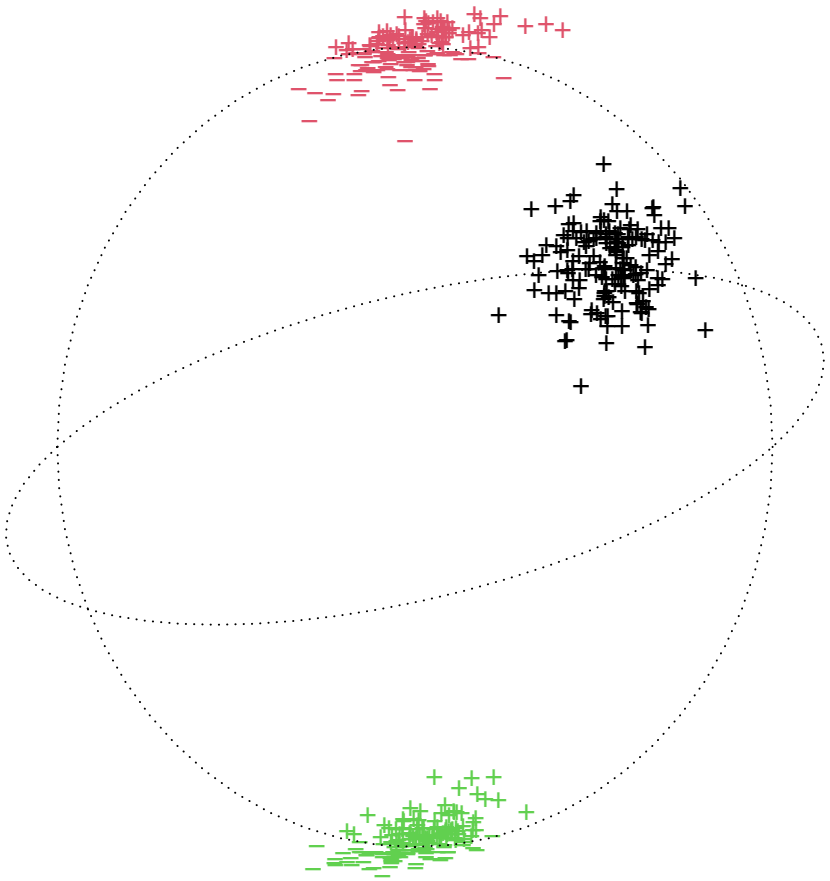
A mixture of von Mises–Fisher producing separated clusters

This scenario works with a data points that come from a mixture which is form by components that follow a von Mises–Fisher distribution. These components produces three different clusters.

```
MvMF_mix <- function(n, n1, n2, n3, mu_mat, kappa = 50) {
  if ((n1 + n2 + n3) != n) {
    stop("(n1 + n2 + n3) must sum 1")
  }
  r_1 <- rotasym::r_vMF(n1, mu = mu_mat[, 1], kappa = kappa)
  r_2 <- rotasym::r_vMF(n2, mu = mu_mat[, 2], kappa = kappa)
  r_3 <- rotasym::r_vMF(n3, mu = mu_mat[, 3], kappa = kappa)
  r_mvmf <- rbind(r_1, r_2, r_3)
  colors <- c(rep(1, n1), rep(2, n2), rep(3, n3))
  return(list("data" = array(r_mvmf, dim = c(dim(r_mvmf), 1)), "colors" = colors))
}
n <- 500
n1 <- ceiling(n / 3)
n2 <- ceiling(n / 3)
n3 <- floor(n / 3)
p <- 2
kappa <- 100
mu_mat <- cbind(c(0, 1, 0), c(0, 0, 1), c(0, 0, -1))
set.seed(42)
mvmf_mix_res <- MvMF_mix(n = n, n1 = n1, n2 = n2, n3 = n3,
                         mu_mat = mu_mat, kappa = kappa)
mvmf_mix_data <- mvmf_mix_res$data
mvmf_mix_colors <- mvmf_mix_res$colors
# shuffle the sample
set.seed(42)
indexes <- sample(1:n)
mvmf_mix_data <- mvmf_mix_data[indexes, , , drop = FALSE]
mvmf_mix_colors <- mvmf_mix_colors[indexes]
```

Since data lie onto the sphere, $\mathbb{S}^2$, then it is possible to see how the distributions involved are graphically:

```
seq_rad <- seq(-pi, pi, by = pi / 30)
meridian <- do.call(rbind, lapply(seq_rad, function(i) c(0, i)))
equator <- do.call(rbind, lapply(seq_rad, function(i) c(i, pi / 2)))
sd3 <- scatterplot3d::scatterplot3d(
  mvmf_mix_data[, , 1], xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  color = mvmf_mix_colors, xlab = "", ylab = "", zlab = "", axis = FALSE,
  pch = c("+", "-")[ifelse(sign(mvmf_mix_data[, 2, 1]) == 1, 1, 2)],
  mar = c(0,0,0,0), grid = FALSE
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 3)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 3)
```

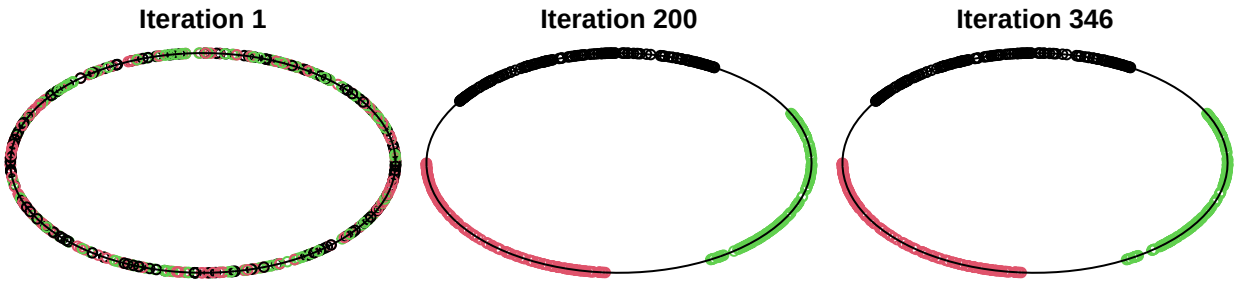


It is possible to see now that the von Mises–Fisher distribution colored with black is approximately located on the north pole and the remainder distributions, that are independently distributed, colored with red and green are placed in the equator, having some points in the south hemisphere and others in the north one. Let's now run the `psc_sne` and see if the algorithm identifies each component of the mixture. But first, let's calculate the rho values based on a perplexity of 30.

```
rho_30_1 <- rho_optim_bst(x = mvmf_mix_data, perp_fixed = 30,
                          num_cores = num_cores_param)
#> Time difference of 3.8693 secs
```

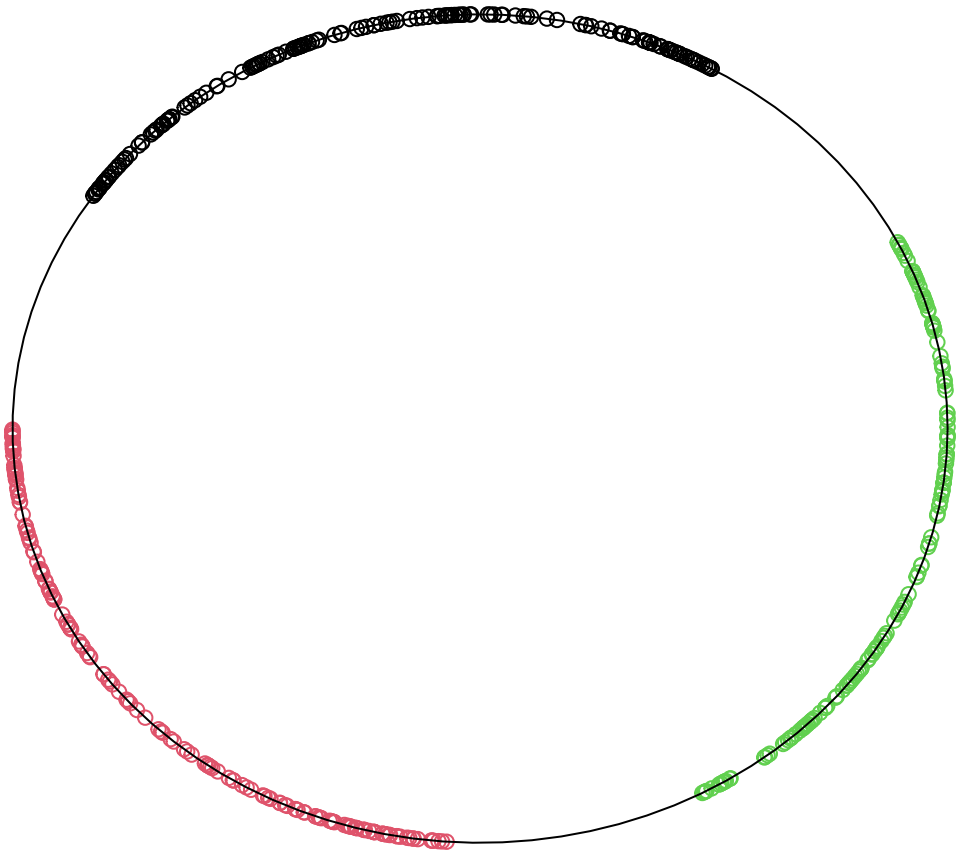
Reduction to $d = 1$ After calculating the rho values, the `psc_sne` is executed with the parameter $d = 1$:

```
res_pscsne_11 <- psc_sne(X = mvmf_mix_data, d = 1, rho_psc_list = rho_30_1,
                         show_prog = TRUE, colors = mvmf_mix_colors, eta = 50,
                         parallel_cores = num_cores_param)
#> It: 1 (best: 1); obj: 1.74e+01 (best: 1.74e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 7.2e-02; mom: 0.0e+00
#> It: 100 (best: 16); obj: 1.29e+01 (best: 1.28e+01); abs: 4.7e-04; rel: 3.7e-05; norm: 1.5e-01; mom: 2.5e+00
#> It: 200 (best: 200); obj: 1.72e+00 (best: 1.72e+00); abs: 1.3e-07; rel: 7.8e-08; norm: 3.7e-05; mom: 1.9e-03
#> It: 300 (best: 300); obj: 1.72e+00 (best: 1.72e+00); abs: 6.5e-09; rel: 3.8e-09; norm: 4.8e-06; mom: 1.1e-03
#> It: 346 (best: 346); obj: 1.72e+00 (best: 1.72e+00); abs: 2.7e-10; rel: 1.6e-10; norm: 9.7e-07; mom: 2.3e-04
#> CONVERGENCE!
```



The best result related with the smallest value of the objective function is:

```
Y <- res_pscsne_11$best_Y
plot(Y[, 1], Y[, 2], col = mvmf_mix_colors, xlim = c(-1, 1), ylim = c(-1, 1),
     xlab = "", ylab = "", axes = FALSE)
th <- seq(0, 2 * pi, length.out = 100)
polygon(x = cos(th), y = sin(th))
```

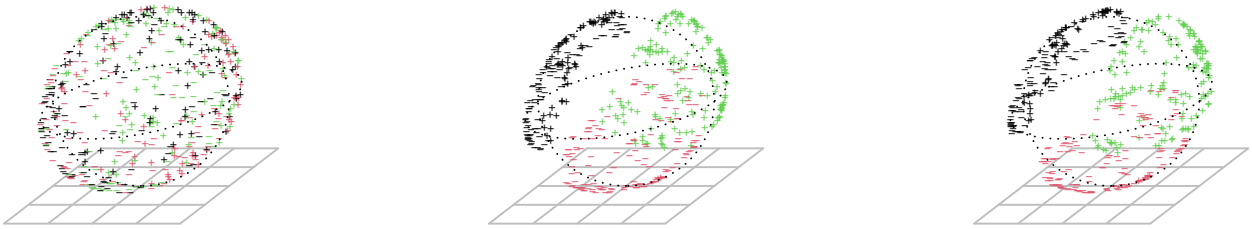


Results are good although not what is expected since the red distribution from the mixture is split in twice. This can be related to the crowding problem that occurs when reducing from a higher dimension to a small

Reduction to $d = 2$ Let's execute `psc_sne` with the parameter $d = 2$ and see if results are more satisfactory:

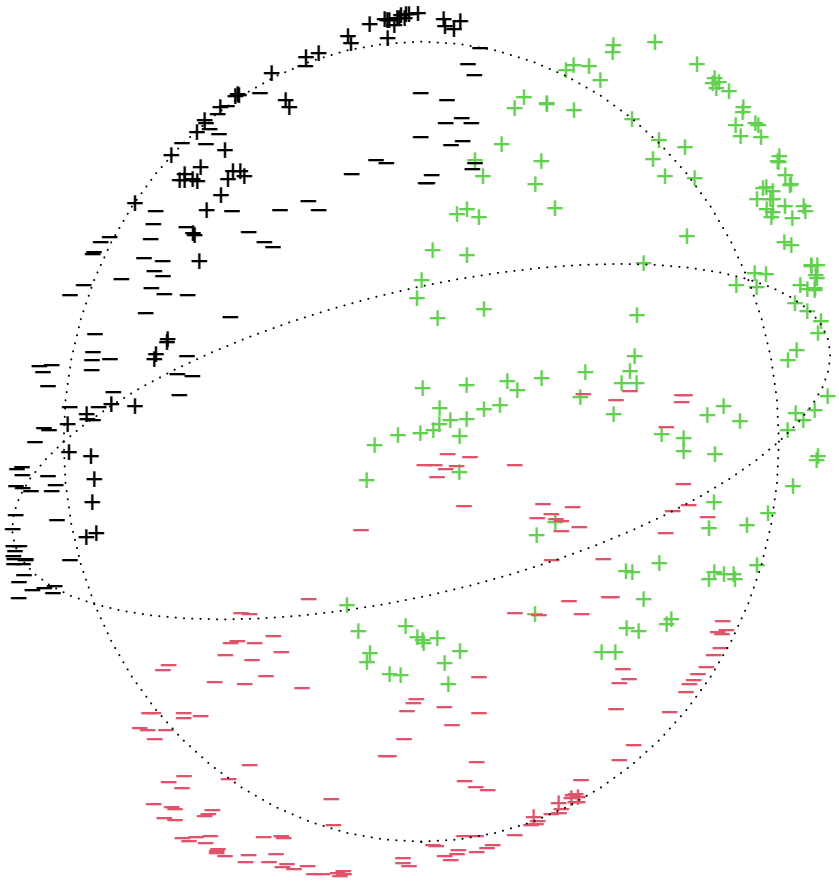
```
res_pscsne_12 <- psc_sne(X = mvmf_mix_data, d = 2, rho_psc_list = rho_30_1,
                         show_prog = TRUE, colors = mvmf_mix_colors,
                         parallel_cores = num_cores_param, eta = 100)
#> It: 1 (best: 1); obj: 1.76e+01 (best: 1.76e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 1.5e-01; mom: 0.0e+00
#> It: 100 (best: 81); obj: 1.44e+01 (best: 1.33e+01); abs: 1.1e+00; rel: 7.9e-02; norm: 4.9e-01; mom: 1.0e+01
#> It: 200 (best: 115); obj: 9.21e-01 (best: 9.18e-01); abs: 6.9e-06; rel: 7.5e-06; norm: 2.8e-02; mom: 9.1e-01
#> It: 300 (best: 300); obj: 9.05e-01 (best: 9.05e-01); abs: 6.5e-09; rel: 7.2e-09; norm: 7.9e-05; mom: 4.2e-03
#> It: 340 (best: 340); obj: 9.05e-01 (best: 9.05e-01); abs: 3.1e-12; rel: 3.5e-12; norm: 9.4e-07; mom: 5.2e-05
#> CONVERGENCE!
```

Iteration 1 **Iteration 200** **Iteration 340**



It is time now to see the best results onto the sphere:

```
Y <- res_pscsne_12$best_Y
sd3 <- scatterplot3d::scatterplot3d(
  Y, xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  color = mvmf_mix_colors, xlab = "", ylab = "", zlab = "", axis = FALSE,
  pch = c("+", "-")[ifelse(sign(Y[, 2]) == 1, 1, 2)], grid = FALSE,
  mar = c(0, 0, 0, 0)
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 3)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 3)
```



In this case, the mixture is properly placed since each data point is located in its original cluster.

Mixtures of von Mises–Fisher with a spherical spiral

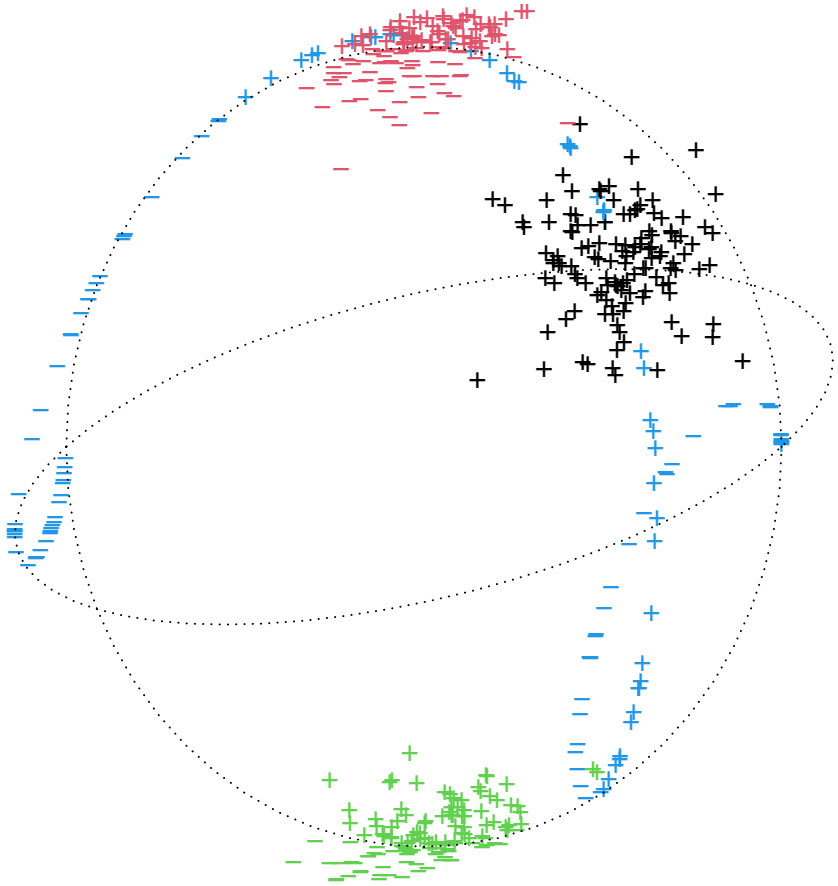
A mixture of von Mises–Fisher along a spherical spiral from north to south is defined using the function created in the previous point and the another function that generates data along a spherical spiral.

```
n <- 500
n1 <- ceiling(n / 4)
n2 <- ceiling(n / 4)
n3 <- floor(n / 4)
n4 <- floor(n / 4)
kappa <- 50
north_pole <- cbind(c(0, 0, 1))
set.seed(42)
spiral_sph <- r_path_s2r(n = n4, r = 1, c = 3, spiral = TRUE, sigma = 0.01,
                         Theta = north_pole)
mu_mat <- cbind(c(0, 1, 0), c(0, 0, 1), c(0, 0, -1))
set.seed(42)

mvmf_mix_res <- MvMF_mix(n = n1 + n2 + n3, n1 = n1, n2 = n2, n3 = n3,
                         mu_mat = mu_mat, kappa = kappa)
mvmf_mix_data <- mvmf_mix_res$data
mvmf_mix_colors <- mvmf_mix_res$colors
mvmf_spiral_mix_data <- abind(mvmf_mix_data, spiral_sph, along = 1)
mvmf_spiral_mix_colors <- c(mvmf_mix_colors, rep(4, times = n4))
# shuffle the sample
```

```
set.seed(42)
indexes <- sample(1:n)
mvmf_spiral_mix_data <- mvmf_spiral_mix_data[indexes, , , drop = FALSE]
mvmf_spiral_mix_colors <- mvmf_spiral_mix_colors[indexes]
```

Since data lie onto the sphere, $\mathbb{S}^2$, then it is possible to see how the distributions involved are graphically:

```
sd3 <- scatterplot3d::scatterplot3d(
  mvmf_spiral_mix_data[, , 1], xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  color = mvmf_spiral_mix_colors, xlab = "", ylab = "", zlab = "", axis = FALSE,
  pch = c("+", "-")[ifelse(sign(mvmf_spiral_mix_data[, 2, 1]) == 1, 1, 2)],
  mar = c(0,0,0,0), grid = FALSE
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 3)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 3)
```

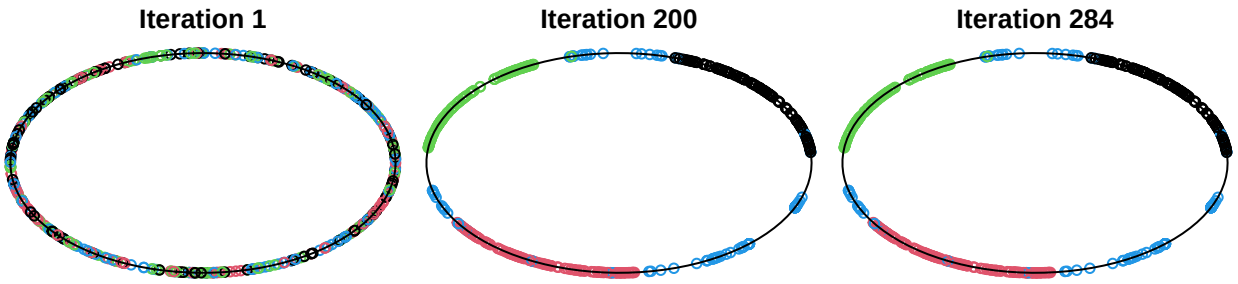


Let's calculate the rho based on a perplexity of 30:

```
rho_30_2 <- rho_optim_bst(x = mvmf_spiral_mix_data, perp_fixed = 30,
                          num_cores = num_cores_param)
#> Time difference of 3.494664 secs
```

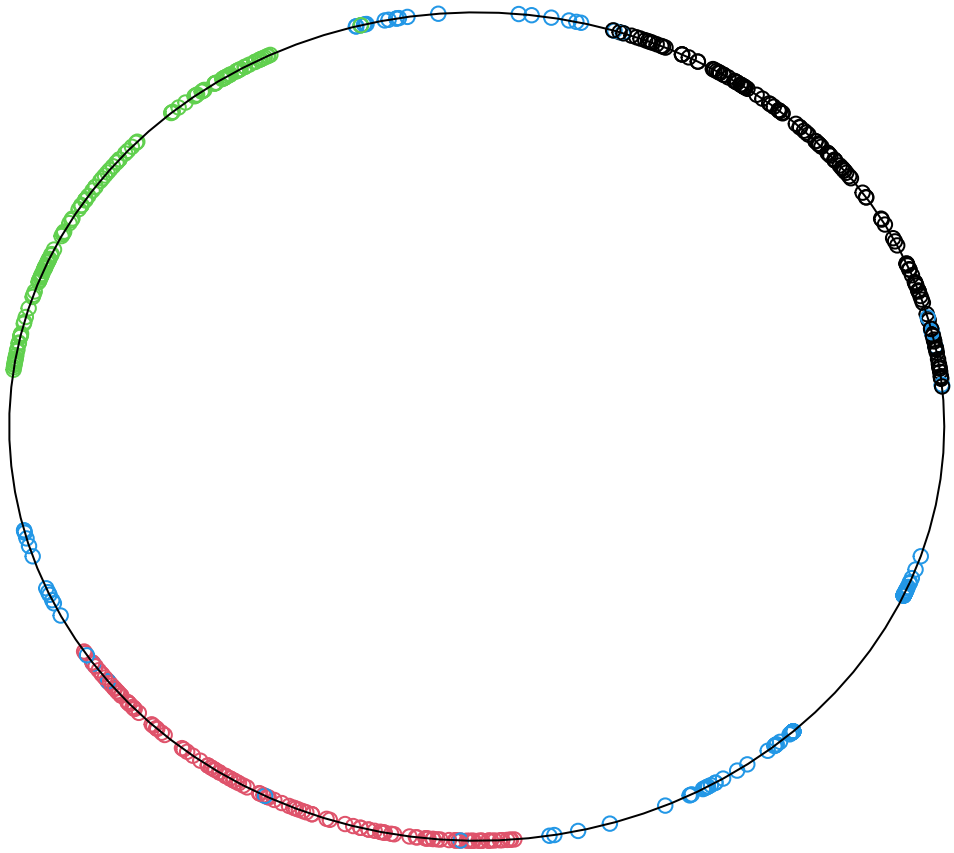
Reduction to $d = 1$ Run `psc_sne` with colors being the groups of variables for $d = 1$:

```
res_pscsne_21 <- psc_sne(X = mvmf_spiral_mix_data, d = 1,
                         rho_psc_list = rho_30_2, eta = 50,
                         colors = mvmf_spiral_mix_colors, show_prog = TRUE,
                         parallel_cores = num_cores_param)
#> It: 1 (best: 1); obj: 1.73e+01 (best: 1.73e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 7.5e-02; mom: 0.0e+00
#> It: 100 (best: 47); obj: 1.29e+01 (best: 1.28e+01); abs: 1.9e-04; rel: 1.5e-05; norm: 1.5e-01; mom: 2.4e+00
#> It: 200 (best: 200); obj: 1.74e+00 (best: 1.74e+00); abs: 6.7e-08; rel: 3.8e-08; norm: 2.6e-05; mom: 1.3e-03
#> It: 284 (best: 284); obj: 1.74e+00 (best: 1.74e+00); abs: 3.2e-10; rel: 1.9e-10; norm: 9.7e-07; mom: 3.0e-04
#> CONVERGENCE!
```



Now, print the best results (lowest objective function value) onto the circumference:

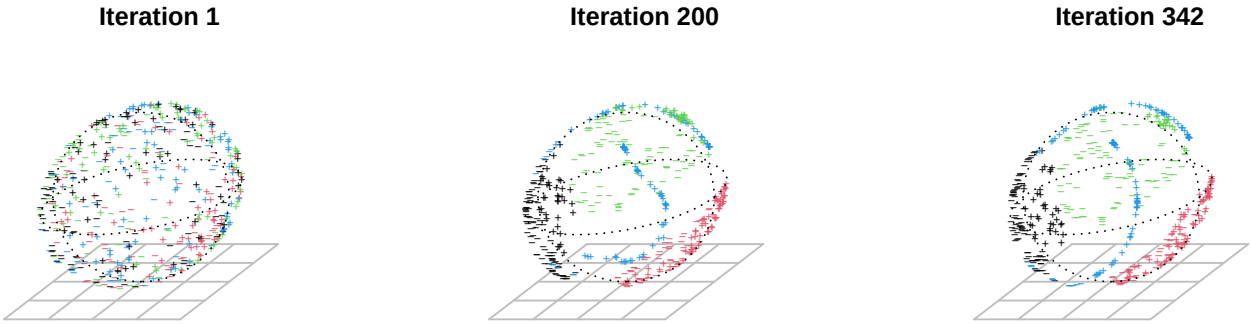
```
Y <- res_pscsne_21$best_Y
plot(Y[, 1], Y[, 2], col = mvmf_spiral_mix_colors, xlim = c(-1, 1),
     ylim = c(-1, 1), axes = FALSE, xlab = "", ylab = "")
th <- seq(0, 2 * pi, length.out = 100)
polygon(x = cos(th), y = sin(th))
```



The `psc_sne` identifies in the reduced dimension on the circumference the three von Mises–Fisher distributions and the spiral distribution which, as it is possible to see, is mixed with the remainder distributions since it collapses with them. This is expected as it works as noise in the data.

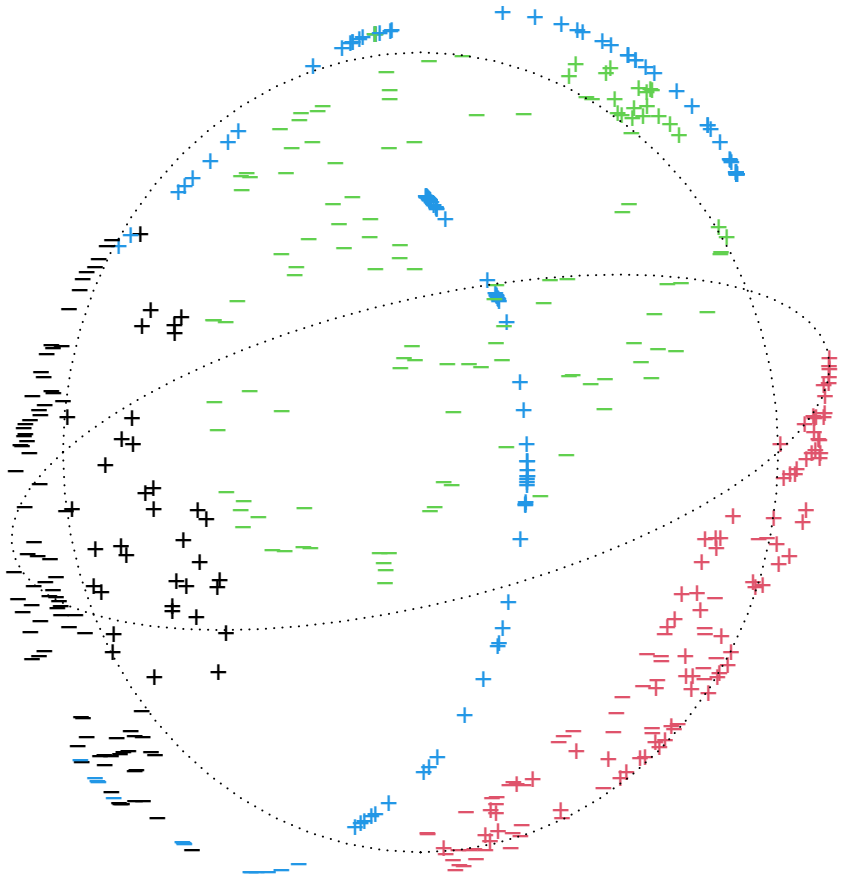
Reduction to $d = 2$ Let's do the same for $d = 2$:

```
psc_sne_res_22 <- psc_sne(mvmf_spiral_mix_data, d = 2, rho_psc_list = rho_30_2,
                          colors = mvmf_spiral_mix_colors, show_prog = TRUE,
                          parallel_cores = num_cores_param, eta = 100)
#> It: 1 (best: 1); obj: 1.77e+01 (best: 1.77e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 1.5e-01; mom: 0.0e+00
#> It: 100 (best: 95); obj: 1.42e+01 (best: 1.30e+01); abs: 1.1e+00; rel: 8.5e-02; norm: 5.0e-01; mom: 1.0e+01
#> It: 200 (best: 200); obj: 9.17e-01 (best: 9.17e-01); abs: 9.1e-05; rel: 1.0e-04; norm: 2.7e-02; mom: 8.7e-01
#> It: 300 (best: 300); obj: 9.04e-01 (best: 9.04e-01); abs: 2.0e-08; rel: 2.2e-08; norm: 7.2e-05; mom: 4.9e-03
#> It: 342 (best: 342); obj: 9.04e-01 (best: 9.04e-01); abs: 1.8e-10; rel: 2.0e-10; norm: 9.8e-07; mom: 2.9e-04
#> CONVERGENCE!
```



The optimization process has converged, discovering that a minimum has been reached. Let's see these points onto the sphere.

```
Y <- psc_sne_res_22$best_Y
sd3 <- scatterplot3d::scatterplot3d(
  Y, xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  color = mvmf_spiral_mix_colors, xlab = "", ylab = "", zlab = "", axis = FALSE,
  pch = c("+", "-")[ifelse(sign(Y[, 2]) == 1, 1, 2)], grid = FALSE,
  mar = c(0, 0, 0, 0)
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 3)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 3)
```



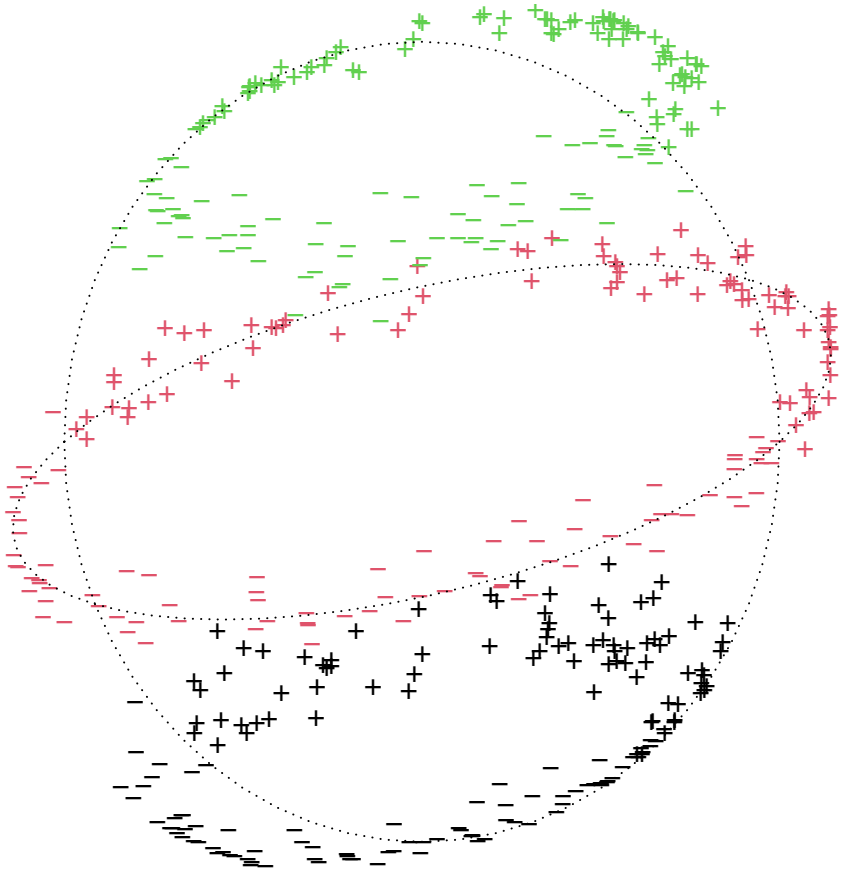
Results onto the sphere are successfully since the three von Mises–Fisher distributions (colors black, red and green) are pretty well identified and the spiral (represented with a blue color) is not located in a group but this is expected since it collapses with many clusters before identified.

Mixtures of small circle distributions at different parallels

```
n <- 500
n1 <- ceiling(n / 3)
n2 <- ceiling(n / 3)
n3 <- floor(n / 3)
set.seed(42)
r_1 <- r_path_s2r(n = n1, r = 1, Theta = rbind(0, 0, 1), sigma = 0.05,
                  t = -0.75)
set.seed(42)
r_2 <- r_path_s2r(n = n2, r = 1, Theta = rbind(0, 0, 1), sigma = 0.05, t = 0)
set.seed(42)
r_3 <- r_path_s2r(n = n3, r = 1, Theta = rbind(0, 0, 1), sigma = 0.05,
                  t = 0.75)
mix_sc <- abind(r_1, r_2, r_3, along = 1)
mix_sc_colors <- c(rep(1, n1), rep(2, n2), rep(3, n3))
set.seed(42)
indexes <- sample(1:n)
mix_sc <- mix_sc[indexes, , , drop = FALSE]
mix_sc_colors <- mix_sc_colors[indexes]
```

Since data lie onto the sphere, $\mathbb{S}^2$, then it is possible to see how the distributions involved are graphically:

```
sd3 <- scatterplot3d::scatterplot3d(
  mix_sc[, , 1], xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  color = mix_sc_colors, xlab = "", ylab = "", zlab = "", axis = FALSE,
  pch = c("+", "-")[ifelse(sign(mix_sc[, 2, 1]) == 1, 1, 2)],
  mar = c(0,0,0,0), grid = FALSE
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 3)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 3)
```

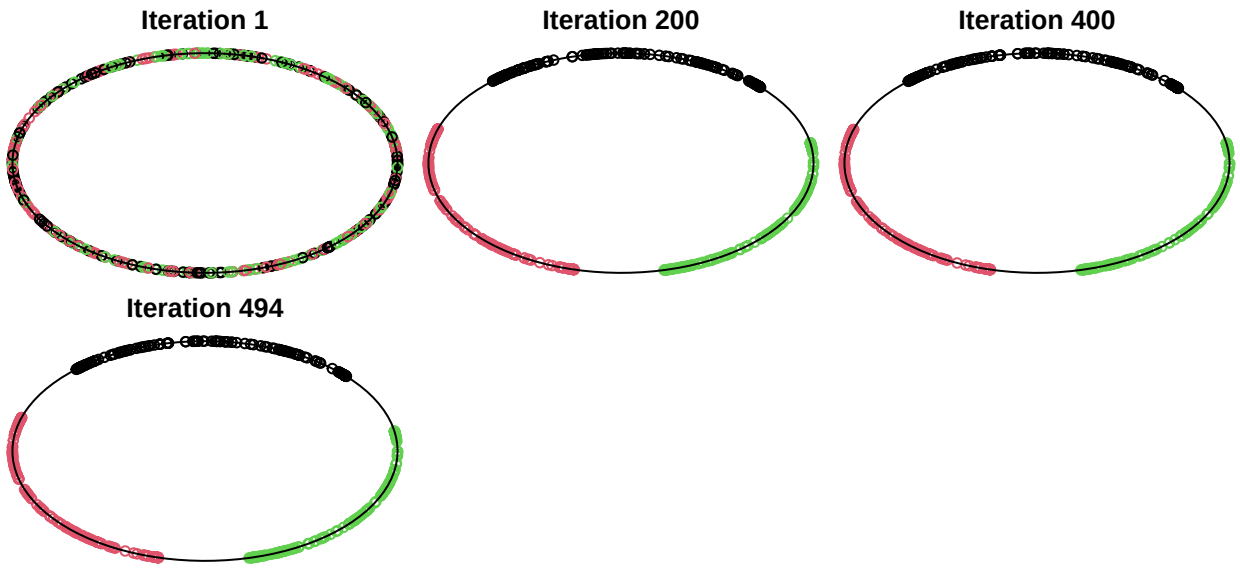


Let's calculate the rho based on a perplexity of 30:

```
rho_30_3 <- rho_optim_bst(x = mix_sc, perp_fixed = 30,
                          num_cores = num_cores_param)
#> Time difference of 3.243842 secs
```

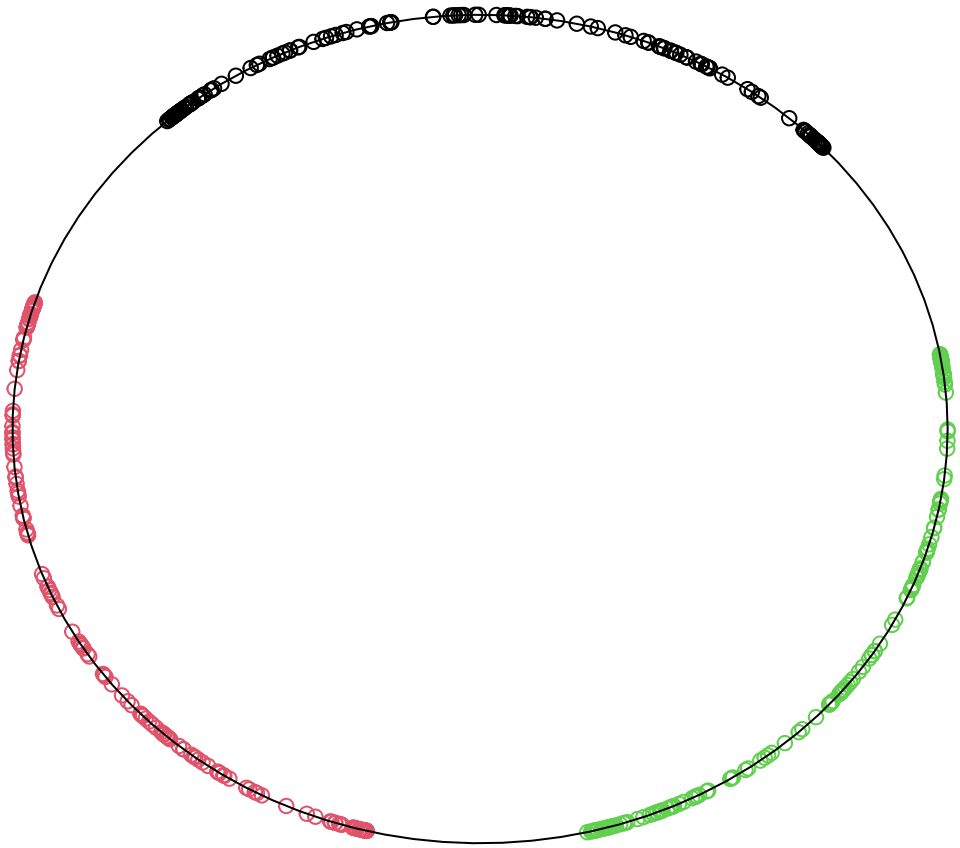
Reduction to $d = 1$ Run `psc_sne()` with colors being the groups of variables for $d = 1$:

```
res_pscsne_31 <- psc_sne(X = mix_sc, d = 1, rho_psc_list = rho_30_3, eta = 50,
                      colors = mix_sc_colors, show_prog = TRUE,
                      perplexity = 30, parallel_cores = num_cores_param)
#> It: 1 (best: 1); obj: 1.77e+01 (best: 1.77e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 6.6e-02; mom: 0.0e+00
#> It: 100 (best: 34); obj: 1.29e+01 (best: 1.28e+01); abs: 5.9e-05; rel: 4.5e-06; norm: 1.3e-01; mom: 2.1e+00
#> It: 200 (best: 200); obj: 1.79e+00 (best: 1.79e+00); abs: 3.4e-07; rel: 1.9e-07; norm: 5.8e-05; mom: 3.0e-03
#> It: 300 (best: 300); obj: 1.79e+00 (best: 1.79e+00); abs: 4.7e-06; rel: 2.6e-06; norm: 1.4e-04; mom: 2.6e-02
#> It: 400 (best: 400); obj: 1.78e+00 (best: 1.78e+00); abs: 2.5e-09; rel: 1.4e-09; norm: 3.1e-06; mom: 6.6e-04
#> It: 494 (best: 494); obj: 1.78e+00 (best: 1.78e+00); abs: 2.6e-10; rel: 1.5e-10; norm: 1.0e-06; mom: 2.1e-04
#> CONVERGENCE!
```



The best iteration results onto the circumference:

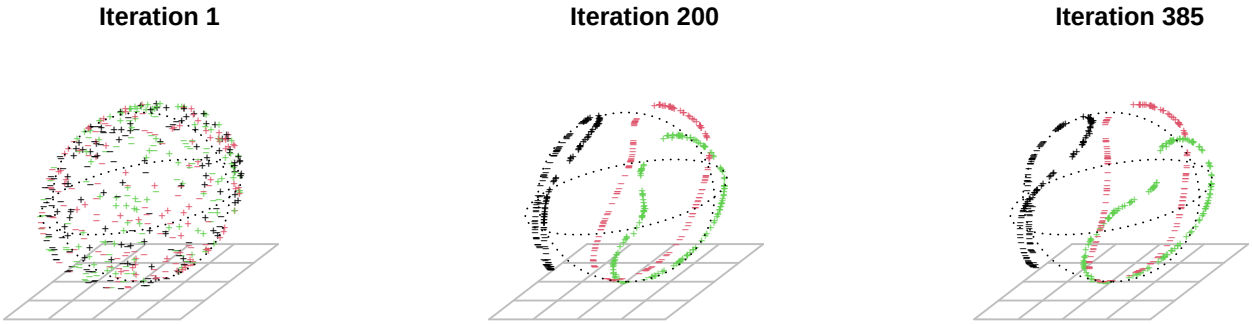
```
Y <- res_pscsne_31$best_Y
plot(Y[, 1], Y[, 2], col = mix_sc_colors, xlim = c(-1, 1), ylim = c(-1, 1),
     axes = FALSE, xlab = "", ylab = "")
th <- seq(0, 2 * pi, length.out = 100)
polygon(x = cos(th), y = sin(th))
```



The `psc_sne` algorithm does identify in the reduced dimension on the circumference the three circles located in each parallel.

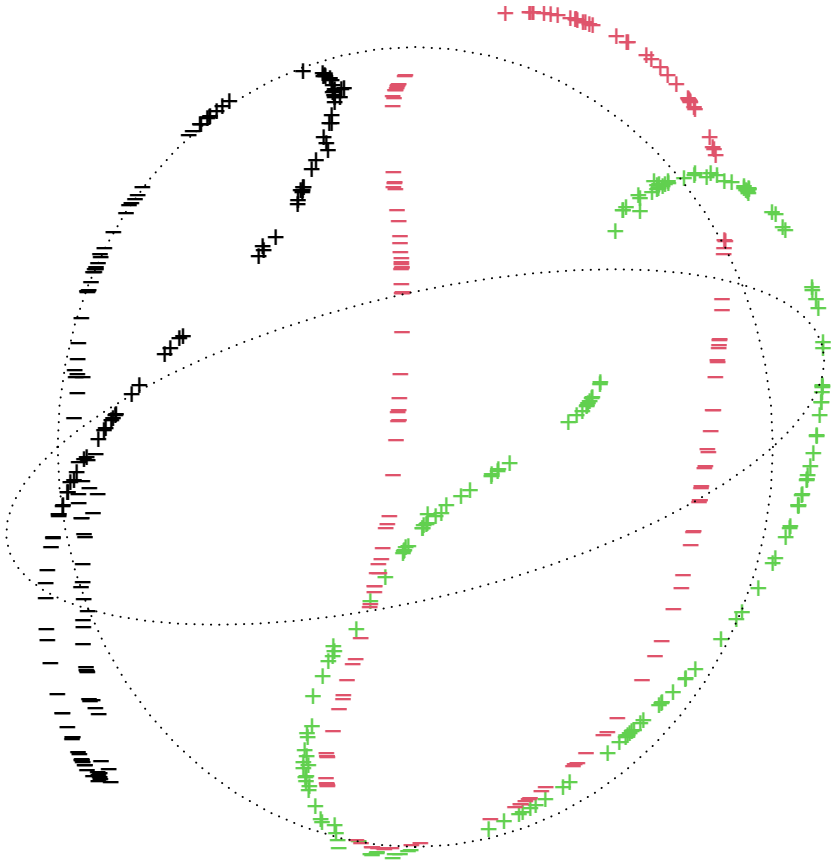
Reduction to $d = 2$ Let's run the psc-SNE algorithm for $d = 2$ and see if results are better:

```
psc_sne_res_32 <- psc_sne(mix_sc, d = 2, rho_psc_list = rho_30_3, eta = 50,
                          colors = mix_sc_colors, show_prog = TRUE,
                          parallel_cores = num_cores_param)
#> It: 1 (best: 1); obj: 1.88e+01 (best: 1.88e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 1.4e-01; mom: 0.0e+00
#> It: 100 (best: 100); obj: 1.17e+01 (best: 1.17e+01); abs: 1.8e-02; rel: 1.5e-03; norm: 5.6e-01; mom: 7.6e+00
#> It: 200 (best: 200); obj: 8.68e-01 (best: 8.68e-01); abs: 7.8e-06; rel: 9.0e-06; norm: 2.8e-04; mom: 1.4e-02
#> It: 300 (best: 300); obj: 8.67e-01 (best: 8.67e-01); abs: 7.9e-07; rel: 9.1e-07; norm: 5.9e-05; mom: 1.4e-02
#> It: 385 (best: 385); obj: 8.67e-01 (best: 8.67e-01); abs: 3.0e-10; rel: 3.5e-10; norm: 9.9e-07; mom: 2.6e-04
#> CONVERGENCE!
```



In this case the scores projected onto the sphere are good since the 3 cicle groups are clearly separated along the sphere. Let's see the best found map points onto the sphere.

```
Y <- psc_sne_res_32$best_Y
sd3 <- scatterplot3d::scatterplot3d(
  Y, xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  color = mix_sc_colors, xlab = "", ylab = "", zlab = "", axis = FALSE,
  pch = c("+", "-")[ifelse(sign(Y[, 2]) == 1, 1, 2)], grid = FALSE,
  mar = c(0, 0, 0, 0)
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 3)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 3)
```



In the graph above, it is possible to distinguish the three circles, colored with the original data: black, red and green.

$(\mathbb{S}^1)^2$

Let us introduce some different scenarios where each of them has some particular distributions for $(\mathbb{S}^1)^2$.

A mixture of von Mises-Fisher that generates separated clusters

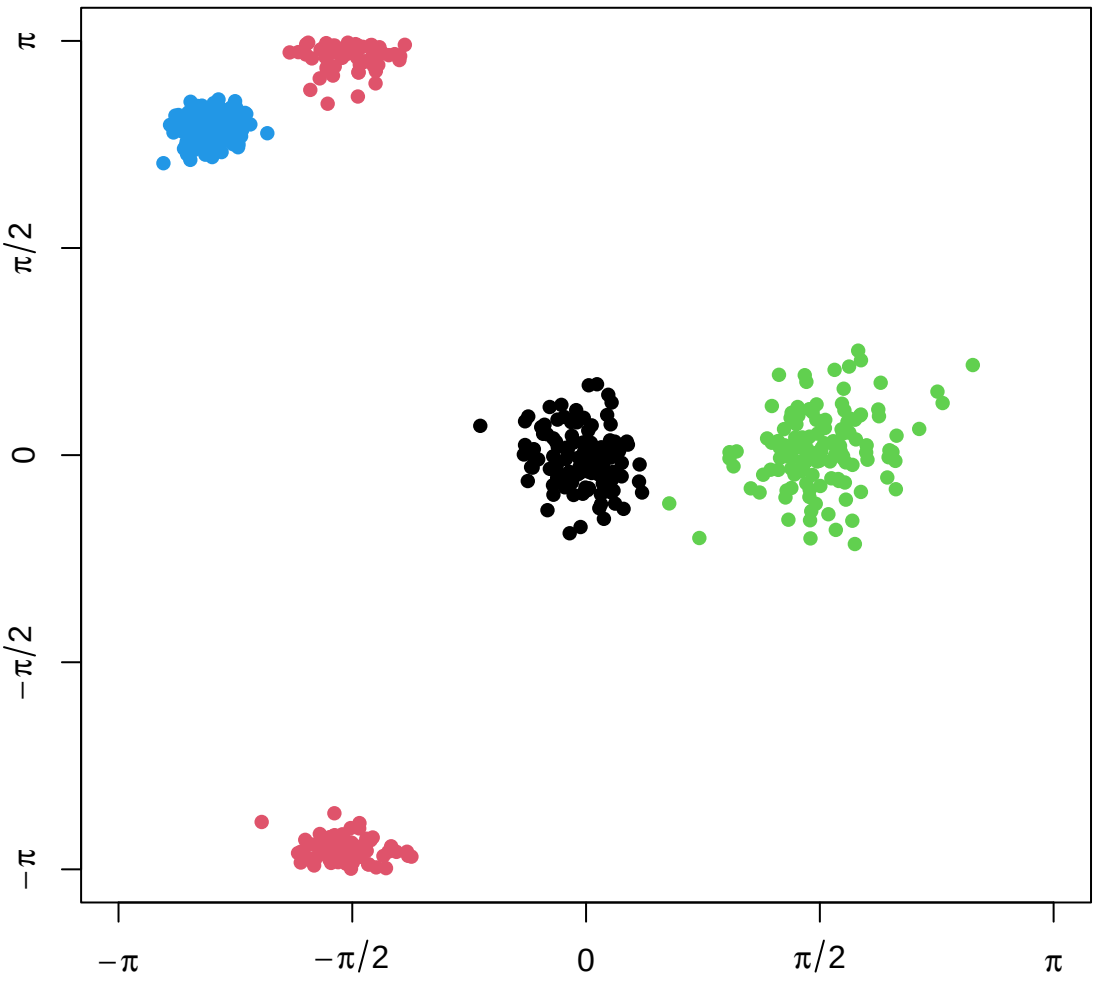
Here there is an approach similar to the first point, a mixture of von Mises–Fisher, that generates different clusters, is implemented.

```
n <- 500
n1 <- ceiling(n / 4)
n2 <- ceiling(n / 4)
n3 <- floor(n / 4)
n4 <- floor(n / 4)
mix_3_mvMF <- function(n, n1, n2, n3, n4, mu_mat, kappa_vec) {
  if ((n1 + n2 + n3 + n4) != n) {
    stop("(n1 + n2 + n3) must sum 1")
  }
  r_1 <- rotasym::r_vMF(n = n1, mu = mu_mat[, 1], kappa = kappa_vec[1])
  r_2 <- rotasym::r_vMF(n = n2, mu = mu_mat[, 2], kappa = kappa_vec[2])
  r_3 <- rotasym::r_vMF(n = n3, mu = mu_mat[, 3], kappa = kappa_vec[3])
  r_4 <- rotasym::r_vMF(n = n4, mu = mu_mat[, 4], kappa = kappa_vec[4])
  data <- rbind(r_1, r_2, r_3, r_4)
  return(array(data, dim = c(dim(data), 1)))
```

```
}
mu_mat_1 <- cbind(drop(DirStats::to_cir(0)),
                  drop(DirStats::to_cir(-pi / 2)),
                  drop(DirStats::to_cir(pi / 2)),
                  drop(DirStats::to_cir(3 * pi / 2.5)))
set.seed(42)
sph_1 <- mix_3_mvMF(n, n1, n2, n3, n4, mu_mat = mu_mat_1,
                    kappa_vec = c(30, 25, 10, 50))
mu_mat_2 <- cbind(drop(DirStats::to_cir(0)),
                  drop(DirStats::to_cir(-pi)),
                  drop(DirStats::to_cir(0)),
                  drop(DirStats::to_cir(-3 * pi / 2.5)))
set.seed(44)
sph_2 <- mix_3_mvMF(n, n1, n2, n3, n4, mu_mat = mu_mat_2,
                    kappa_vec = c(25, 32, 12, 100))
data <- abind(sph_1, sph_2, along = 3)
set.seed(42)
indexes <- sample(1:n)
mix_3_mvMF_data <- data[indexes, , , drop = FALSE]
mix_3_mvMF_cols <- c(rep(1, n1), rep(2, n2), rep(3, n3), rep(4, n4))
mix_3_mvMF_cols <- mix_3_mvMF_cols[indexes]
```

Since input data lies on $(\mathbb{S}^1)^2$ is easy to be viewed in the torus as follows:

```
mix_3_mvMF_torus <- sdetorus::toPiInt(cbind(
  DirStats::to_rad(mix_3_mvMF_data[, , 1]),
  DirStats::to_rad(mix_3_mvMF_data[, , 2])))

plot(mix_3_mvMF_torus, xlim = c(-pi, pi), ylim = c(-pi, pi), axes = FALSE,
     col = rep(mix_3_mvMF_cols, 2), pch = 16,
     xlab = "", ylab = "")
sdetorus::torusAxis()
```

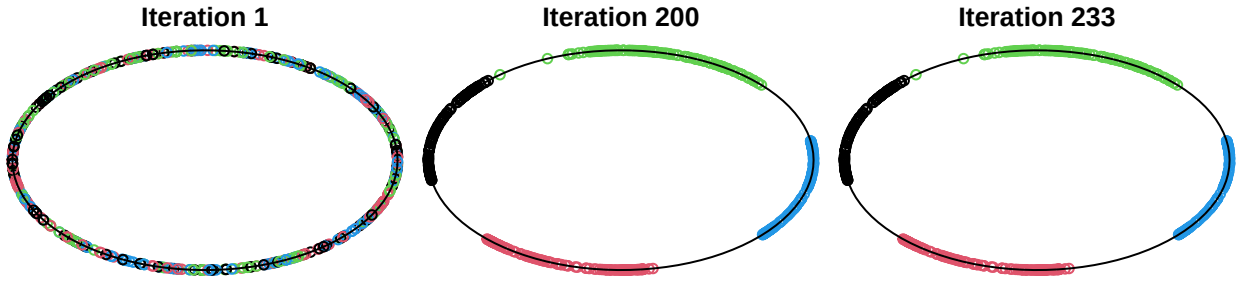


Let's calculate the rho based on a perplexity of 30:

```
rho_30_4 <- rho_optim_bst(x = mix_3_mvMF_data, perp_fixed = 30,
                          num_cores = num_cores_param)
#> Time difference of 5.774905 secs
```

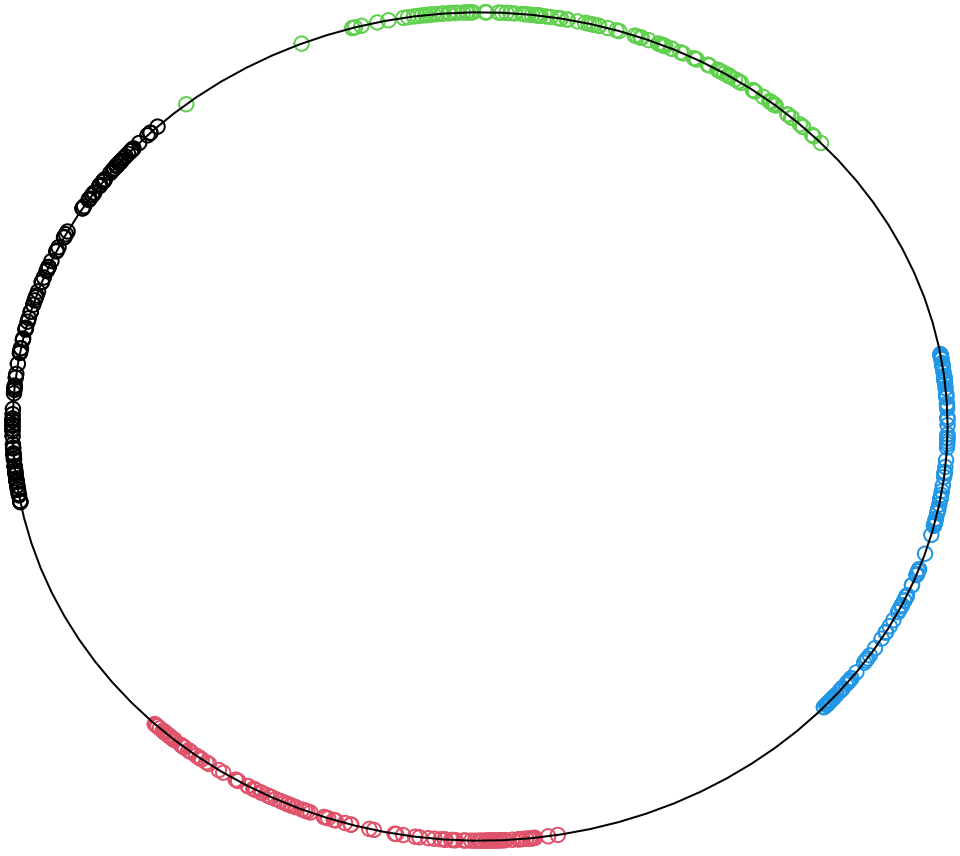
Reduction to $d = 1$ Run `psc_sne()` with colors being the groups of variables for $d = 1$:

```
res_pscsne_41 <- psc_sne(X = mix_3_mvMF_data, d = 1, rho_psc_list = rho_30_4,
                         eta = 50, colors = mix_3_mvMF_cols, show_prog = TRUE,
                         parallel_cores = num_cores_param)
#> It: 1 (best: 1); obj: 1.73e+01 (best: 1.73e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 7.8e-02; mom: 0.0e+00
#> It: 100 (best: 22); obj: 1.29e+01 (best: 1.29e+01); abs: 1.5e-04; rel: 1.1e-05; norm: 1.7e-01; mom: 2.7e+00
#> It: 200 (best: 200); obj: 1.74e+00 (best: 1.74e+00); abs: 3.0e-09; rel: 1.7e-09; norm: 5.4e-06; mom: 3.0e-04
#> It: 233 (best: 233); obj: 1.74e+00 (best: 1.74e+00); abs: 9.8e-11; rel: 5.7e-11; norm: 9.8e-07; mom: 5.4e-05
#> CONVERGENCE!
```



The best iteration results onto the circumference:

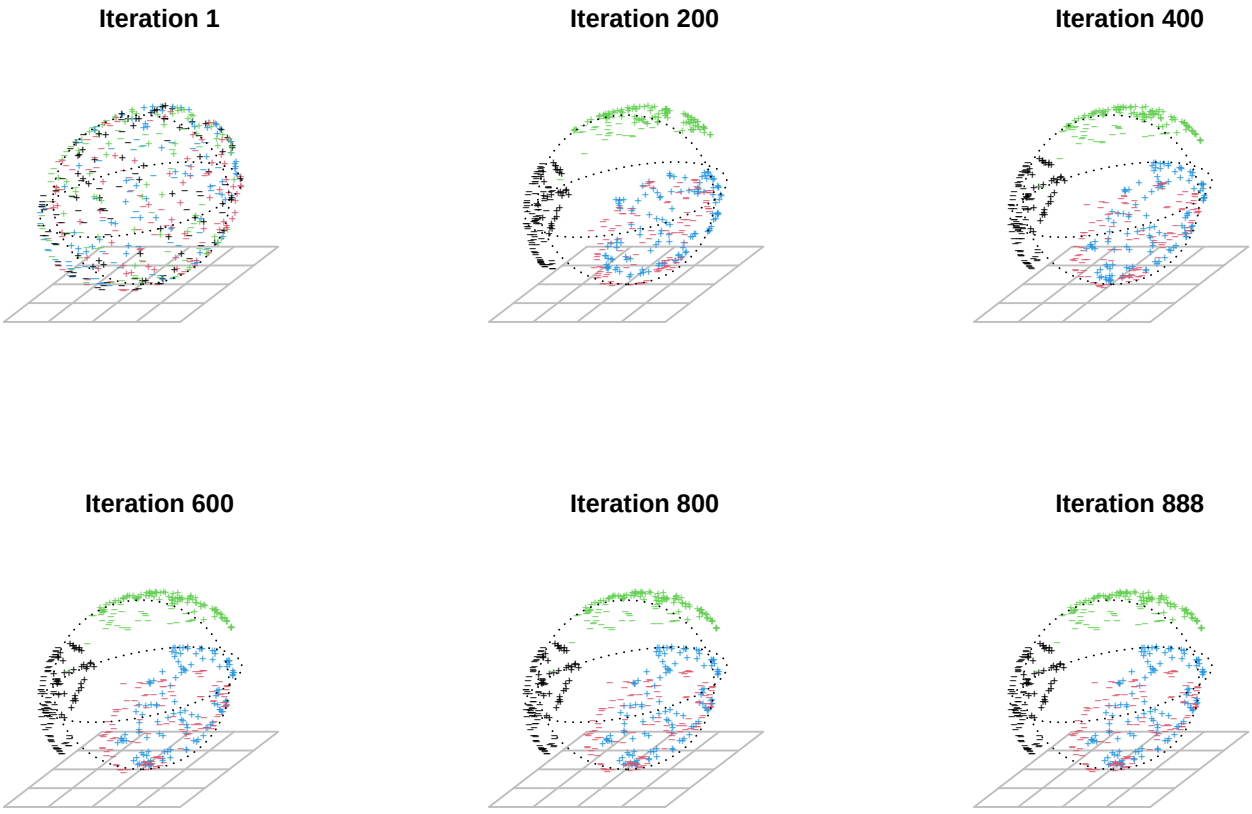
```r
Y <- res_pscsne_41$best_Y
plot(Y[, 1], Y[, 2], col = mix_3_mvMF_cols, xlim = c(-1, 1), ylim = c(-1, 1),
     axes = FALSE, xlab = "", ylab = "")
th <- seq(0, 2 * pi, length.out = 100)
polygon(x = cos(th), y = sin(th))
```



As we can see in the plot above, the three components of the von Mises–Fisher mixture are identified.

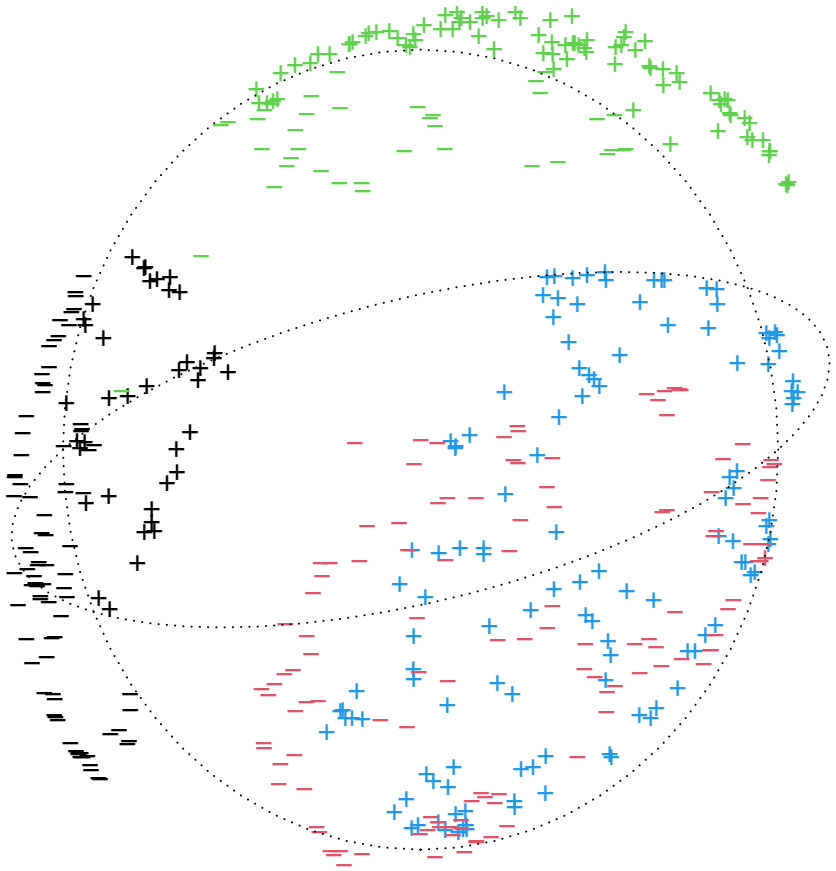
Reduction to $d = 2$ Let's run the psc-SNE algorithm for $d = 2$ and see if results are good as well:

```r
psc_sne_res_42 <- psc_sne(mix_3_mvMF_data, d = 2, rho_psc_list = rho_30_4,
                          eta = 25, colors = mix_3_mvMF_cols,
                          show_prog = TRUE, parallel_cores = num_cores_param)
#> It: 1 (best: 1); obj: 1.88e+01 (best: 1.88e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 1.6e-01; mom: 0.0e+00
#> It: 100 (best: 10); obj: 1.12e+01 (best: 1.11e+01); abs: 7.2e-04; rel: 6.4e-05; norm: 3.1e-01; mom: 2.5e+00
#> It: 200 (best: 200); obj: 9.62e-01 (best: 9.62e-01); abs: 9.4e-06; rel: 9.8e-06; norm: 4.3e-04; mom: 1.1e-02
#> It: 300 (best: 300); obj: 9.62e-01 (best: 9.62e-01); abs: 7.5e-06; rel: 7.8e-06; norm: 2.5e-04; mom: 2.4e-02
#> It: 400 (best: 400); obj: 9.61e-01 (best: 9.61e-01); abs: 3.5e-06; rel: 3.7e-06; norm: 1.6e-04; mom: 1.8e-02
#> It: 500 (best: 500); obj: 9.61e-01 (best: 9.61e-01); abs: 3.7e-07; rel: 3.8e-07; norm: 5.3e-05; mom: 5.6e-03
#> It: 600 (best: 600); obj: 9.61e-01 (best: 9.61e-01); abs: 5.6e-08; rel: 5.8e-08; norm: 2.1e-05; mom: 2.2e-03
#> It: 700 (best: 700); obj: 9.61e-01 (best: 9.61e-01); abs: 7.3e-09; rel: 7.6e-09; norm: 7.5e-06; mom: 7.9e-04
#> It: 800 (best: 800); obj: 9.61e-01 (best: 9.61e-01); abs: 8.7e-10; rel: 9.1e-10; norm: 2.6e-06; mom: 2.7e-04
#> It: 888 (best: 888); obj: 9.61e-01 (best: 9.61e-01); abs: 1.3e-10; rel: 1.3e-10; norm: 9.9e-07; mom: 1.0e-04
#> CONVERGENCE!
```



The best results on the sphere are:

```
Y <- psc_sne_res_42$best_Y
sd3 <- scatterplot3d::scatterplot3d(
  Y, xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  color = mix_3_mvMF_cols, xlab = "", ylab = "", zlab = "", axis = FALSE,
  pch = c("+", "-")[ifelse(sign(Y[, 2]) == 1, 1, 2)], grid = FALSE,
  mar = c(0, 0, 0, 0)
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 3)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 3)
```



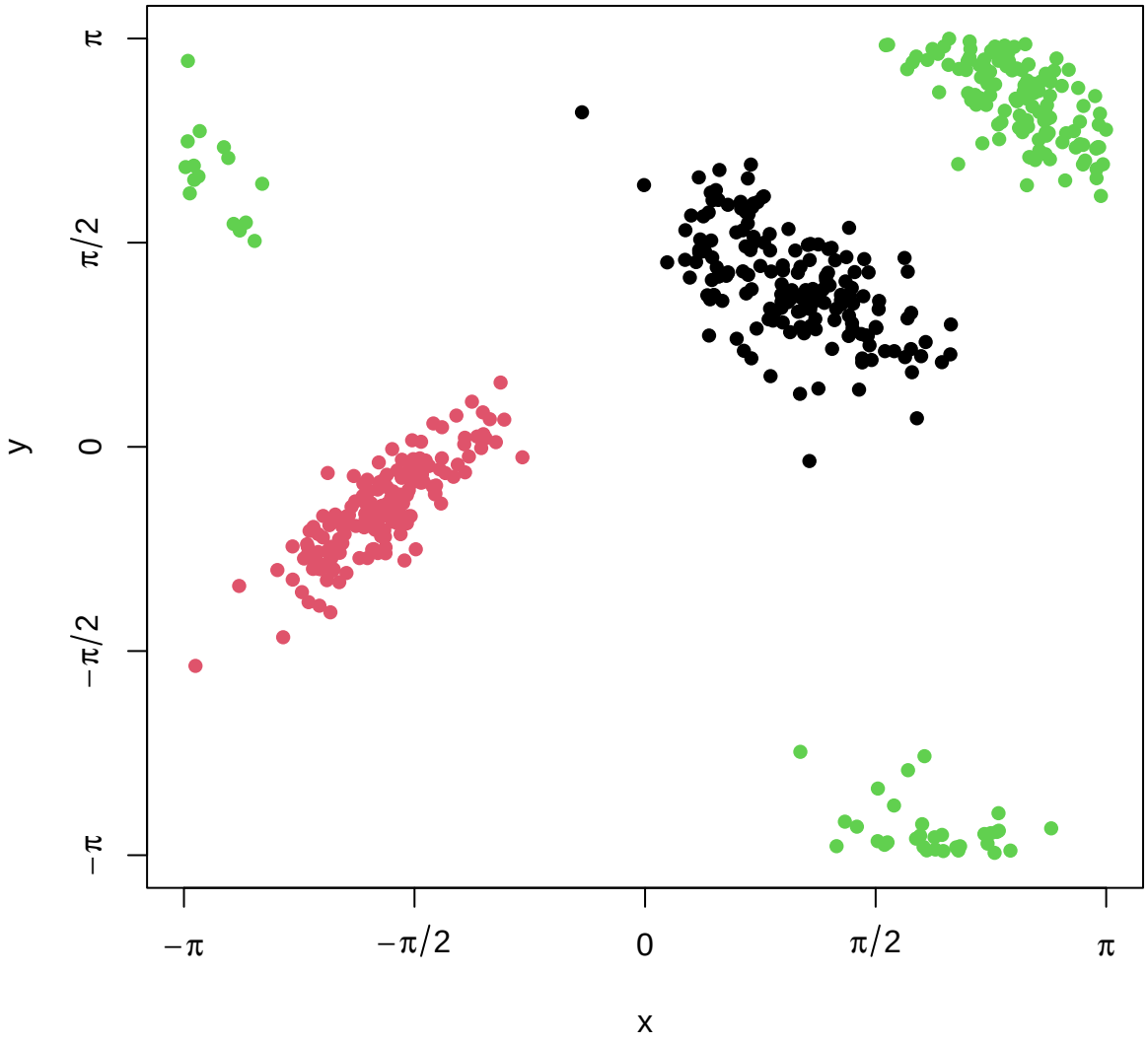
As it happened for $d = 1$, all components were found in four different clusters.

Mixtures of bivariate wrapped normals to have controlled clusters with correlations

This case is as simple as generate data twice from the bivariate normal distribution and later move these values into $[-\pi, \pi]$. First, let's represent this data in the torus:

```
n <- 500
n1 <- ceiling(n / 3)
n2 <- ceiling(n / 3)
n3 <- floor(n / 3)
mean1 <- c(1, 1.25)
mean2 <- c(-1.8, -0.5)
mean3 <- c(2.5, 2.75)
sigma1 <- matrix(c(0.2, -0.12, -0.12, 0.2), byrow = TRUE, nrow = 2)
sigma2 <- toeplitz(c(0.12, 0.1))
sigma3 <- toeplitz(c(0.2, -0.15))
mix_3_mvnorm <- function(n, n1, n2, n3, mean_mat, sigma1, sigma2, sigma3,
                         kappa = 10) {
  if ((n1 + n2 + n3) != n) {
    stop("(n1 + n2 + n3) must sum 1")
  }
  r_1 <- mvtnorm::rmvnorm(n = n1, mean = mean_mat[, 1], sigma = sigma1)
  r_2 <- mvtnorm::rmvnorm(n = n2, mean = mean_mat[, 2], sigma = sigma2)
  r_3 <- mvtnorm::rmvnorm(n = n3, mean = mean_mat[, 3], sigma = sigma3)
  data <- rbind(r_1, r_2, r_3)
  return(data)
```

```
}
set.seed(5)
x_5_bvnorm <- mix_3_mvnorm(n = n, n1 = n1, n2 = n2, n3 = n3,
                           mean_mat = cbind(mean1, mean2, mean3),
                           sigma1 = sigma1, sigma2 = sigma2, sigma3 = sigma3)
x_5_torus <- sdetorus::toPiInt(x_5_bvnorm)

samp_5 <- abind(DirStats::to_cir(x_5_torus[, 1]),
                DirStats::to_cir(x_5_torus[, 2]), along = 3)
samp_5_cols <- c(rep(1, n1), rep(2, n2), rep(3, n3))

plot(x_5_torus, xlim = c(-pi, pi), ylim = c(-pi, pi), axes = FALSE,
     col = samp_5_cols, pch = 16,
     xlab = "x", ylab = "y")
sdetorus::torusAxis()
```



First, the arrangement of the dataset is altered in order to be more complicated for the algorithm to discover the clusters:

```
set.seed(42)
indexes <- sample(1:n)
```

```
samp_5 <- samp_5[indexes, , , drop = FALSE]
samp_5_cols <- samp_5_cols[indexes]
```

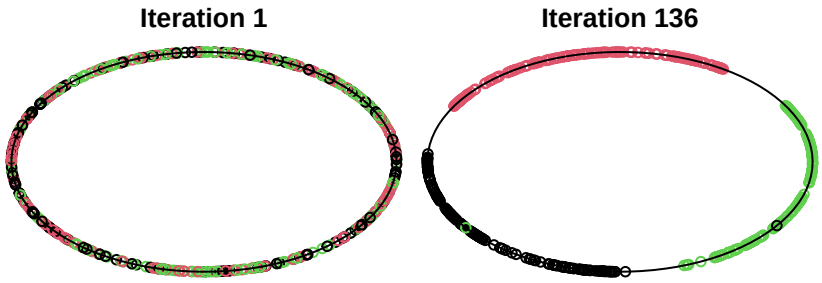
Let's calculate rho based on a fixed perplexity of 30:

```
rho_30_5 <- rho_optim_bst(x = samp_5, perp_fixed = 30,
                          num_cores = num_cores_param)
#> Time difference of 5.516747 secs
```

Now, everything is ready to reduce the dimension into $d = 1$ and $d = 2$:

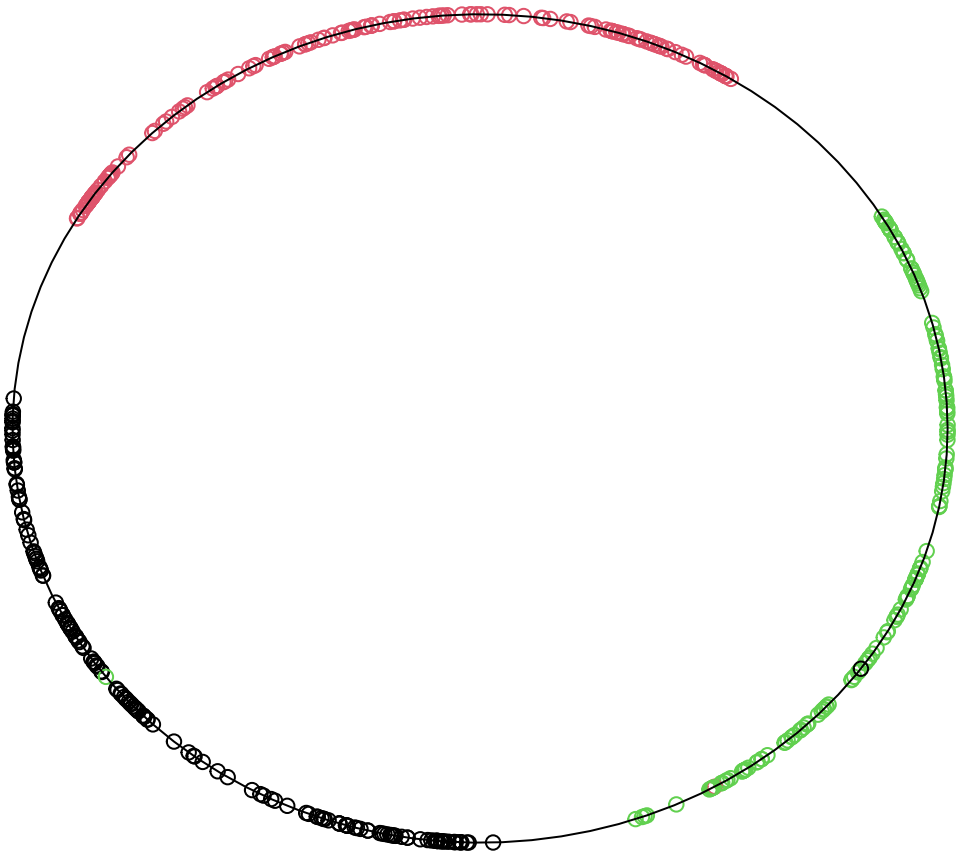
Reduction to $d = 1$ Run `psc_sne()` with colors being the groups of variables for $d = 1$:

```
res_pscsne_51 <- psc_sne(X = samp_5, d = 1, rho_psc_list = rho_30_5,
                         eta = 100, colors = samp_5_cols, show_prog = TRUE,
                         parallel_cores = num_cores_param)
#> It: 1 (best: 1); obj: 1.73e+01 (best: 1.73e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 6.7e-02; mom: 0.0e+00
#> It: 100 (best: 46); obj: 1.38e+01 (best: 1.38e+01); abs: 2.0e-03; rel: 1.4e-04; norm: 2.6e-01; mom: 7.1e+00
#> It: 136 (best: 136); obj: 1.73e+00 (best: 1.73e+00); abs: 7.3e-12; rel: 4.2e-12; norm: 7.3e-07; mom: 4.3e-05
#> CONVERGENCE!
```



The best iteration results onto the circumference:

```
Y <- res_pscsne_51$best_Y
plot(Y[, 1], Y[, 2], col = samp_5_cols, xlim = c(-1, 1), ylim = c(-1, 1),
     axes = FALSE, xlab = "", ylab = "")
th <- seq(0, 2 * pi, length.out = 100)
polygon(x = cos(th), y = sin(th))
```

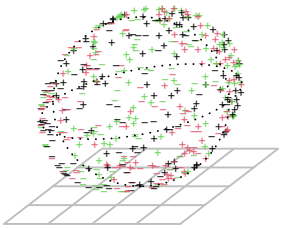


As we can see in the plot above, the three components of the mixture are identified although there are a few points that are wrongly associated to the green and black cluster. This can be explained due to the proximity of the points of both clusters in the original data. #### Reduction to $d = 2$

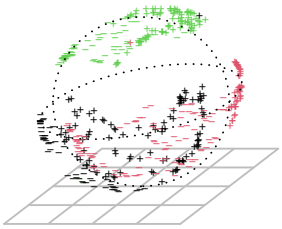
Let's run the psc-SNE algorithm for $d = 2$ and see if results are good as well:

```
psc_sne_res_52 <- psc_sne(samp_5, d = 2, rho_psc_list = rho_30_5,
                          eta = 50, colors = samp_5_cols,
                          show_prog = TRUE, parallel_cores = num_cores_param)
#> It: 1 (best: 1); obj: 1.84e+01 (best: 1.84e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 1.5e-01; mom: 0.0e+00
#> It: 100 (best: 10); obj: 1.23e+01 (best: 1.20e+01); abs: 1.4e-04; rel: 1.1e-05; norm: 5.7e-01; mom: 7.6e+00
#> It: 200 (best: 200); obj: 9.47e-01 (best: 9.47e-01); abs: 6.3e-07; rel: 6.7e-07; norm: 7.9e-05; mom: 4.0e-03
#> It: 300 (best: 300); obj: 9.47e-01 (best: 9.47e-01); abs: 9.4e-08; rel: 1.0e-07; norm: 1.9e-05; mom: 4.2e-03
#> It: 400 (best: 400); obj: 9.47e-01 (best: 9.47e-01); abs: 2.6e-09; rel: 2.8e-09; norm: 3.2e-06; mom: 6.9e-04
#> It: 472 (best: 472); obj: 9.47e-01 (best: 9.47e-01); abs: 2.6e-10; rel: 2.8e-10; norm: 1.0e-06; mom: 2.2e-04
#> CONVERGENCE!
```

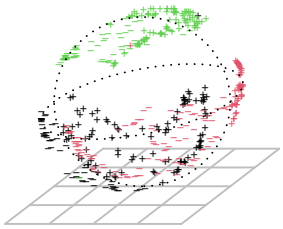
Iteration 1



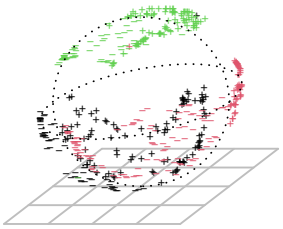
Iteration 200



Iteration 400

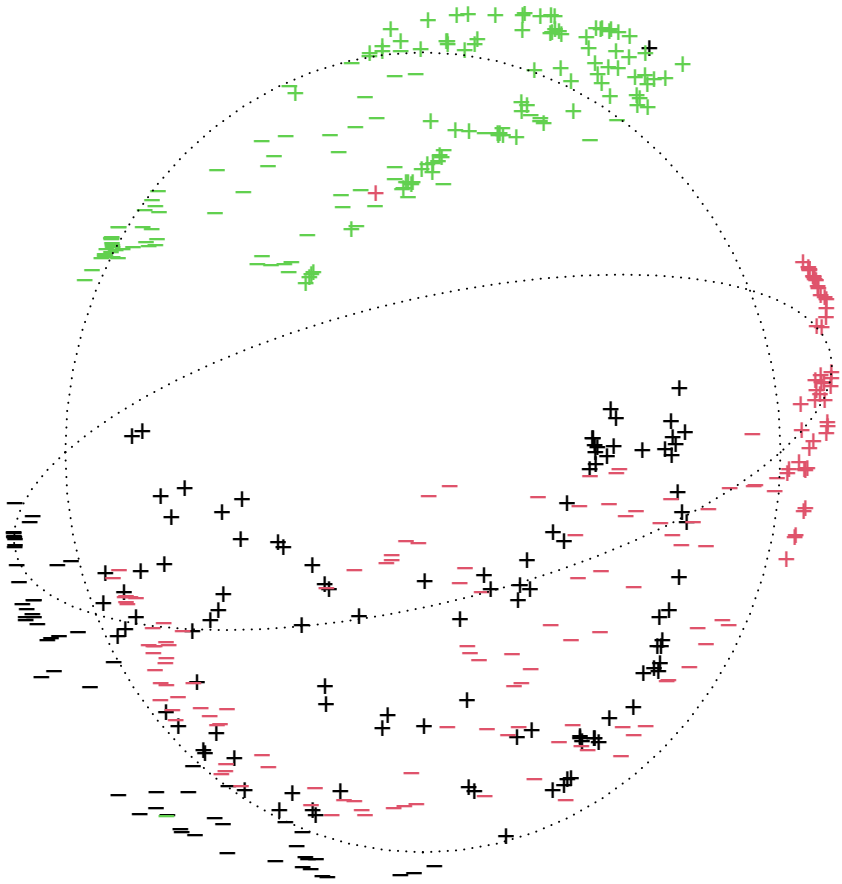


Iteration 472



The best results on the sphere are:

```
Y <- psc_sne_res_52$best_Y
sd3 <- scatterplot3d::scatterplot3d(
  Y, xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  color = samp_5_cols, xlab = "", ylab = "", zlab = "", axis = FALSE,
  pch = c("+", "-")[ifelse(sign(Y[, 2]) == 1, 1, 2)], grid = FALSE,
  mar = c(0, 0, 0, 0)
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 3)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 3)
```

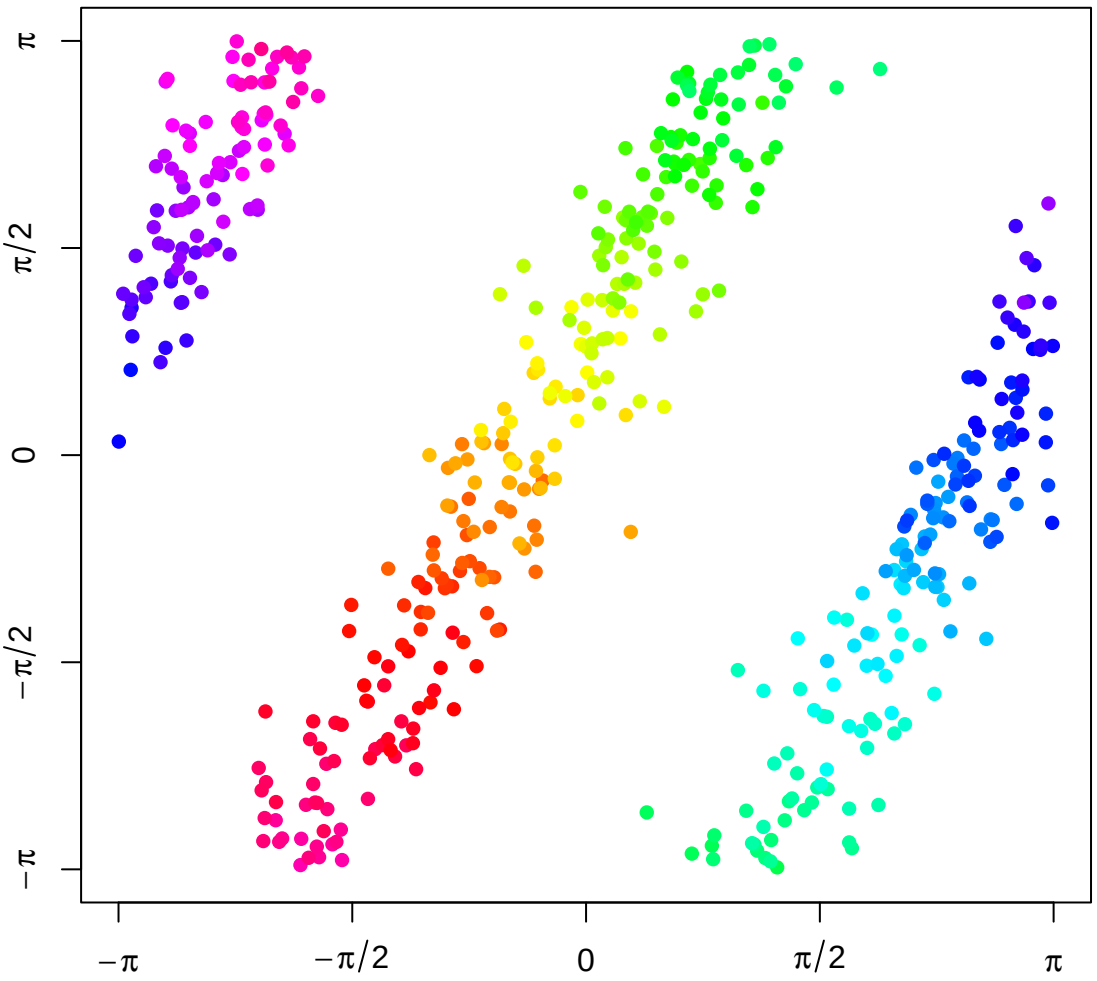


As it happened for $d = 1$, all components were found in three different clusters. Nevertheless, it happened the same that occured in $d = 1$, a few points are wrongly associated to the black and red clusters.

Data along a continuous path that crosses the boundaries

As it happened before, the structure of the data is possible to see thanks to the torus. We will represent this continuous path with the colors of the rainbow, it will help to identify if the reduction is properly done later on.

```
n <- 500
set.seed(3)
samp_1 <- r_path_s1r(n = n, r = 2, k = c(1, 2), angles = TRUE)
plot(samp_1, xlim = c(-pi, pi), ylim = c(-pi, pi), col = rainbow(n, alpha = 1),
     axes = FALSE, xlab = "", ylab = "", pch = 16)
sdetorus::torusAxis()
```



Let's convert the torus data into the polysphere $(\mathbb{S}^1)^2$ (Cartesian coordinates) and the index of the dataset is altered to complicate the search of the path:

```
x_1 <- DirStats::to_cir(samp_1[, 1])
x_2 <- DirStats::to_cir(samp_1[, 2])
x_path_1 <- abind(x_1, x_2, along = 3)
x_path_1_cols <- rainbow(n, alpha = 1)
set.seed(42)
indexes <- sample(1:n)
x_path_1 <- x_path_1[indexes, , , drop = FALSE]
x_path_1_cols <- x_path_1_cols[indexes]
```

Let's calculate now the values of rho, $\boldsymbol{\rho}$, based on a fixed perplexity of 30:

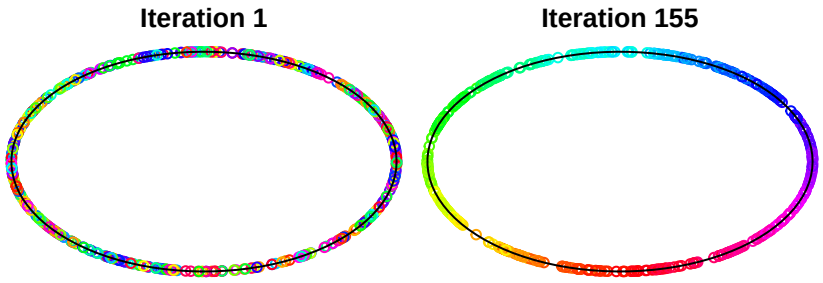
```
rho_30_6 <- rho_optim_bst(x = x_path_1, perp_fixed = 30,
                          num_cores = num_cores_param)
#> Time difference of 5.31267 secs
```

After all of this done, everything is prepared to reduce the dimension of the data to $d = 1$ and $d = 2$:

Reduction to $d = 1$ Run `psc_sne()` with colors being the groups of variables for $d = 1$:

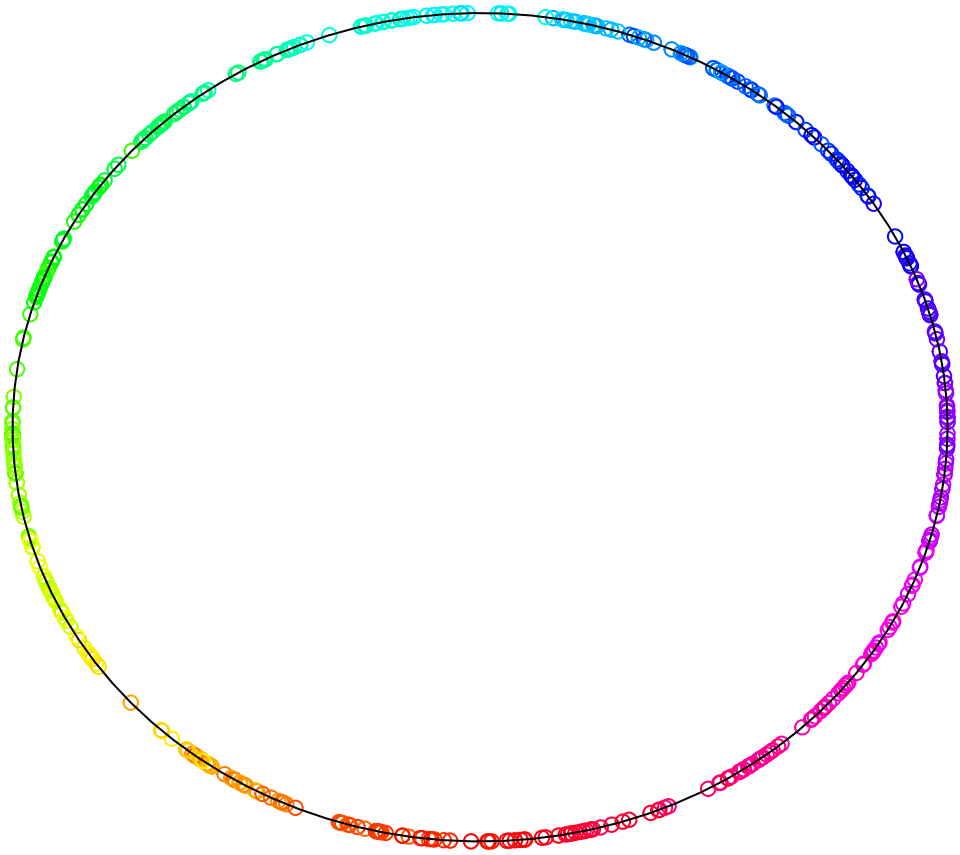
```
set.seed(42)
res_pscsne_61 <- psc_sne(X = x_path_1, d = 1, rho_psc_list = rho_30_6,
                         eta = 25, colors = x_path_1_cols, show_prog = TRUE,
                         parallel_cores = num_cores_param, init = "random")
#> It: 1 (best: 1); obj: 1.78e+01 (best: 1.78e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 6.9e-02; mom: 0.0e+00
```

```
#> It: 100 (best: 100); obj: 1.26e+01 (best: 1.26e+01); abs: 8.8e-08; rel: 7.0e-09; norm: 9.5e-07; mom: 3.1e-05
#> It: 155 (best: 155); obj: 1.76e+00 (best: 1.76e+00); abs: 4.6e-11; rel: 2.6e-11; norm: 9.2e-07; mom: 3.0e-05
#> CONVERGENCE!
```



The best iteration results onto the circumference:

```
Y <- res_pscsne_61$best_Y
plot(Y[, 1], Y[, 2], col = x_path_1_cols, xlim = c(-1, 1), ylim = c(-1, 1),
     axes = FALSE, xlab = "", ylab = "")
th <- seq(0, 2 * pi, length.out = 100)
polygon(x = cos(th), y = sin(th))
```



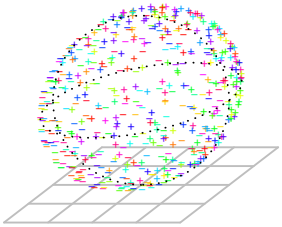
It is possible to see an appropriate arrangement of the path thanks to configuring the initialization of the zero-iteration solution randomly instead of the evenly separated points when the solution did not identify properly the structure of the path. Let's see what happens when reducing to $d = 2$.

Reduction to $d = 2$ Let's run the psc-SNE algorithm for $d = 2$ and see if results are better:

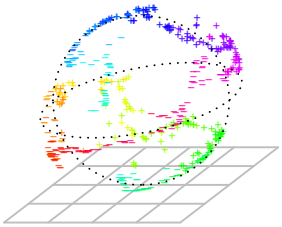
```
psc_sne_res_62 <- psc_sne(x_path_1, d = 2, rho_psc_list = rho_30_6,
                          eta = 15, colors = x_path_1_cols,
                          show_prog = TRUE, parallel_cores = num_cores_param)
```

```
#> It: 1 (best: 1); obj: 1.94e+01 (best: 1.94e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 1.5e-01; mom: 0.0e+00
#> It: 100 (best: 47); obj: 9.70e+00 (best: 9.60e+00); abs: 1.2e-03; rel: 1.2e-04; norm: 7.0e-04; mom: 1.1e-02
#> It: 200 (best: 200); obj: 9.63e-01 (best: 9.63e-01); abs: 2.1e-05; rel: 2.2e-05; norm: 8.3e-04; mom: 1.3e-02
#> It: 300 (best: 300); obj: 9.61e-01 (best: 9.61e-01); abs: 1.9e-05; rel: 2.0e-05; norm: 5.1e-04; mom: 3.0e-02
#> It: 400 (best: 400); obj: 9.60e-01 (best: 9.60e-01); abs: 7.1e-06; rel: 7.4e-06; norm: 3.1e-04; mom: 1.8e-02
#> It: 500 (best: 500); obj: 9.58e-01 (best: 9.58e-01); abs: 4.0e-07; rel: 4.2e-07; norm: 7.1e-05; mom: 4.7e-03
#> It: 600 (best: 600); obj: 9.58e-01 (best: 9.58e-01); abs: 9.5e-09; rel: 9.9e-09; norm: 1.1e-05; mom: 7.1e-04
#> It: 700 (best: 700); obj: 9.58e-01 (best: 9.58e-01); abs: 6.6e-10; rel: 6.9e-10; norm: 2.9e-06; mom: 1.8e-04
#> It: 795 (best: 795); obj: 9.58e-01 (best: 9.58e-01); abs: 7.6e-11; rel: 8.0e-11; norm: 9.9e-07; mom: 6.3e-05
#> CONVERGENCE!
```

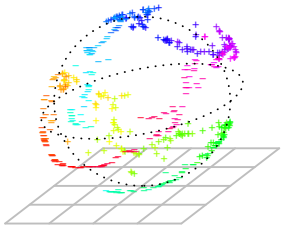
Iteration 1



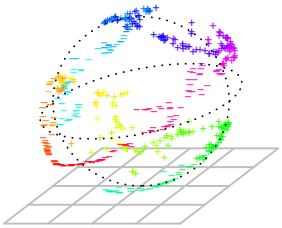
Iteration 200



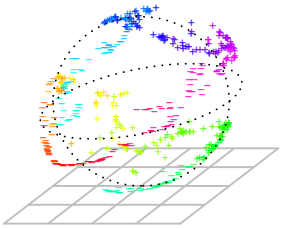
Iteration 400



Iteration 600

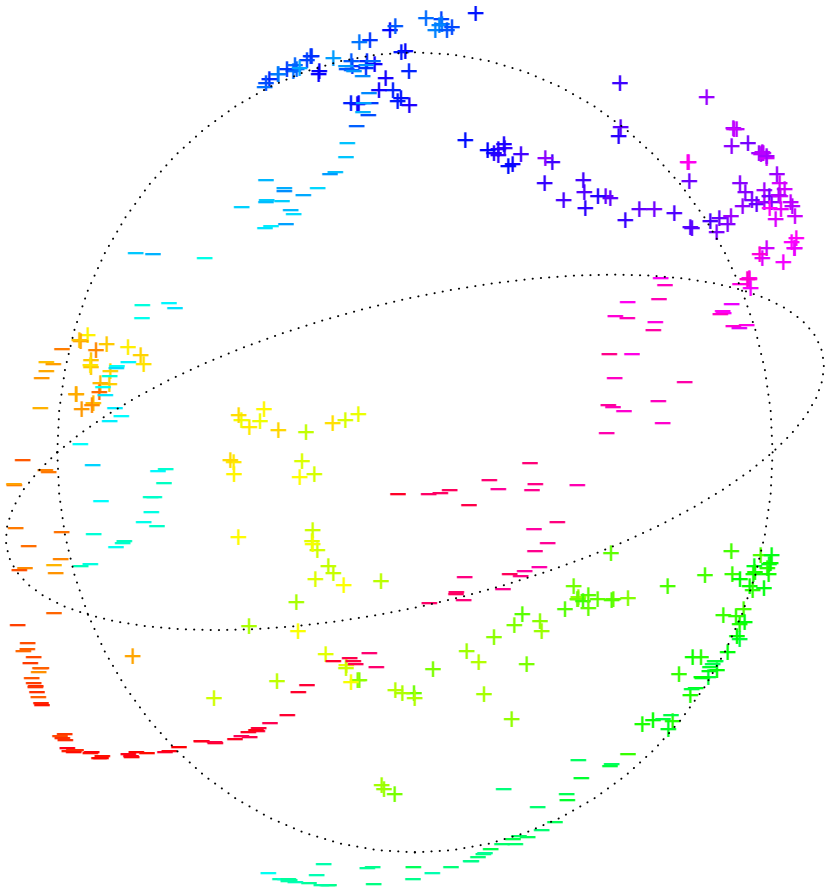


Iteration 795



The best results on the sphere are:

```
Y <- psc_sne_res_62$best_Y
sd3 <- scatterplot3d::scatterplot3d(
  Y, xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  color = x_path_1_cols, xlab = "", ylab = "", zlab = "", axis = FALSE,
  pch = c("+", "-")[ifelse(sign(Y[, 2]) == 1, 1, 2)], grid = FALSE,
  mar = c(0, 0, 0, 0)
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 3)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 3)
```



The path track were properly identified, having a continuous path visualized it with the rainbow gradient color.

$(\mathbb{S}^1)^3$

Let us introduce some different scenarios where each of them has some particular distributions for $(\mathbb{S}^1)^3$.

Mixtures of von Mises–Fisher that generate separated clusters.

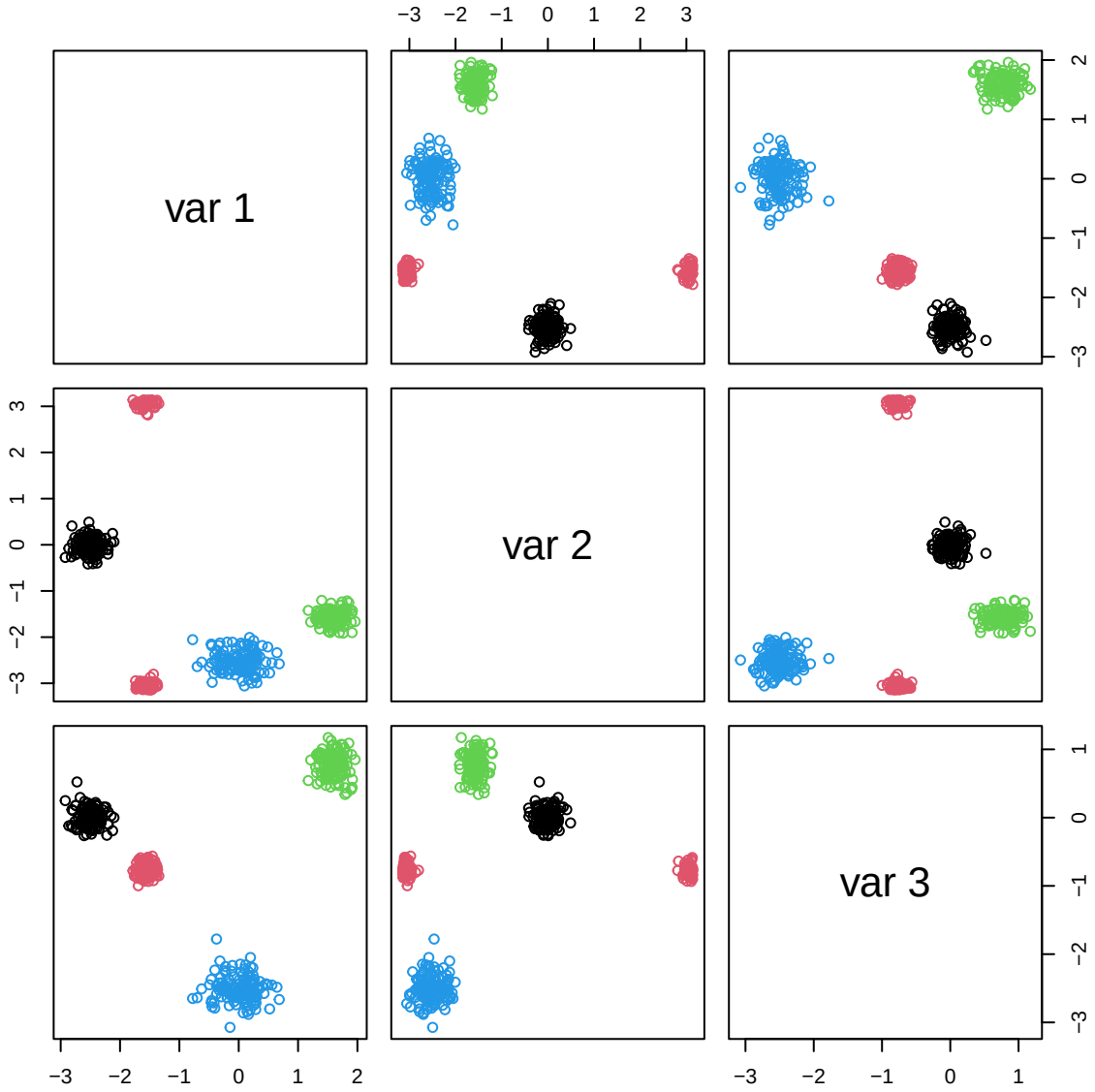
The process to obtain this input data is similar to what was done in the first part of the previous point but in this case we have three spheres in the polysphere instead of just two.

```
n <- 500
n1 <- ceiling(n / 4)
n2 <- ceiling(n / 4)
n3 <- floor(n / 4)
n4 <- floor(n / 4)
mu_mat_1 <- cbind(drop(DirStats::to_cir(3 * pi / 2.5)),
                  drop(DirStats::to_cir(-pi / 2)),
                  drop(DirStats::to_cir(pi / 2)),
                  drop(DirStats::to_cir(0)))
kappa_vec1 <- c(50, 100, 30, 15)
set.seed(3)
sph_1 <- mix_3_mvMF(n, n1 = n1,  n2 = n2, n3 = n3, n4 = n4,
                    mu_mat = mu_mat_1, kappa_vec = kappa_vec1)
mu_mat_2 <- cbind(drop(DirStats::to_cir(0)),
```

```
                drop(DirStats::to_cir(-pi)),
                drop(DirStats::to_cir(-pi / 2)),
                drop(DirStats::to_cir(3 * pi / 2.5)))
kappa_vec2 <- c(40, 85, 35, 20)
set.seed(4)
sph_2 <- mix_3_mvMF(n, n1 = n1,  n2 = n2, n3 = n3, n4 = n4,
                    mu_mat = mu_mat_2, kappa_vec = kappa_vec2)
mu_mat_3 <- cbind(drop(DirStats::to_cir(0)),
                drop(DirStats::to_cir(-pi / 4)),
                drop(DirStats::to_cir(pi / 4)),
                drop(DirStats::to_cir(3 * pi / 2.5)))
kappa_vec3 <- c(65, 105, 28, 25)
set.seed(5)
sph_3 <- mix_3_mvMF(n, n1 = n1,  n2 = n2, n3 = n3, n4 = n4,
                    mu_mat = mu_mat_3, kappa_vec = kappa_vec3)
data <- abind(sph_1, sph_2, sph_3, along = 3)
set.seed(42)
indexes <- sample(1:n)
mix_3_mvMF_data_7 <- data[indexes, , , drop = FALSE]
mix_3_mvMF_cols_7 <- c(rep(1, n1), rep(2, n2), rep(3, n3), rep(4, n4))
mix_3_mvMF_cols_7 <- mix_3_mvMF_cols_7[indexes]
```

Since input data lies on $(\mathbb{S}^1)^3$ is easy to be viewed, first in a diagram of pairs:

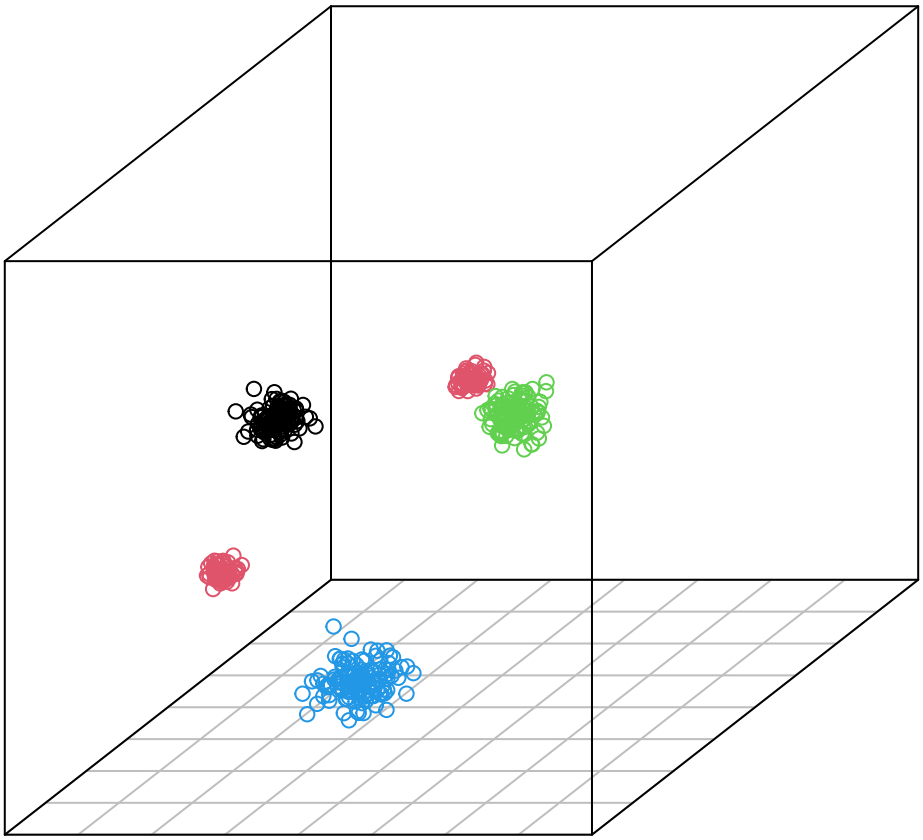
```
mix_3_mvMF_3dtorus <- sdetorus::toPiInt(cbind(
  DirStats::to_rad(mix_3_mvMF_data_7[, , 1]),
  DirStats::to_rad(mix_3_mvMF_data_7[, , 2]),
  DirStats::to_rad(mix_3_mvMF_data_7[, , 3])))

pairs(mix_3_mvMF_3dtorus, col = mix_3_mvMF_cols_7)
```



Or in a 3d-torus:

And it is also possible to see in the 3d-torus:

```
scatterplot3d::scatterplot3d(mix_3_mvMF_3dtorus, xlim = c(-pi, pi),
                             ylim = c(-pi, pi), zlim = c(-pi, pi),
                             color = mix_3_mvMF_cols_7, xlab = "", ylab = "",
                             zlab = "", tick.marks = FALSE)
```

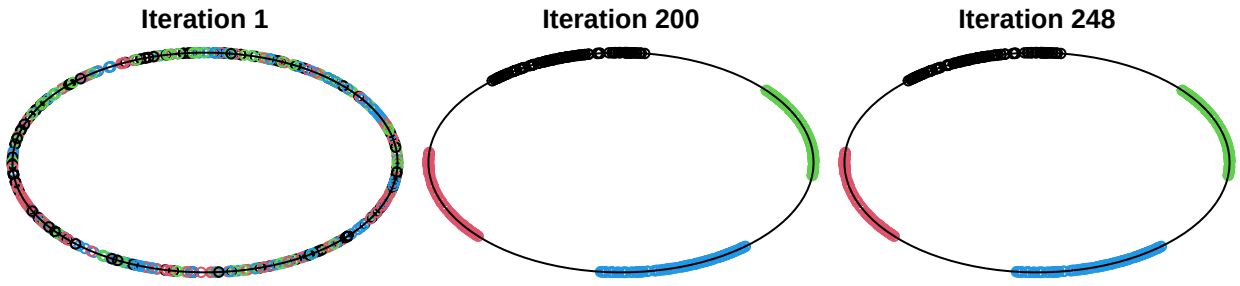


Let's calculate the rho based on a perplexity of 30:

```
rho_30_7 <- rho_optim_bst(x = mix_3_mvMF_data_7, perp_fixed = 30,
                          num_cores = num_cores_param)
#> Time difference of 7.845296 secs
```

Reduction to $d = 1$ Run `psc_sne()` with colors being the groups of variables for $d = 1$:

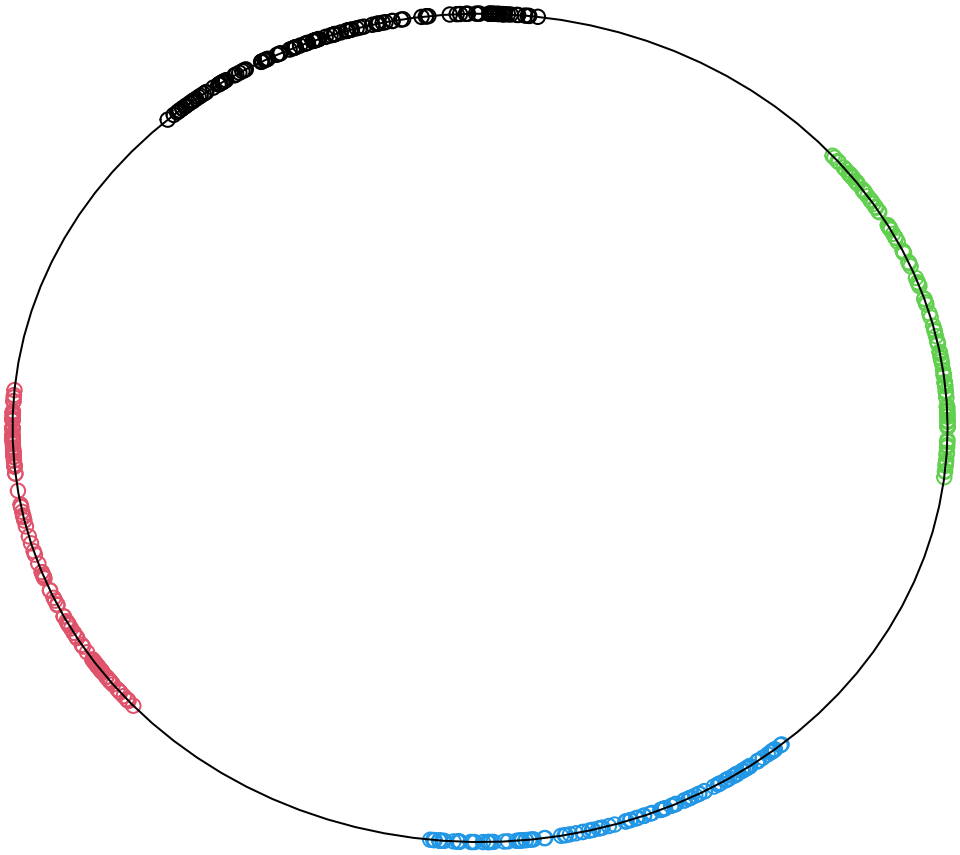
```
res_pscsne_71 <- psc_sne(X = mix_3_mvMF_data_7, d = 1, rho_psc_list = rho_30_7,
                         eta = 50, colors = mix_3_mvMF_cols_7, show_prog = TRUE,
                         parallel_cores = num_cores_param)
#> It: 1 (best: 1); obj: 1.74e+01 (best: 1.74e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 7.2e-02; mom: 0.0e+00
#> It: 100 (best: 18); obj: 1.28e+01 (best: 1.28e+01); abs: 1.8e-06; rel: 1.4e-07; norm: 2.0e-01; mom: 3.1e+00
#> It: 200 (best: 200); obj: 1.73e+00 (best: 1.73e+00); abs: 7.2e-08; rel: 4.2e-08; norm: 2.6e-05; mom: 1.5e-03
#> It: 248 (best: 248); obj: 1.73e+00 (best: 1.73e+00); abs: 1.0e-10; rel: 5.8e-11; norm: 9.8e-07; mom: 5.6e-05
#> CONVERGENCE!
```



The best iteration results onto the circumference:

```
Y <- res_pscsne_71$best_Y
plot(Y[, 1], Y[, 2], col = mix_3_mvMF_cols_7, xlim = c(-1, 1), ylim = c(-1, 1),
     axes = FALSE, xlab = "", ylab = "")
```

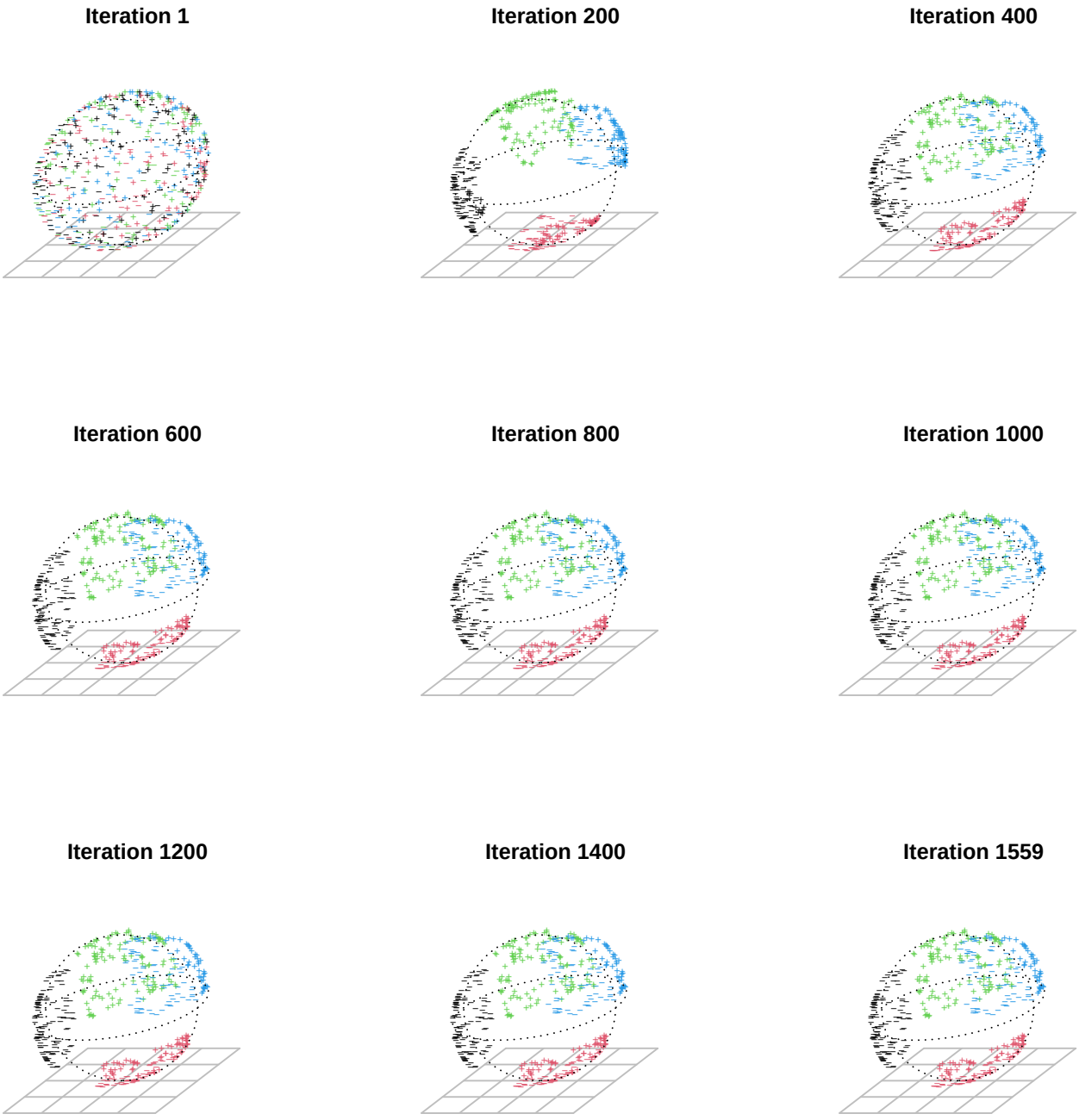
```
th <- seq(0, 2 * pi, length.out = 100)
polygon(x = cos(th), y = sin(th))
```



As we can see in the plot above, the three components of the von Mises–Fisher mixture are identified.

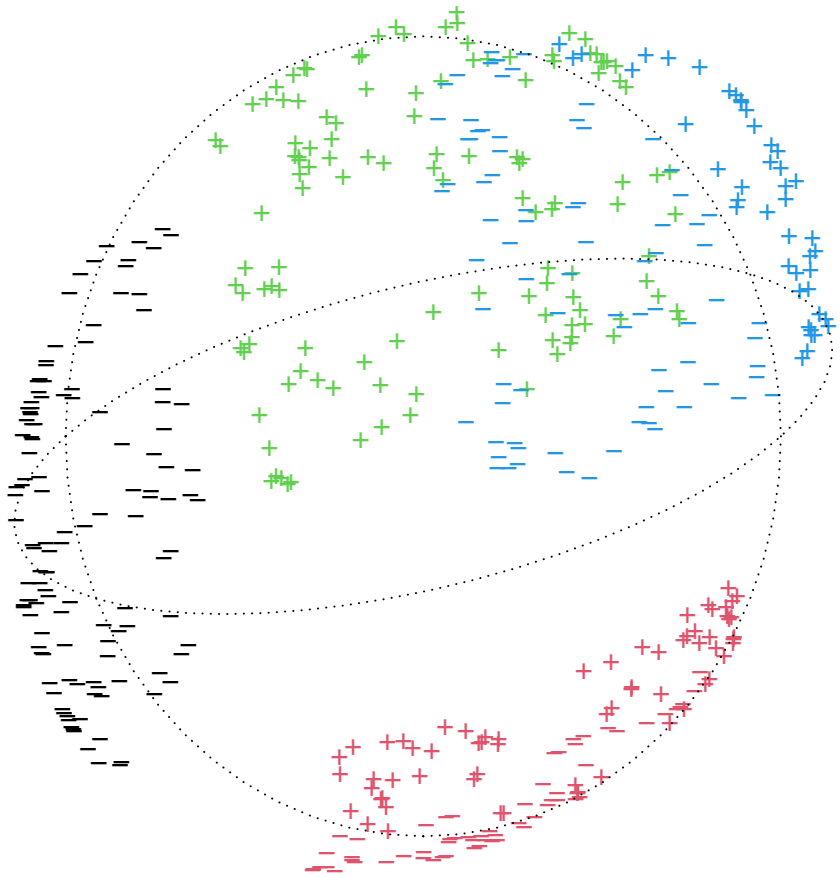
Reduction to $d = 2$ Let's run the psc-SNE algorithm for $d = 2$ and see if results are good as well:

```
set.seed(42)
psc_sne_res_72 <- psc_sne(mix_3_mvMF_data_7, d = 2, rho_psc_list = rho_30_7,
                          eta = 10, colors = mix_3_mvMF_cols_7,
                          show_prog = TRUE, parallel_cores = num_cores_param,
                          maxit = 2000)
#> It: 1 (best: 1); obj: 1.92e+01 (best: 1.92e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 1.6e-01; mom: 0.0e+00
#> It: 100 (best: 100); obj: 1.13e+01 (best: 1.13e+01); abs: 7.5e-05; rel: 6.6e-06; norm: 1.0e-03; mom: 1.0e-02
#> It: 200 (best: 200); obj: 1.05e+00 (best: 1.05e+00); abs: 3.0e-04; rel: 2.8e-04; norm: 3.9e-03; mom: 3.8e-02
#> It: 300 (best: 300); obj: 1.02e+00 (best: 1.02e+00); abs: 3.5e-06; rel: 3.4e-06; norm: 2.5e-04; mom: 1.2e-02
#> It: 400 (best: 400); obj: 1.02e+00 (best: 1.02e+00); abs: 5.1e-07; rel: 5.0e-07; norm: 9.9e-05; mom: 4.2e-03
#> It: 500 (best: 500); obj: 1.02e+00 (best: 1.02e+00); abs: 9.6e-08; rel: 9.4e-08; norm: 4.3e-05; mom: 1.8e-03
#> It: 600 (best: 600); obj: 1.02e+00 (best: 1.02e+00); abs: 2.4e-08; rel: 2.4e-08; norm: 2.2e-05; mom: 9.0e-04
#> It: 700 (best: 700); obj: 1.02e+00 (best: 1.02e+00); abs: 8.6e-09; rel: 8.4e-09; norm: 1.3e-05; mom: 5.3e-04
#> It: 800 (best: 800); obj: 1.02e+00 (best: 1.02e+00); abs: 3.8e-09; rel: 3.7e-09; norm: 8.6e-06; mom: 3.5e-04
#> It: 900 (best: 900); obj: 1.02e+00 (best: 1.02e+00); abs: 1.9e-09; rel: 1.8e-09; norm: 6.1e-06; mom: 2.5e-04
#> It: 1000 (best: 1000); obj: 1.02e+00 (best: 1.02e+00); abs: 1.0e-09; rel: 9.9e-10; norm: 4.5e-06; mom: 1.8e-04
#> It: 1100 (best: 1100); obj: 1.02e+00 (best: 1.02e+00); abs: 5.7e-10; rel: 5.6e-10; norm: 3.4e-06; mom: 1.4e-04
#> It: 1200 (best: 1200); obj: 1.02e+00 (best: 1.02e+00); abs: 3.3e-10; rel: 3.2e-10; norm: 2.6e-06; mom: 1.0e-04
#> It: 1300 (best: 1300); obj: 1.02e+00 (best: 1.02e+00); abs: 1.9e-10; rel: 1.9e-10; norm: 2.0e-06; mom: 8.0e-05
#> It: 1400 (best: 1400); obj: 1.02e+00 (best: 1.02e+00); abs: 1.2e-10; rel: 1.1e-10; norm: 1.5e-06; mom: 6.1e-05
#> It: 1500 (best: 1500); obj: 1.02e+00 (best: 1.02e+00); abs: 6.8e-11; rel: 6.7e-11; norm: 1.2e-06; mom: 4.7e-05
#> It: 1559 (best: 1559); obj: 1.02e+00 (best: 1.02e+00); abs: 5.0e-11; rel: 4.9e-11; norm: 1.0e-06; mom: 4.0e-05
#> CONVERGENCE!
```



The best results on the sphere are:

```
Y <- psc_sne_res_72$best_Y
sd3 <- scatterplot3d::scatterplot3d(
  Y, xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  color = mix_3_mvMF_cols_7, xlab = "", ylab = "", zlab = "", axis = FALSE,
  pch = c("+", "-")[ifelse(sign(Y[, 2]) == 1, 1, 2)], grid = FALSE,
  mar = c(0, 0, 0, 0)
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 3)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 3)
```



As it happened for $d = 1$, all components were found in three different clusters.

Mixtures of trivariate wrapped normals to have controlled clusters with correlations

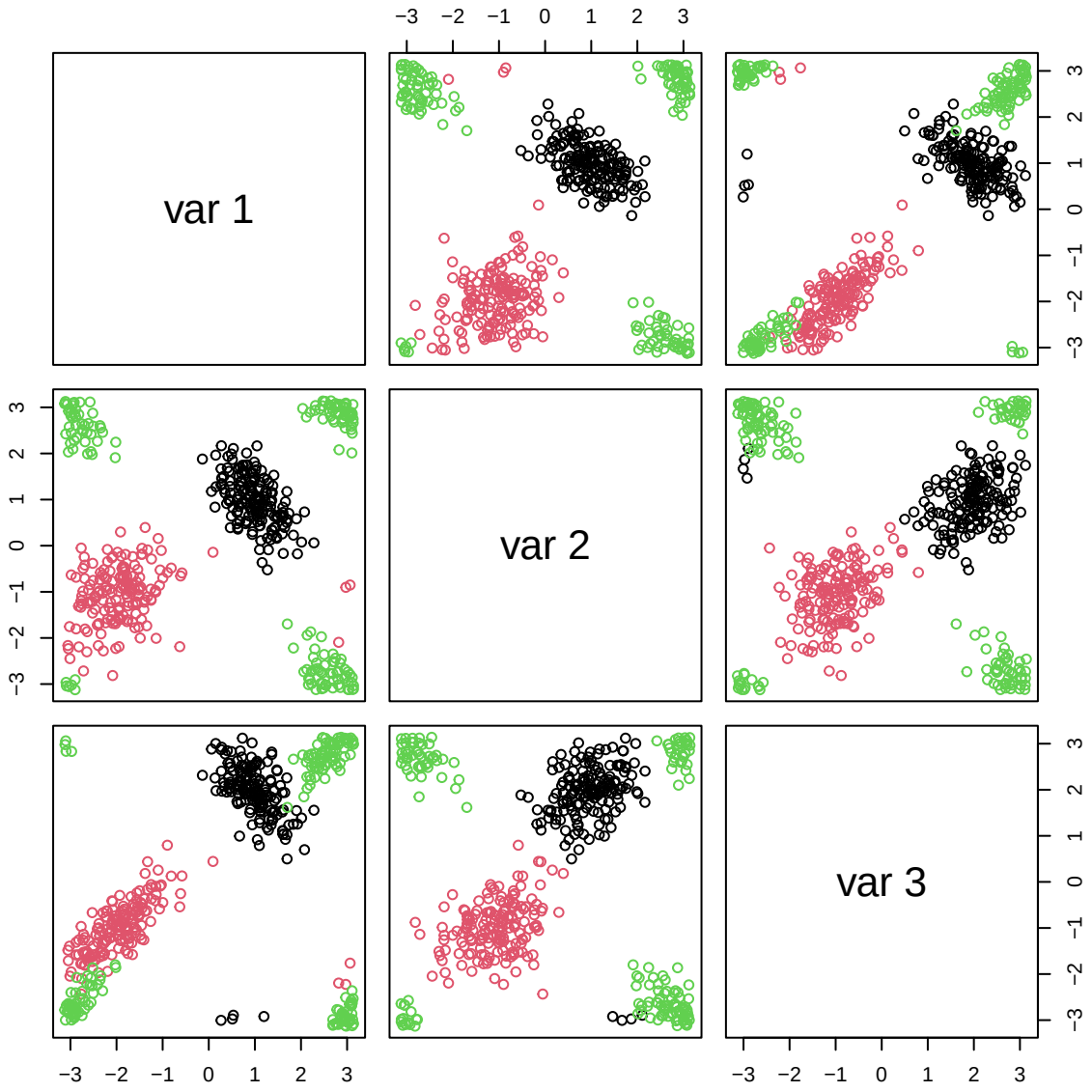
This case is similar to the bivariate one commented before with the difference that the trivariate normal are defined by a vector of means with three elements and a covariance matrix of size three. First, let's represent this data in the 3d-torus:

```
n <- 500
mean1 <- c(1, 1, 2)
mean2 <- c(-2, -1, -1)
mean3 <- c(3, 3, 3.25)
sigma1 <- matrix(c(0.22, -0.10, -0.15, -0.10, 0.28, 0.10, -0.15, 0.10, 0.30),
                 byrow = TRUE, nrow = 3)
sigma2 <- toeplitz(c(0.32, 0.1, 0.25))
sigma3 <- toeplitz(c(0.25, -0.15, 0.21))
n1 <- ceiling(n / 3)
n2 <- ceiling(n / 3)
n3 <- floor(n / 3)
set.seed(42)
x_8_bvnorm <- mix_3_mvnorm(n = n, n1 = n1, n2 = n2, n3 = n3,
                           mean_mat = cbind(mean1, mean2, mean3),
                           sigma1 = sigma1, sigma2 = sigma2, sigma3 = sigma3)
x_8_torus <- sdetorus::toPiInt(x_8_bvnorm)

samp_8 <- abind(DirStats::to_cir(x_8_torus[, 1]),
                DirStats::to_cir(x_8_torus[, 2]),
```

```
                DirStats::to_cir(x_8_torus[, 3]), along = 3)
samp_8_cols <- c(rep(1, n1), rep(2, n2), rep(3, n3))
```

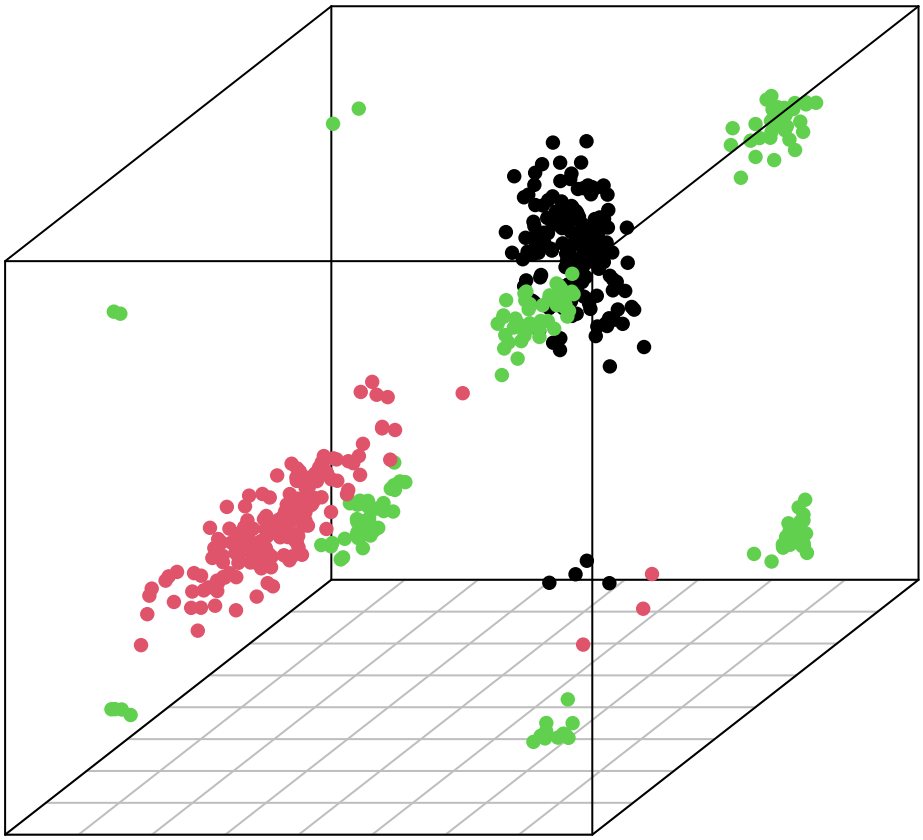
Let's visualize the data in the pairs plot:

```
pairs(x_8_torus, col = samp_8_cols)
```



Since data lie in $(\mathbb{S}^1)^3$ can also be viewed in the 3d-torus:

```
scatterplot3d::scatterplot3d(x_8_torus, xlim = c(-pi, pi), ylim = c(-pi, pi),
                             zlim = c(-pi, pi), xlab = "", ylab = "", zlab = "",
                             color = samp_8_cols, pch = 16, tick.marks = FALSE)
```



First, the arrangement of the dataset is altered in order to be more complicated for the algorithm to discover the clusters:

```
set.seed(42)
indexes <- sample(1:n)
samp_8 <- samp_8[indexes, , , drop = FALSE]
samp_8_cols <- samp_8_cols[indexes]
```

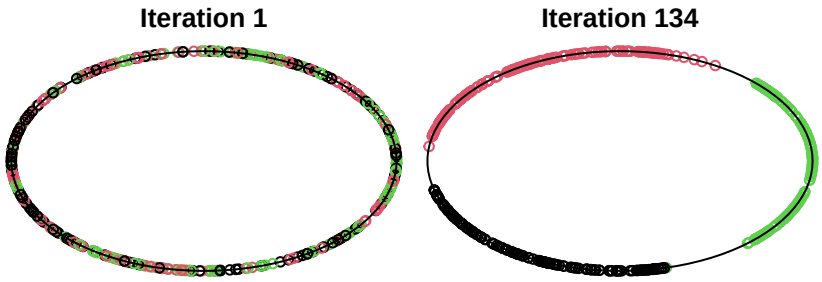
Let's calculate rho based on a fixed perplexity of 30:

```
rho_30_8 <- rho_optim_bst(x = samp_8, perp_fixed = 30,
                          num_cores = num_cores_param)
#> Time difference of 7.347097 secs
```

Now, everything is ready to reduce the dimension into $d = 1$ and $d = 2$:

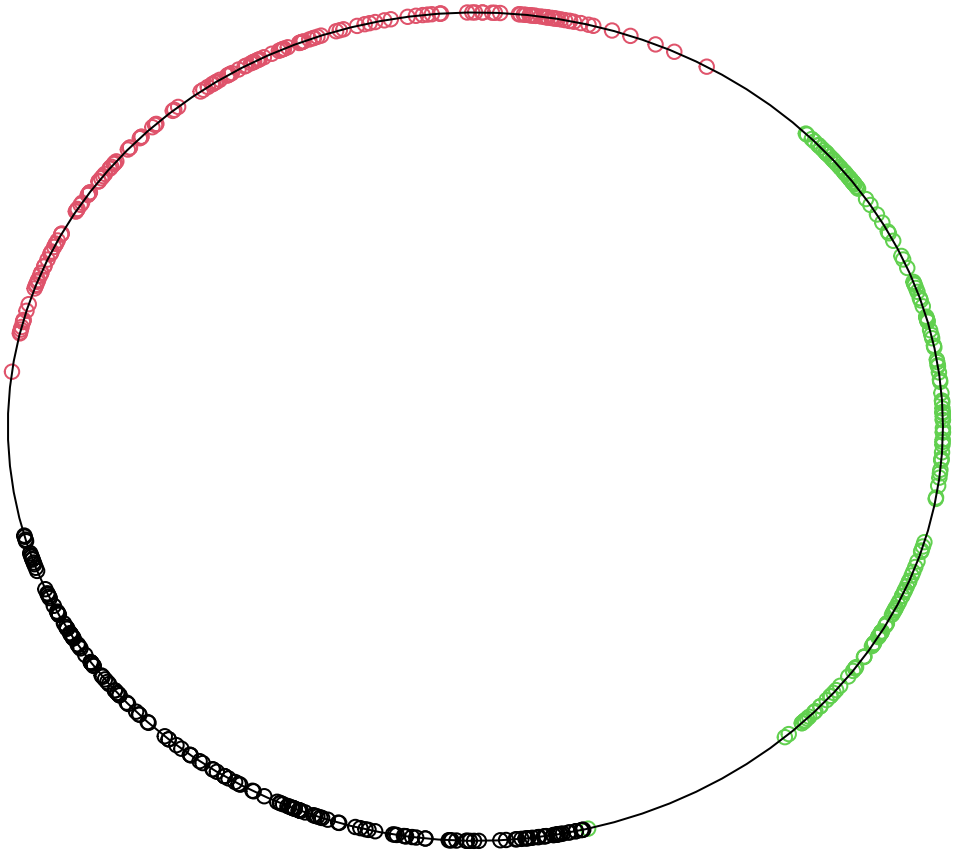
Reduction to $d = 1$ Run `psc_sne()` with colors being the groups of variables for $d = 1$:

```
set.seed(42)
res_pscsne_81 <- psc_sne(X = samp_8, d = 1, rho_psc_list = rho_30_8,
                         eta = 50, colors = samp_8_cols, show_prog = TRUE,
                         parallel_cores = num_cores_param, init = "random")
#> It: 1 (best: 1); obj: 1.74e+01 (best: 1.74e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 7.7e-02; mom: 0.0e+00
#> It: 100 (best: 30); obj: 1.30e+01 (best: 1.30e+01); abs: 1.6e-03; rel: 1.2e-04; norm: 1.9e-01; mom: 3.0e+00
#> It: 134 (best: 134); obj: 1.74e+00 (best: 1.74e+00); abs: 3.4e-11; rel: 2.0e-11; norm: 8.9e-07; mom: 4.7e-05
#> CONVERGENCE!
```



The best iteration results onto the circumference:

```
Y <- res_pscsne_81$best_Y
plot(Y[, 1], Y[, 2], col = samp_8_cols, xlim = c(-1, 1), ylim = c(-1, 1),
     axes = FALSE, xlab = "", ylab = "")
th <- seq(0, 2 * pi, length.out = 100)
polygon(x = cos(th), y = sin(th))
```



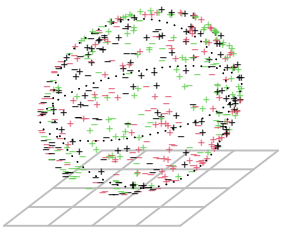
In the previous plot is possible to see that the three components of the mixture are identified properly. The init parameter is set to `random` to avoid the not satisfactory results of the default behavior since there was an isolated point point from the green cluster in the opposite side of the data points related to this group.

Reduction to $d = 2$ Let's run the psc-SNE algorithm for $d = 2$ and see if results are good as well:

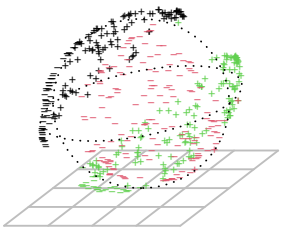
```
psc_sne_res_82 <- psc_sne(samp_8, d = 2, rho_psc_list = rho_30_8,
                          eta = 50, colors = samp_8_cols,
                          show_prog = TRUE, parallel_cores = num_cores_param)
#> It: 1 (best: 1); obj: 1.84e+01 (best: 1.84e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 1.5e-01; mom: 0.0e+00
#> It: 100 (best: 14); obj: 1.28e+01 (best: 1.26e+01); abs: 1.3e-02; rel: 1.0e-03; norm: 5.8e-01; mom: 7.6e+00
#> It: 200 (best: 200); obj: 9.95e-01 (best: 9.95e-01); abs: 7.4e-07; rel: 7.5e-07; norm: 8.6e-05; mom: 4.3e-03
#> It: 300 (best: 300); obj: 9.95e-01 (best: 9.95e-01); abs: 4.1e-07; rel: 4.1e-07; norm: 3.9e-05; mom: 8.6e-03
```

```
#> It: 400 (best: 400); obj: 9.94e-01 (best: 9.94e-01); abs: 4.5e-09; rel: 4.6e-09; norm: 4.1e-06; mom: 9.3e-04
#> It: 457 (best: 457); obj: 9.94e-01 (best: 9.94e-01); abs: 2.6e-10; rel: 2.6e-10; norm: 9.8e-07; mom: 2.2e-04
#> CONVERGENCE!
```

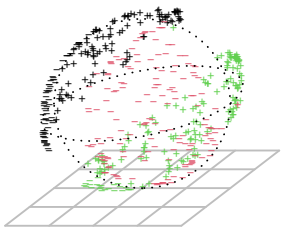
Iteration 1



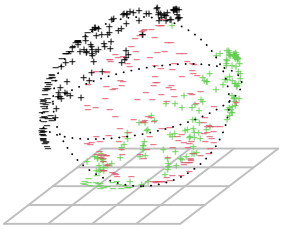
Iteration 200



Iteration 400

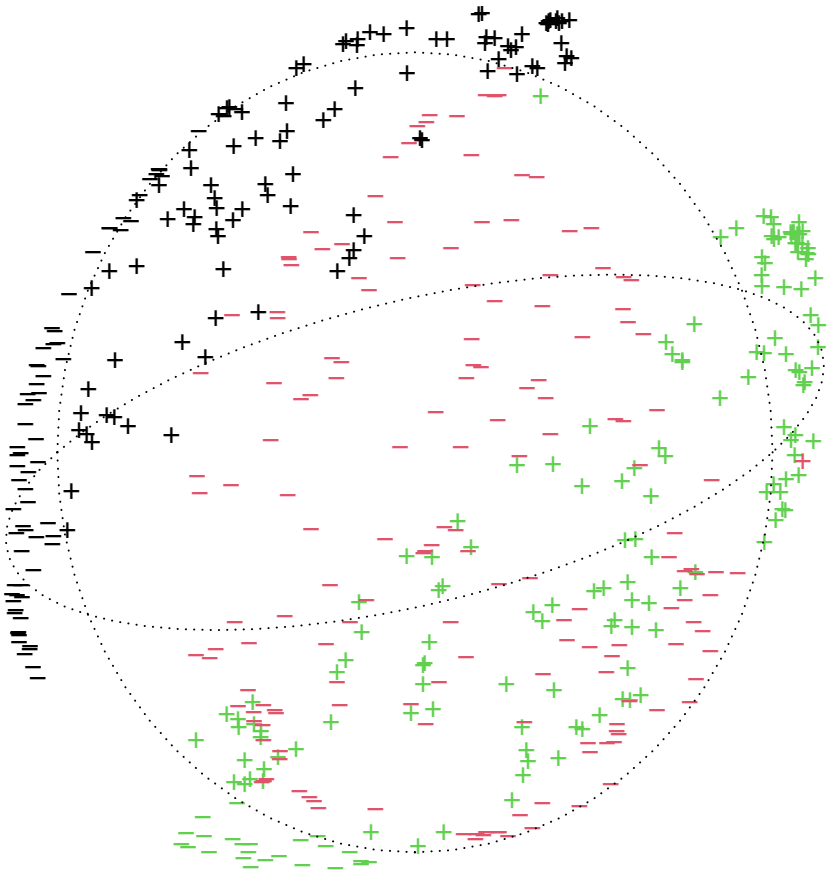


Iteration 457



The best results on the sphere are:

```
Y <- psc_sne_res_82$best_Y
sd3 <- scatterplot3d::scatterplot3d(
  Y, xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  color = samp_8_cols, xlab = "", ylab = "", zlab = "", axis = FALSE,
  pch = c("+", "-")[ifelse(sign(Y[, 2]) == 1, 1, 2)], grid = FALSE,
  mar = c(0, 0, 0, 0)
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 3)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 3)
```



As it happened for $d = 1$, all components were found in three different clusters.

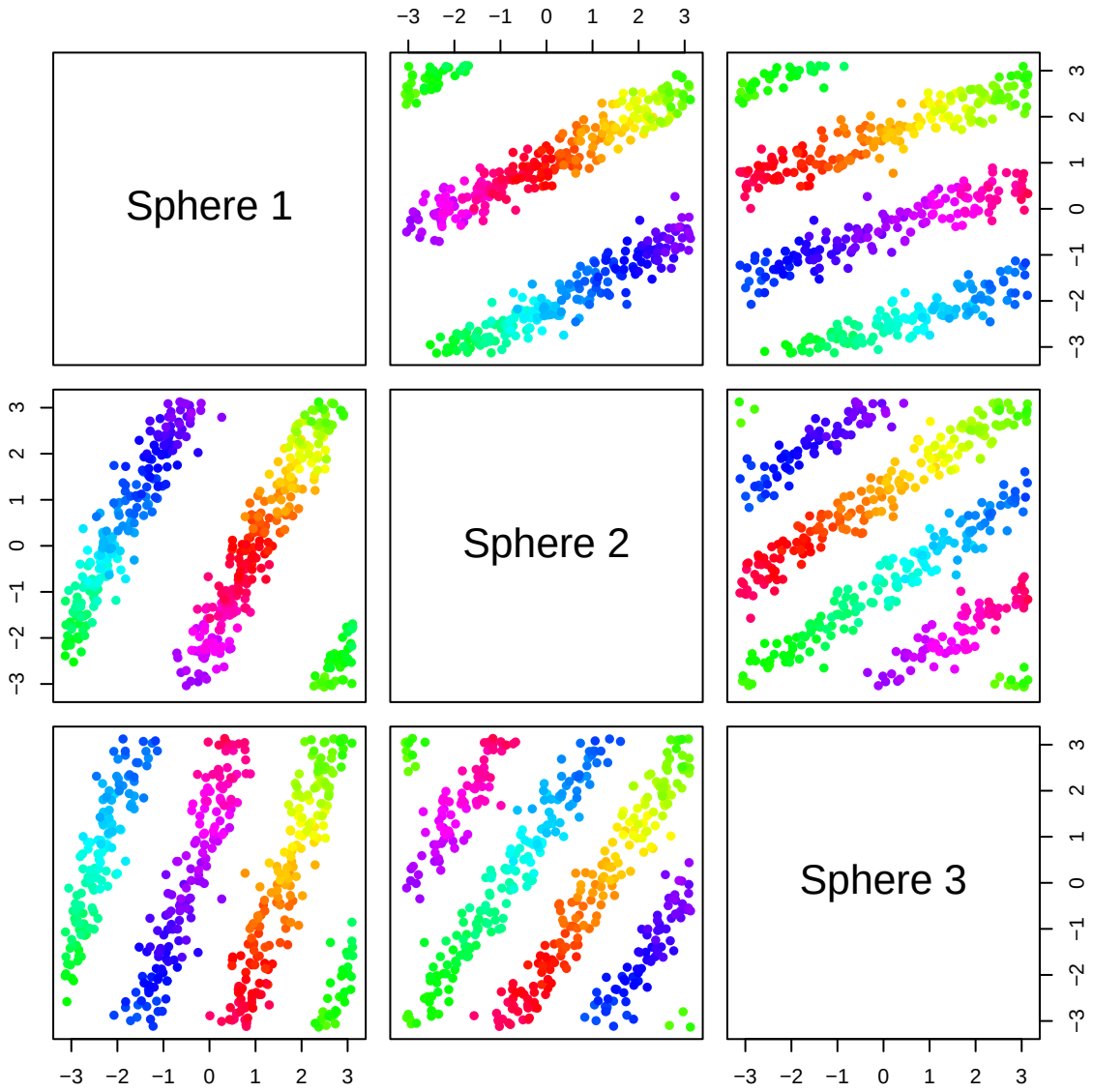
Data along a continuous path that crosses the boundaries

As it happened before, data's structure is possible to see thanks to the 3d-torus. We will represent this continuous path with the colors of the rainbow, it will help to identify if the reduction is properly done later on.

```
n <- 500
set.seed(4)
samp_9 <- r_path_s1r(n = n, r = 3, angles = TRUE, k = c(1, 2, 3))
samp_9_cols <- rainbow(n, alpha = 1)
```

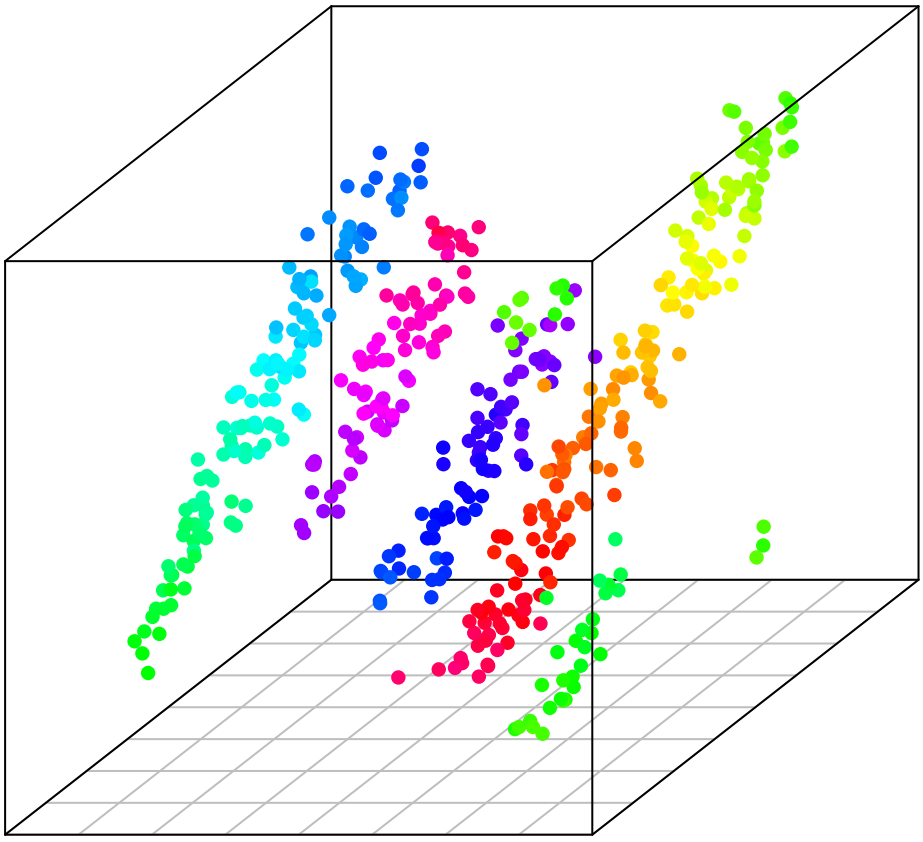
A pairs plot is shown in order to see the structure of the continuous path as follows.

```
pairs(samp_9, xlim = c(-pi, pi), ylim = c(-pi, pi), col = samp_9_cols,
      pch = 16, labels = c("Sphere 1", "Sphere 2", "Sphere 3"))
```



Data can also be visualized in the 3d-torus:

```
scatterplot3d::scatterplot3d(samp_9, xlim = c(-pi, pi), ylim = c(-pi, pi),
                             zlim = c(-pi, pi), xlab = "", ylab = "", zlab = "",
                             color = samp_9_cols, pch = 16, tick.marks = FALSE)
```



Let's convert the torus data into the polysphere $(\mathbb{S}^1)^2$ (Cartesian coordinates) and the index of the dataset is altered to complicate the search of the path:

```
x_1 <- DirStats::to_cir(samp_9[, 1])
x_2 <- DirStats::to_cir(samp_9[, 2])
x_3 <- DirStats::to_cir(samp_9[, 3])
x_path_2 <- abind(x_1, x_2, x_3, along = 3)
set.seed(42)
indexes <- sample(1:n)
x_path_2 <- x_path_2[indexes, , , drop = FALSE]
samp_9_cols <- samp_9_cols[indexes]
```

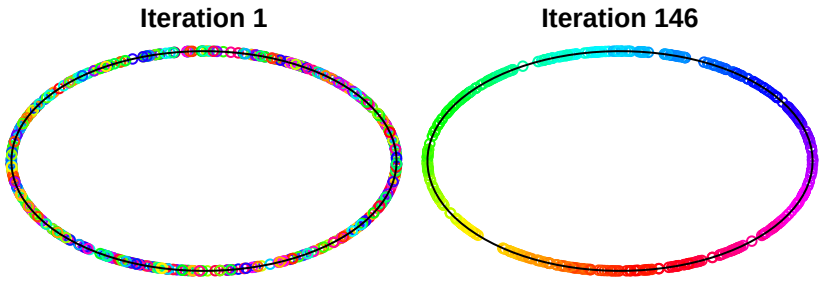
Let's calculate now the values of rho, $\boldsymbol{\rho}$, based on a fixed perplexity of 30:

```
rho_30_9 <- rho_optim_bst(x = x_path_2, perp_fixed = 30,
                          num_cores = num_cores_param)
#> Time difference of 7.04356 secs
```

After all of this done, everything is prepared to reduce the dimension of the data to $d = 1$ and $d = 2$:

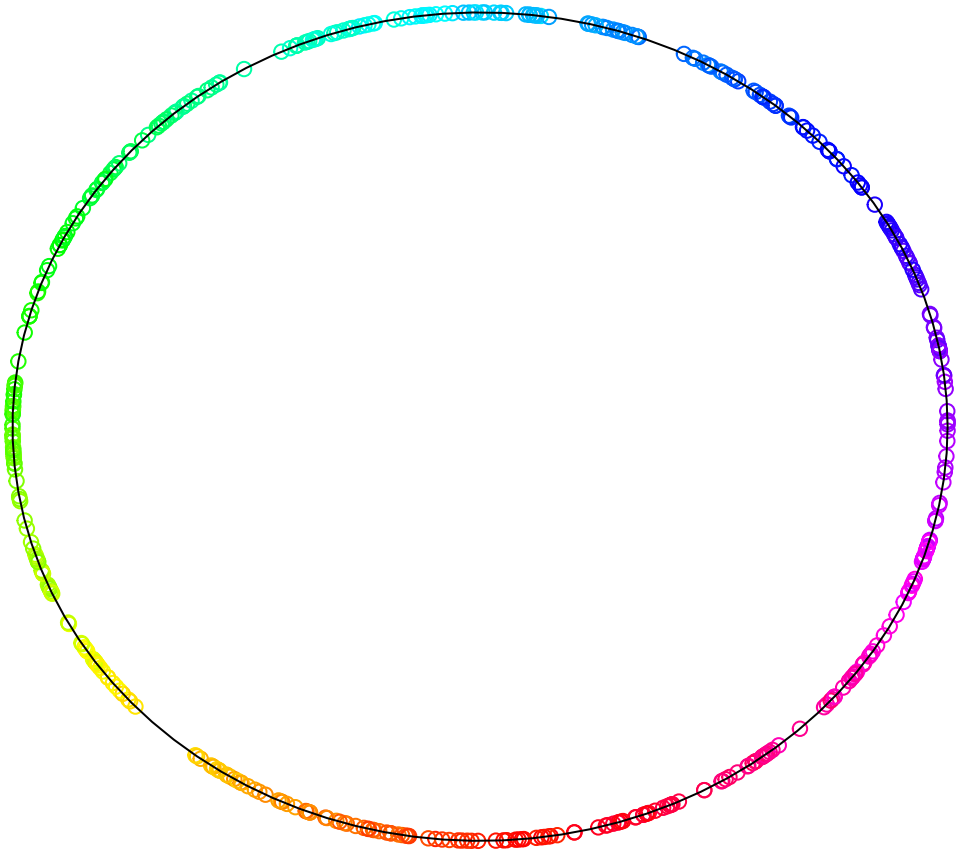
Reduction to $d = 1$ Run `psc_sne()` with colors being the groups of variables for $d = 1$:

```
set.seed(42)
res_pscsne_91 <- psc_sne(X = x_path_2, d = 1, rho_psc_list = rho_30_9,
                         eta = 25, colors = samp_9_cols, show_prog = TRUE,
                         parallel_cores = num_cores_param, init = "random")
#> It: 1 (best: 1); obj: 1.78e+01 (best: 1.78e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 6.5e-02; mom: 0.0e+00
#> It: 100 (best: 85); obj: 1.25e+01 (best: 1.25e+01); abs: 1.7e-07; rel: 1.3e-08; norm: 1.2e-05; mom: 4.0e-04
#> It: 146 (best: 146); obj: 1.73e+00 (best: 1.73e+00); abs: 5.1e-11; rel: 3.0e-11; norm: 9.7e-07; mom: 3.2e-05
#> CONVERGENCE!
```



The best iteration results onto the circumference:

```
Y <- res_pscsne_91$best_Y
plot(Y[, 1], Y[, 2], col = samp_9_cols, xlim = c(-1, 1), ylim = c(-1, 1),
     axes = FALSE, xlab = "", ylab = "")
th <- seq(0, 2 * pi, length.out = 100)
polygon(x = cos(th), y = sin(th))
```



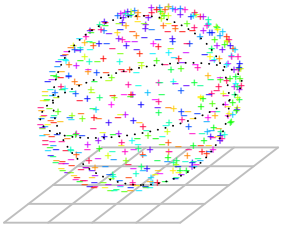
It is possible to see a perfect arrangement of the path. This could be achieved defining the initialization of the zero-iteration solution randomly uniform. Let's see if results are as good when reducing to $d = 2$.

Reduction to $d = 2$ Let's run the psc-SNE algorithm for $d = 2$ and see if results are better:

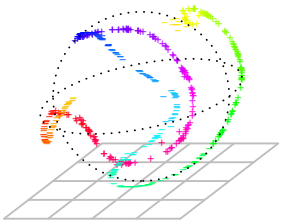
```
psc_sne_res_92 <- psc_sne(x_path_2, d = 2, rho_psc_list = rho_30_9,
                          eta = 10, colors = samp_9_cols,
                          show_prog = TRUE, parallel_cores = num_cores_param,
                          maxit = 3000)
#> It: 1 (best: 1); obj: 1.95e+01 (best: 1.95e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 1.4e-01; mom: 0.0e+00
#> It: 100 (best: 100); obj: 9.44e+00 (best: 9.44e+00); abs: 7.7e-03; rel: 8.1e-04; norm: 1.2e-02; mom: 1.4e-01
#> It: 200 (best: 200); obj: 8.91e-01 (best: 8.91e-01); abs: 2.6e-05; rel: 2.9e-05; norm: 1.1e-03; mom: 1.2e-02
#> It: 300 (best: 300); obj: 8.88e-01 (best: 8.88e-01); abs: 2.8e-05; rel: 3.2e-05; norm: 7.4e-04; mom: 3.1e-02
#> It: 400 (best: 400); obj: 8.87e-01 (best: 8.87e-01); abs: 8.5e-06; rel: 9.6e-06; norm: 4.1e-04; mom: 1.7e-02
```

```
#> It: 500 (best: 500); obj: 8.86e-01 (best: 8.86e-01); abs: 8.9e-07; rel: 1.0e-06; norm: 1.3e-04; mom: 5.5e-03
#> It: 600 (best: 600); obj: 8.86e-01 (best: 8.86e-01); abs: 2.8e-07; rel: 3.1e-07; norm: 7.4e-05; mom: 3.0e-03
#> It: 700 (best: 700); obj: 8.86e-01 (best: 8.86e-01); abs: 1.8e-07; rel: 2.0e-07; norm: 6.0e-05; mom: 2.4e-03
#> It: 800 (best: 800); obj: 8.86e-01 (best: 8.86e-01); abs: 1.4e-07; rel: 1.6e-07; norm: 5.3e-05; mom: 2.1e-03
#> It: 900 (best: 900); obj: 8.86e-01 (best: 8.86e-01); abs: 1.1e-07; rel: 1.3e-07; norm: 4.8e-05; mom: 1.9e-03
#> It: 1000 (best: 1000); obj: 8.86e-01 (best: 8.86e-01); abs: 8.8e-08; rel: 9.9e-08; norm: 4.2e-05; mom: 1.7e-03
#> It: 1100 (best: 1100); obj: 8.86e-01 (best: 8.86e-01); abs: 6.5e-08; rel: 7.4e-08; norm: 3.6e-05; mom: 1.5e-03
#> It: 1200 (best: 1200); obj: 8.86e-01 (best: 8.86e-01); abs: 4.7e-08; rel: 5.3e-08; norm: 3.1e-05; mom: 1.2e-03
#> It: 1300 (best: 1300); obj: 8.86e-01 (best: 8.86e-01); abs: 3.3e-08; rel: 3.8e-08; norm: 2.6e-05; mom: 1.0e-03
#> It: 1400 (best: 1400); obj: 8.86e-01 (best: 8.86e-01); abs: 2.3e-08; rel: 2.6e-08; norm: 2.1e-05; mom: 8.6e-04
#> It: 1500 (best: 1500); obj: 8.86e-01 (best: 8.86e-01); abs: 1.6e-08; rel: 1.8e-08; norm: 1.8e-05; mom: 7.1e-04
#> It: 1600 (best: 1600); obj: 8.86e-01 (best: 8.86e-01); abs: 1.0e-08; rel: 1.2e-08; norm: 1.4e-05; mom: 5.8e-04
#> It: 1700 (best: 1700); obj: 8.86e-01 (best: 8.86e-01); abs: 6.8e-09; rel: 7.6e-09; norm: 1.2e-05; mom: 4.7e-04
#> It: 1800 (best: 1800); obj: 8.86e-01 (best: 8.86e-01); abs: 4.4e-09; rel: 4.9e-09; norm: 9.3e-06; mom: 3.8e-04
```

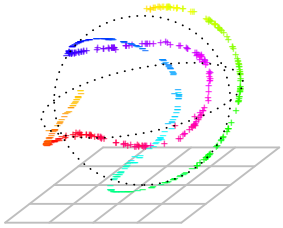
Iteration 1



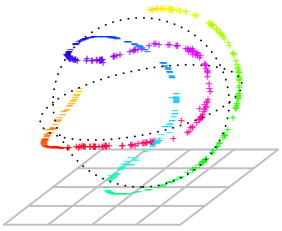
Iteration 200



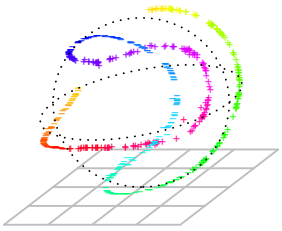
Iteration 400



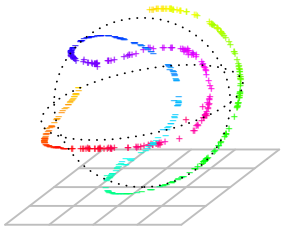
Iteration 600



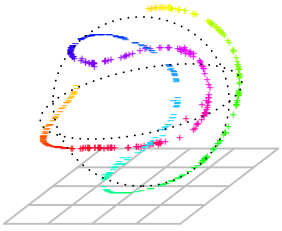
Iteration 800



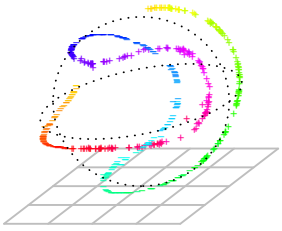
Iteration 1000



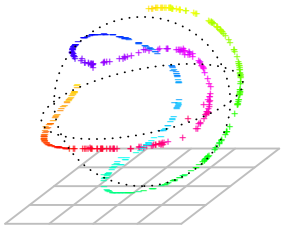
Iteration 1200



Iteration 1400



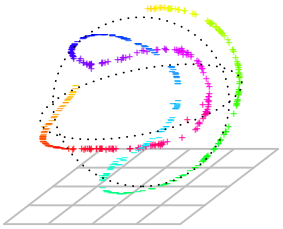
Iteration 1600



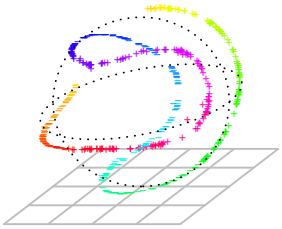
```
#> It: 1900 (best: 1900); obj: 8.86e-01 (best: 8.86e-01); abs: 2.8e-09; rel: 3.2e-09; norm: 7.5e-06; mom: 3.0e-04
#> It: 2000 (best: 2000); obj: 8.86e-01 (best: 8.86e-01); abs: 1.8e-09; rel: 2.0e-09; norm: 6.0e-06; mom: 2.4e-04
#> It: 2100 (best: 2100); obj: 8.86e-01 (best: 8.86e-01); abs: 1.1e-09; rel: 1.3e-09; norm: 4.7e-06; mom: 1.9e-04
```

```
#> It: 2200 (best: 2200); obj: 8.86e-01 (best: 8.86e-01); abs: 7.2e-10; rel: 8.1e-10; norm: 3.8e-06; mom: 1.5e-04
#> It: 2300 (best: 2300); obj: 8.86e-01 (best: 8.86e-01); abs: 4.5e-10; rel: 5.1e-10; norm: 3.0e-06; mom: 1.2e-04
#> It: 2400 (best: 2400); obj: 8.86e-01 (best: 8.86e-01); abs: 2.8e-10; rel: 3.2e-10; norm: 2.4e-06; mom: 9.6e-05
#> It: 2500 (best: 2500); obj: 8.86e-01 (best: 8.86e-01); abs: 1.8e-10; rel: 2.0e-10; norm: 1.9e-06; mom: 7.6e-05
#> It: 2600 (best: 2600); obj: 8.86e-01 (best: 8.86e-01); abs: 1.1e-10; rel: 1.2e-10; norm: 1.5e-06; mom: 6.0e-05
#> It: 2700 (best: 2700); obj: 8.86e-01 (best: 8.86e-01); abs: 6.9e-11; rel: 7.8e-11; norm: 1.2e-06; mom: 4.7e-05
#> It: 2767 (best: 2767); obj: 8.86e-01 (best: 8.86e-01); abs: 5.0e-11; rel: 5.7e-11; norm: 1.0e-06; mom: 4.0e-05
#> CONVERGENCE!
```

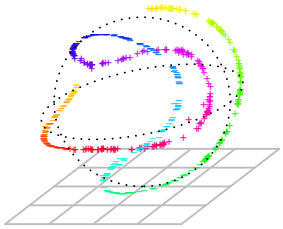
Iteration 1800



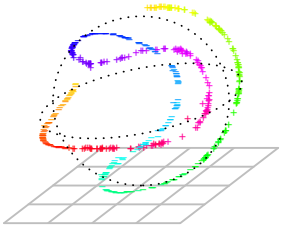
Iteration 2000



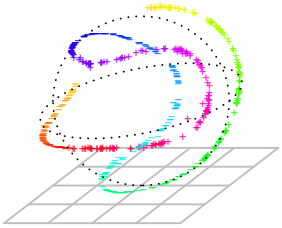
Iteration 2200



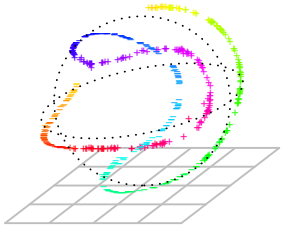
Iteration 2400



Iteration 2600

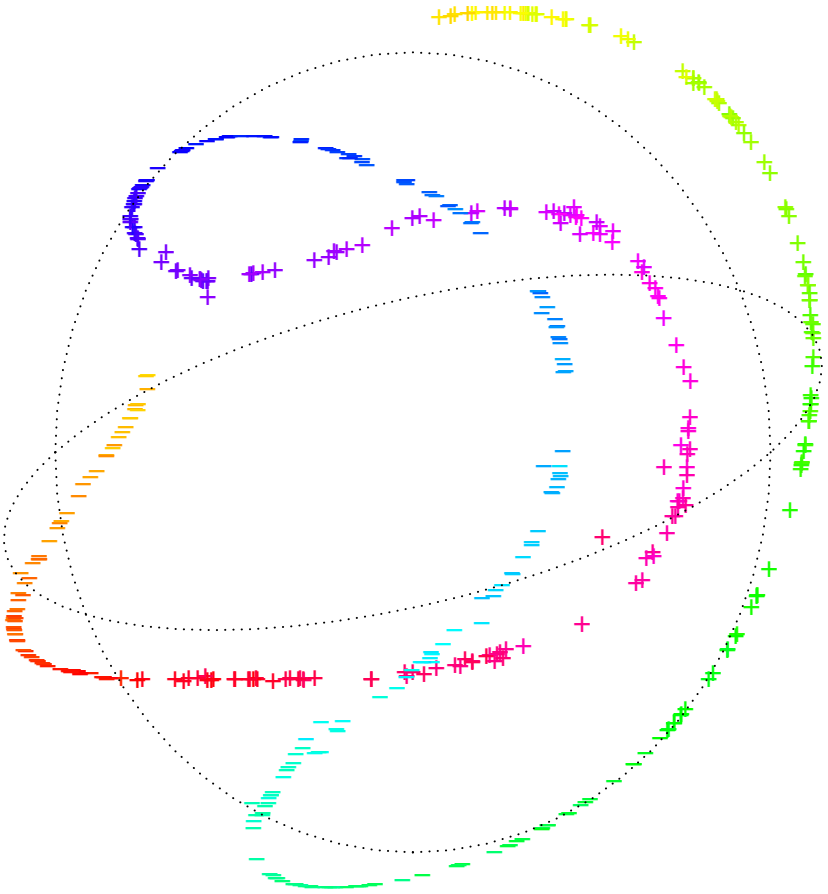


Iteration 2767



The best results on the sphere are:

```
Y <- psc_sne_res_92$best_Y
sd3 <- scatterplot3d::scatterplot3d(
  Y, xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  color = samp_9_cols, xlab = "", ylab = "", zlab = "", axis = FALSE,
  pch = c("+", "-")[ifelse(sign(Y[, 2]) == 1, 1, 2)], grid = FALSE,
  mar = c(0, 0, 0, 0)
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 3)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 3)
```



As it happened with the reduction to $d = 1$, the path track were properly identified, having a continuous path visualized it with the rainbow gradient color. Nevertheless, the total number of iterations has been increased and the learning rate has been decreased to 10 to find convergence.