Project

2025-03-30

Normal Mixture model tests

helper functions

```
library(mvtnorm)
library(stats)
library(ggplot2)
library(proxy)
```

```
##
## Attaching package: 'proxy'
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

```
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
library(clue)

source("em_general.R")

generate_GMM = function(n, k=3, dim = 2, mu_list = list(c(0,0), c(0,2), c(2,0)), prob = rep(1/3, 3), I2
  # n: number of samples
  # k: number of components
  # dim: # of dimension of the sample, mu = c(0,0) should be dim 2
  # prob: list of numbers(int/float) of probability of each cluster, should be of length k
  # I2_list: list of covariate matrices for rmvnorm(), list should be of length k, matrix should be dim

  #sigma2 <- 1 / beta  # Variance = 1/beta
  #I2 <- diag(1/beta, dim) #* sigma2  # Covariance matrix

  # Define means for the three clusters
  #mu_list <- list(c(0,0), c(0,2), c(2,0))

  # Generate categorical assignments Z
  Z <- sample(1:k, size = n, replace = TRUE, prob = prob)
  #print(length(Z))

  #print(I2_list)

  # Generate X given Z
  X <- matrix(0, n, dim)
  for (i in 1:n) {
```

```
    X[i, ] <- rmvnorm(1, mean = mu_list[[Z[i]]], sigma = I2_list[[Z[i]]])
  }
  list(X,Z)
}

eval_EM = function(res, mu_list, dim, I2_list, threshold = 0.1) {
  mu_predict = matrix(unlist(res[["mu"]]), ncol = dim, byrow = TRUE)
  mu_true = matrix(unlist(mu_list), ncol = dim, byrow = TRUE)

  #print(length(mu_predict))
  #print(length(mu_true))

  if (length(mu_predict) != length(mu_true)) {
    message("predicted wrong number of clusters")
    return(-1)
    }

  dist_matrix <- proxy::dist(mu_true, mu_predict,  method = "Euclidean")
  #dist_matrix
  assignment <- clue::solve_LSAP(as.matrix(dist_matrix), maximum = FALSE)
  #assignment
  matched_distances <- dist_matrix[cbind(1:length(assignment), assignment)]
  accuracy_score = sum(matched_distances < threshold) / length(assignment)
  #print(assignment)

  sigma_pred = res[["Sigma"]]
  sigma_true = I2_list

  sigma_error <- function(true_sigma, pred_sigma) {
    sqrt(sum((true_sigma - pred_sigma)^2))  # Frobenius norm
  }

  k = length(sigma_pred)
  errors <- numeric(k)
  for (i in 1:k) {
    errors[i] <- sigma_error(sigma_true[[i]], sigma_pred[[assignment[i]]])
  }

  return(list(mu_error = mean(matched_distances), score = accuracy_score, sigma_error = mean(errors)))
}


plot_ellipse_n_center = function(t_coords, res, mu_list, C_pred){

  plot(x = t_coords[1,], y = t_coords[2,],
       col = adjustcolor(col = "black" ,alpha.f = 0.5), pch = 19)

  # Ellipse
  for (i in 1:C_pred) {
    draw_ellipse(res, i)
  }

  mu_true = matrix(unlist(mu_list), ncol = dim, byrow = TRUE)
```

```
  points(x = mu_true[,1], y = mu_true[,2], col = "blue", pch = 19)

}
```

2d sample with 3 mixtures

correct guess for centers

robust

another 2d sample with 3 mixtures

correct guess for centers

robust

2d sample with 5 mixtures

correct guess for centers

robust

3d sample with 3 mixtures

correct guess for centers

robust

varing centers

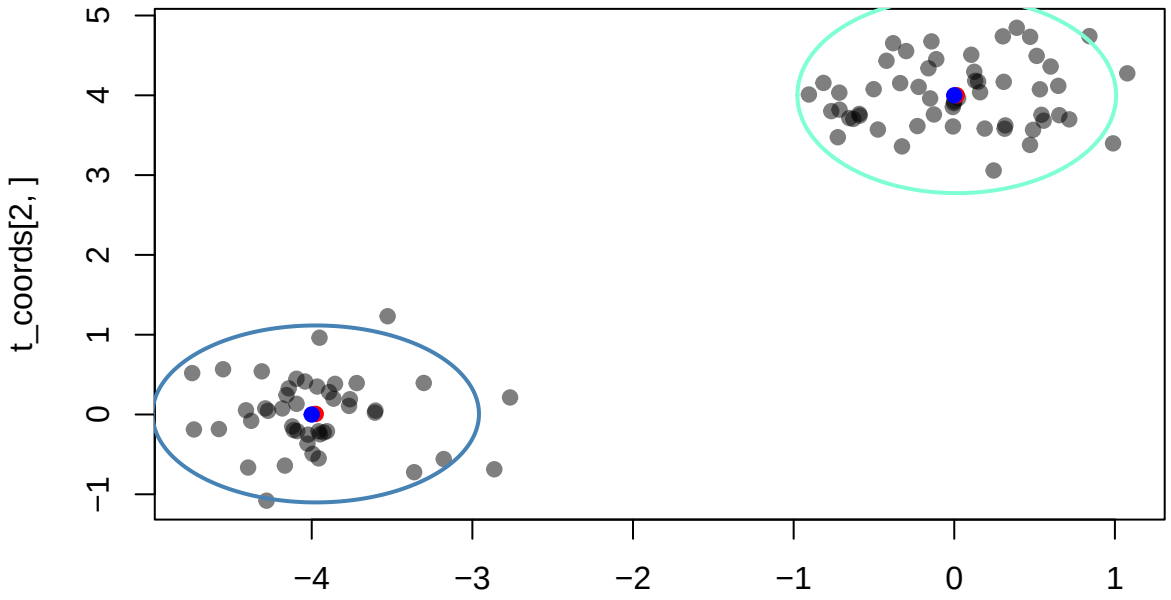
mean mu error is the mean euclidean distance between the true and predicted centers. smaller the better
center accuracy score is the percentage of predicted centers that are with in the 0.1 threshold of true centers (which means they are close), closer to one the better
mean sigma error is the mean euclidean distance between the true and predicted covariance matrix. smaller the better

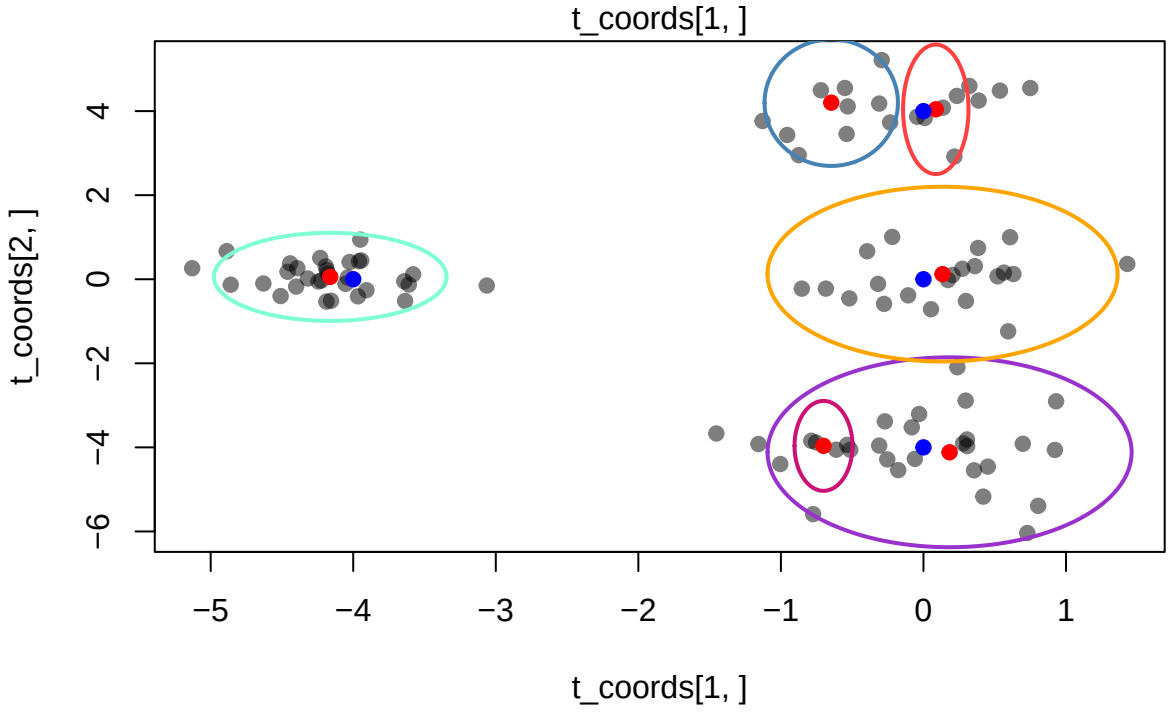
varing sample size

```
set.seed(3)
n_seq = seq(from = 100, to = 700, by = 200)
wrong_cluster_count = numeric(length(n_seq))

for (j in 1:length(n_seq)){

  m = 30 # NUMBER OF TESTS
  n <- n_seq[j]   # Number of data points
  dim = 2

  wrong_ks = 0
  robust_mu_error_list = numeric(m)
  robust_score_list = numeric(m)
  robust_sigma_error_list = numeric(m)

  # original EM
```

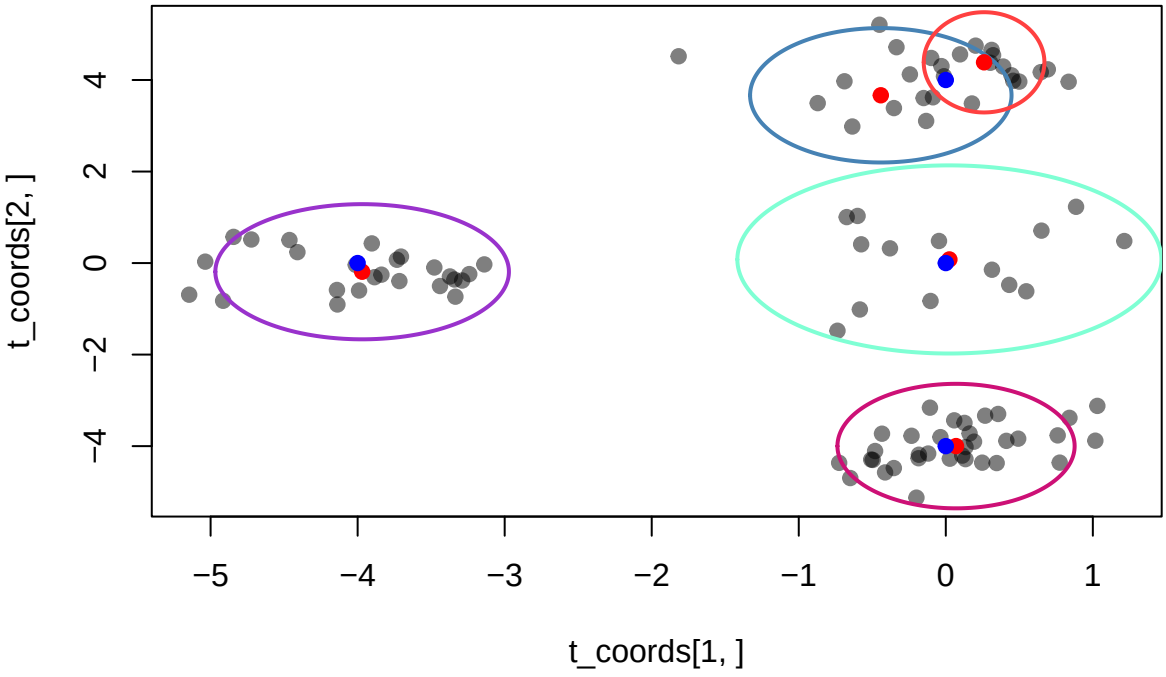
```
#original_mu_error_list = numeric(m)
#original_score_list = numeric(m)
#original_sigma_error_list = numeric(m)

for (i in 1:m) {
  k = sample(2:5, size = 1) # number of clusters
  mu_complete_list = list(c(0,0), c(0,4), c(4,0), c(-4,0), c(0,-4))
  mu_list = sample(mu_complete_list, size = k)
  prob = abs(rnorm(k, mean = 0, sd = 1)) + 1
  I2_variance = abs(rnorm(k, mean = 0, sd = 0.3))+0.1
  I2_list = lapply(I2_variance, function(x) diag(x,dim))

  #XZ = generate_GMM(n)
  XZ = generate_GMM(n, k =k, dim = dim, mu_list = mu_list, prob = prob, I2_list=I2_list)

  coords = XZ[[1]]
  group = XZ[[2]]
  dat = cbind(coords,group)
  colnames(dat) = c("X", "Y", "Group")

  # Plot the generated data
  ggplot(data = dat) +
    geom_point(aes(x = X, y = Y, colour = factor(Group)))
  #plot(X, col = Z, pch = 16, main = "Generated Data")

  C = k #number of centers
  Z <- sample(1:C, n, replace = T)
  t_coords = t(coords)

  #EM_res <- EM(X=t_coords, C=C, Z=Z, tol=1e-10, m_iter=1e5) # always true number of clusters

  #original_results = eval_EM(res = EM_res, mu_list = mu_list, dim = dim, I2_list=I2_list)

  #original_mu_error_list[i] = original_results[["mu_error"]]
  #original_score_list[i] = original_results[["score"]]
  #original_sigma_error_list[i] = original_results[["sigma_error"]]

  ########################

  EMR_res <- EM_Robust(t_coords, n-1)
  #EMR_res

  C_pred = length(EMR_res[["mu"]])
  plot_ellipse_n_center(t_coords = t_coords, res = EMR_res, mu_list = mu_list, C_pred = C_pred)

  robust_results = eval_EM(res = EMR_res, mu_list = mu_list, dim = dim, I2_list=I2_list)

  if (robust_results[1] != -1) {
    robust_mu_error_list[i] = robust_results[["mu_error"]]
    robust_score_list[i] = robust_results[["score"]]
    robust_sigma_error_list[i] = robust_results[["sigma_error"]]

  } else {
```

```
      wrong_ks = wrong_ks + 1
      #robust_mu_error_list[i] = -1
      #robust_score_list[i] = -1
      #robust_sigma_error_list[i] = -1
    }

  }


  wrong_cluster_count[j] = wrong_ks
}
```

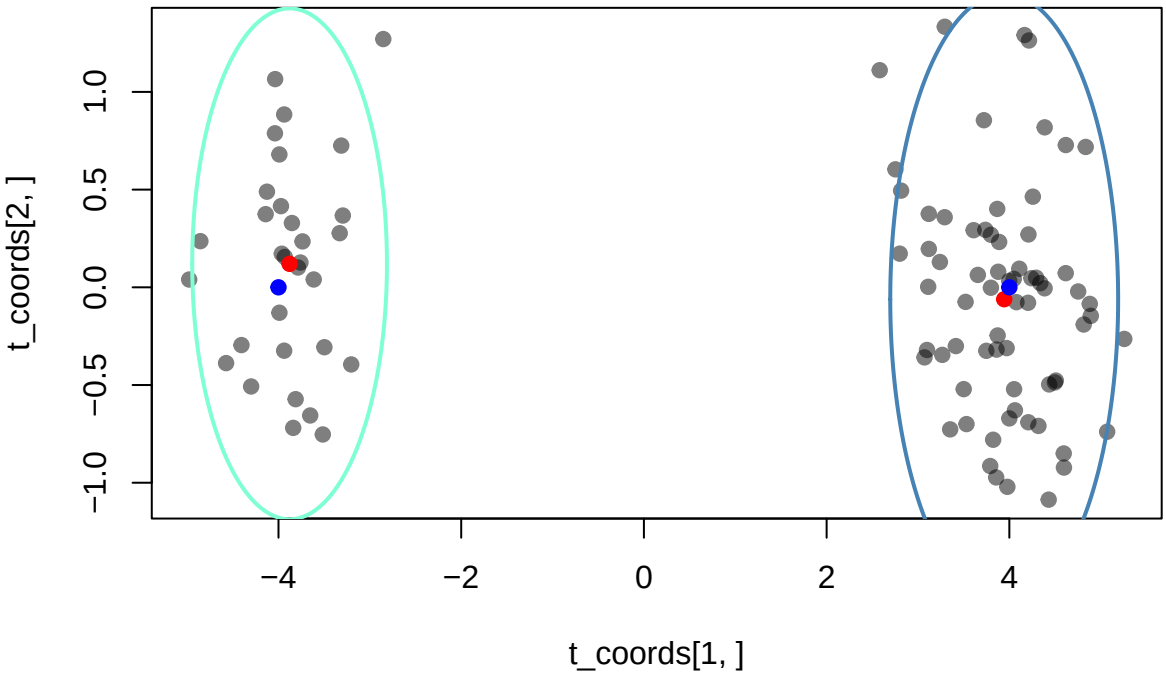


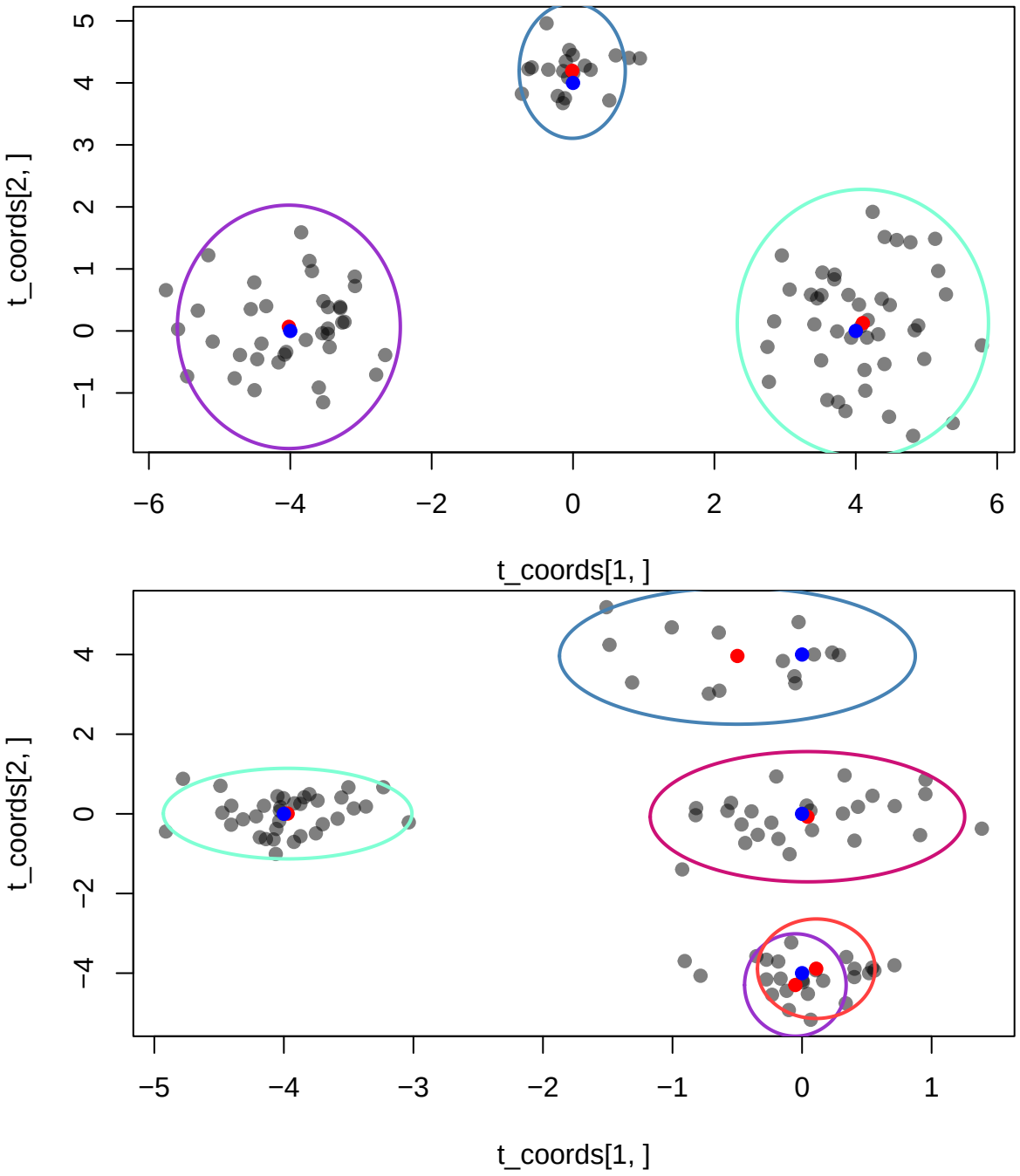


```
## predicted wrong number of clusters
```

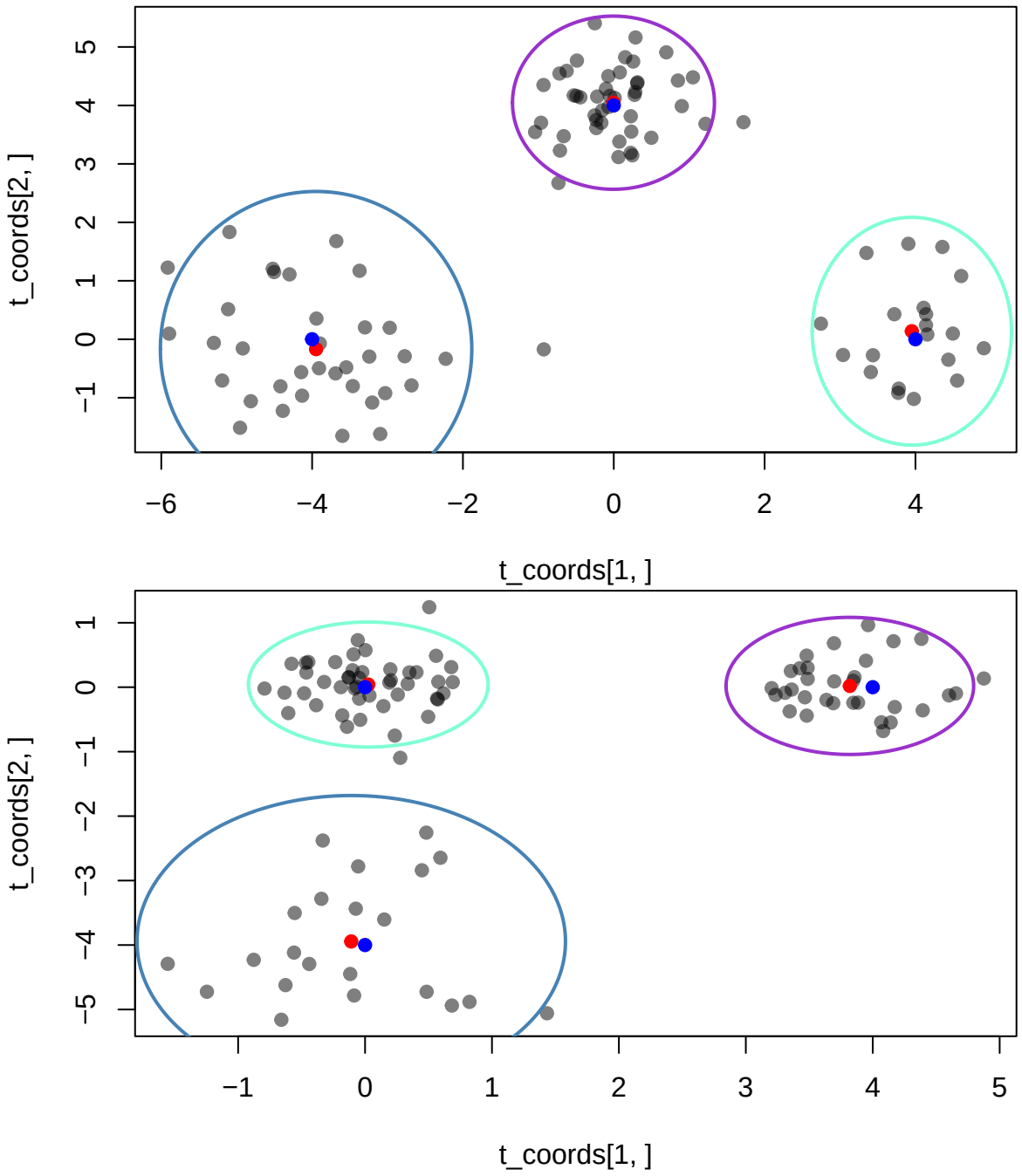


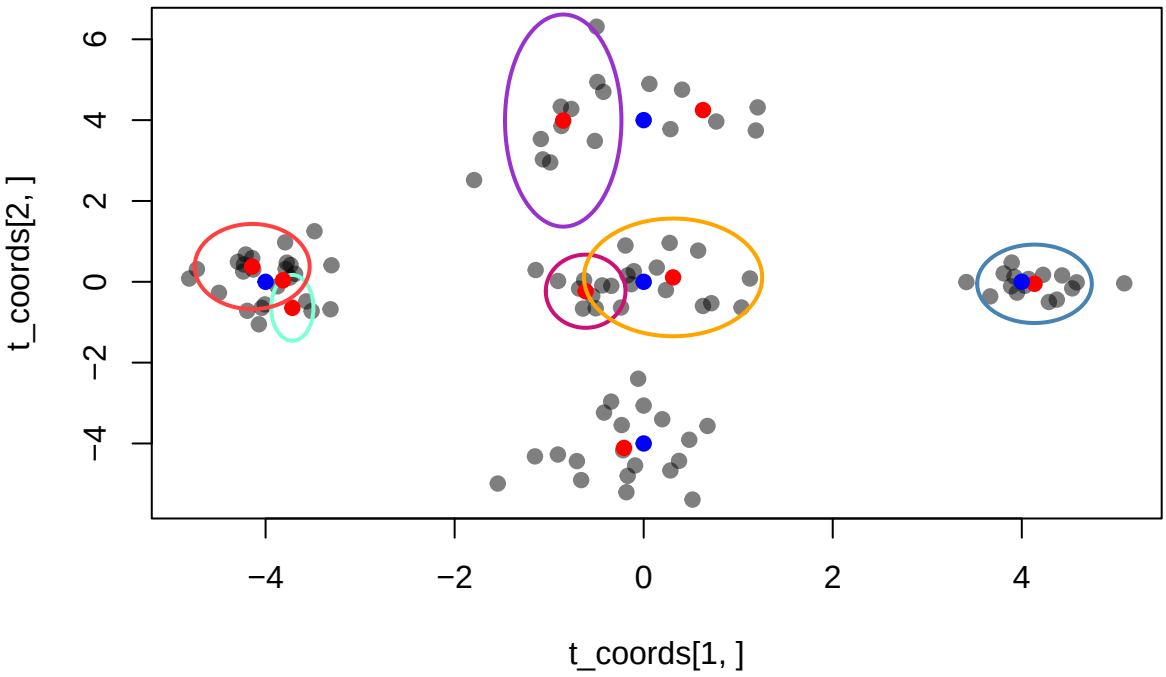

```
## predicted wrong number of clusters
```



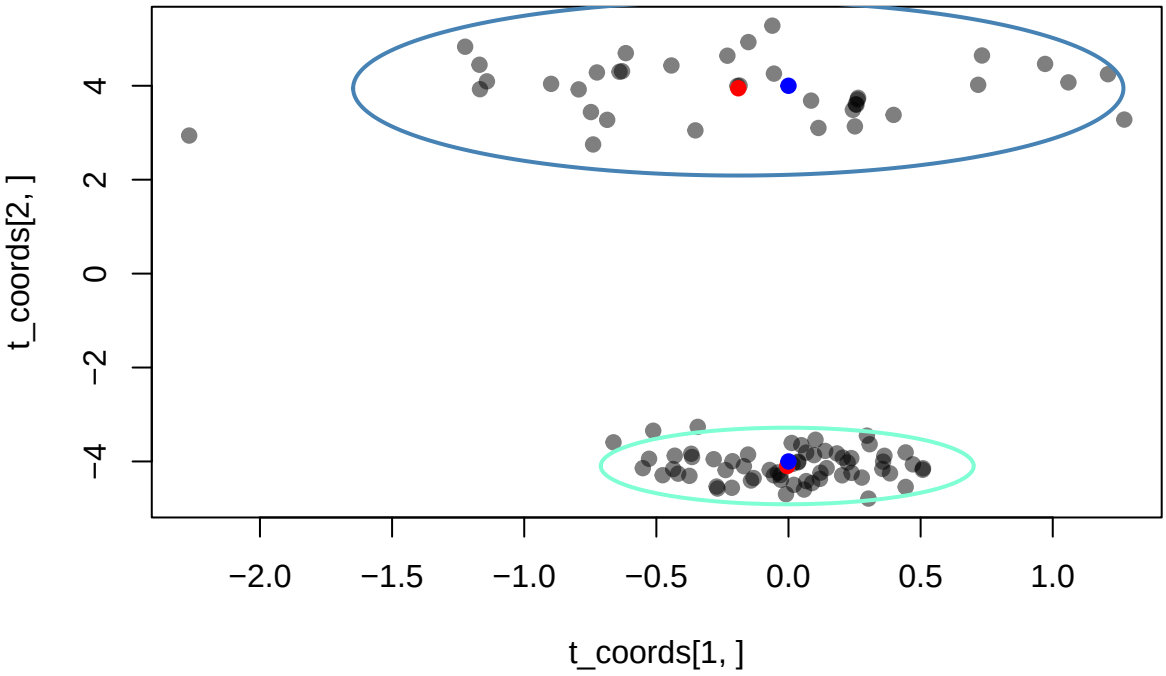


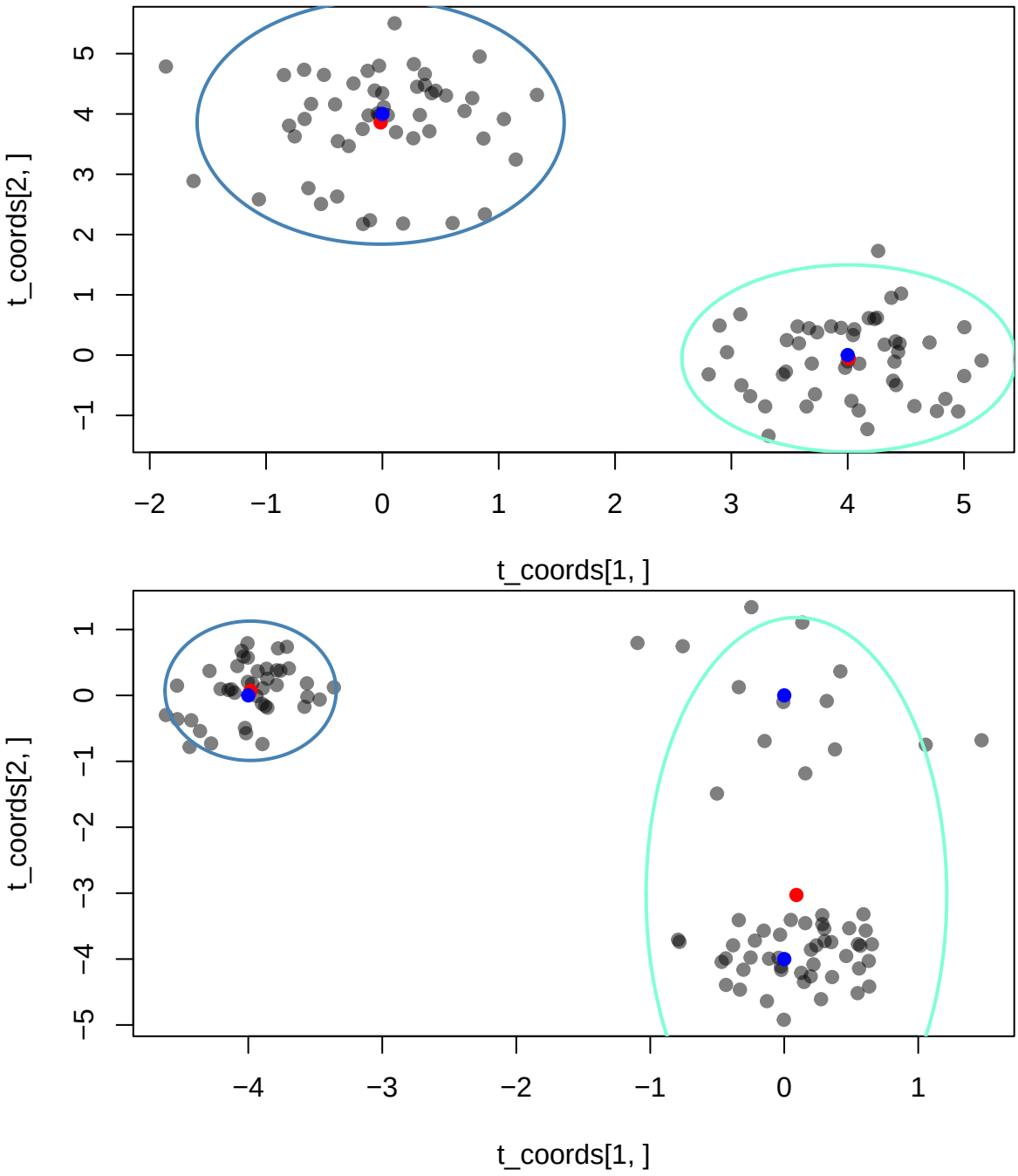

```
## predicted wrong number of clusters
```


t_coords[2,]
5
4
3
2
1
0
−1
−6
−4
−2
0
2
4
t_coords[1,]
t_coords[2,]
1
0
−1
−2
−3
−4
−5
−1
0
1
2
3
4
5
t_coords[1,]

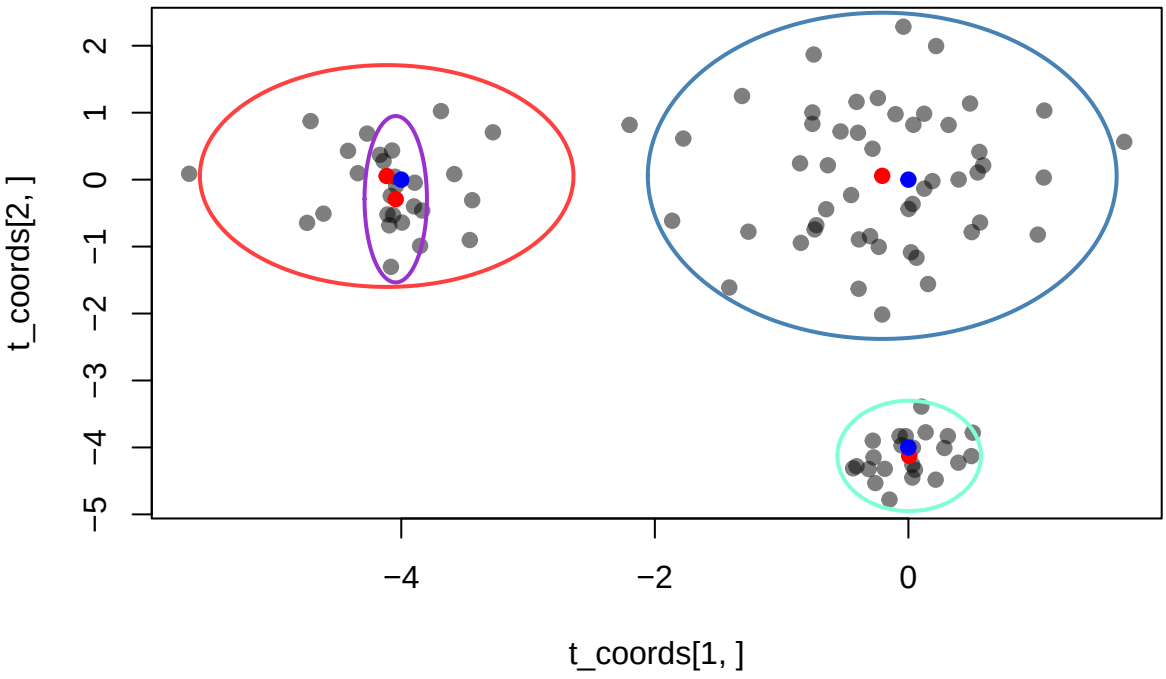



```
## predicted wrong number of clusters
```

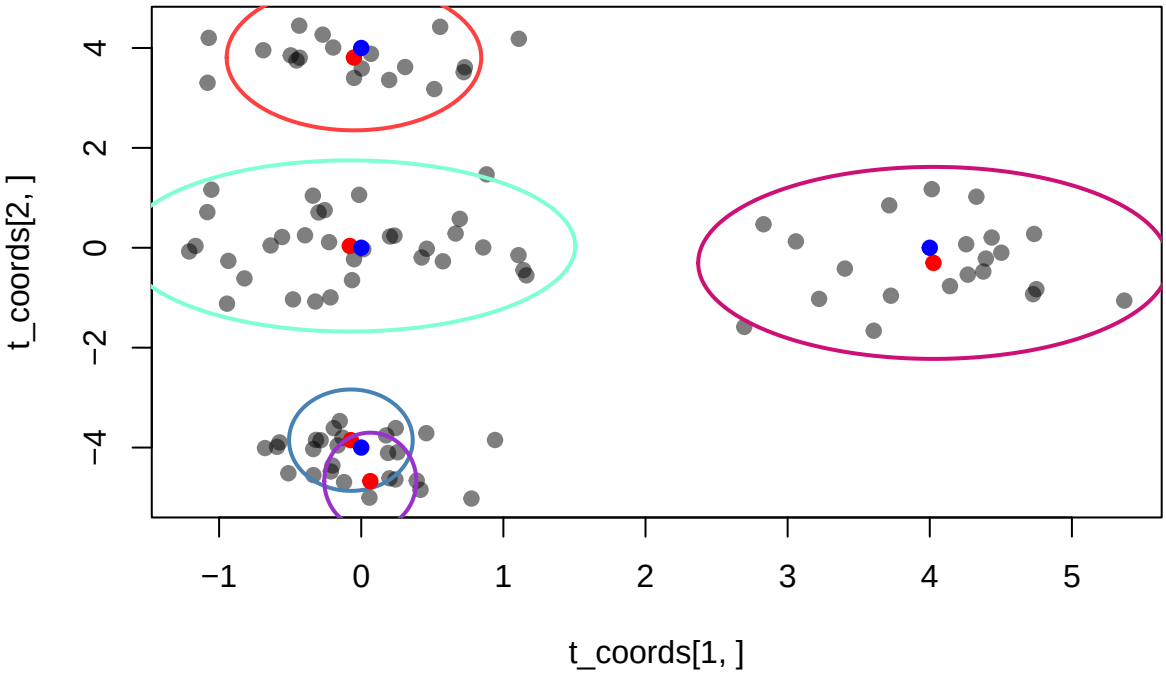




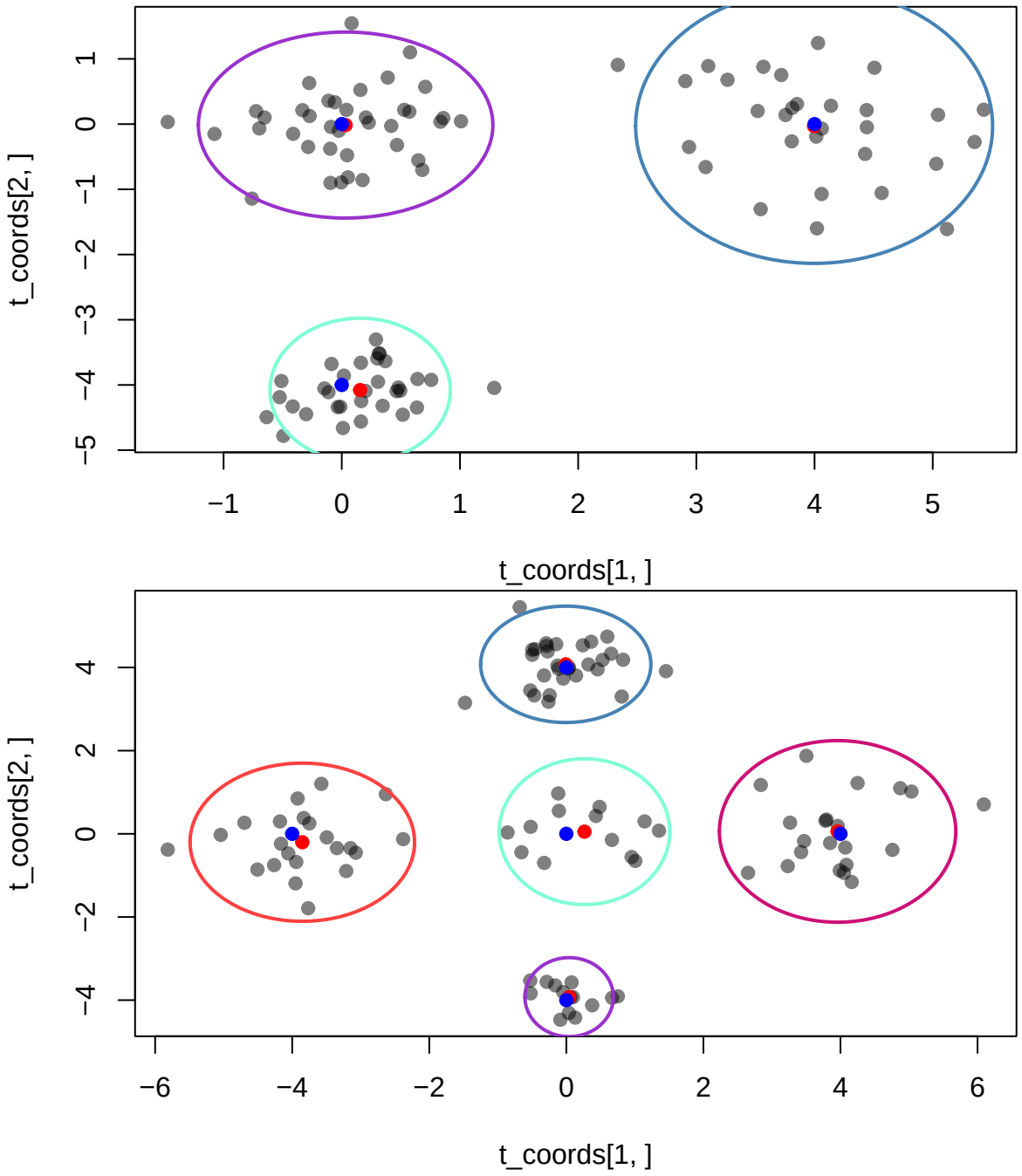

```
## predicted wrong number of clusters
```



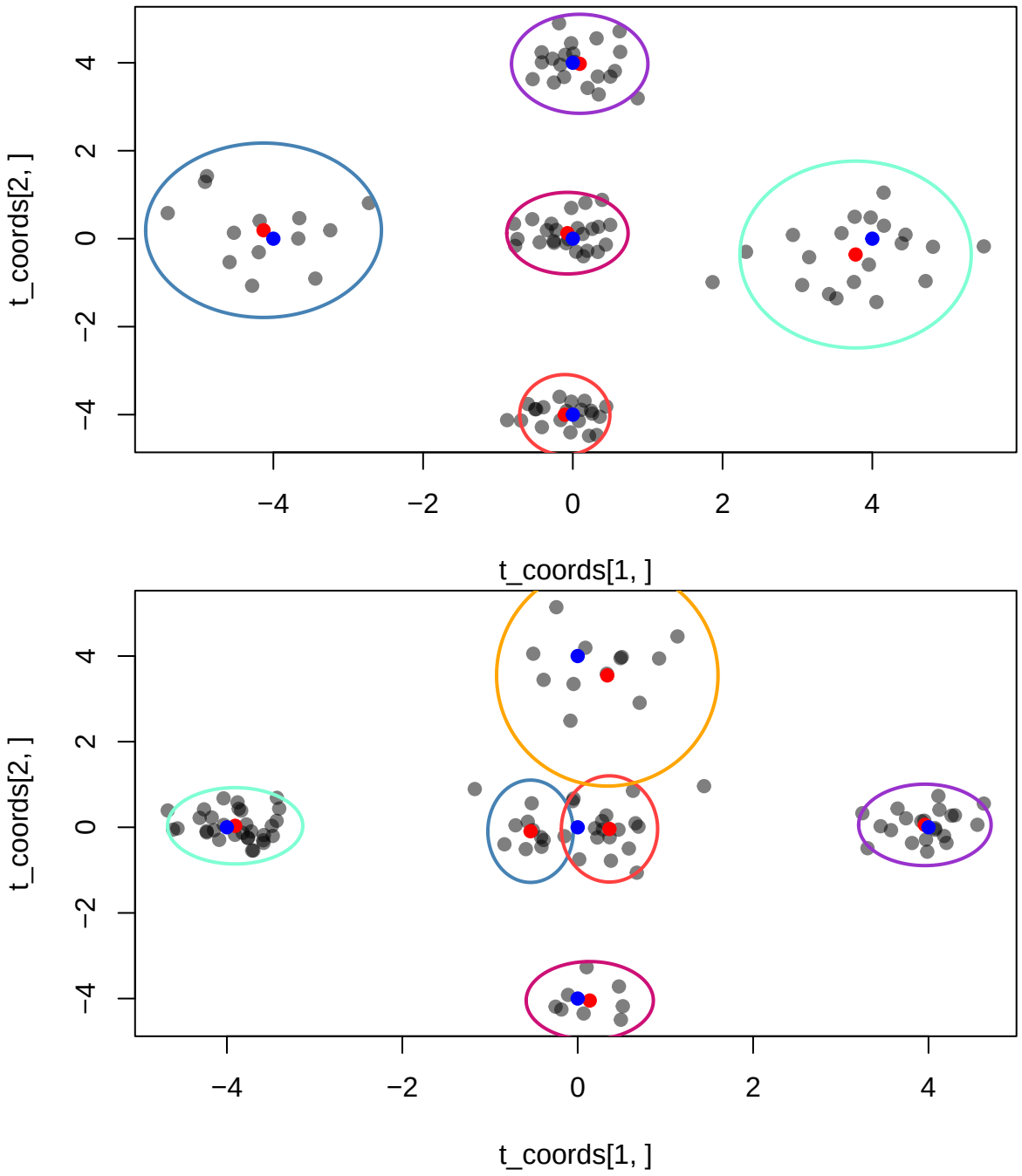

```
## predicted wrong number of clusters
```



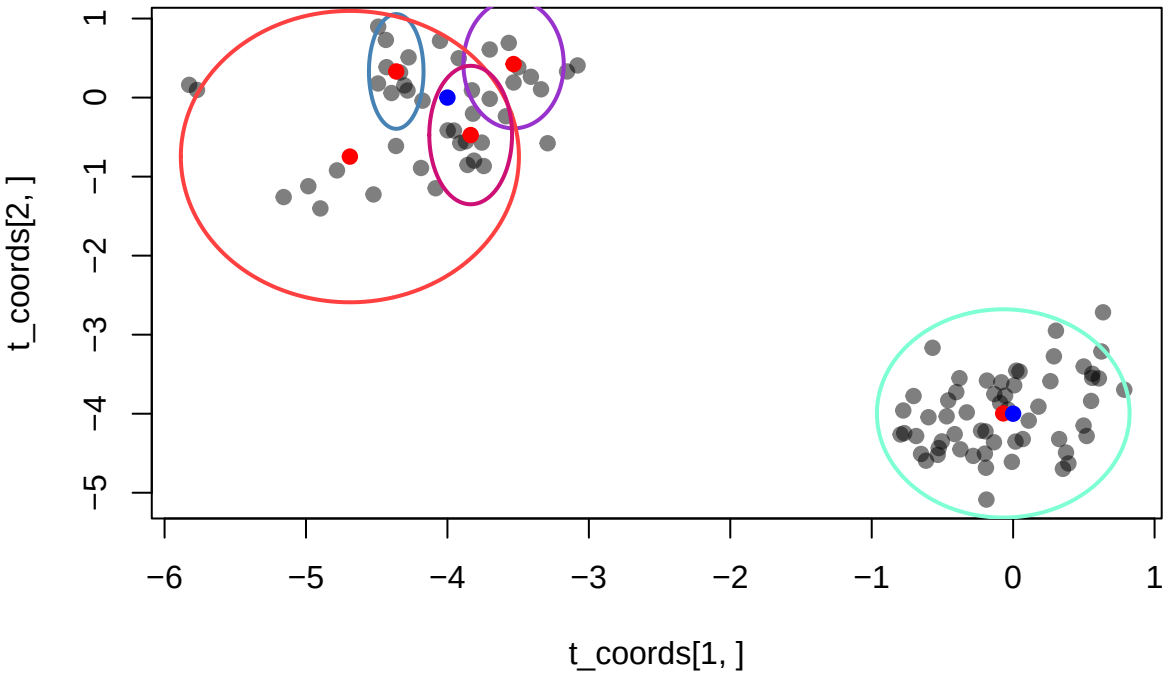

```
## predicted wrong number of clusters
```


t_coords[2,]
t_coords[1,]
−1
0
1
2
3
4
5
−5
−4
−3
−2
−1
0
1

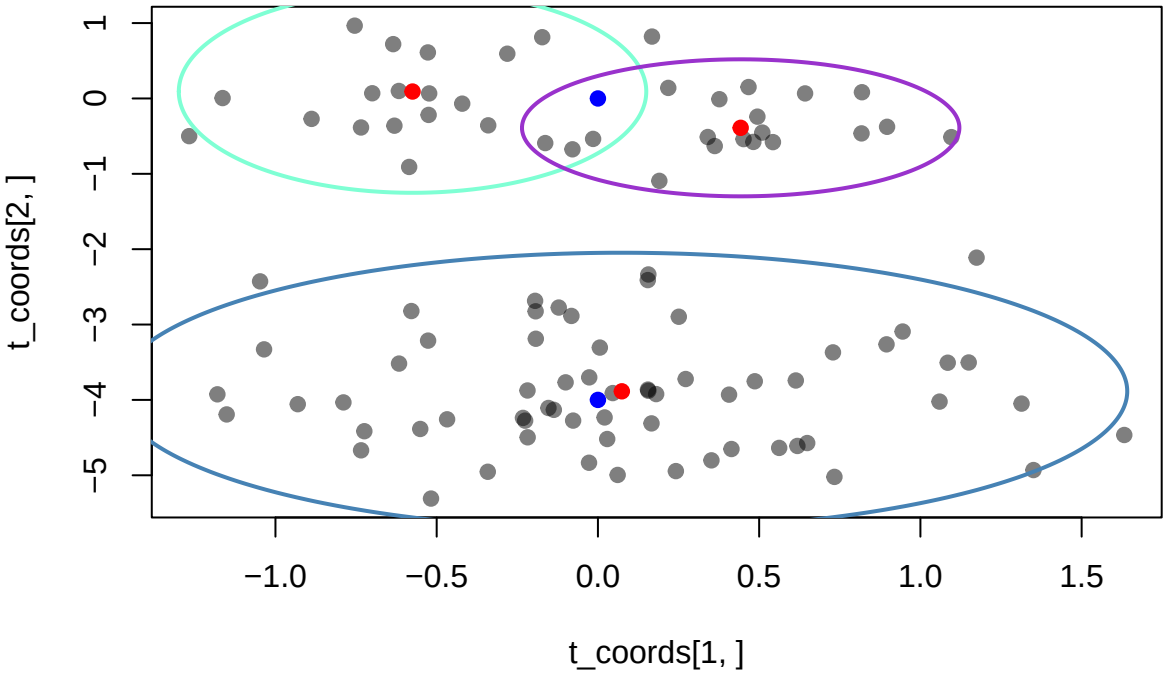
t_coords[2,]
t_coords[1,]
−6
−4
−2
0
2
4
6
−4
−2
0
2
4



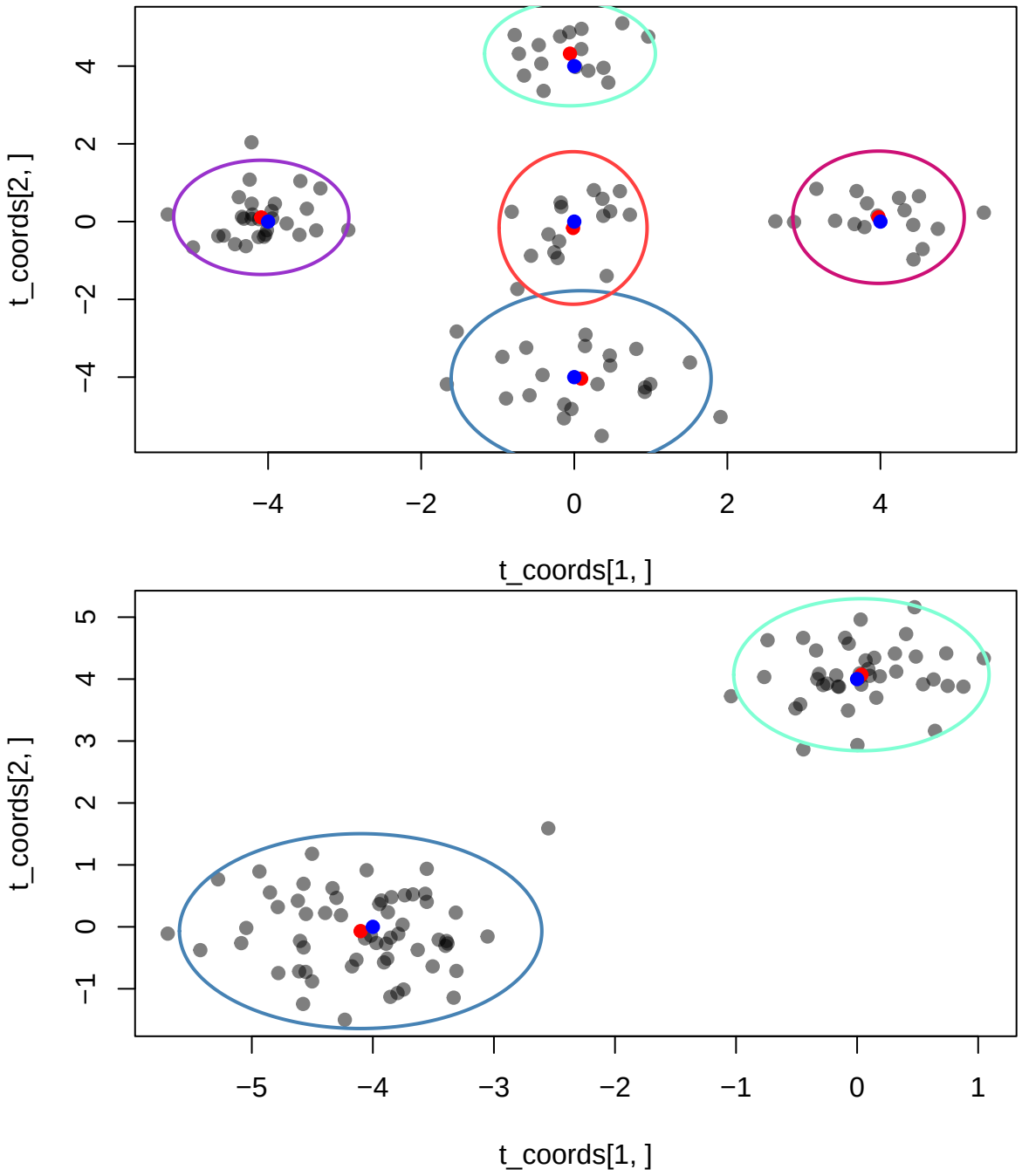

```
## predicted wrong number of clusters
```

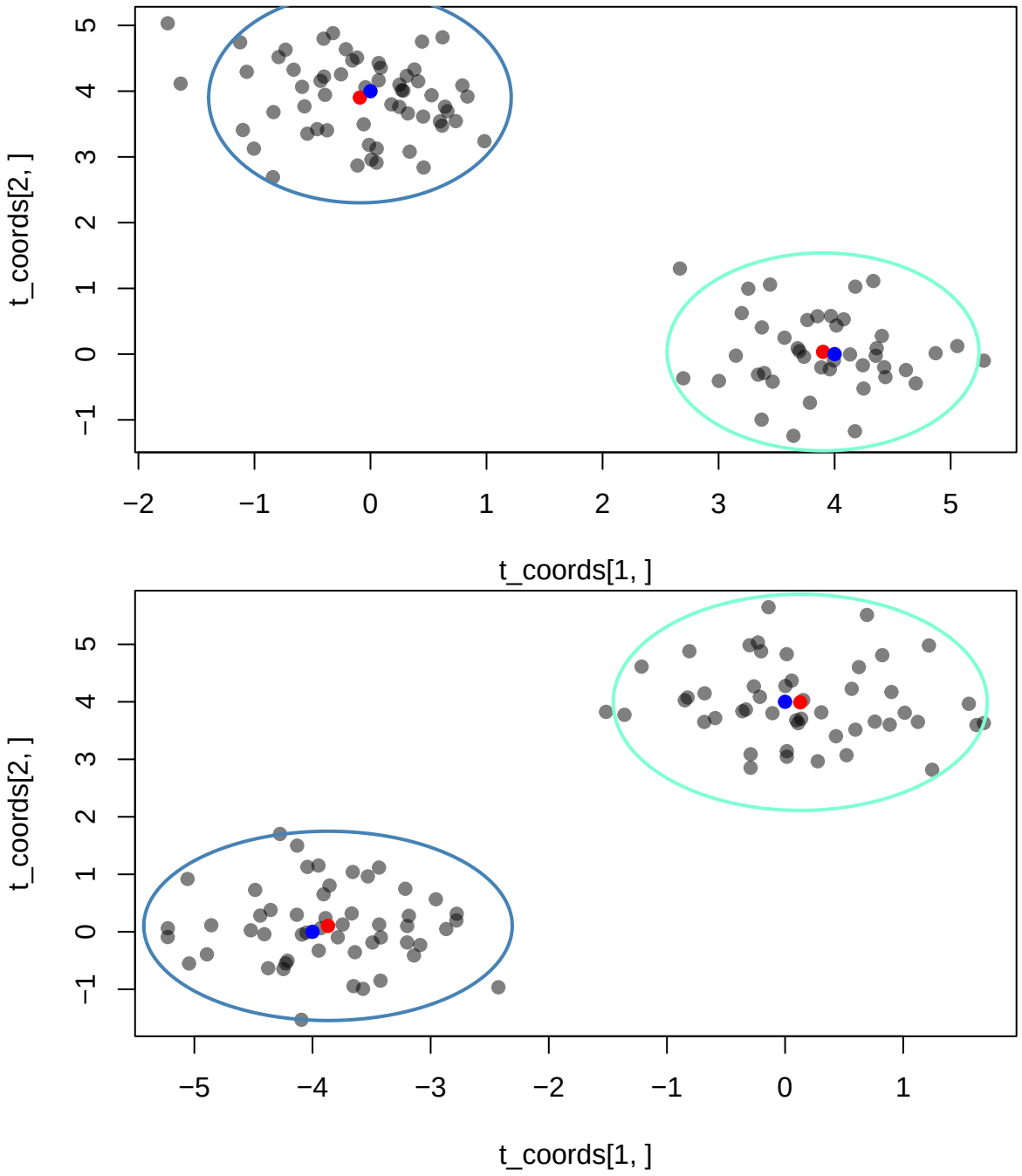



```
## predicted wrong number of clusters
```

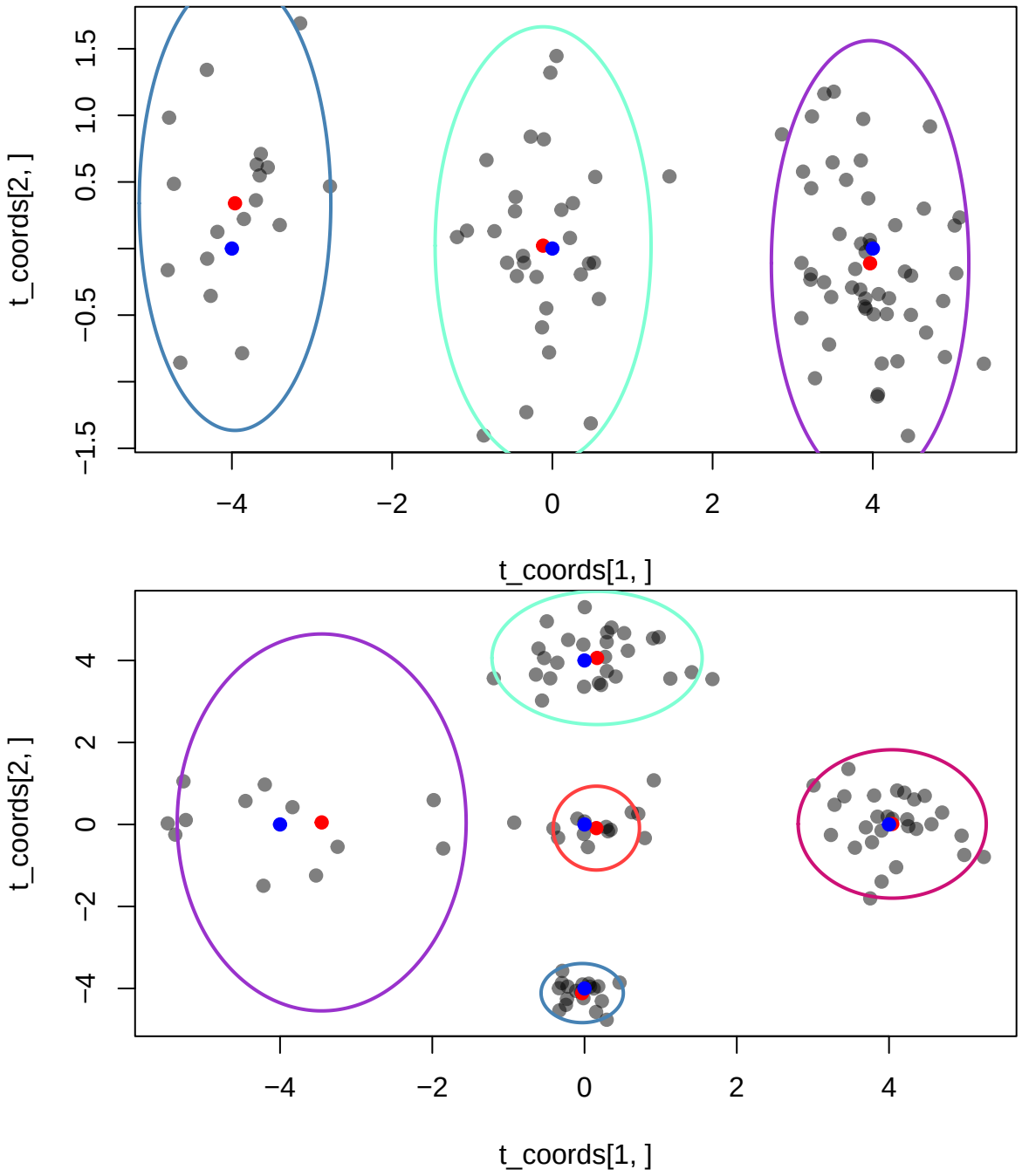


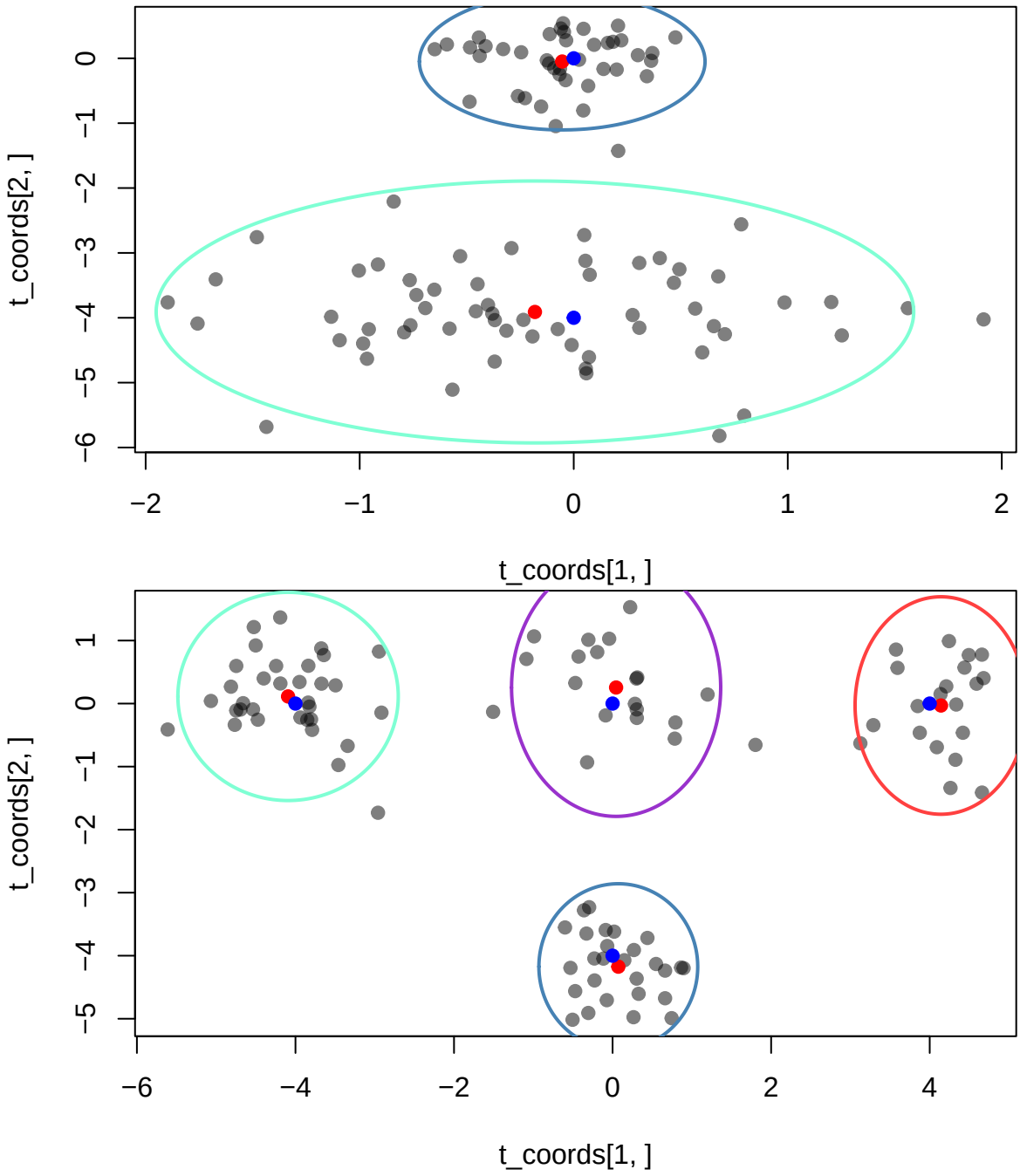

```
## predicted wrong number of clusters
```

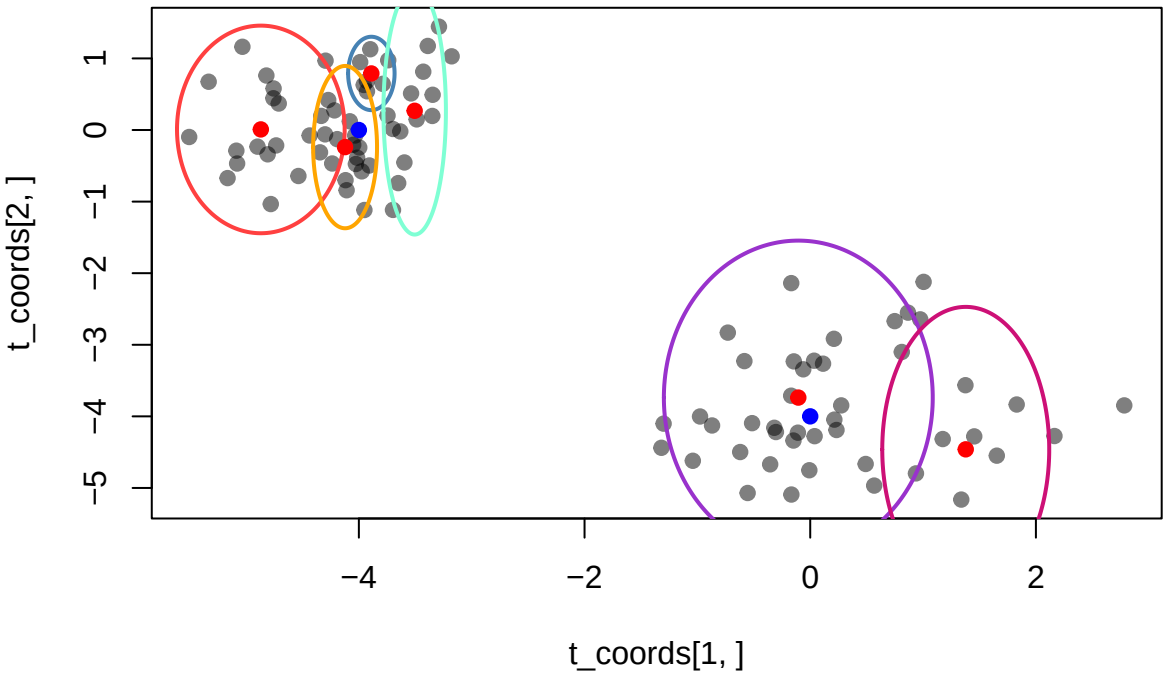

t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
t_coords[1,]
t_coords[2,]
5
4
3
2
1
0
−1
−5
−4
−3
−2
−1
0
1
t_coords[1,]


t_coords[2,]
t_coords[1,]
−2
−1
0
1
2
3
4
5
−1
0
1
2
3
4
5

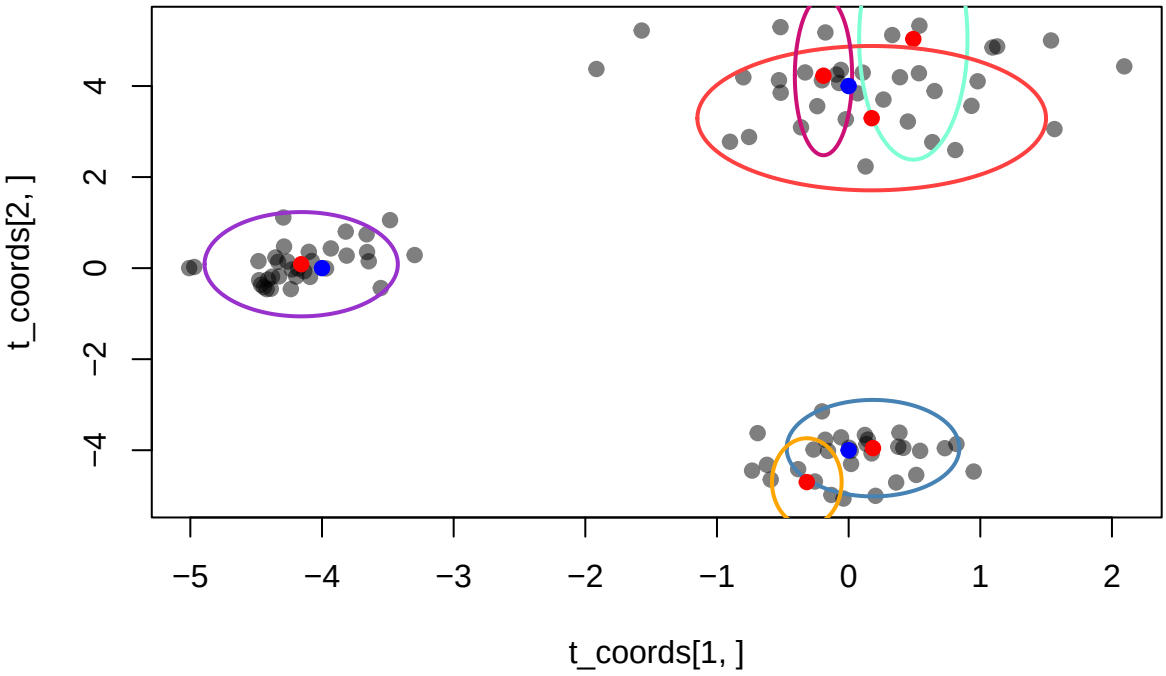
t_coords[2,]
t_coords[1,]
−5
−4
−3
−2
−1
0
1
−1
0
1
2
3
4
5


t_coords[2,]
1.5
1.0
0.5
−0.5
−1.5
−4
−2
0
2
4
t_coords[1,]
t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
t_coords[1,]

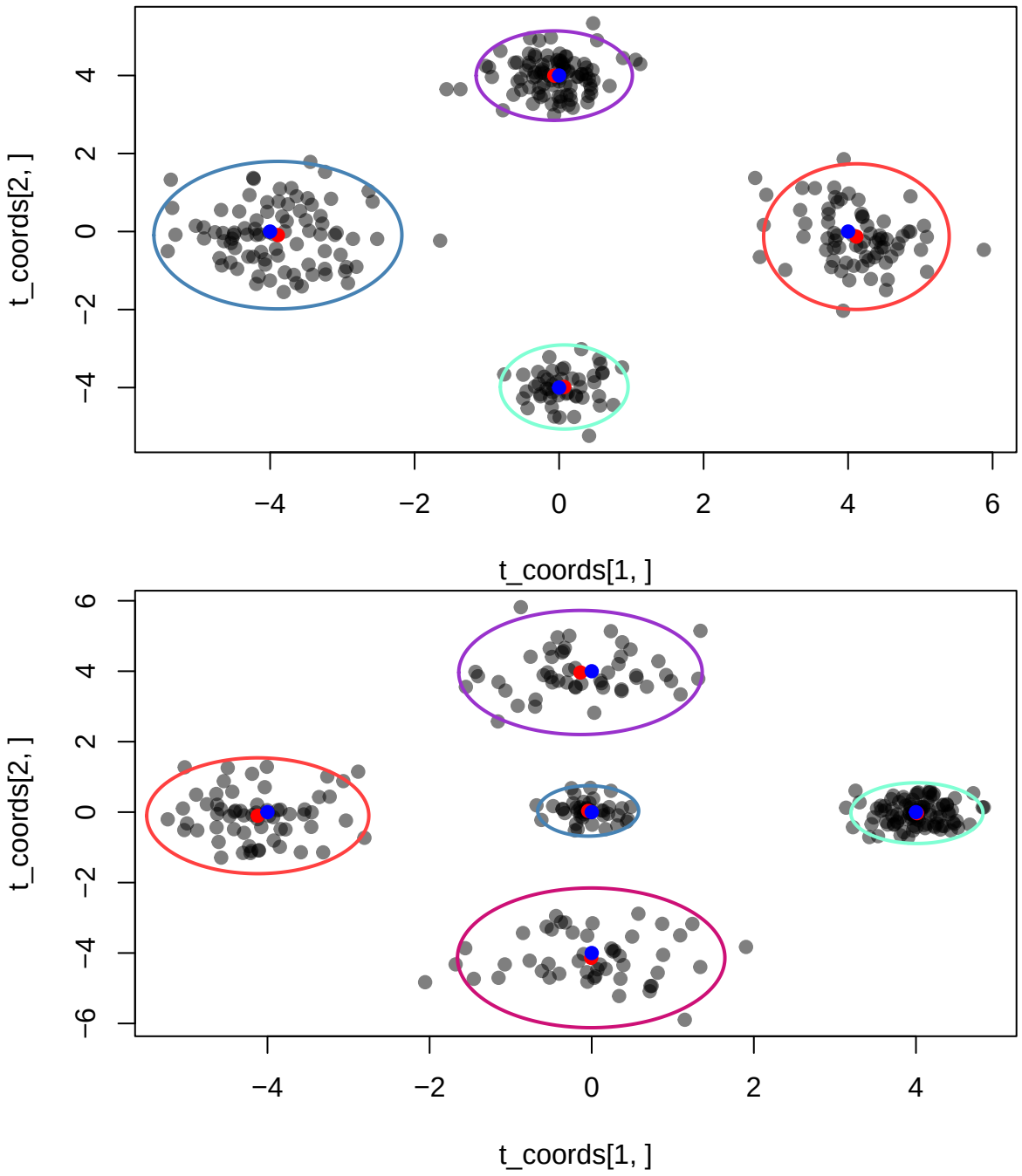

t_coords[2,]
t_coords[1,]
t_coords[2,]
t_coords[1,]

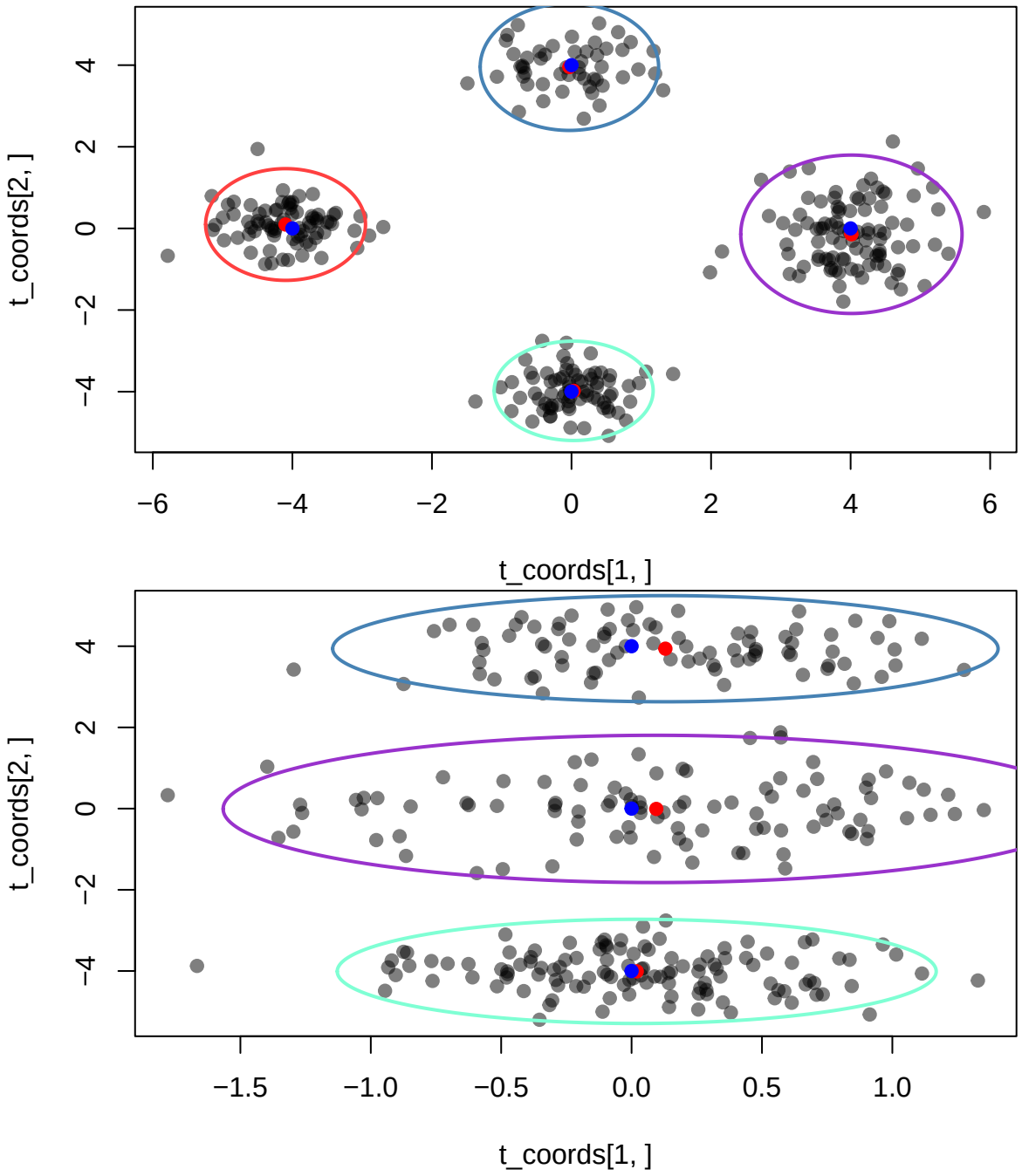


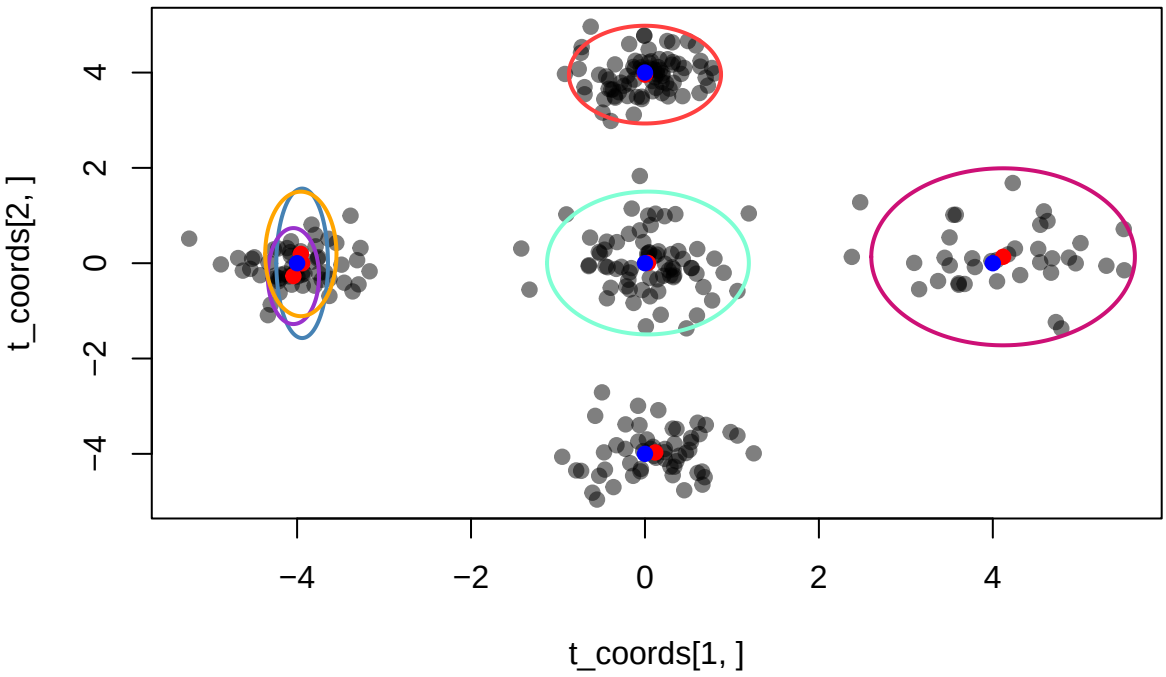

```
## predicted wrong number of clusters
```



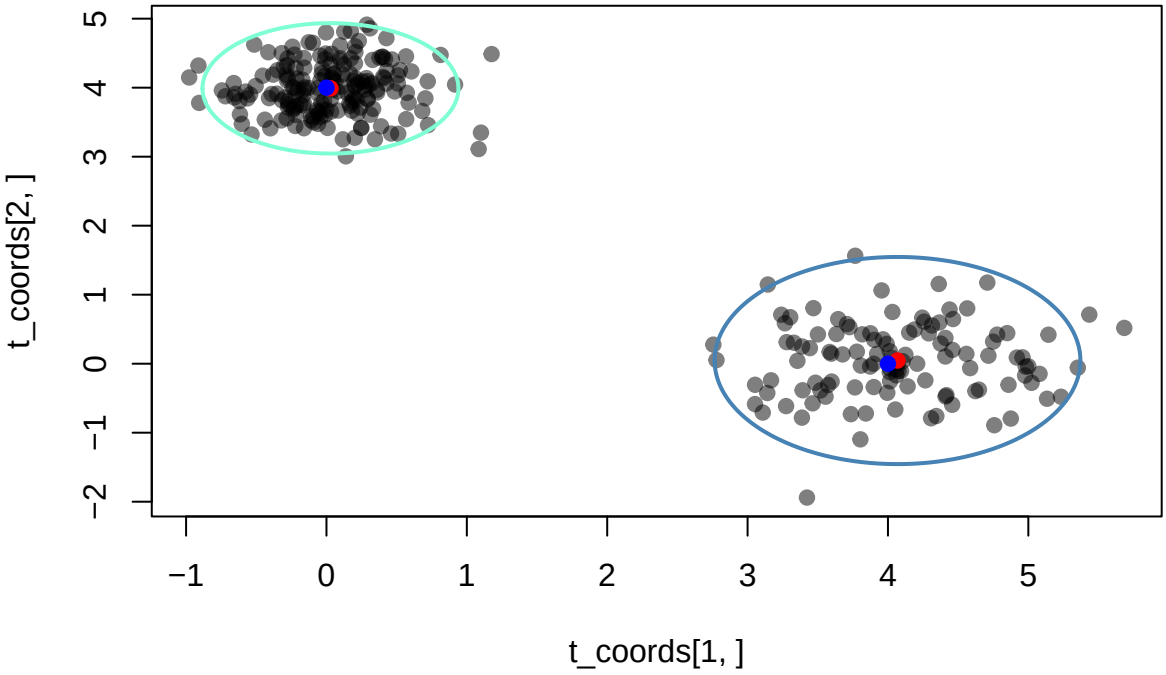

```
## predicted wrong number of clusters
```

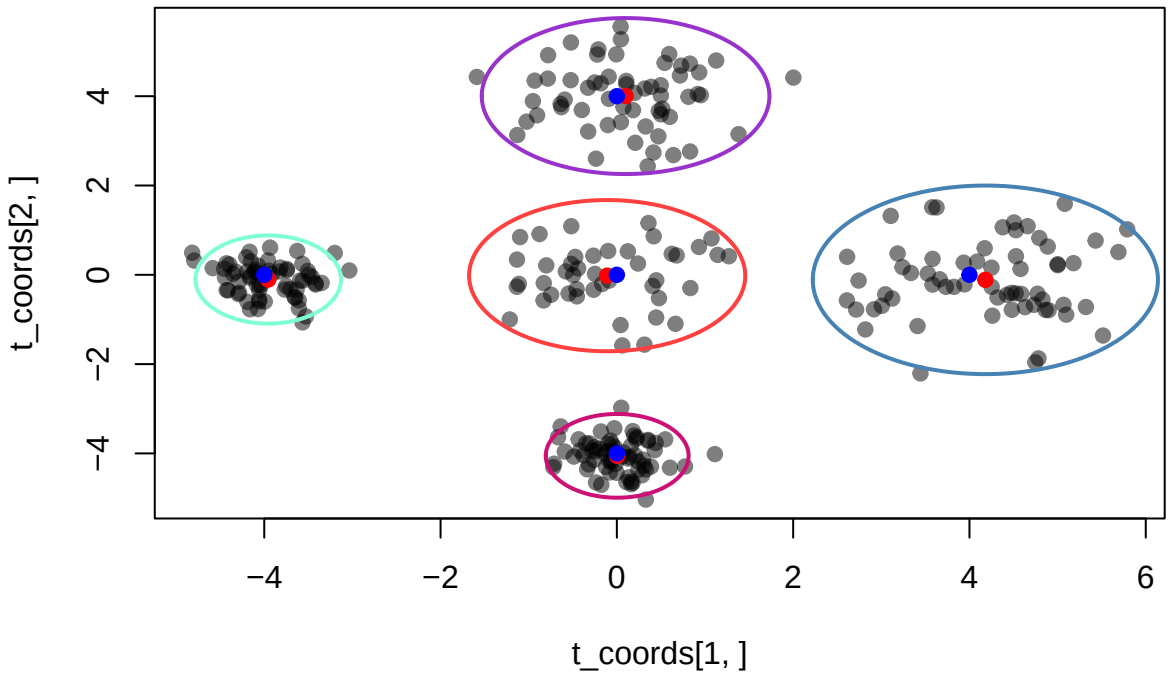

t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
6
t_coords[1,]
t_coords[2,]
6
4
2
0
−2
−4
−6
−4
−2
0
2
4
t_coords[1,]

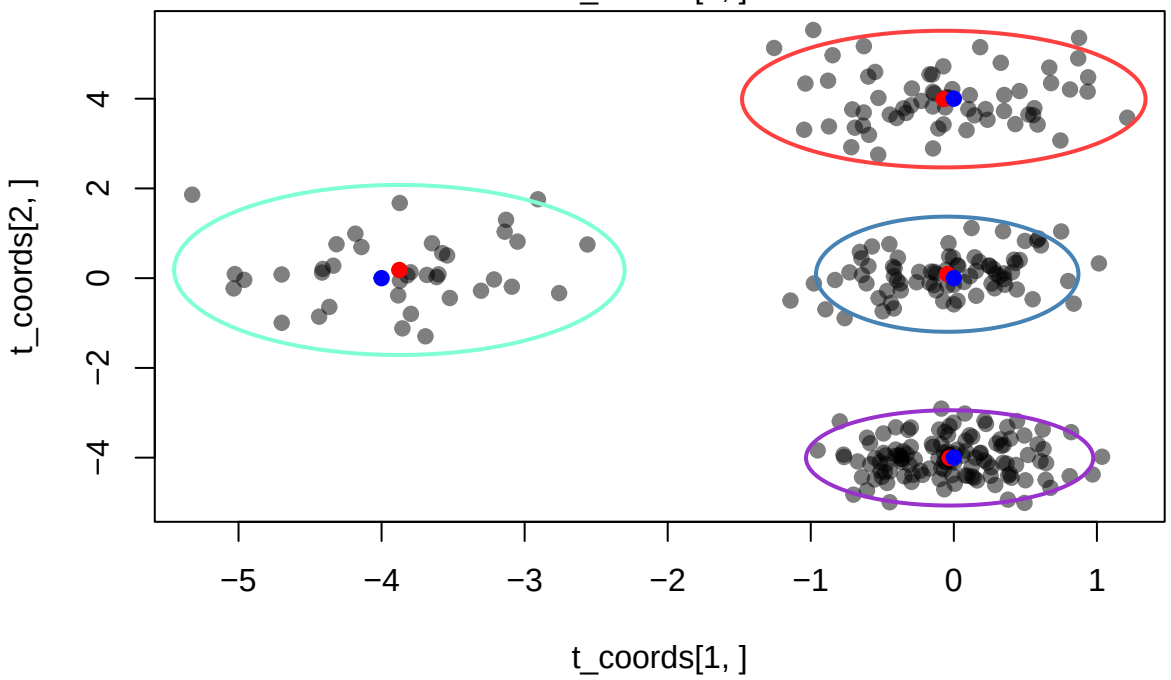

t_coords[2,]
4
2
0
−2
−4
−6
−4
−2
0
2
4
6
t_coords[1,]
t_coords[2,]
4
2
0
−2
−4
−1.5
−1.0
−0.5
0.0
0.5
1.0
t_coords[1,]

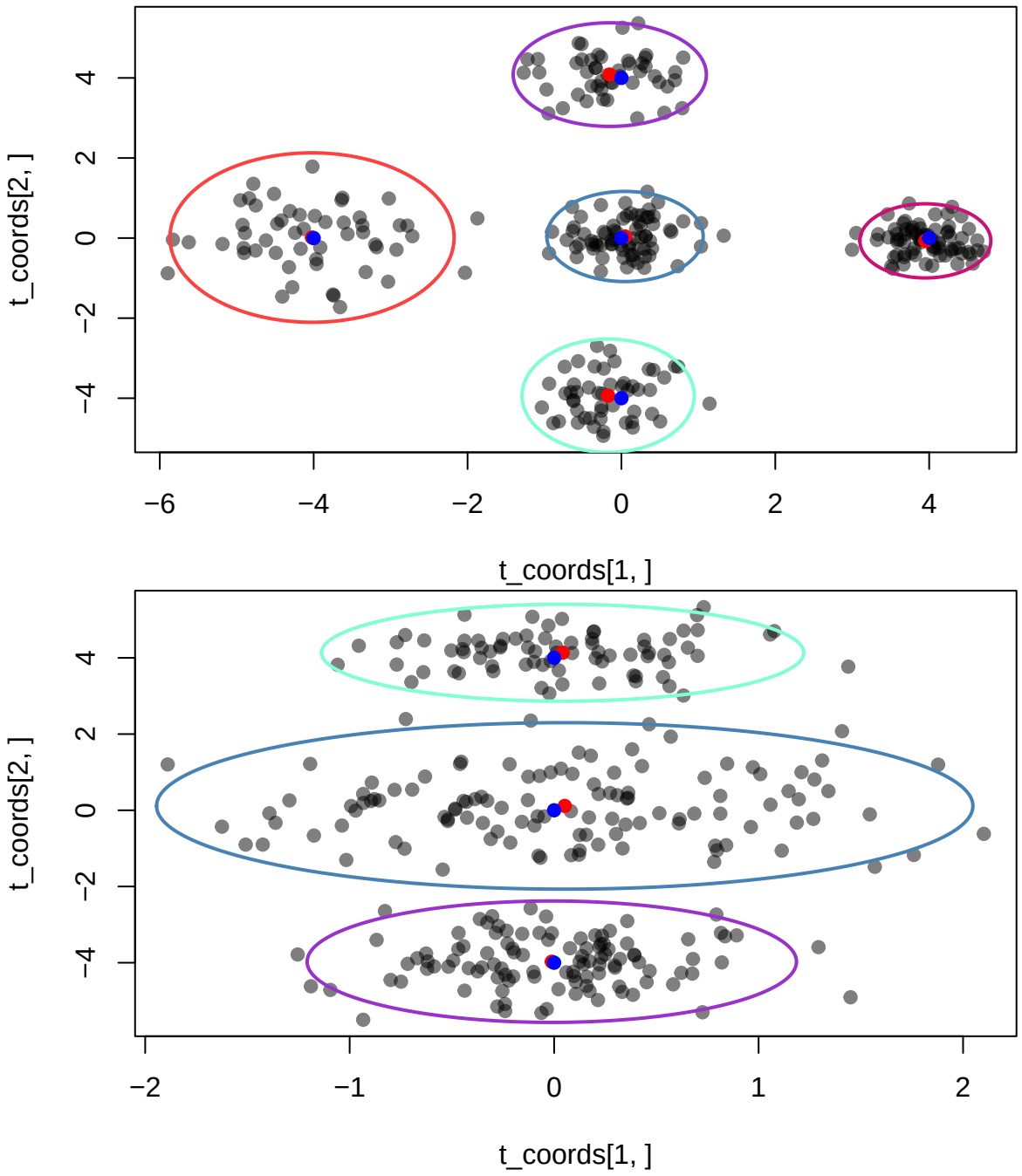


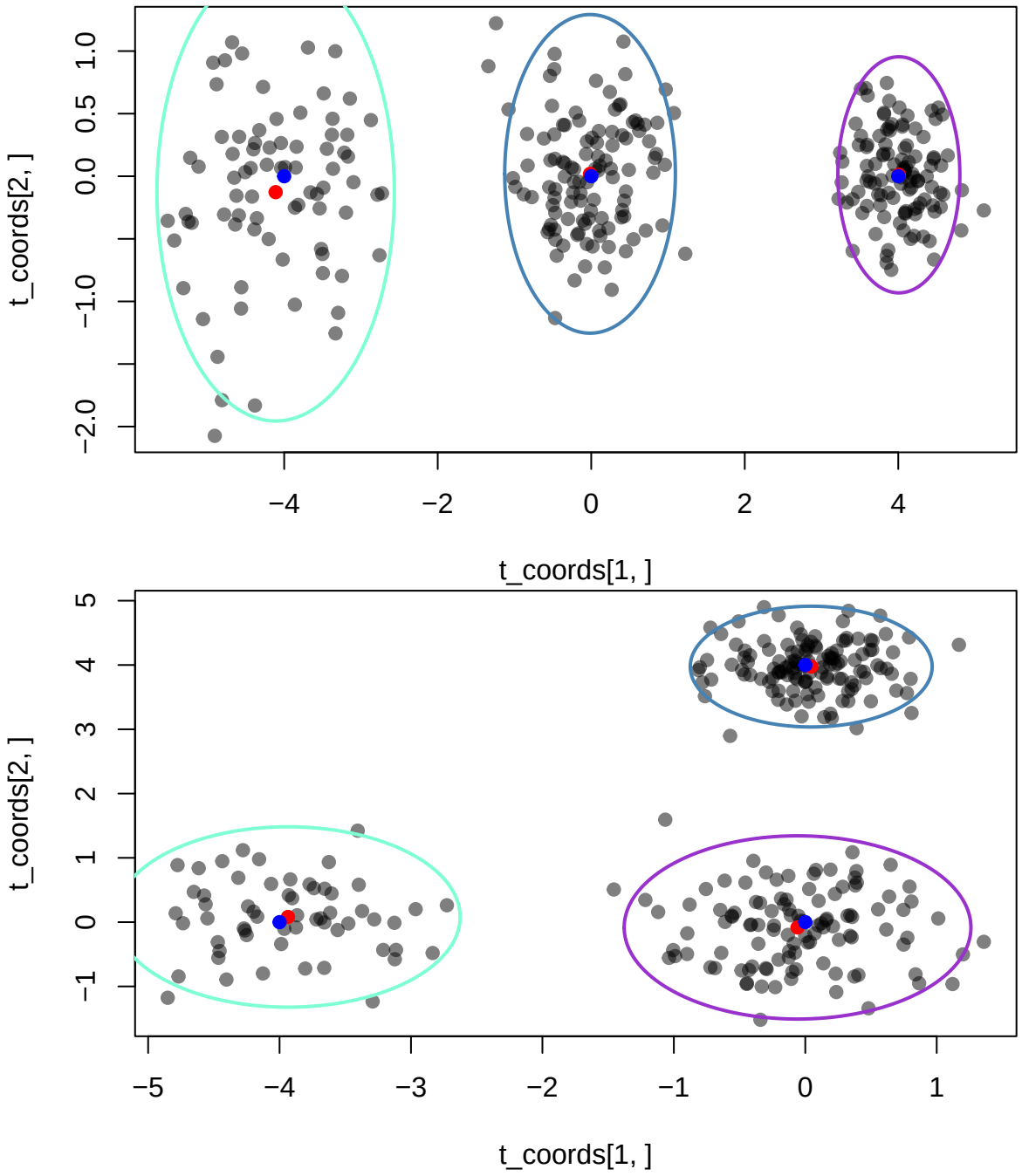

```
## predicted wrong number of clusters
```

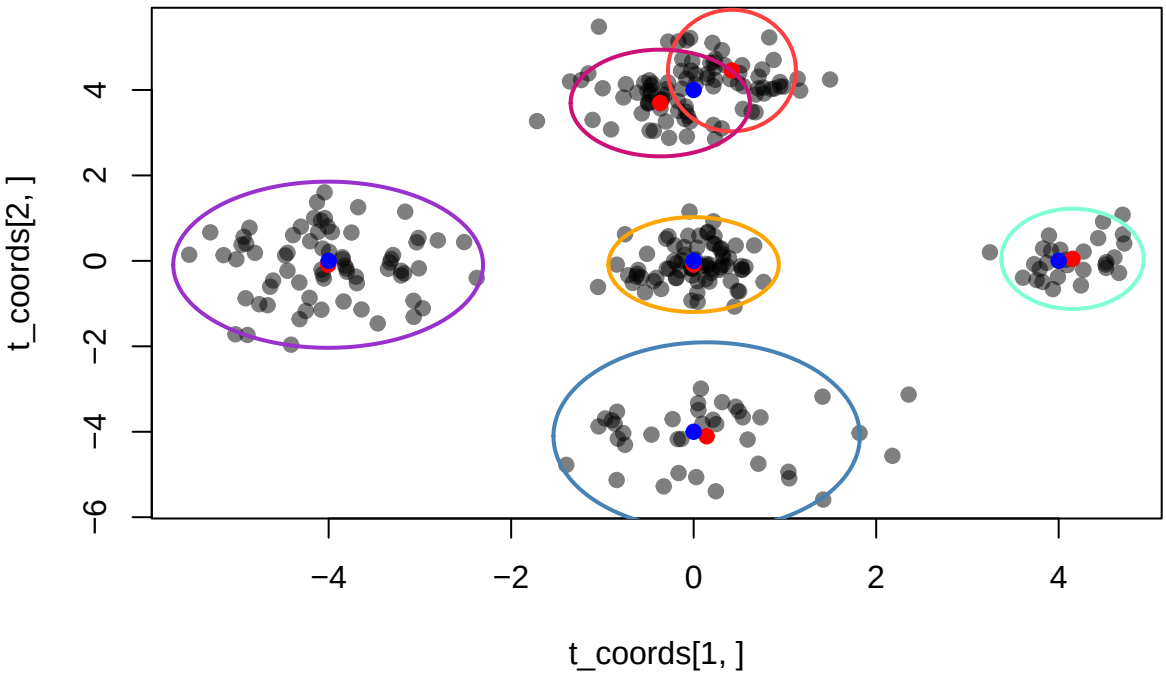



t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
6
t_coords[1,]

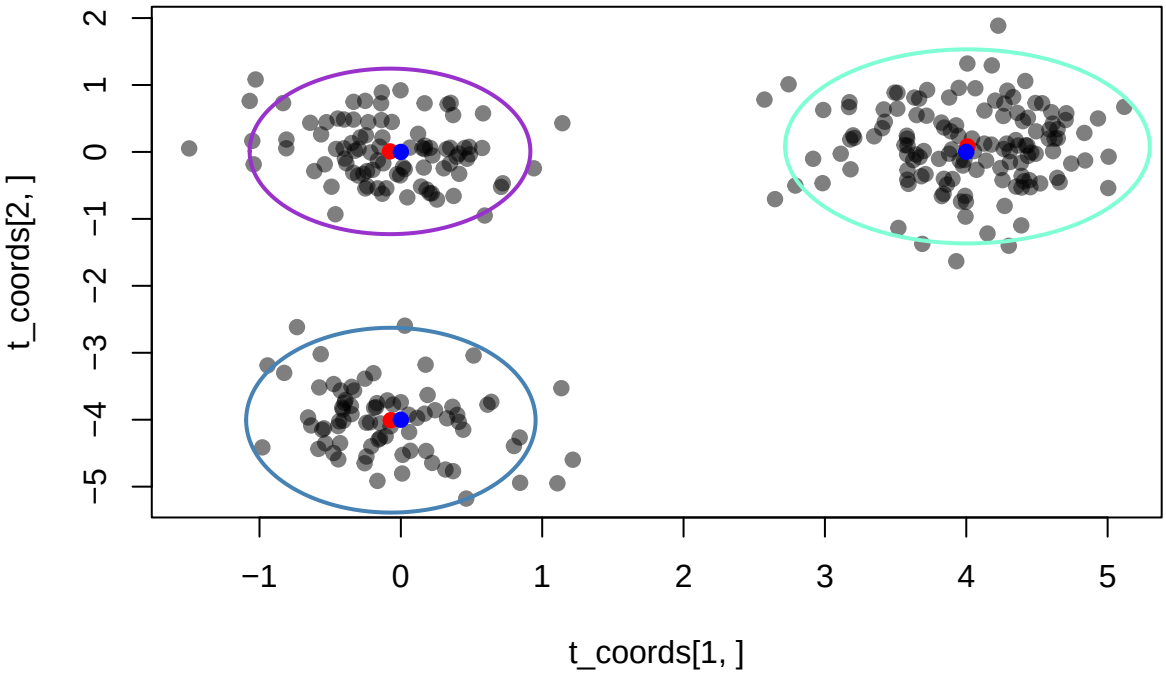

t_coords[2,]
4
2
0
−2
−4
−5
−4
−3
−2
−1
0
1
t_coords[1,]

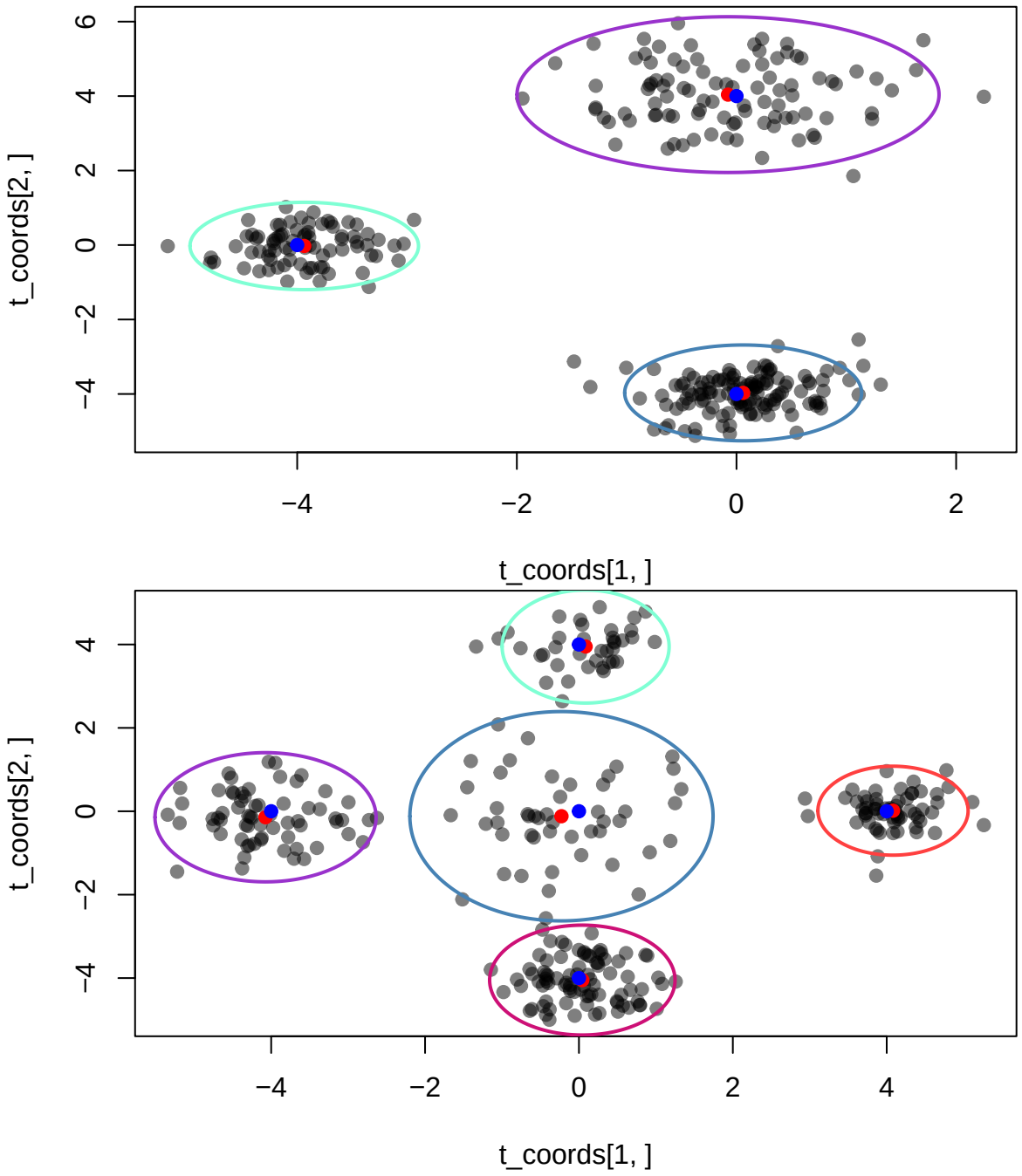

t_coords[2,]
4
2
0
−2
−4
−6
−4
−2
0
2
4
t_coords[1,]
t_coords[2,]
4
2
0
−2
−4
−2
−1
0
1
2
t_coords[1,]

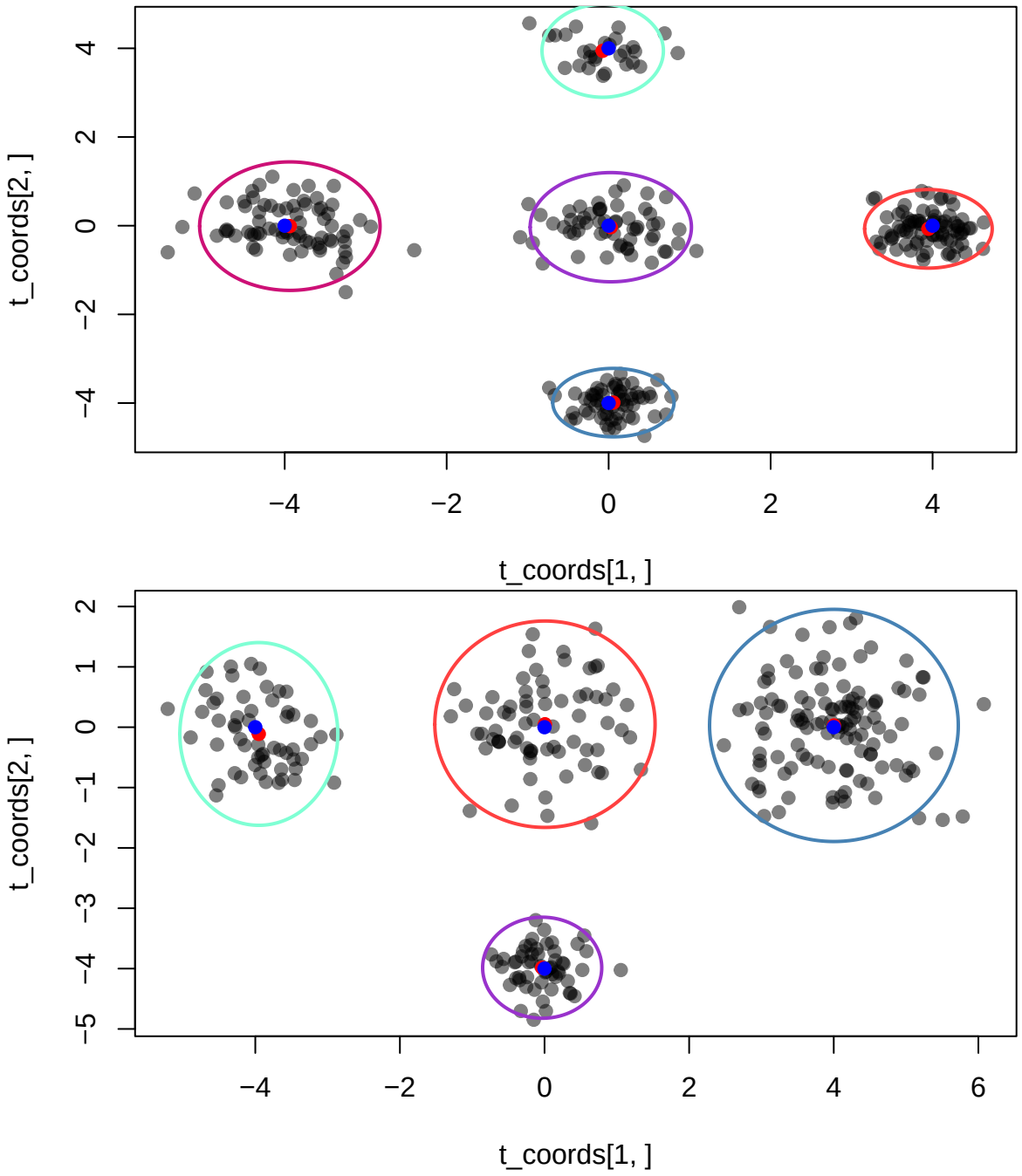

t_coords[2,]
1.0
0.5
0.0
−1.0
−2.0
−4
−2
0
2
4
t_coords[1,]
t_coords[2,]
5
4
3
2
1
0
−1
−5
−4
−3
−2
−1
0
1
t_coords[1,]

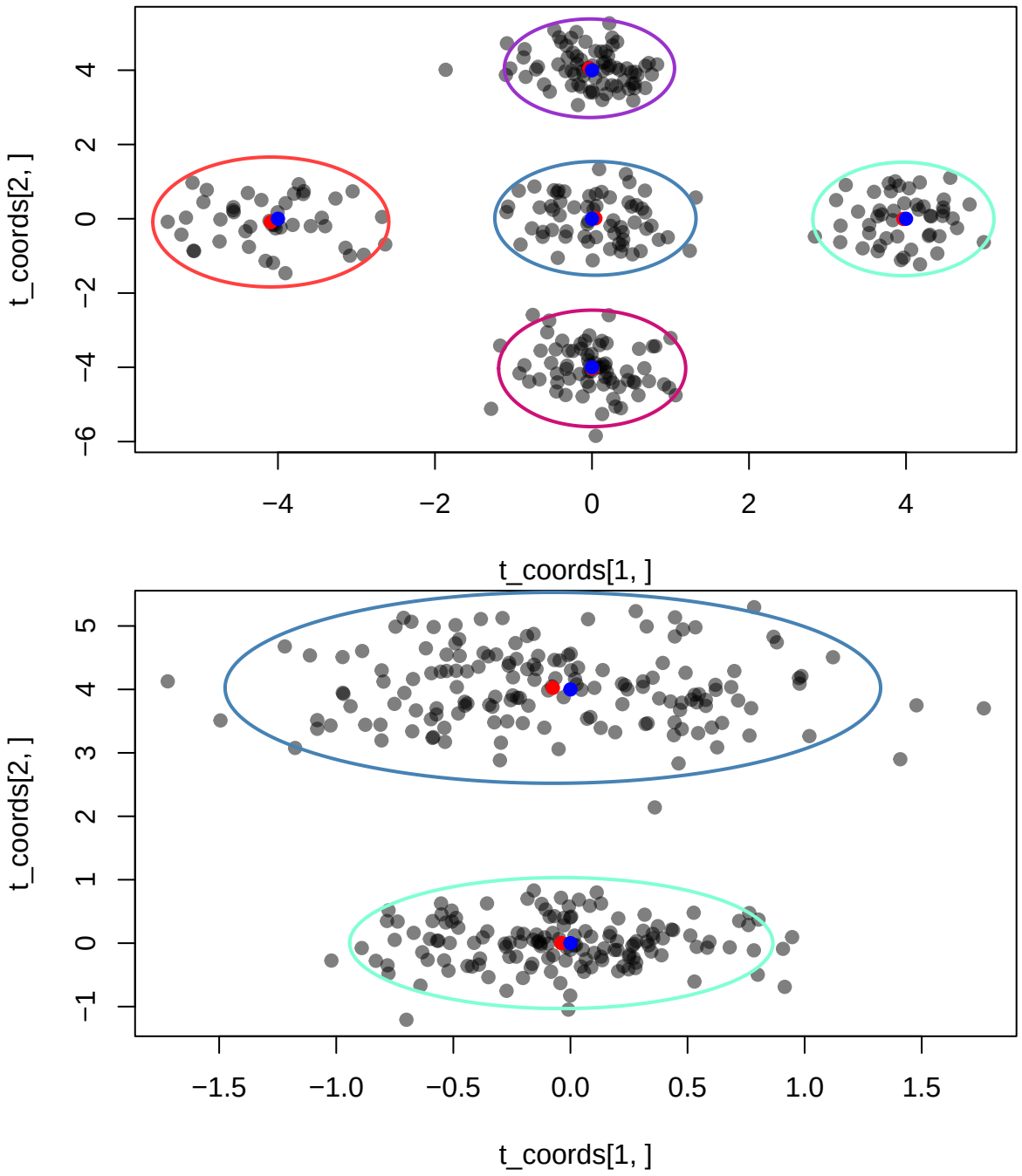


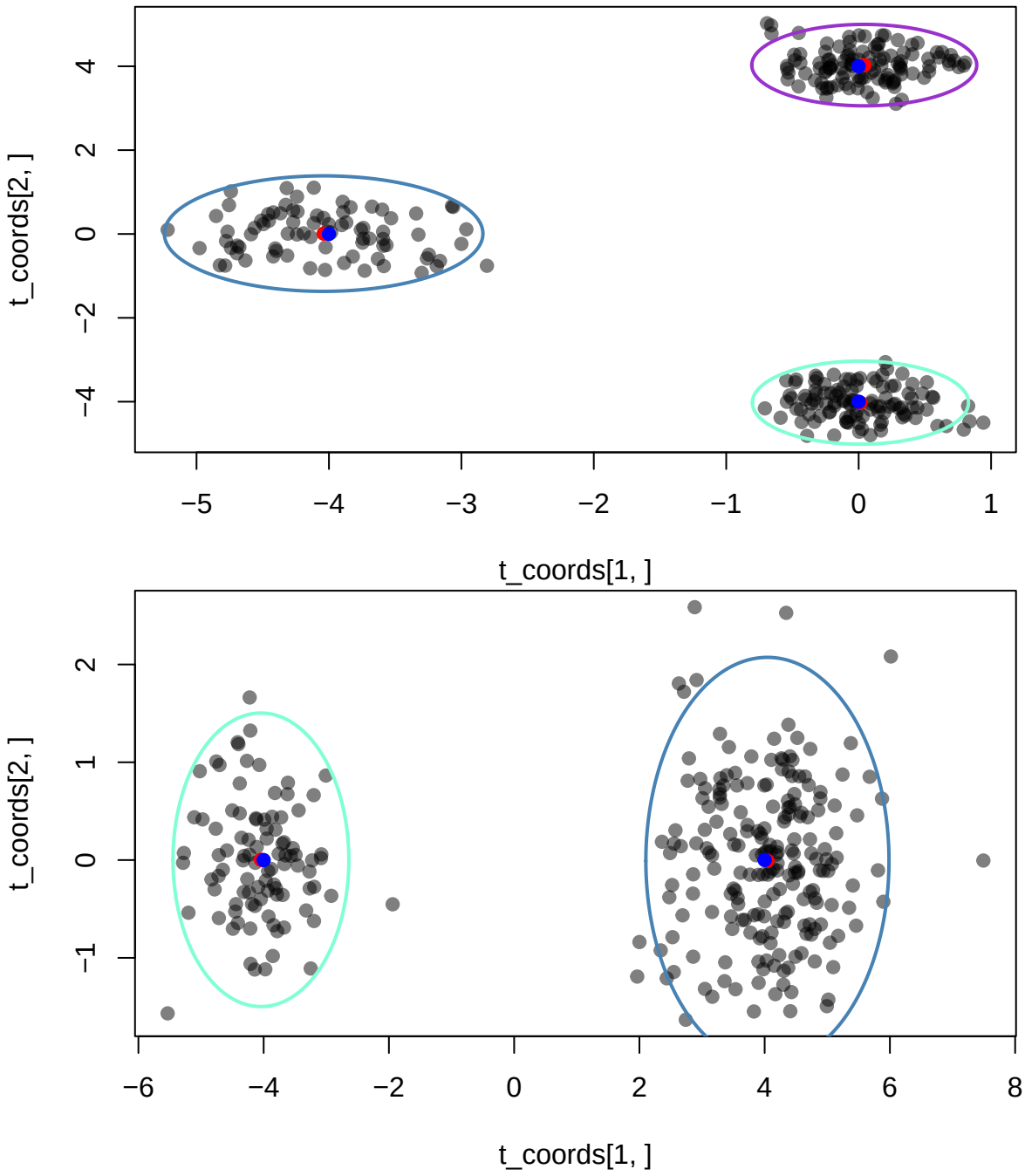

```
## predicted wrong number of clusters
```

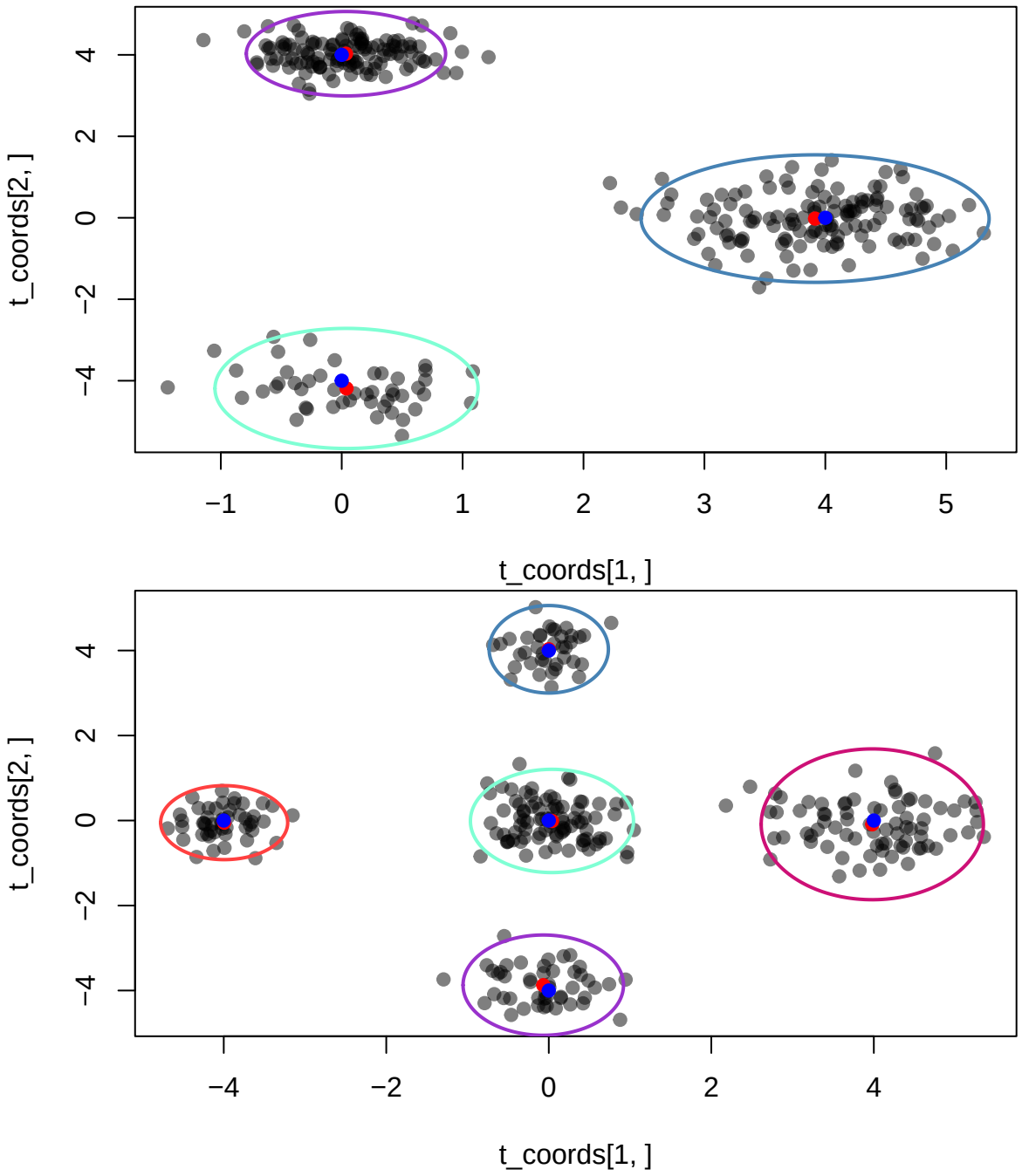


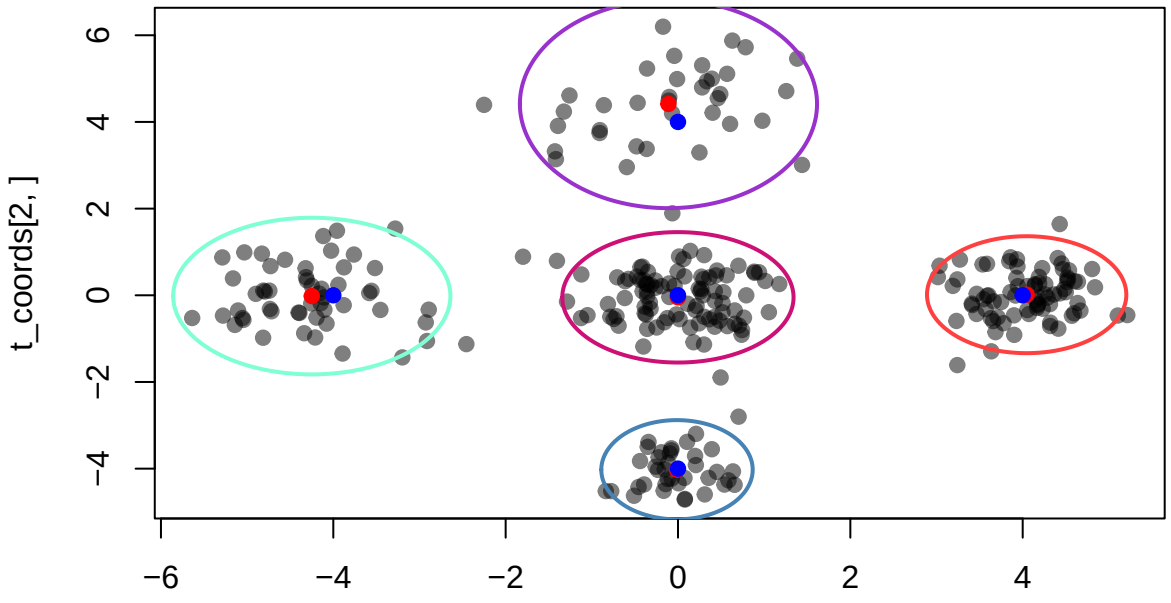

t_coords[2,]
6
4
2
0
−2
−4
−4
−2
0
2
t_coords[1,]
t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
t_coords[1,]

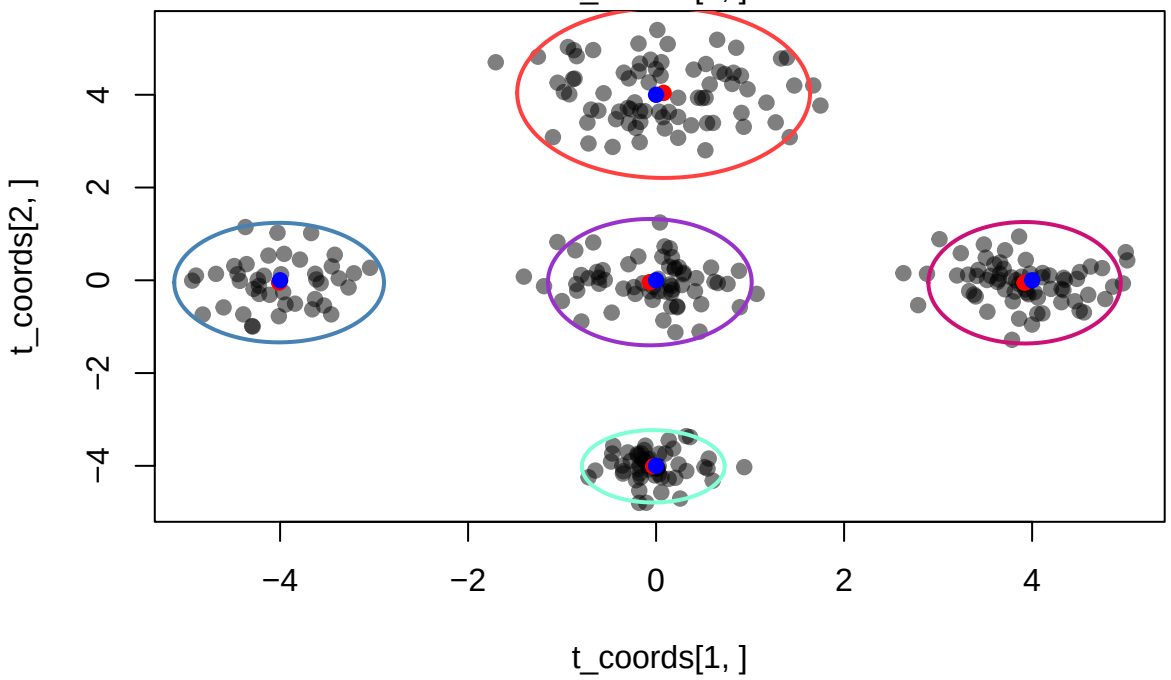

t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
t_coords[1,]
t_coords[2,]
2
1
0
−1
−2
−3
−4
−5
−4
−2
0
2
4
6
t_coords[1,]

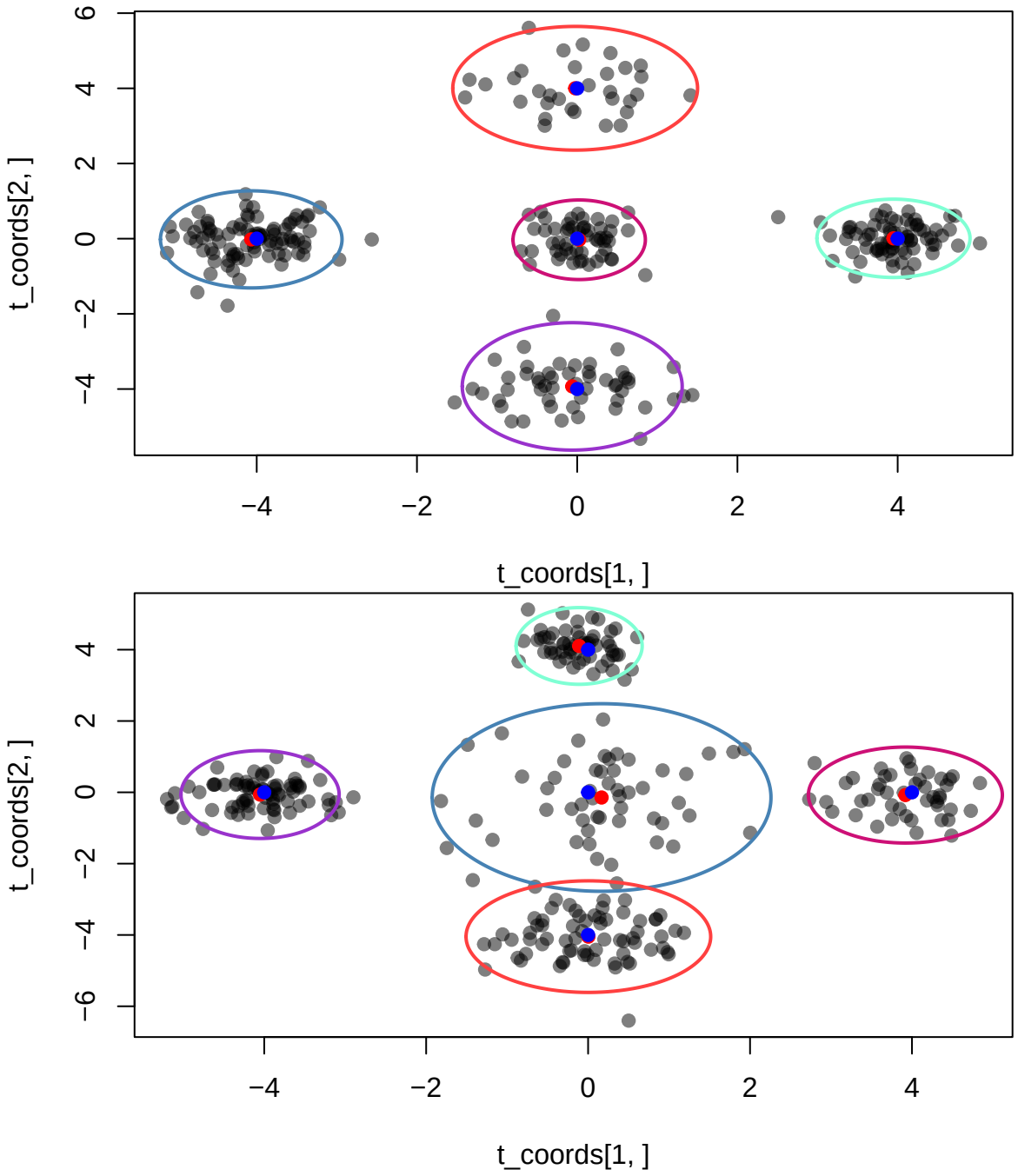

t_coords[2,]
4
2
0
−2
−4
−6
−4
−2
0
2
4
t_coords[1,]
t_coords[2,]
5
4
3
2
1
0
−1
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
t_coords[1,]

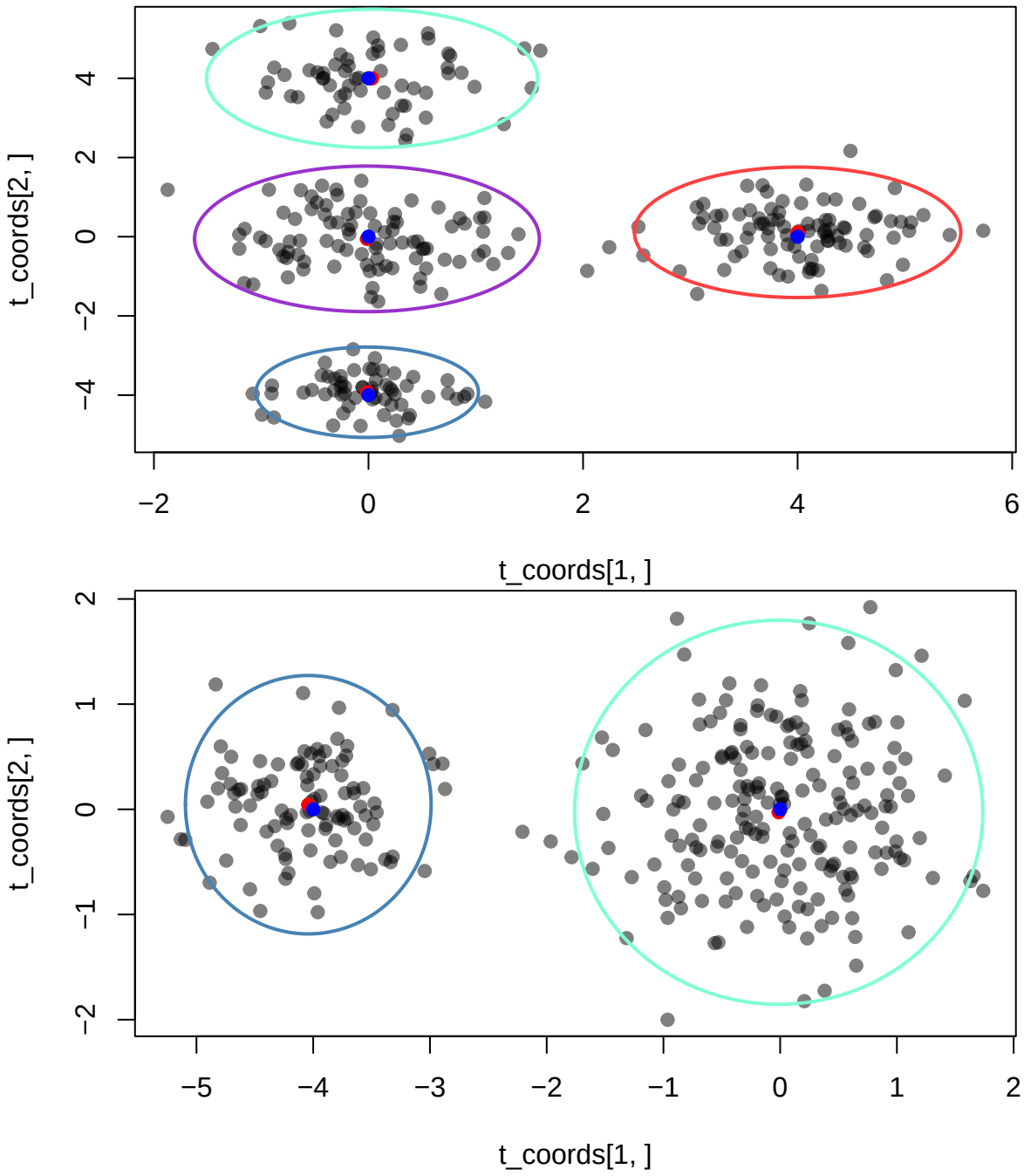

t_coords[2,]
4
2
0
−2
−4
−5
−4
−3
−2
−1
0
1
t_coords[1,]
t_coords[2,]
2
1
0
−1
−6
−4
−2
0
2
4
6
8
t_coords[1,]

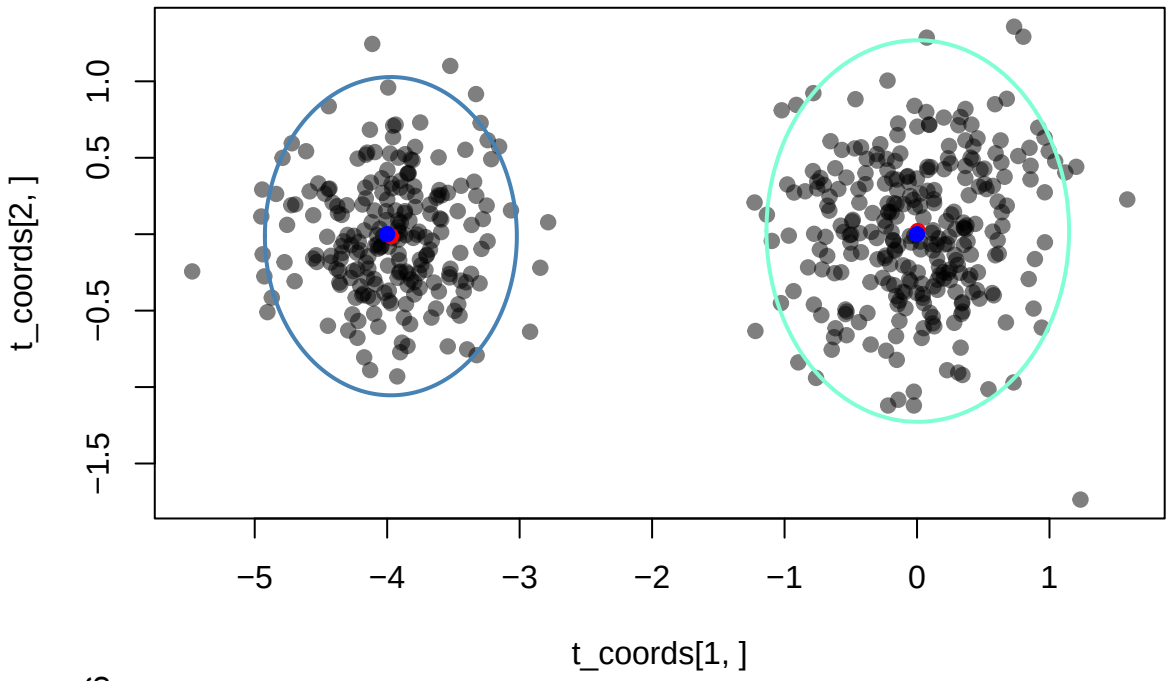

t_coords[2,]
4
2
0
−2
−4
−1
0
1
2
3
4
5
t_coords[1,]
t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
t_coords[1,]

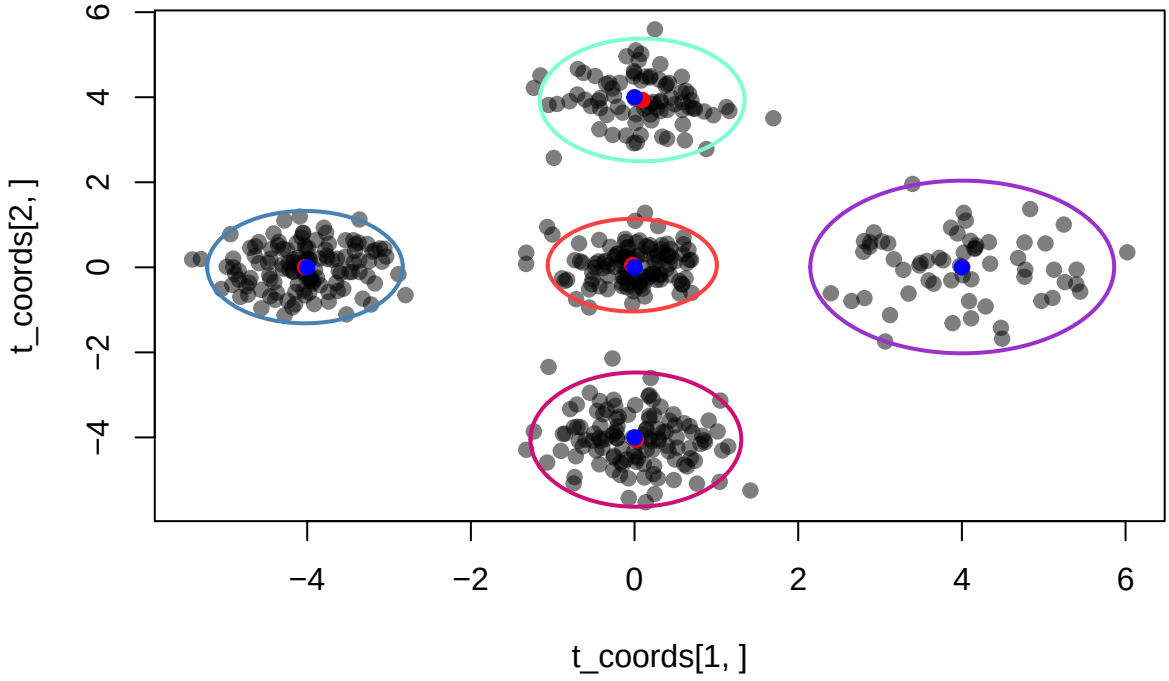

t_coords[2,]
6
4
2
0
−2
−4
−6
−4
−2
0
2
4
t_coords[1,]

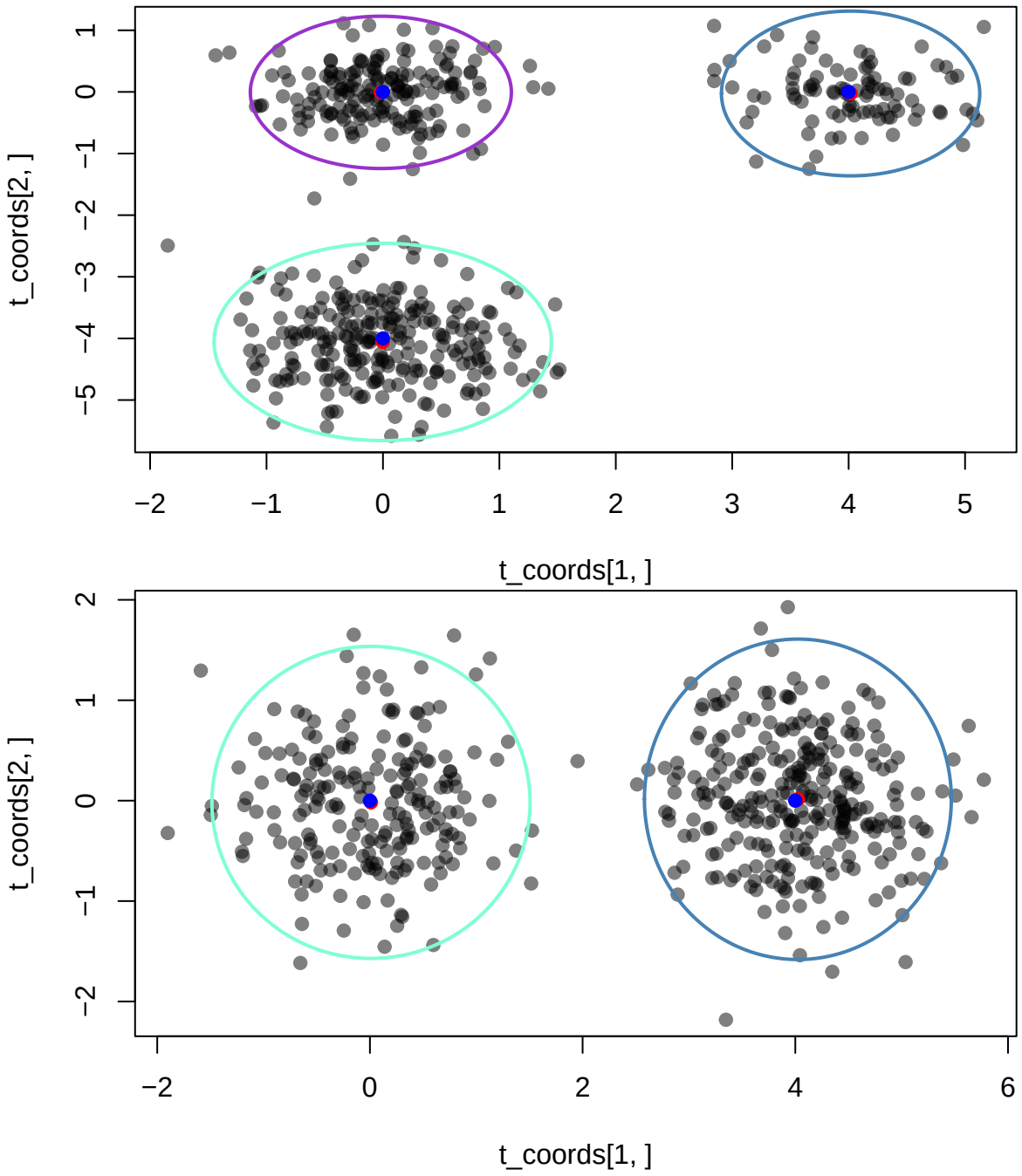

t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
t_coords[1,]

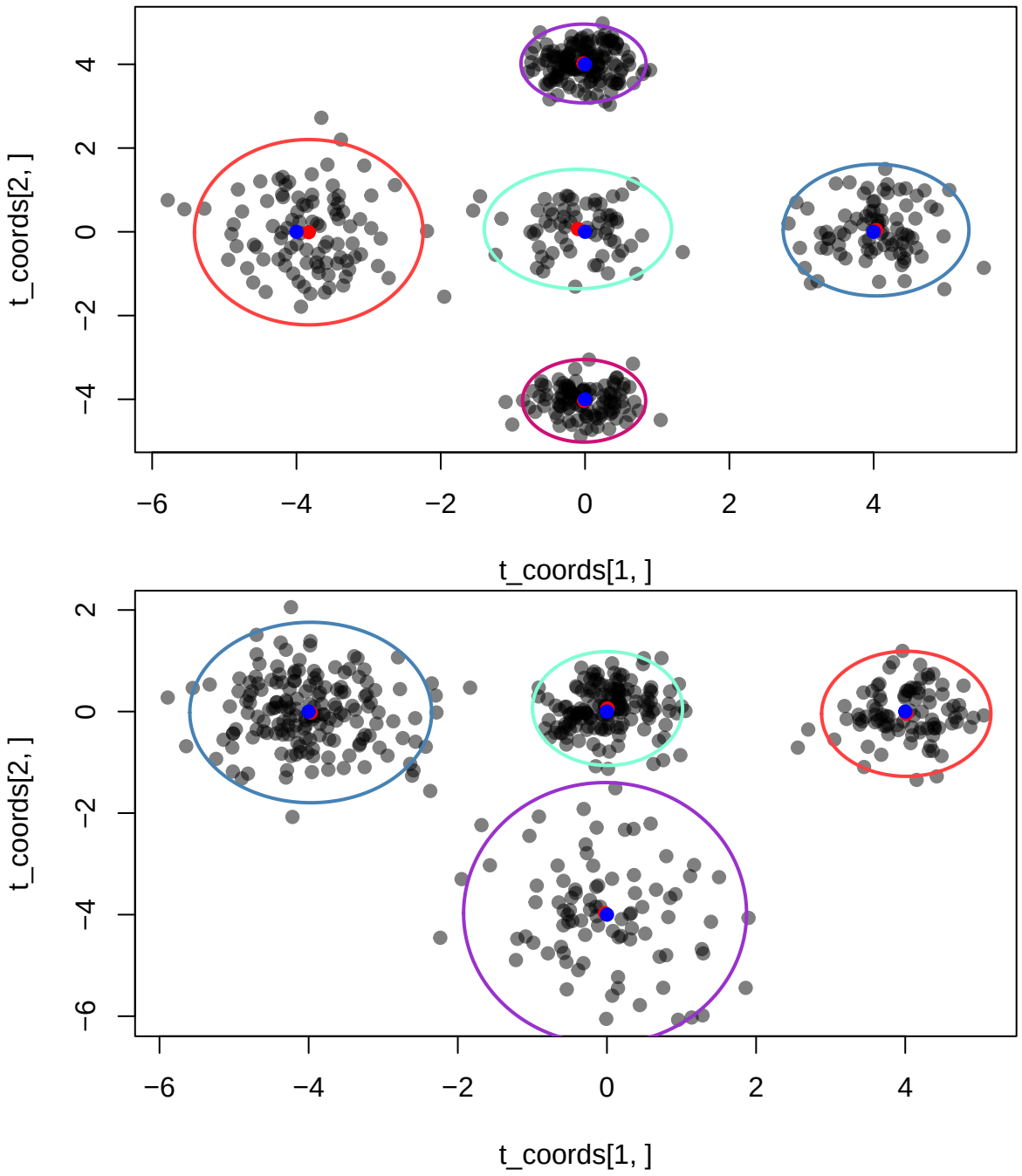

t_coords[2,]
6
4
2
0
−2
−4
−4
−2
0
2
4
t_coords[1,]
t_coords[2,]
4
2
0
−2
−4
−6
−4
−2
0
2
4
t_coords[1,]

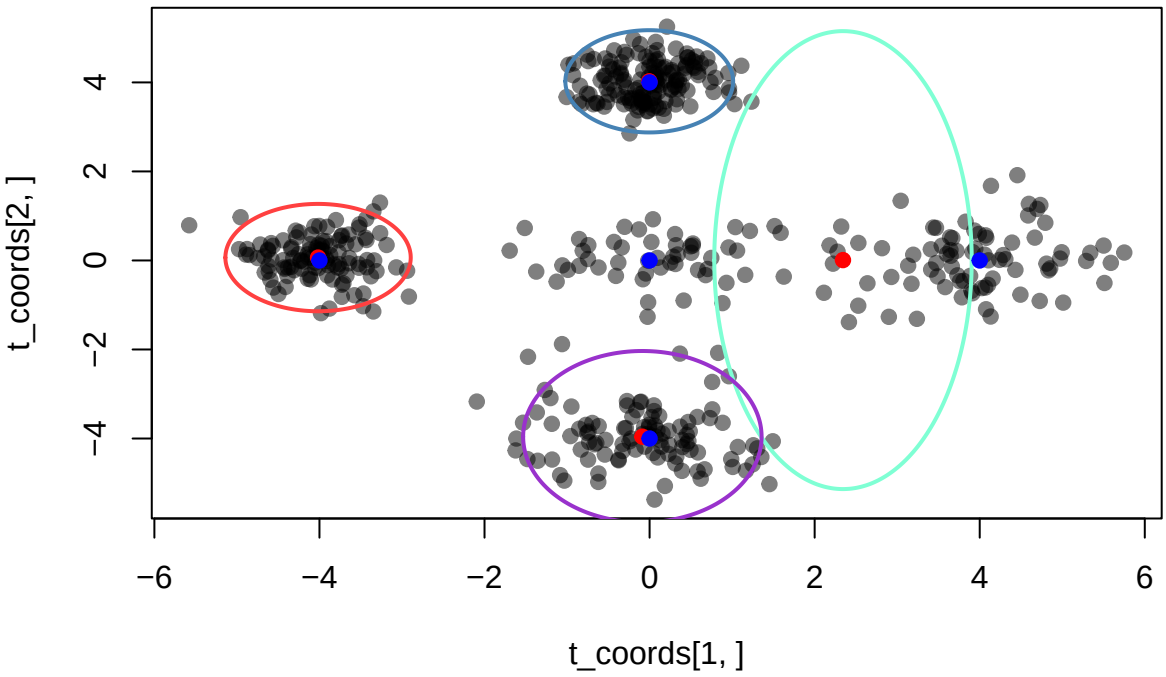

t_coords[2,]
4
2
0
−2
−4
−2
0
2
4
6
t_coords[1,]
t_coords[2,]
2
1
0
−1
−2
−5
−4
−3
−2
−1
0
1
2
t_coords[1,]


t_coords[2,]
1.0
0.5
−0.5
−1.5
−5
−4
−3
−2
−1
0
1
t_coords[1,]

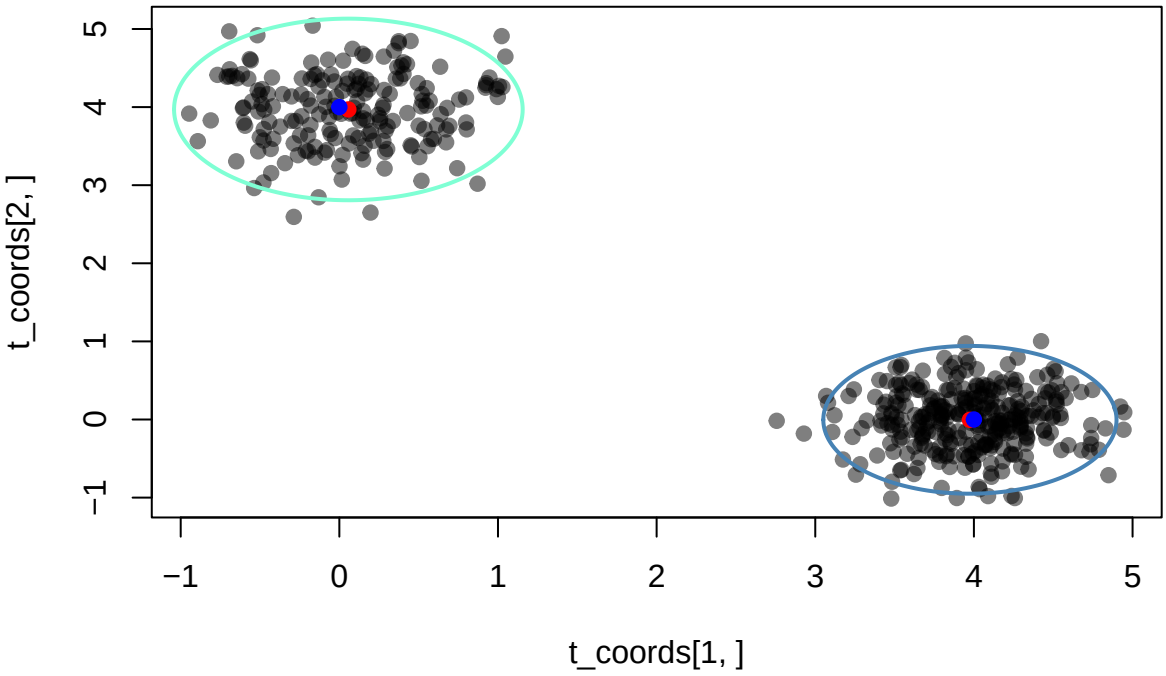

t_coords[2,]
6
4
2
0
−2
−4
−4
−2
0
2
4
6
t_coords[1,]

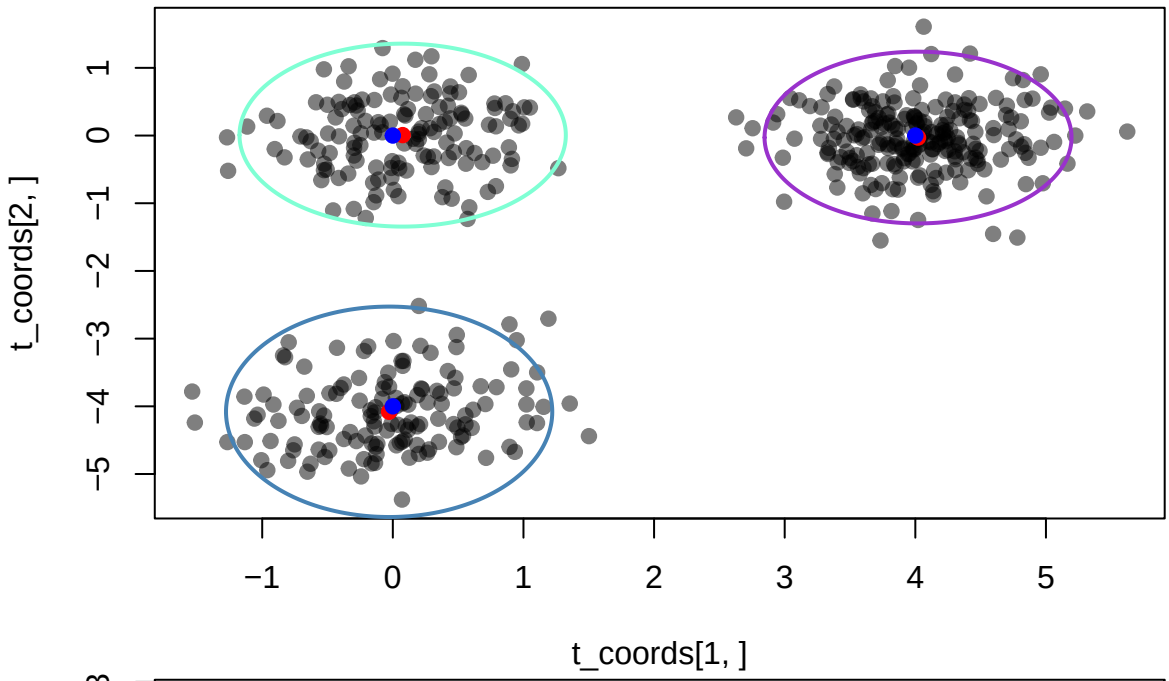

t_coords[2,]
1
0
−1
−2
−3
−4
−5
−2
−1
0
1
2
3
4
5
t_coords[1,]
t_coords[2,]
2
1
0
−1
−2
−2
0
2
4
6
t_coords[1,]

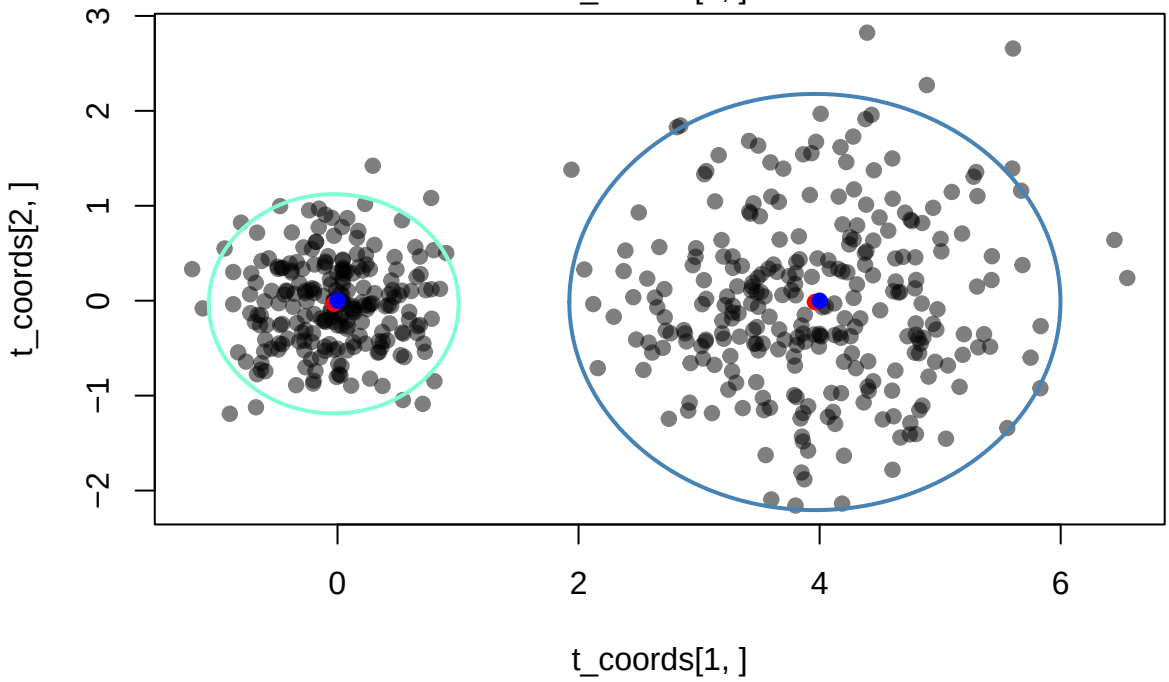

t_coords[2,]
4
2
0
−2
−4
−6
−4
−2
0
2
4
t_coords[1,]
t_coords[2,]
2
0
−2
−4
−6
−6
−4
−2
0
2
4
t_coords[1,]

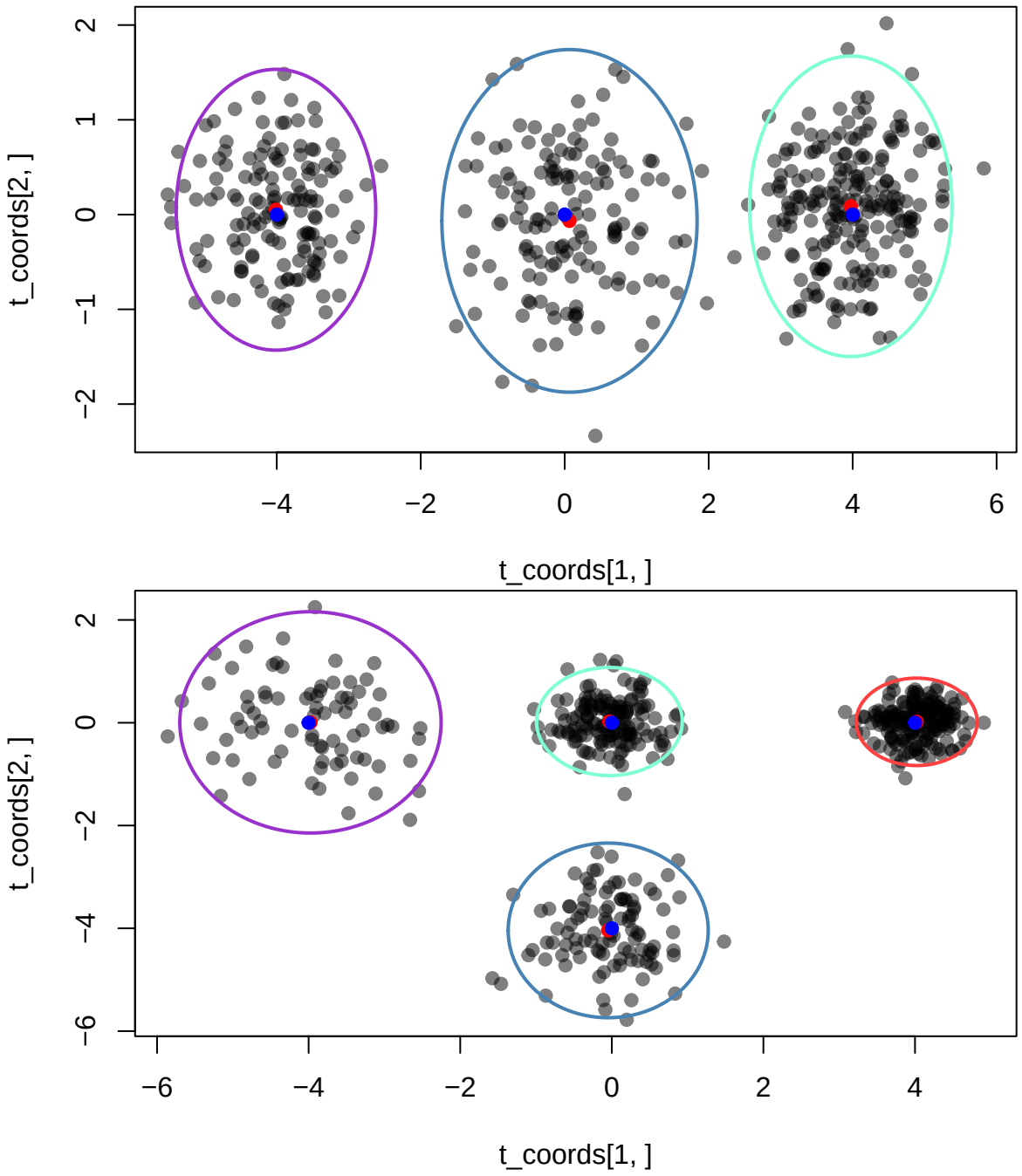


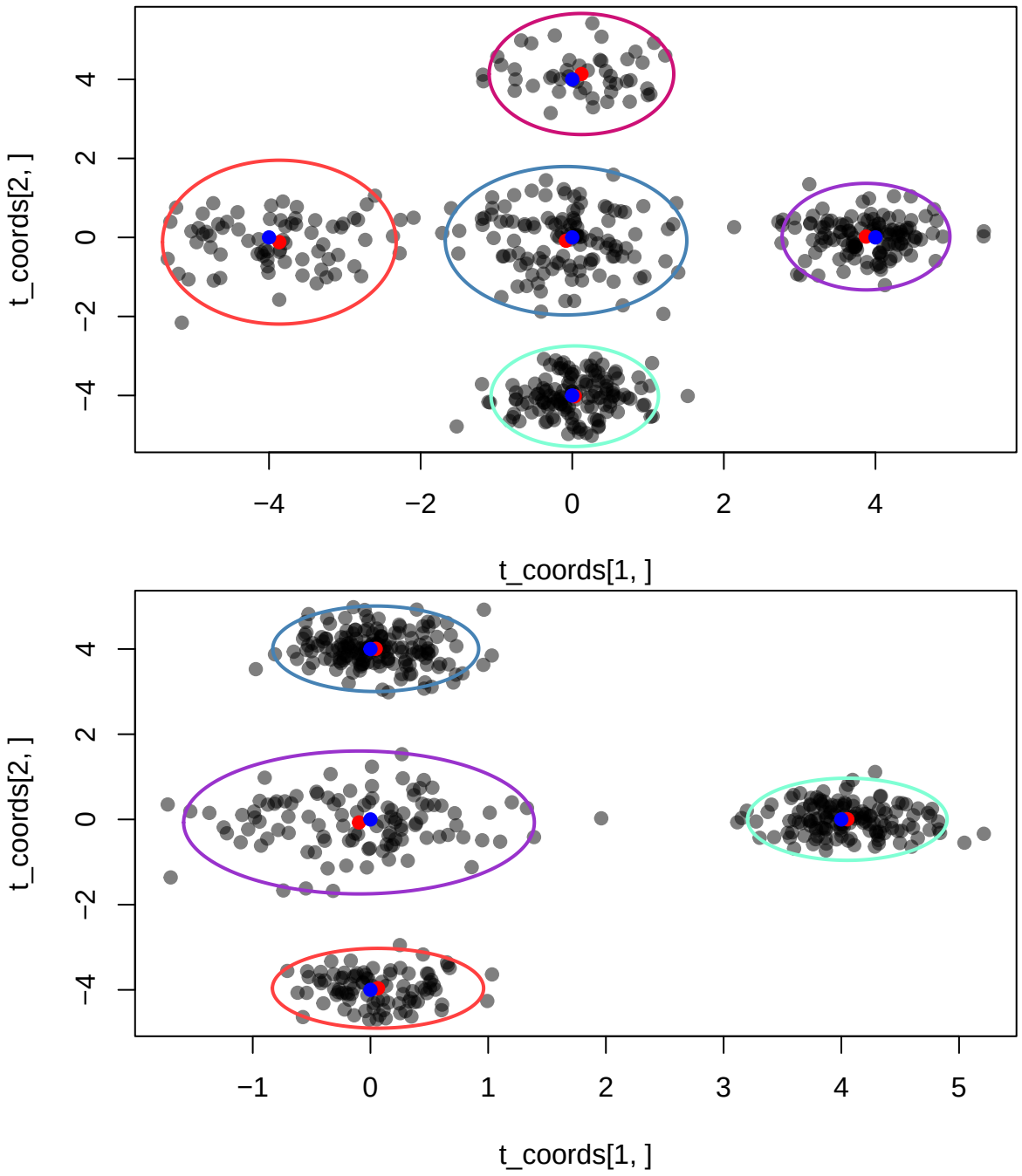

```
## predicted wrong number of clusters
```

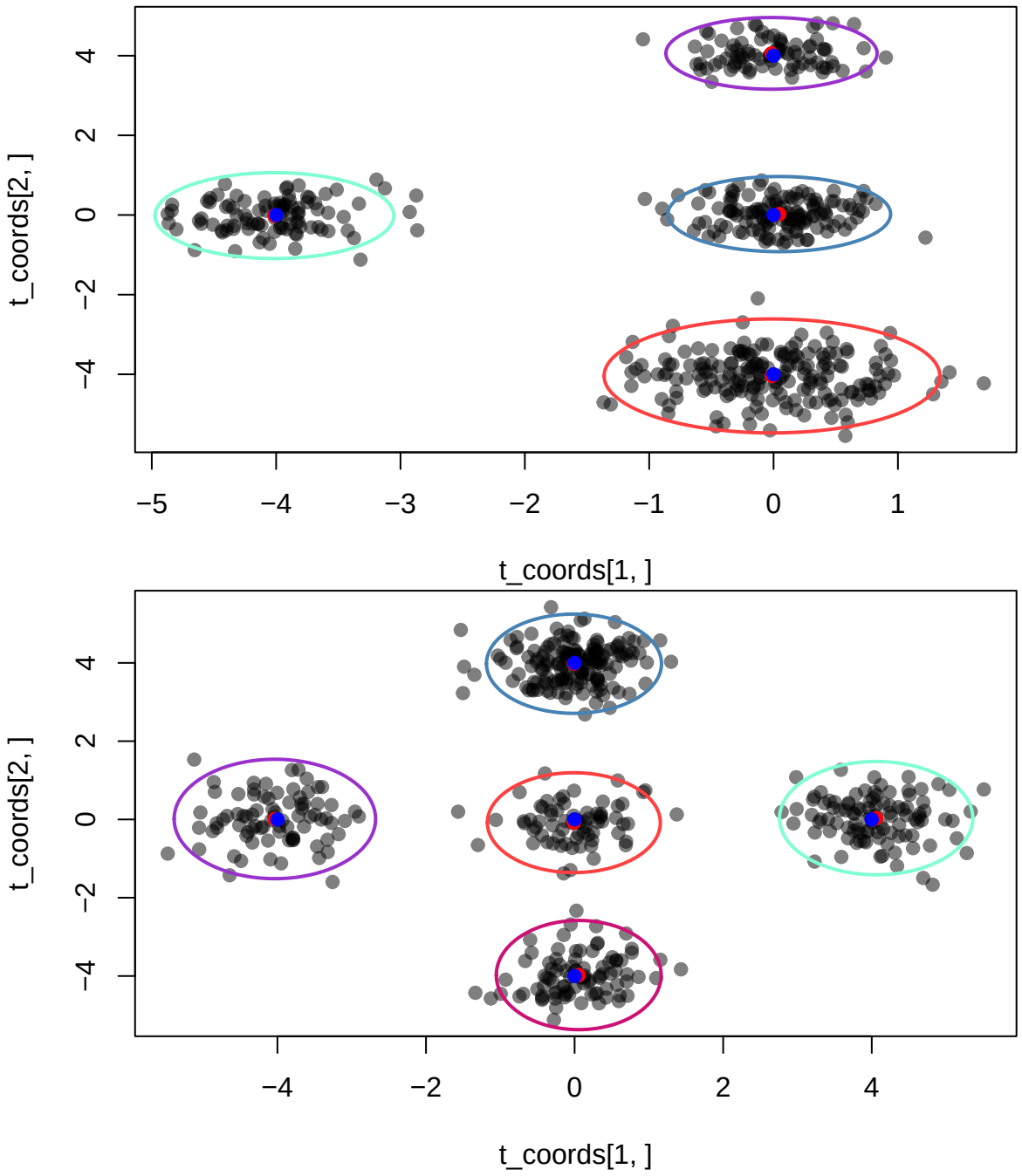


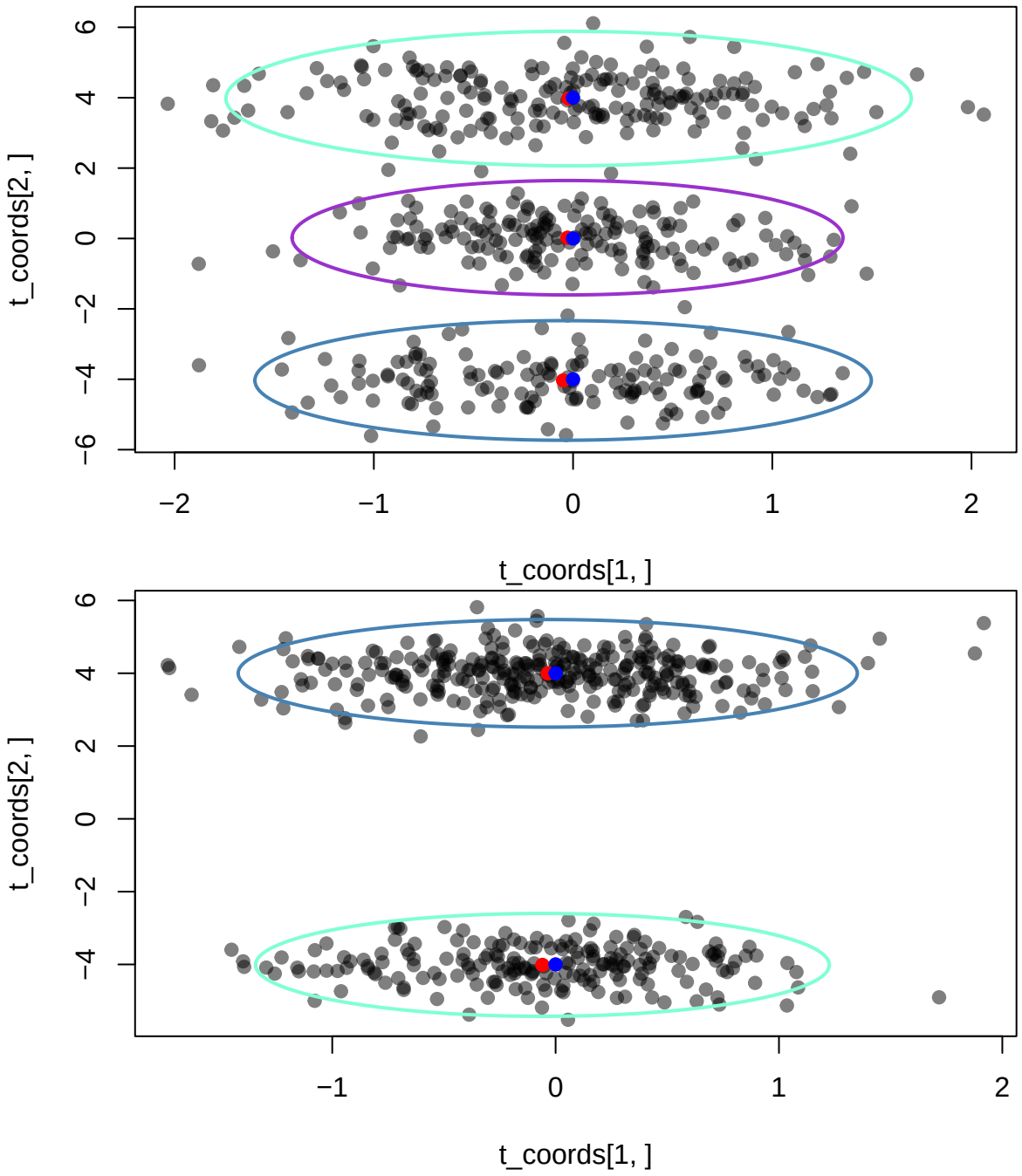

t_coords[2,]
1
0
−1
−2
−3
−4
−5
−1
0
1
2
3
4
5
t_coords[1,]

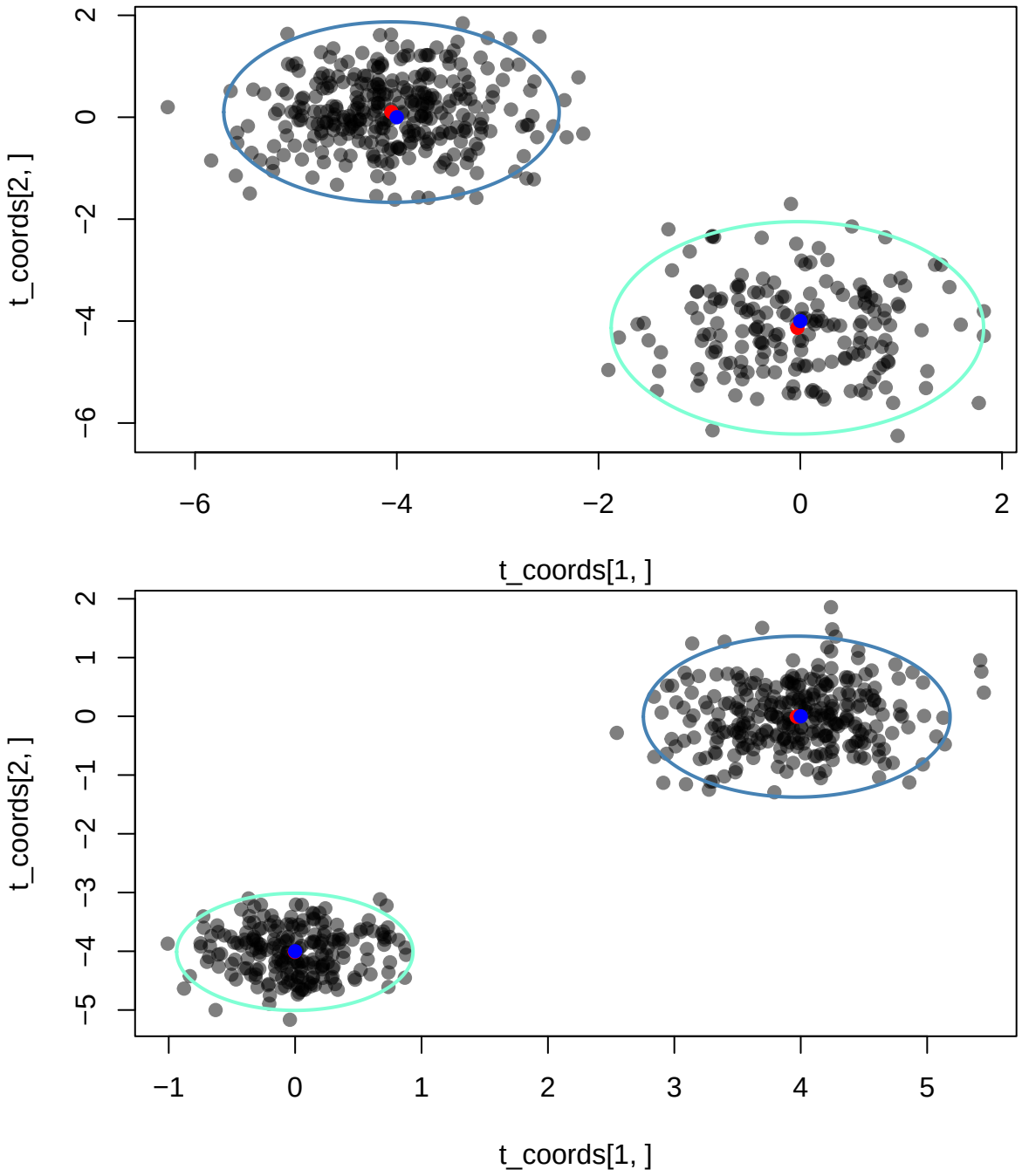

t_coords[2,]
3
2
1
0
−1
−2
0
2
4
6
t_coords[1,]

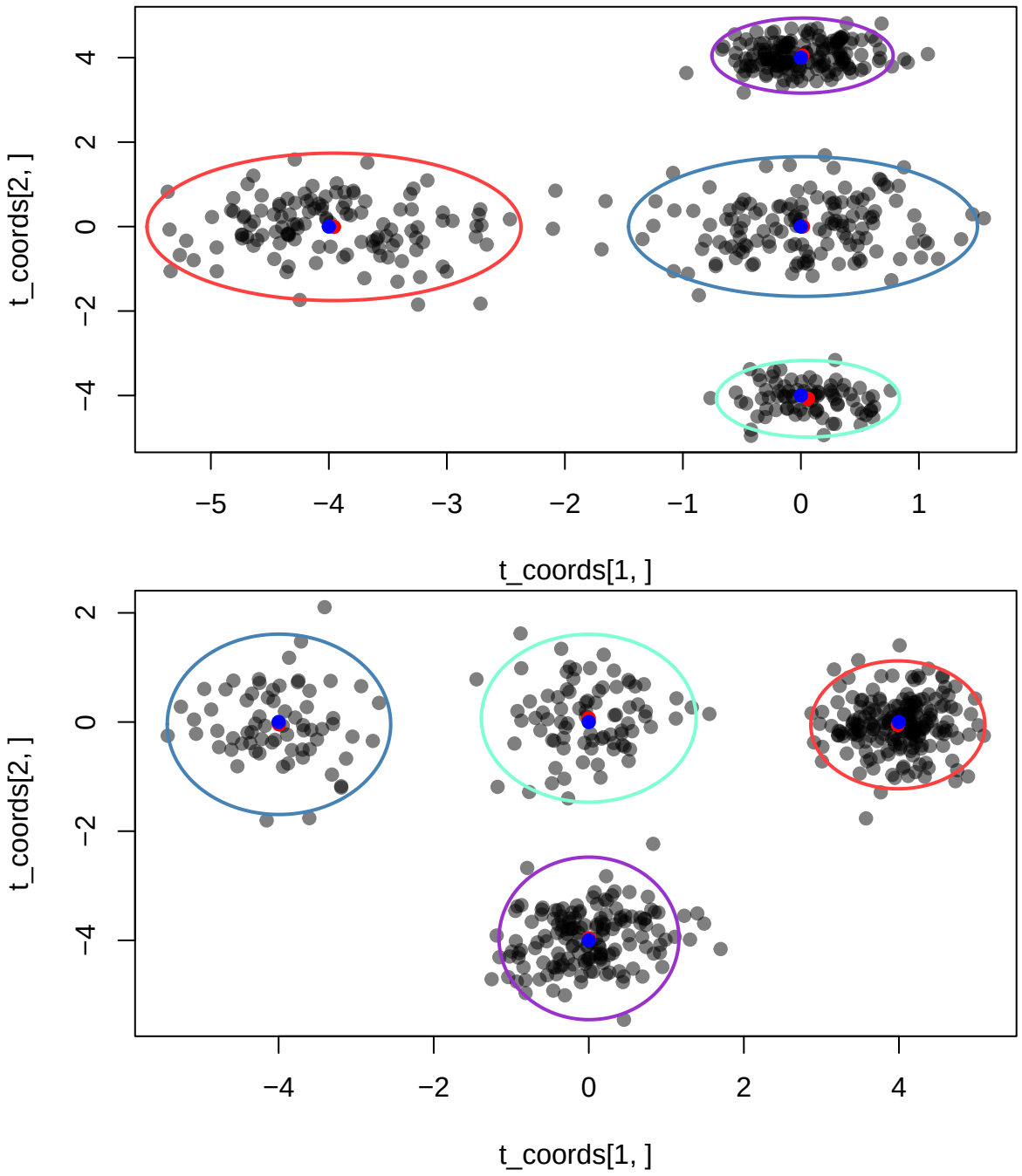

t_coords[2,]
2
1
0
−1
−2
−4
−2
0
2
4
6
t_coords[1,]
t_coords[2,]
2
0
−2
−4
−6
−6
−4
−2
0
2
4
t_coords[1,]

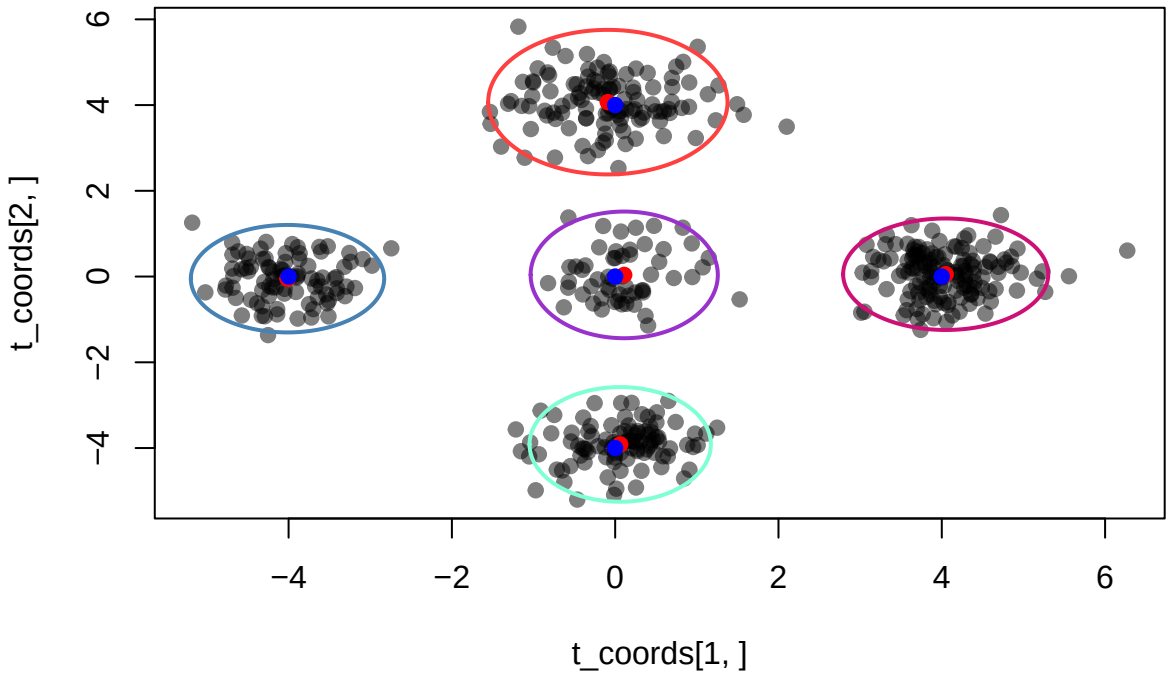

t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
t_coords[1,]
t_coords[2,]
4
2
0
−2
−4
−1
0
1
2
3
4
5
t_coords[1,]

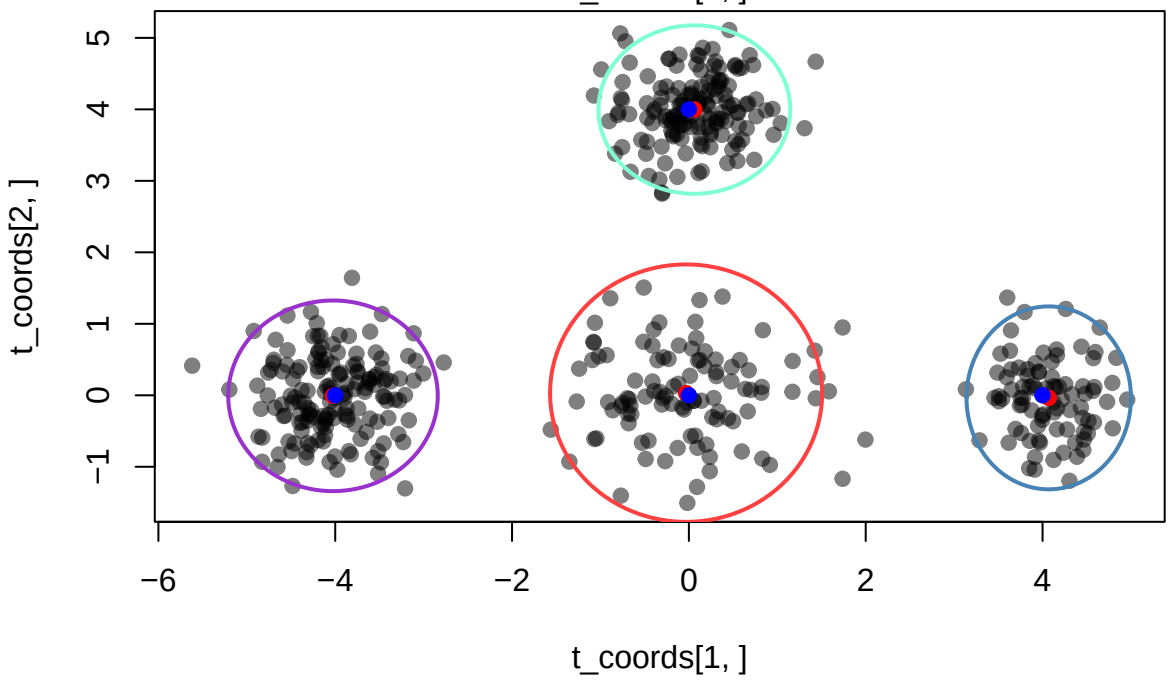

t_coords[2,]
4
2
0
−2
−4
−5
−4
−3
−2
−1
0
1
t_coords[1,]
t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
t_coords[1,]

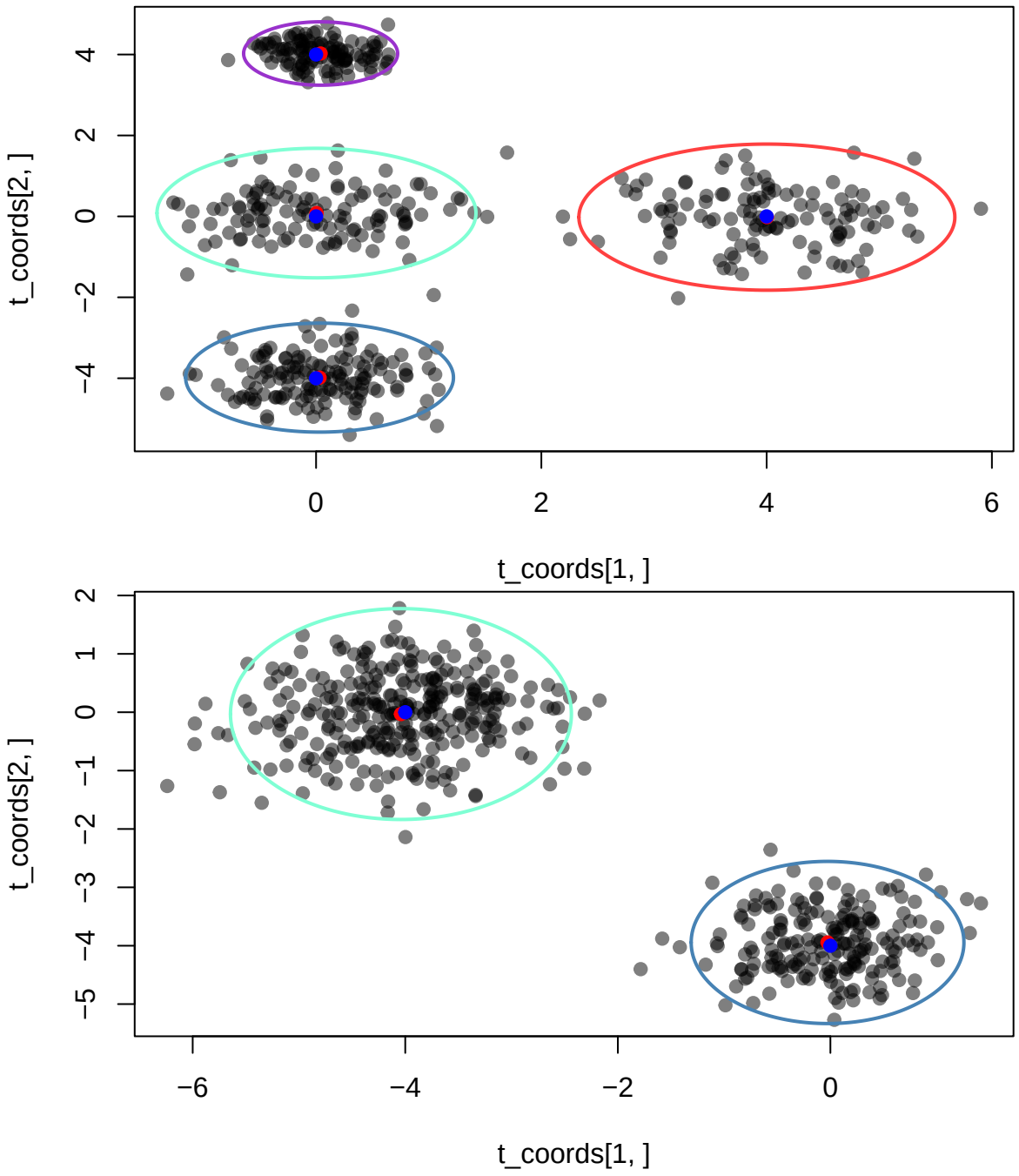

t_coords[2,]
6
4
2
0
−2
−4
−6
−2
−1
0
1
2
t_coords[1,]
t_coords[2,]
6
4
2
0
−2
−4
−1
0
1
2
t_coords[1,]

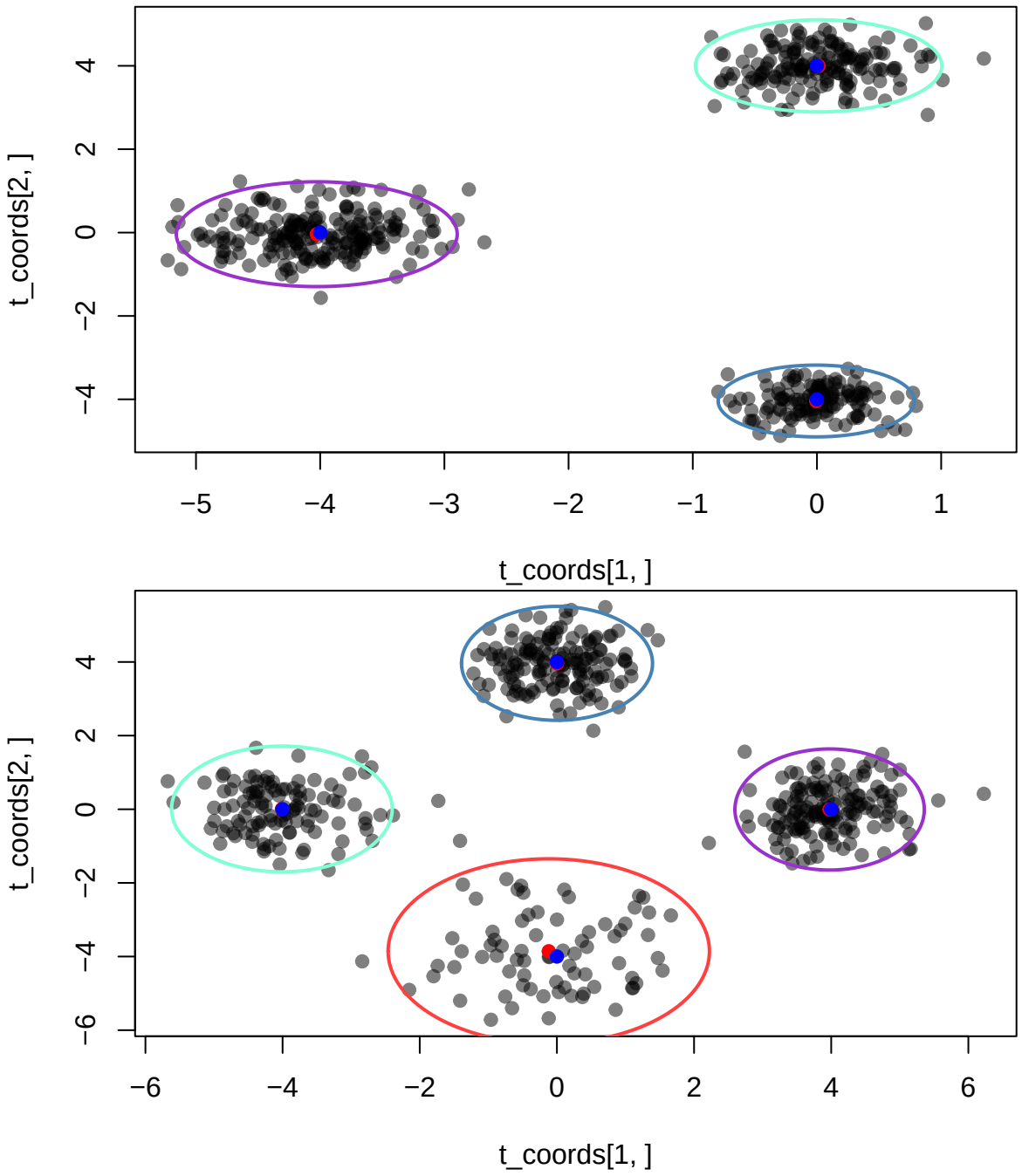

t_coords[2,]
2
0
−2
−4
−6
−6
−4
−2
0
2
t_coords[1,]
t_coords[2,]
2
1
0
−1
−2
−3
−4
−5
−1
0
1
2
3
4
5
t_coords[1,]

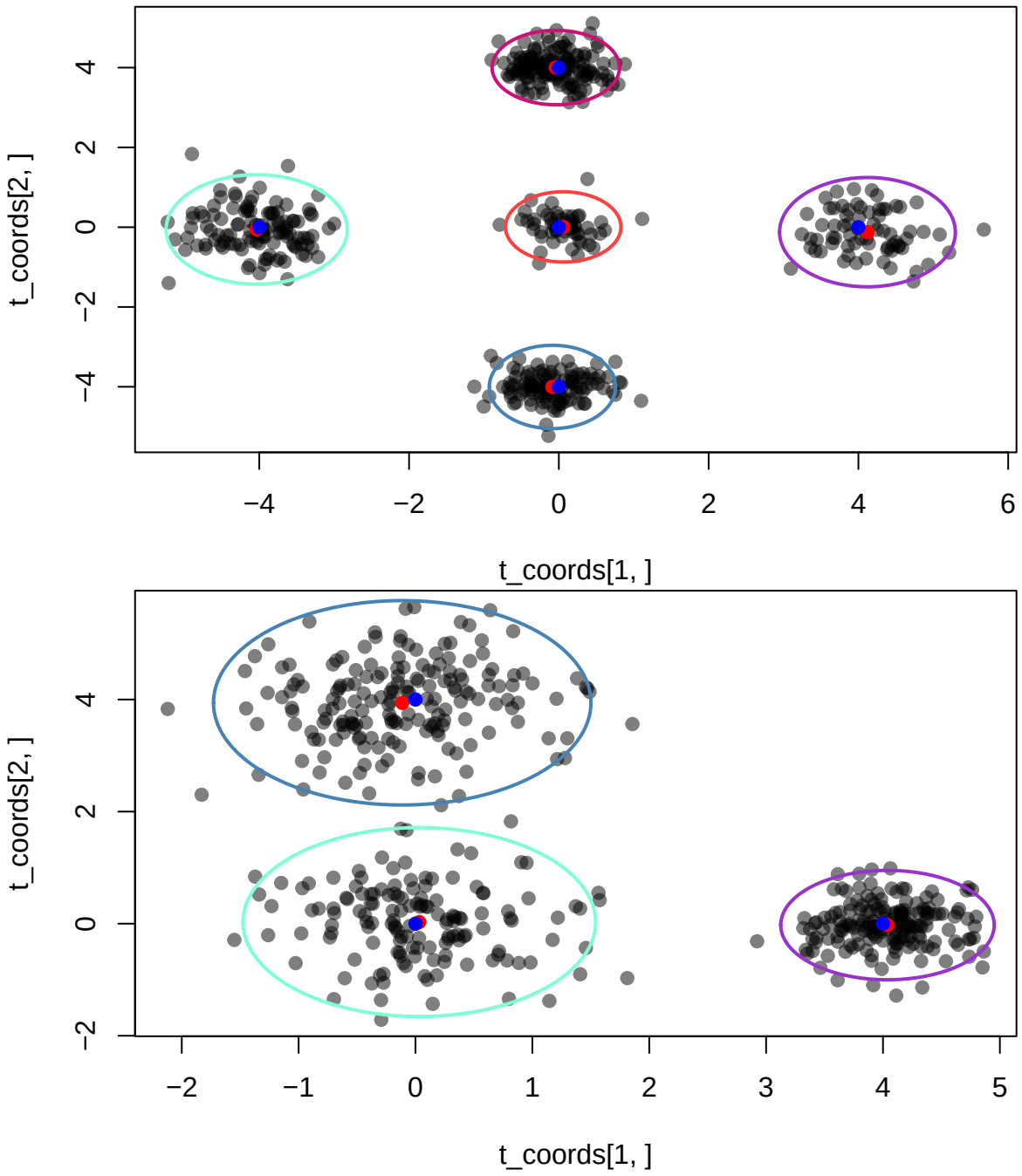

t_coords[2,]
4
2
0
−2
−4
−5
−4
−3
−2
−1
0
1
t_coords[1,]
t_coords[2,]
2
0
−2
−4
−4
−2
0
2
4
t_coords[1,]

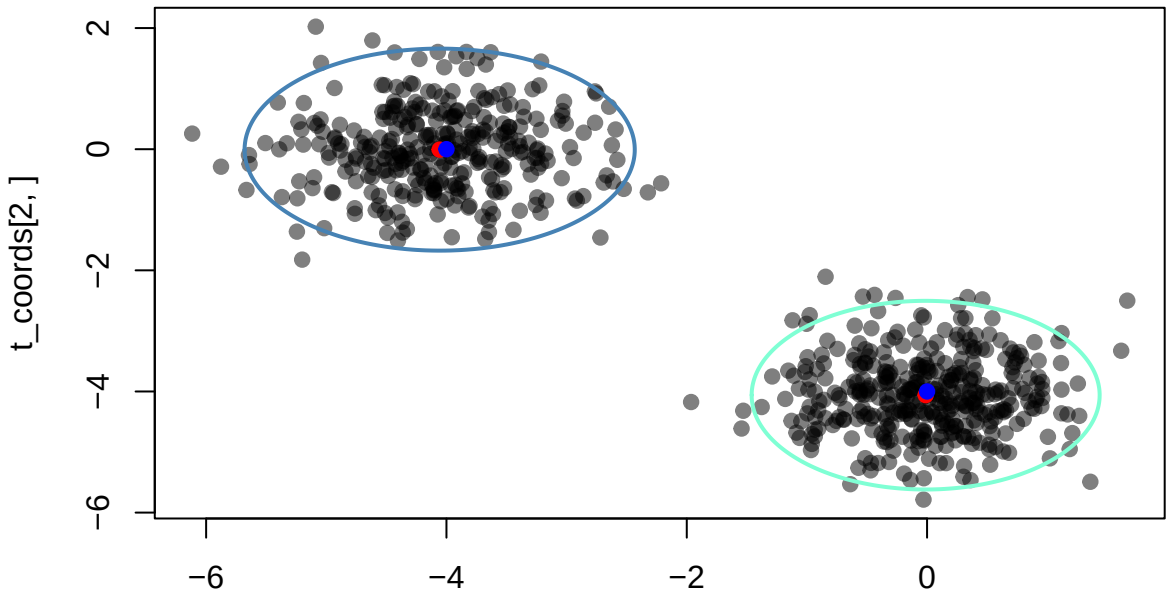

t_coords[2,]
6
4
2
0
−2
−4
−4
−2
0
2
4
6
t_coords[1,]

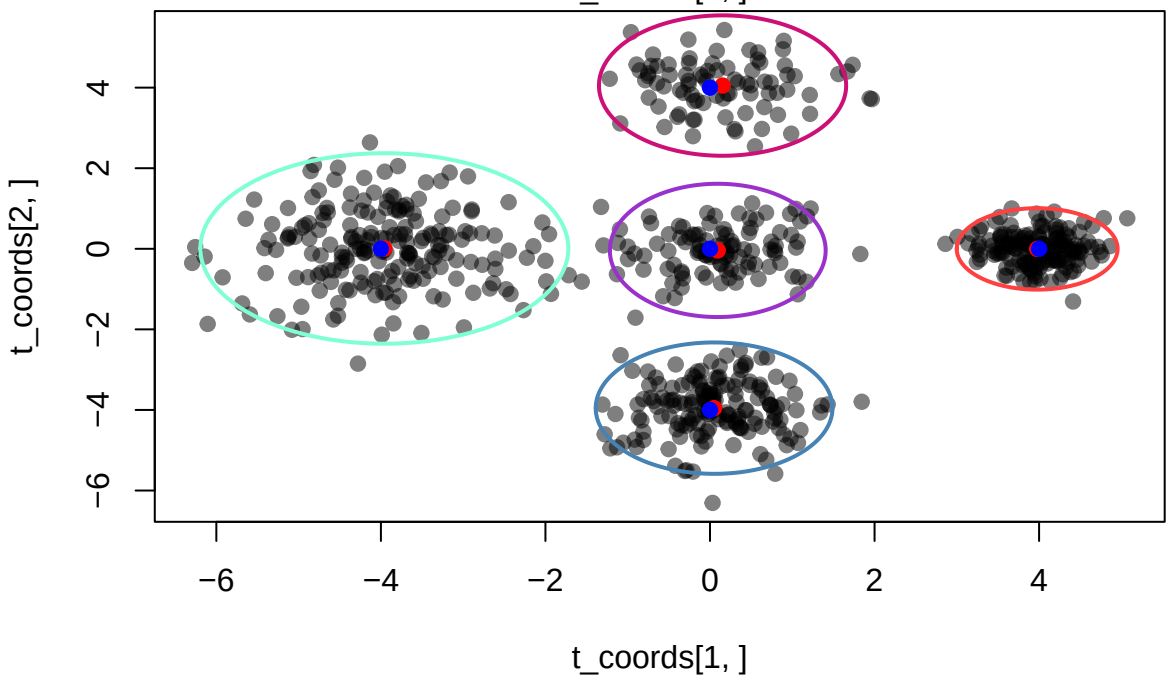

t_coords[2,]
5
4
3
2
1
0
−1
−6
−4
−2
0
2
4
t_coords[1,]

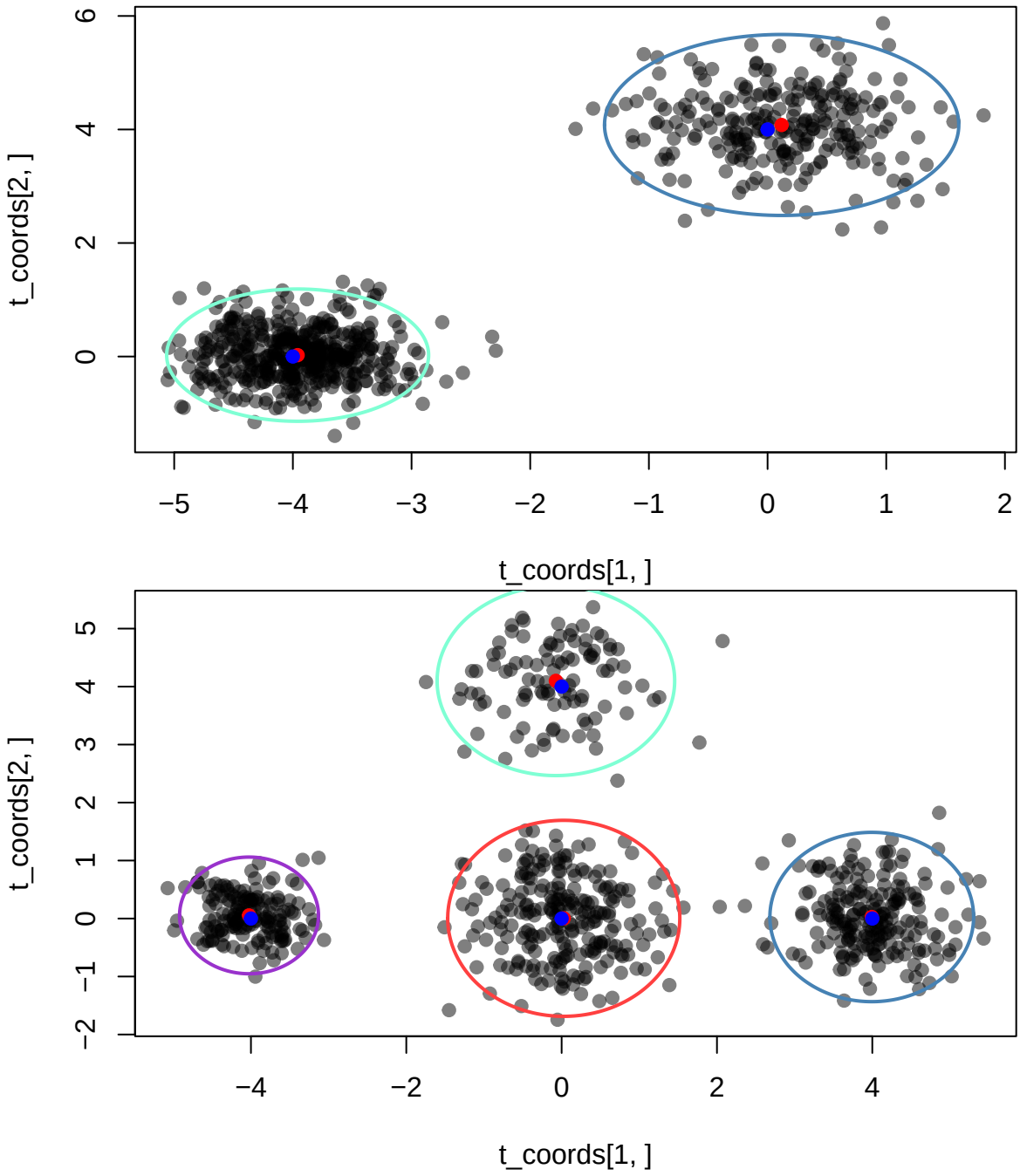

t_coords[2,]
4
2
0
−2
−4
0
2
4
6
t_coords[1,]
t_coords[2,]
2
1
0
−1
−2
−3
−4
−5
−6
−4
−2
0
t_coords[1,]

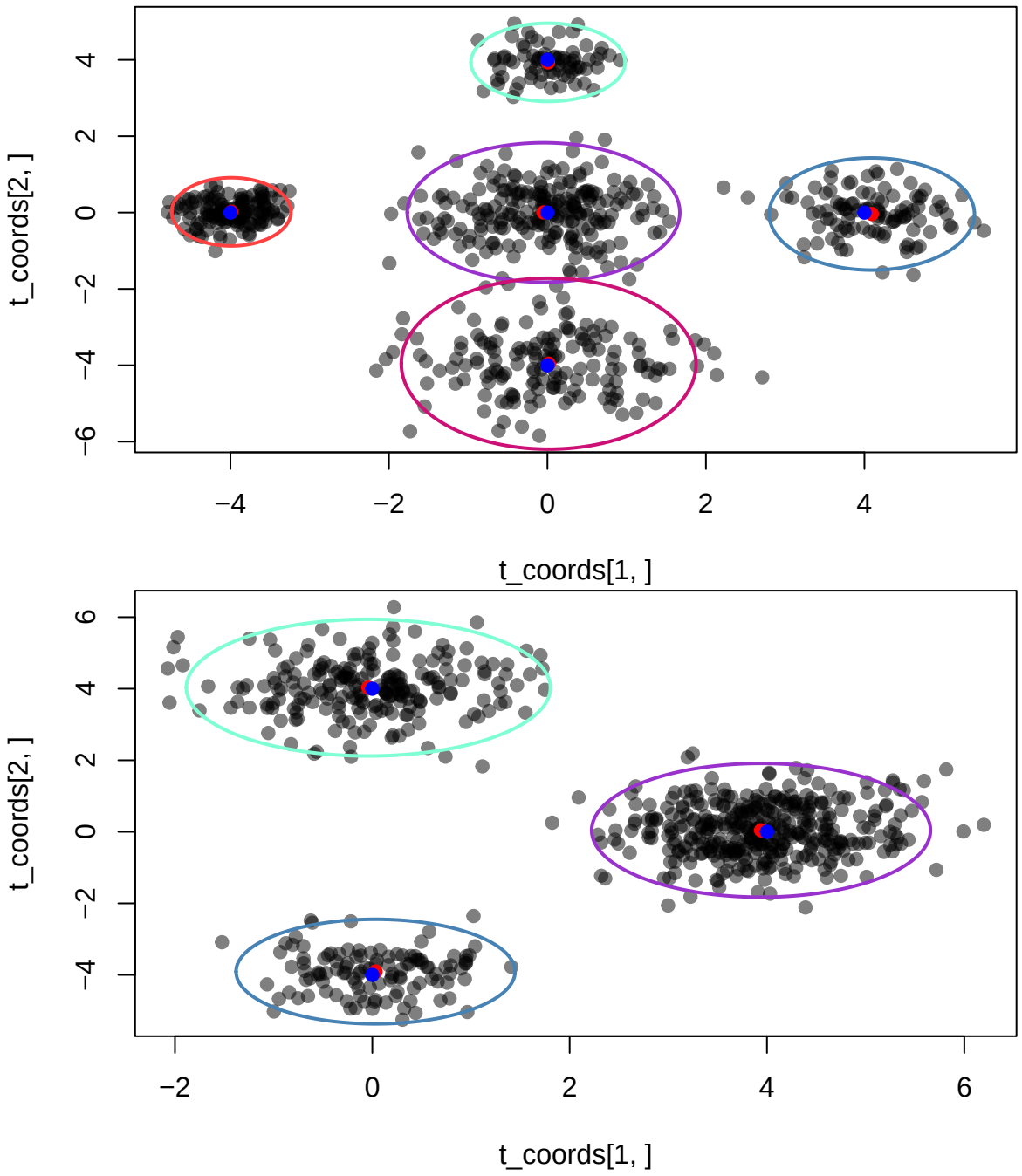

t_coords[2,]
4
2
0
−2
−4
−5
−4
−3
−2
−1
0
1
t_coords[1,]
t_coords[2,]
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
t_coords[1,]

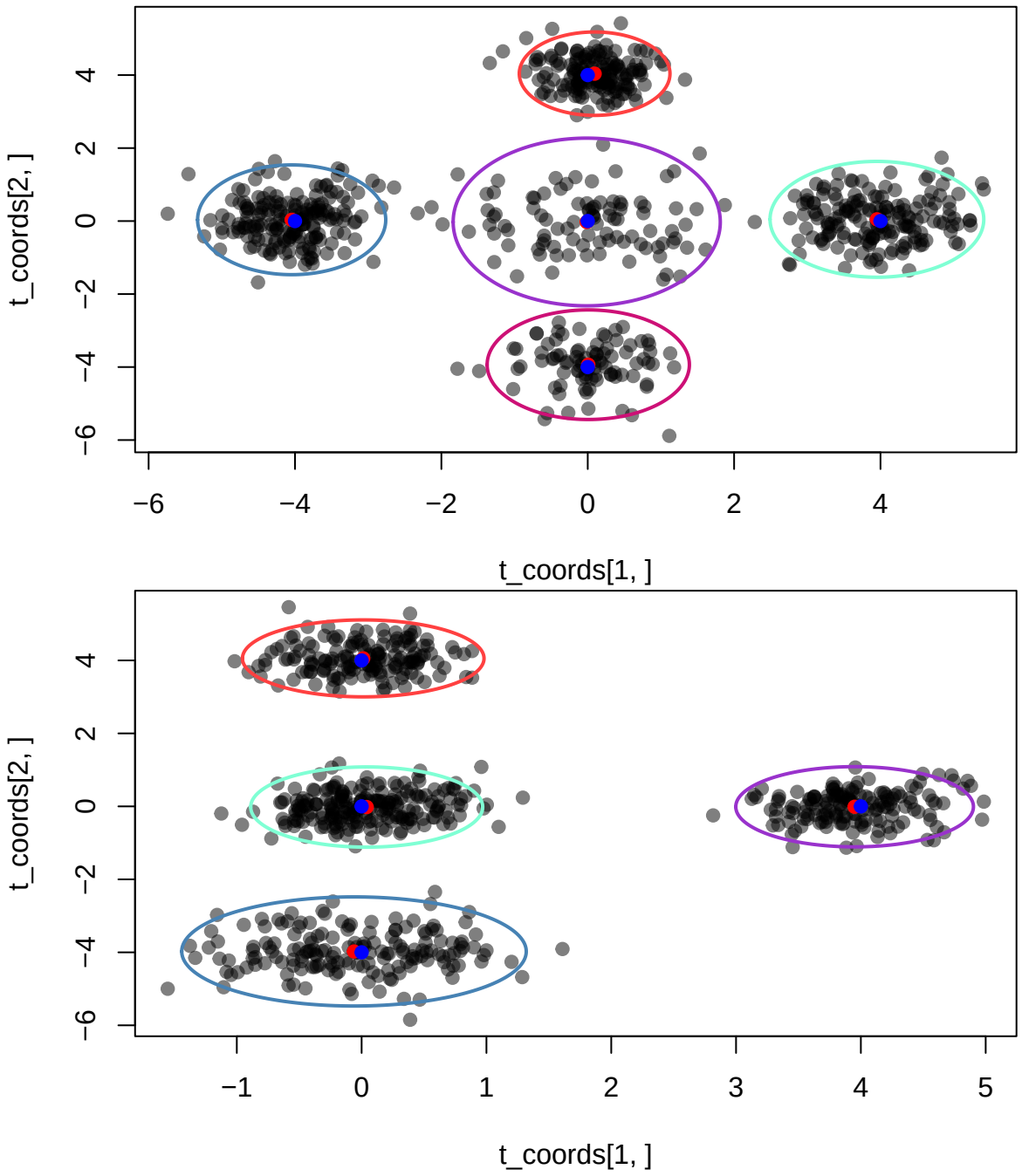

t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
6
t_coords[1,]
t_coords[2,]
4
2
0
−2
−2
−1
0
1
2
3
4
5
t_coords[1,]

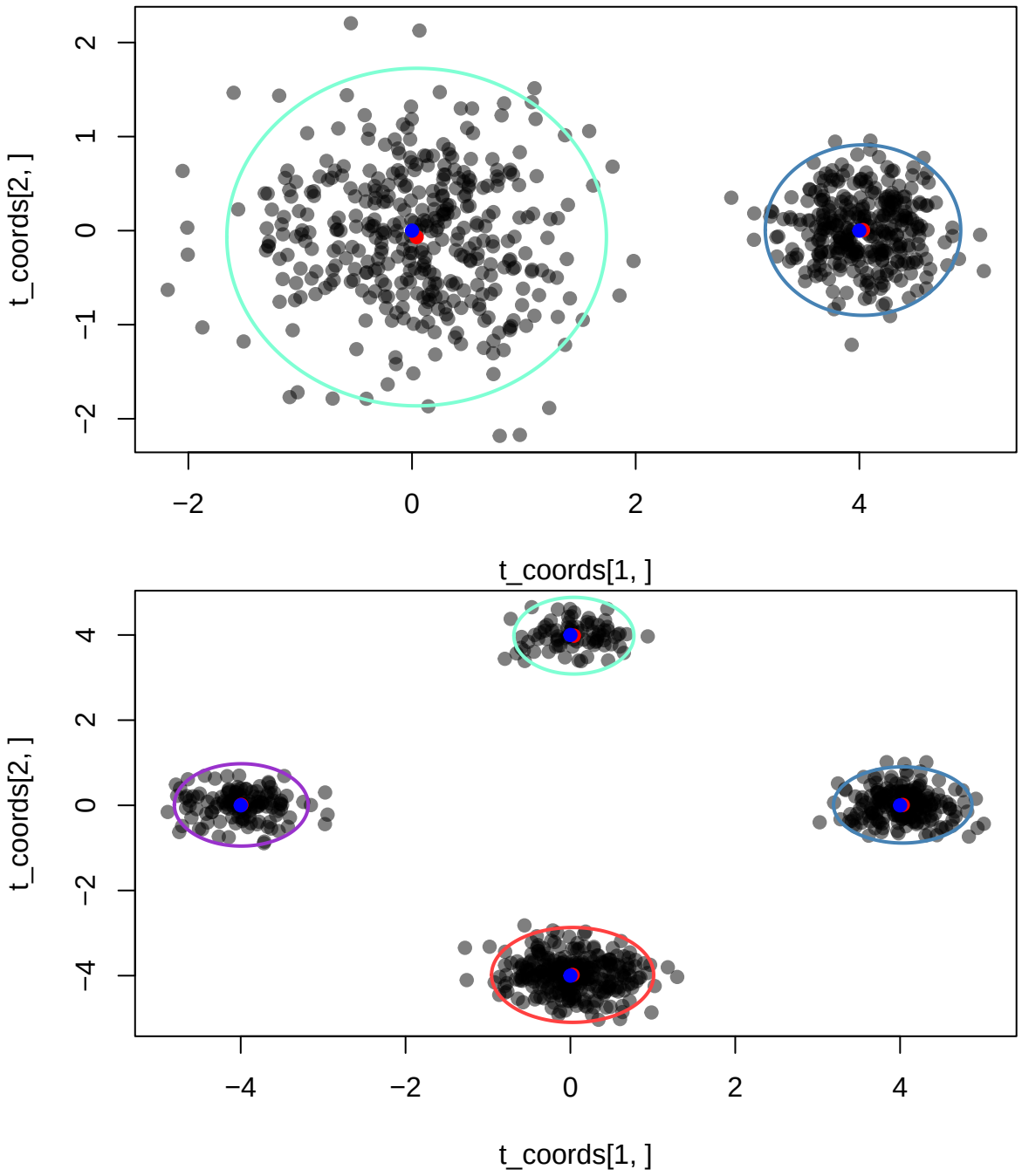

t_coords[2,]
2
0
−2
−4
−6
−6
−4
−2
0
t_coords[1,]

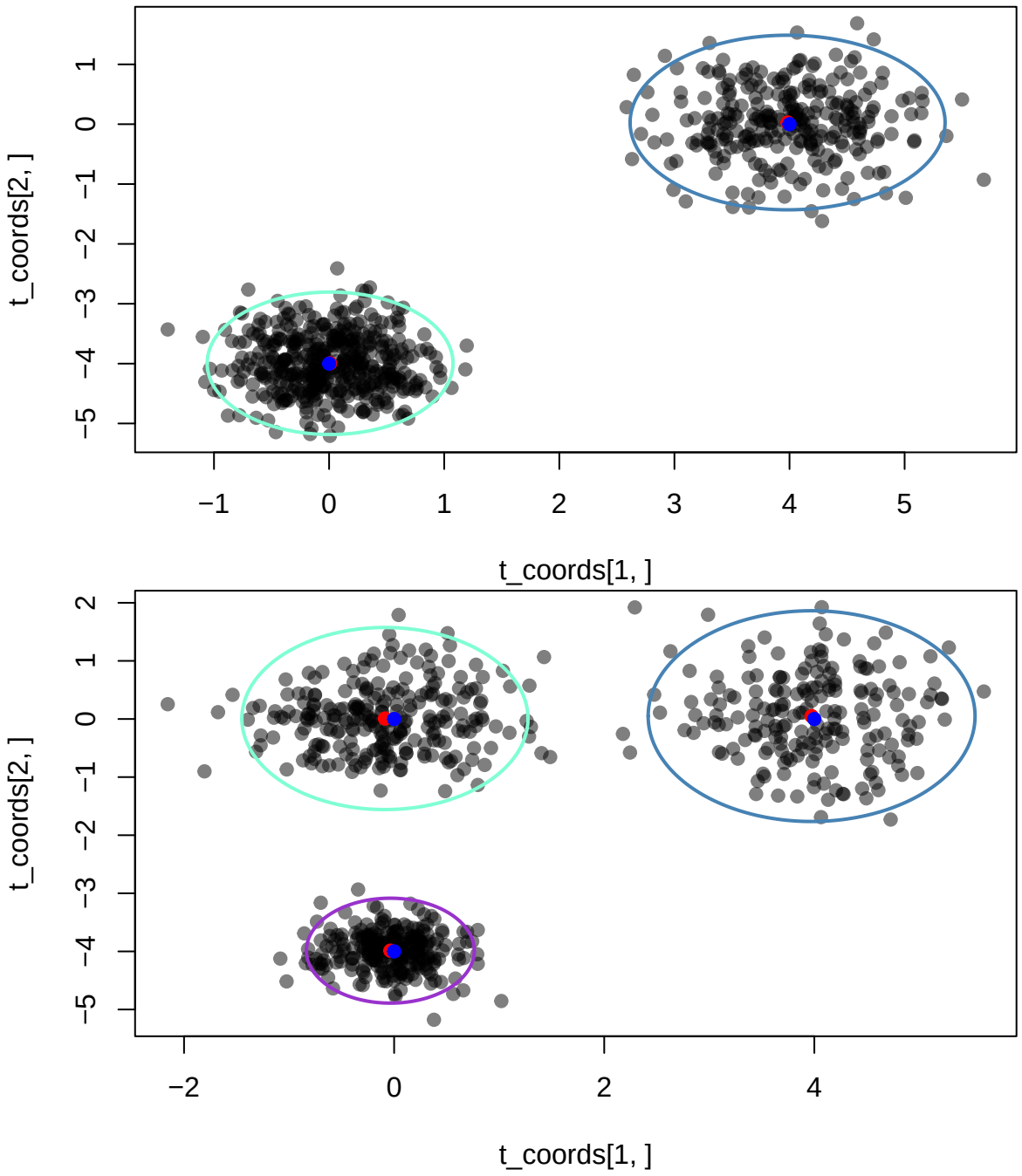

t_coords[2,]
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
t_coords[1,]

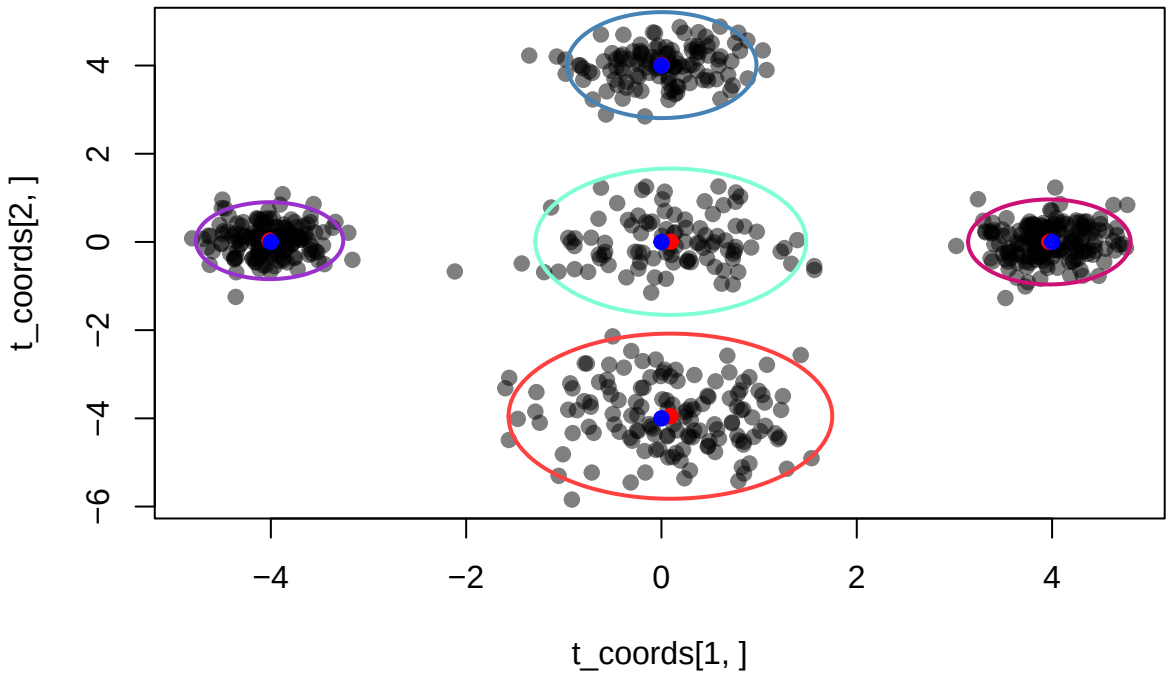

t_coords[2,]
6
4
2
0
−5
−4
−3
−2
−1
0
1
2
t_coords[1,]
t_coords[2,]
5
4
3
2
1
0
−1
−2
−4
−2
0
2
4
t_coords[1,]

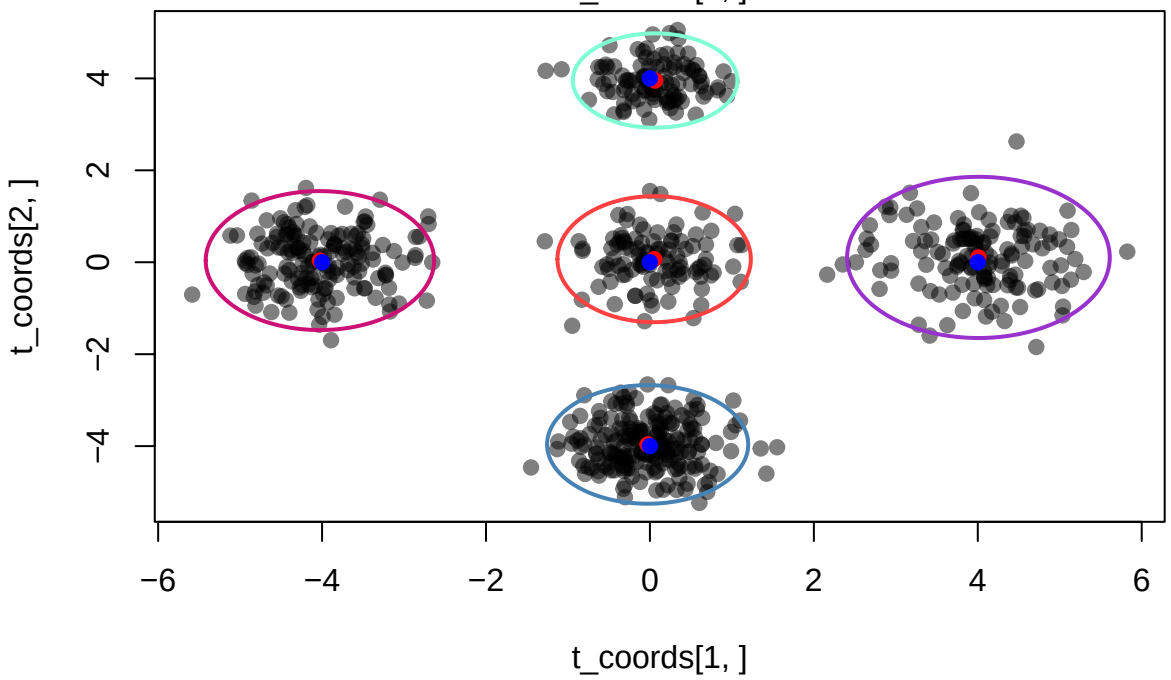

t_coords[2,]
4
2
0
−2
−4
−6
−4
−2
0
2
4
t_coords[1,]
t_coords[2,]
6
4
2
0
−2
−4
−2
0
2
4
6
t_coords[1,]

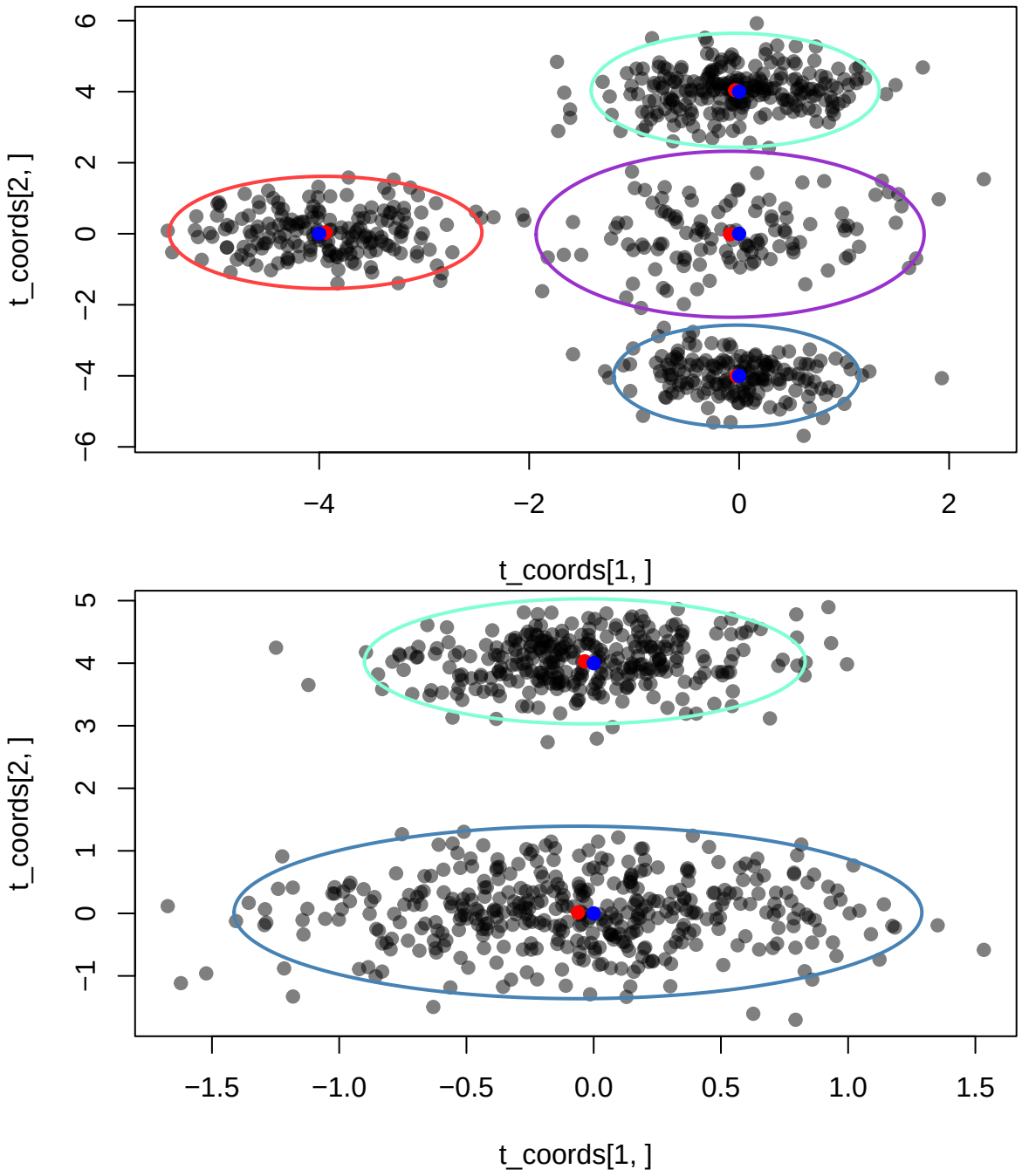

t_coords[2,]
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
t_coords[1,]
t_coords[2,]
4
2
0
−2
−4
−6
−1
0
1
2
3
4
5
t_coords[1,]

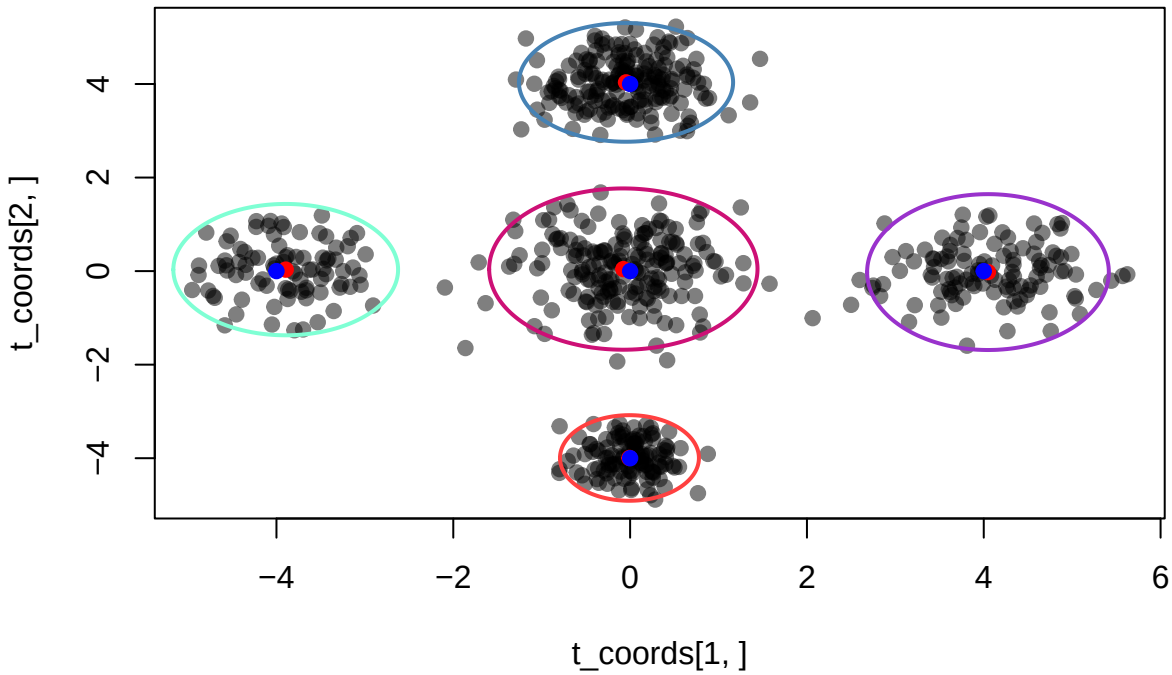

t_coords[2,]
2
1
0
−1
−2
−2
0
2
4
t_coords[1,]
t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
t_coords[1,]

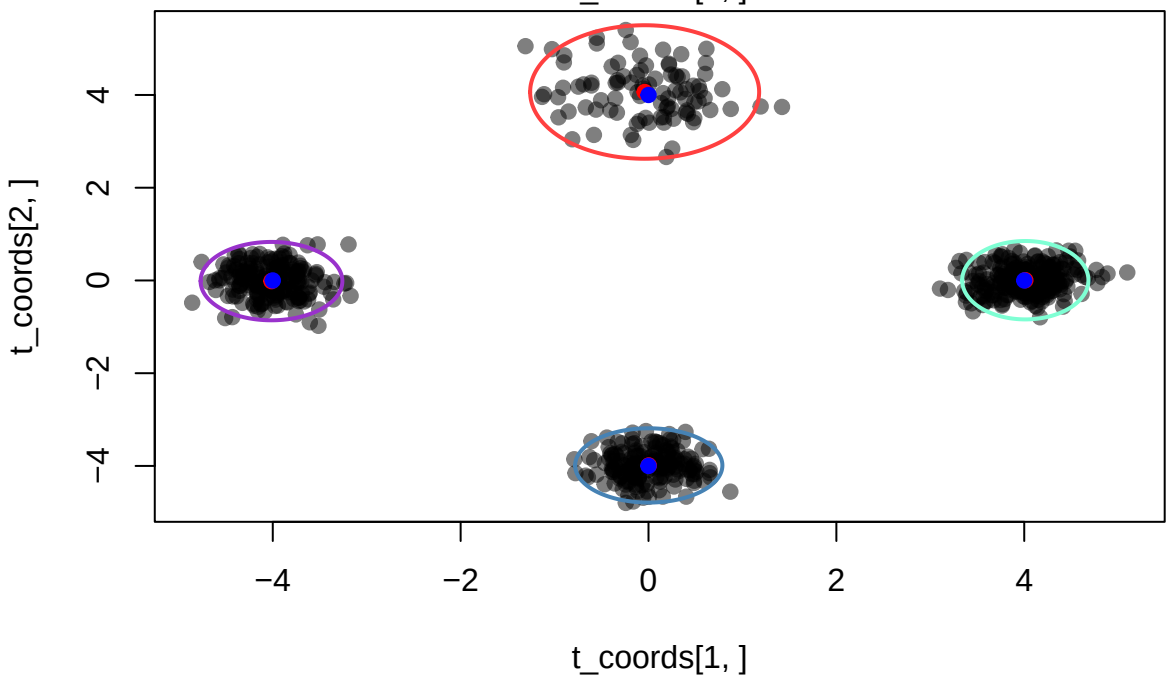

t_coords[2,]
t_coords[1,]
t_coords[2,]
t_coords[1,]

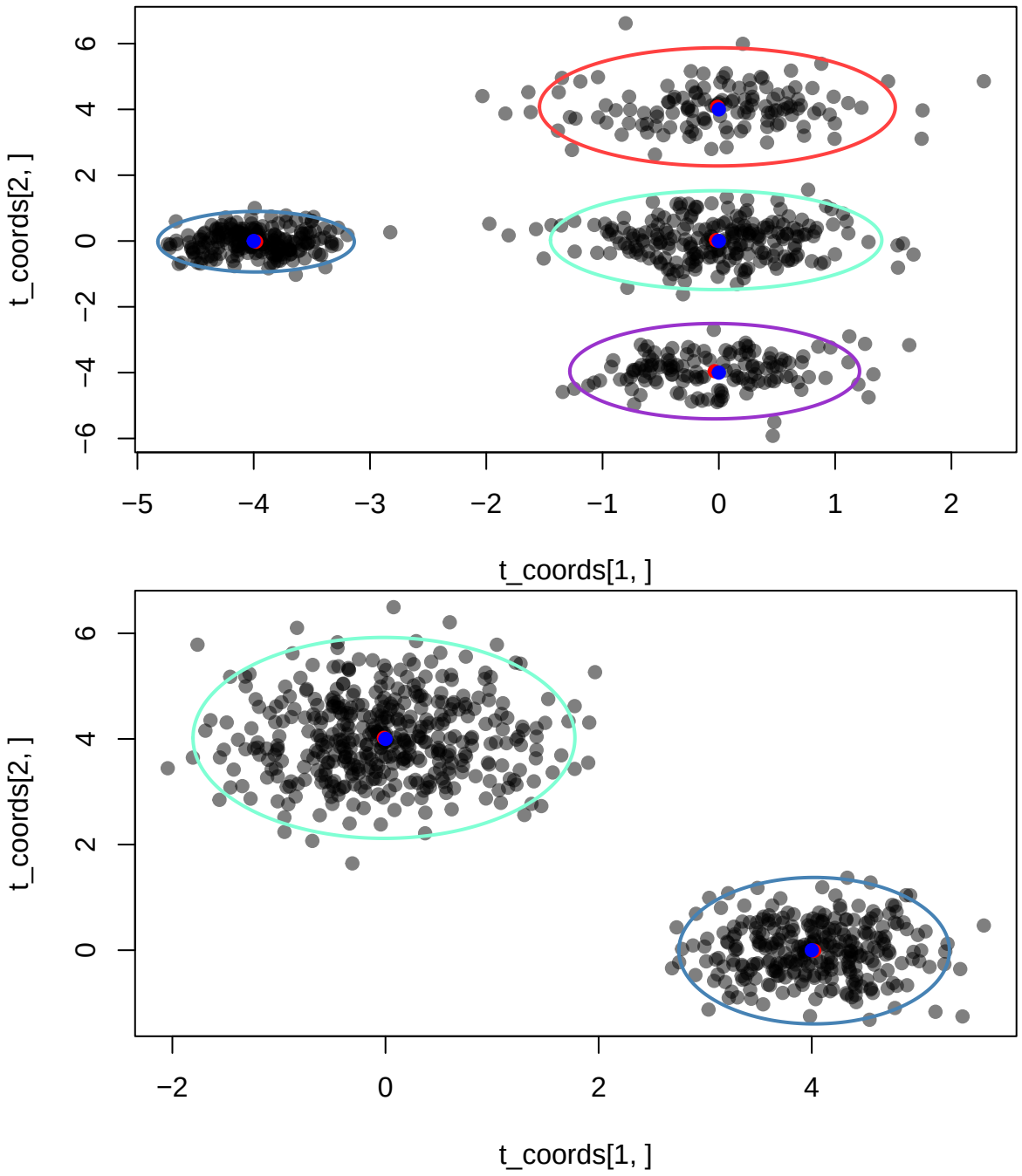

t_coords[2,]
4
2
0
−2
−4
−6
−4
−2
0
2
4
t_coords[1,]

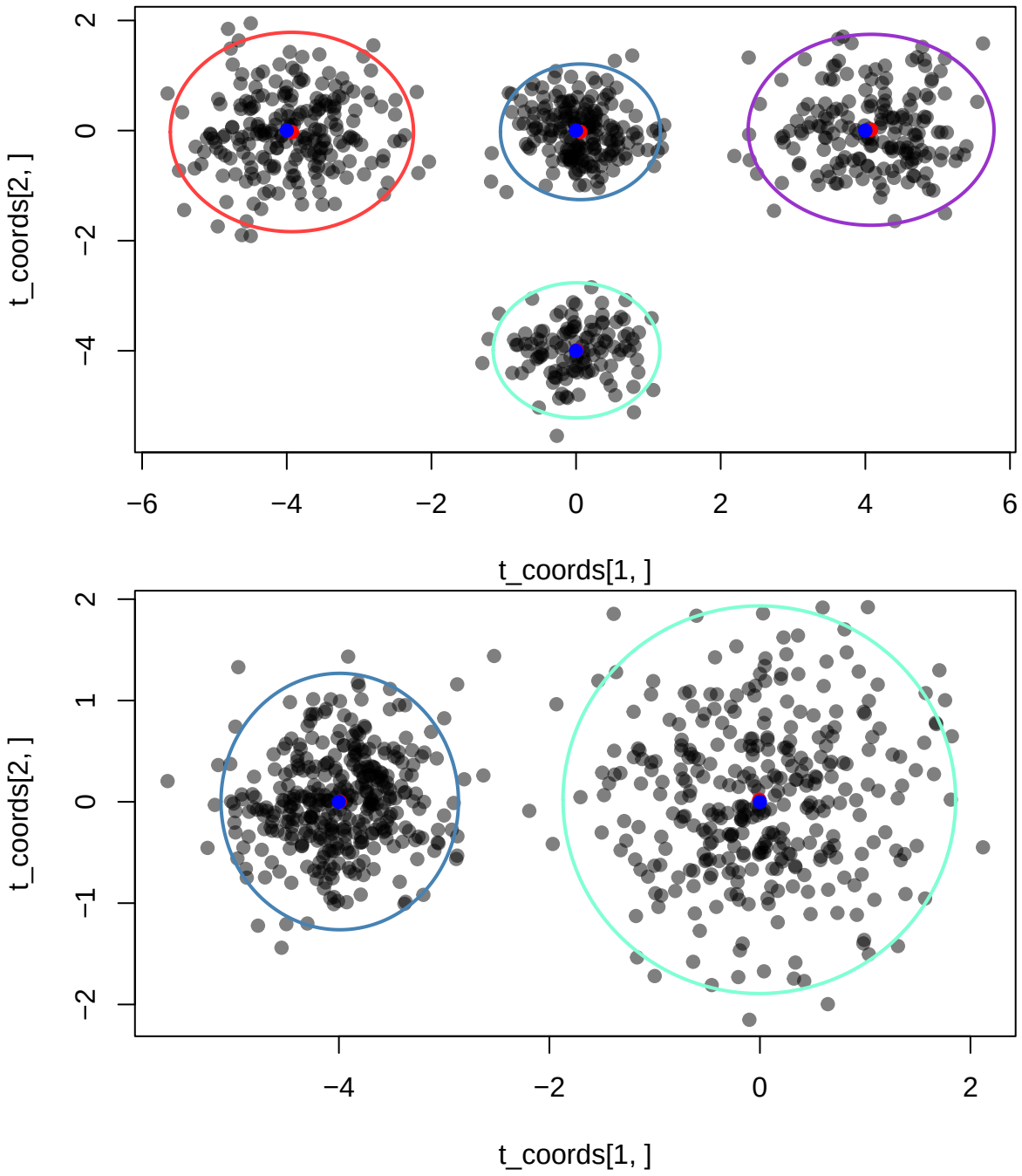

t_coords[2,]
4
2
0
−2
−4
−6
−4
−2
0
2
4
6
t_coords[1,]

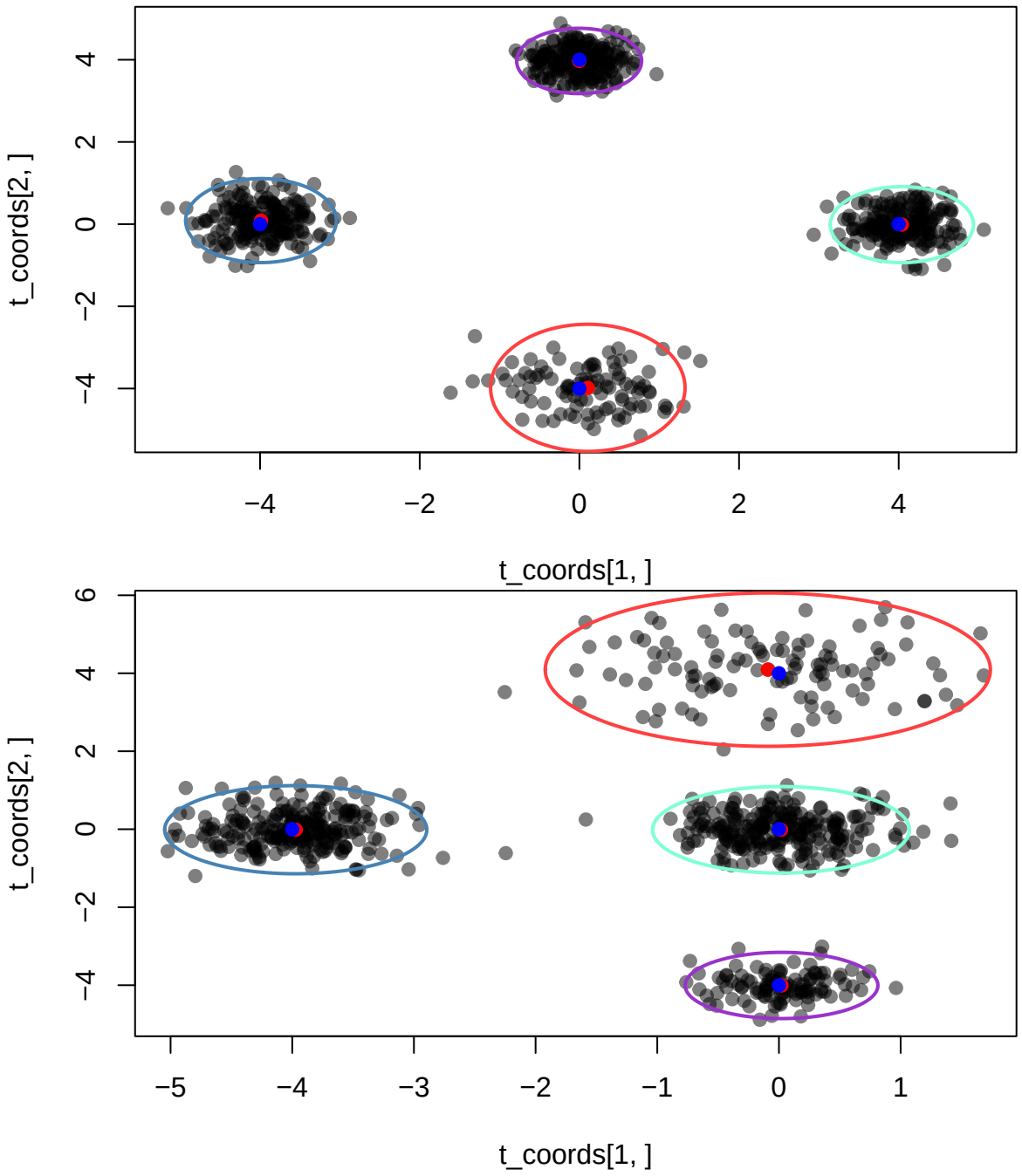

t_coords[2,]
6
4
2
0
−2
−4
−6
−4
−2
0
2
t_coords[1,]
t_coords[2,]
5
4
3
2
1
0
−1
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
t_coords[1,]

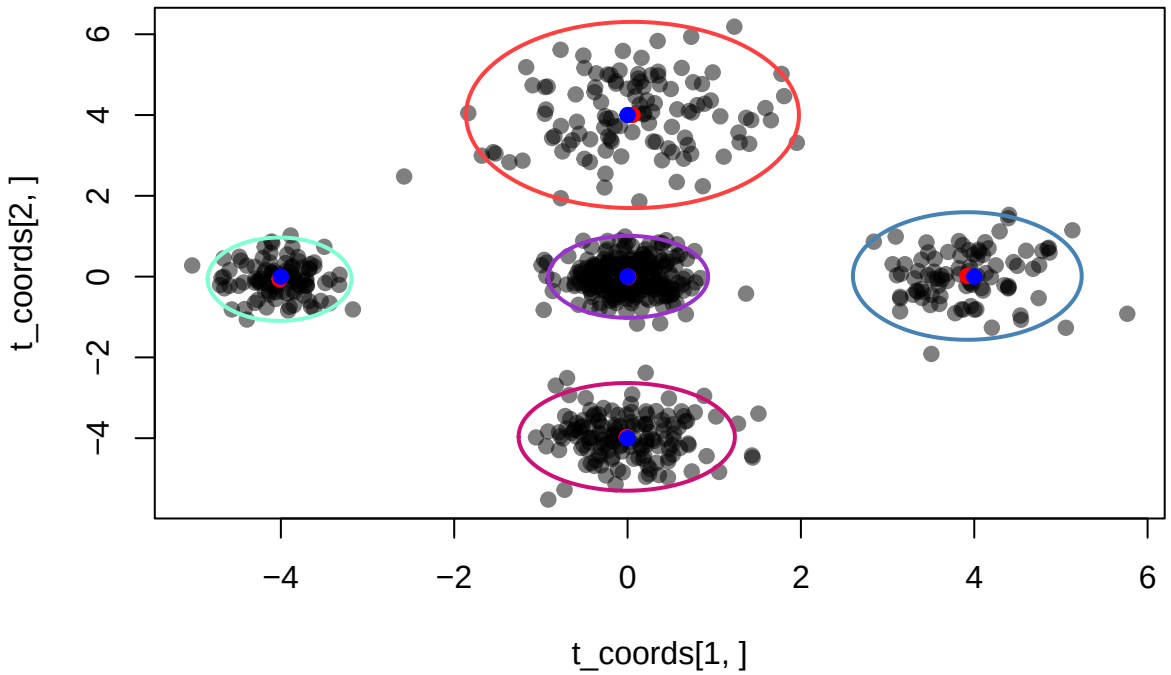

t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
6
t_coords[1,]

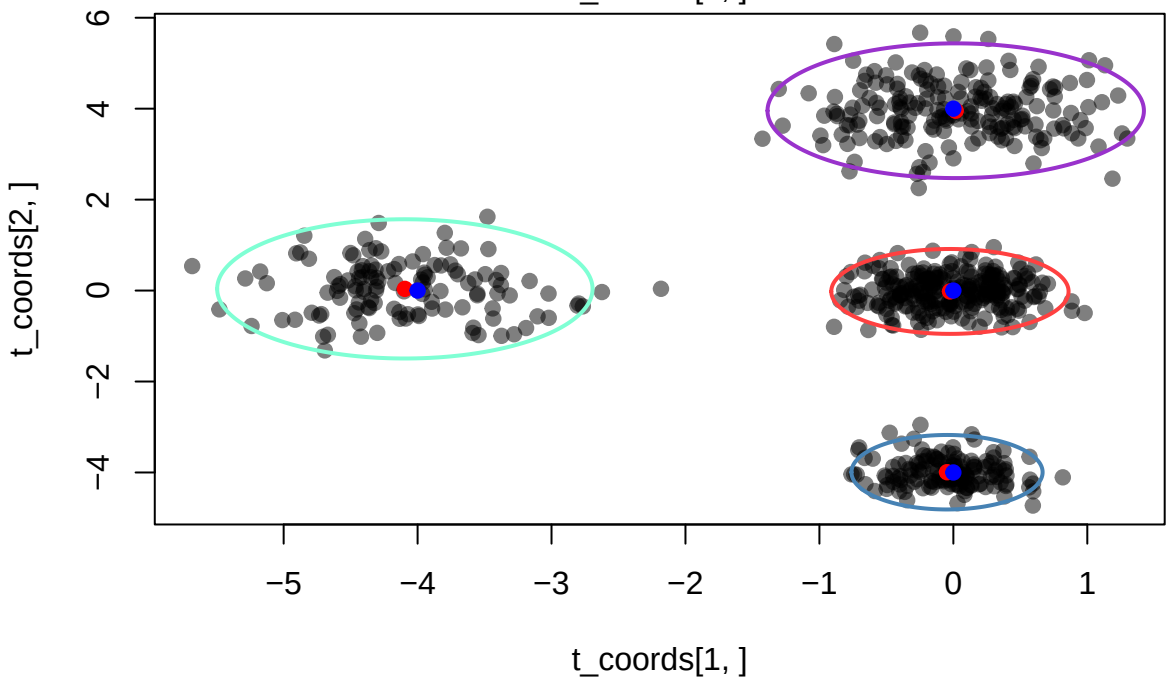

t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
t_coords[1,]

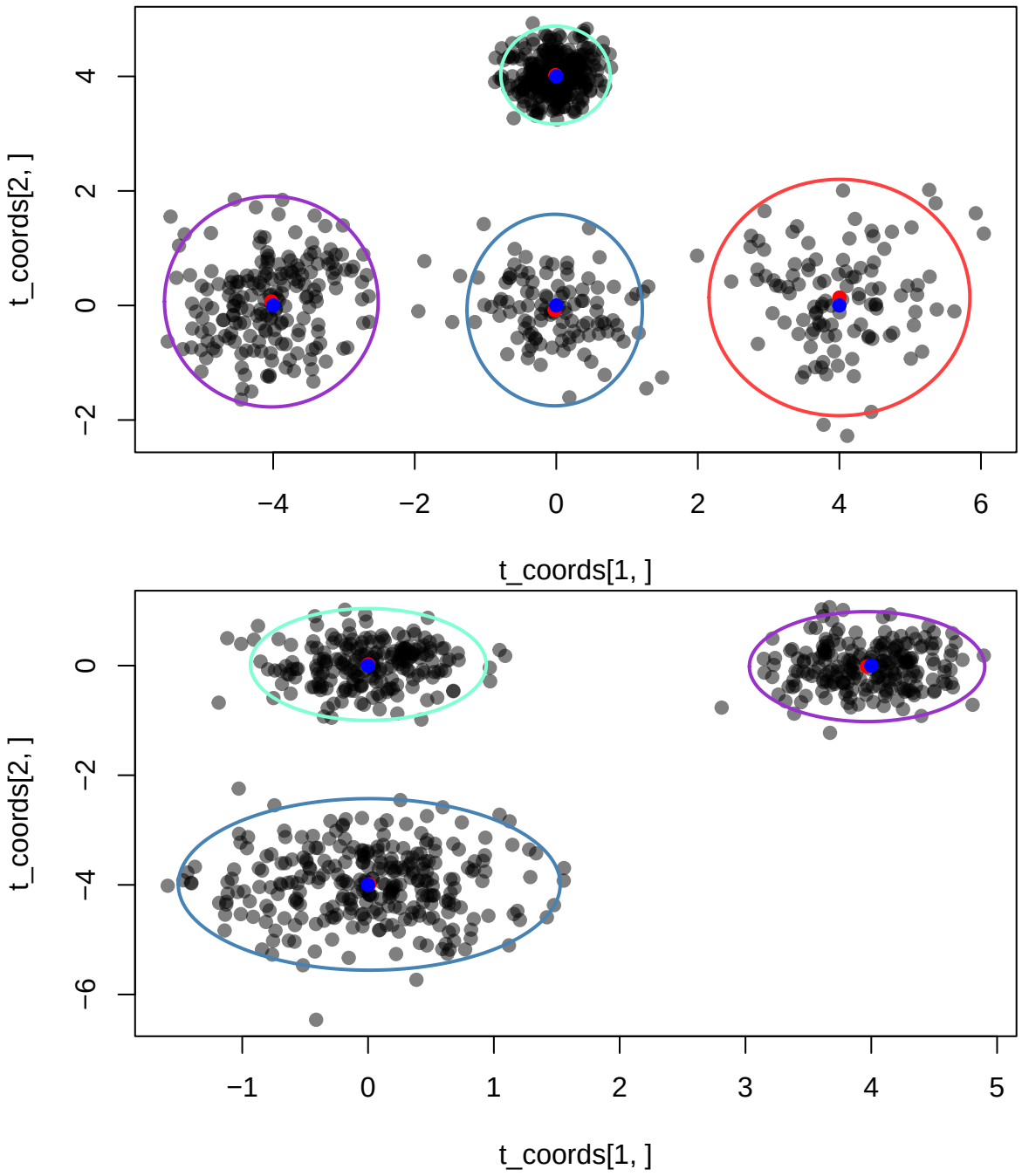

t_coords[2,]
6
4
2
0
−2
−4
−6
−5
−4
−3
−2
−1
0
1
2
t_coords[1,]
t_coords[2,]
6
4
2
0
−2
0
2
4
t_coords[1,]

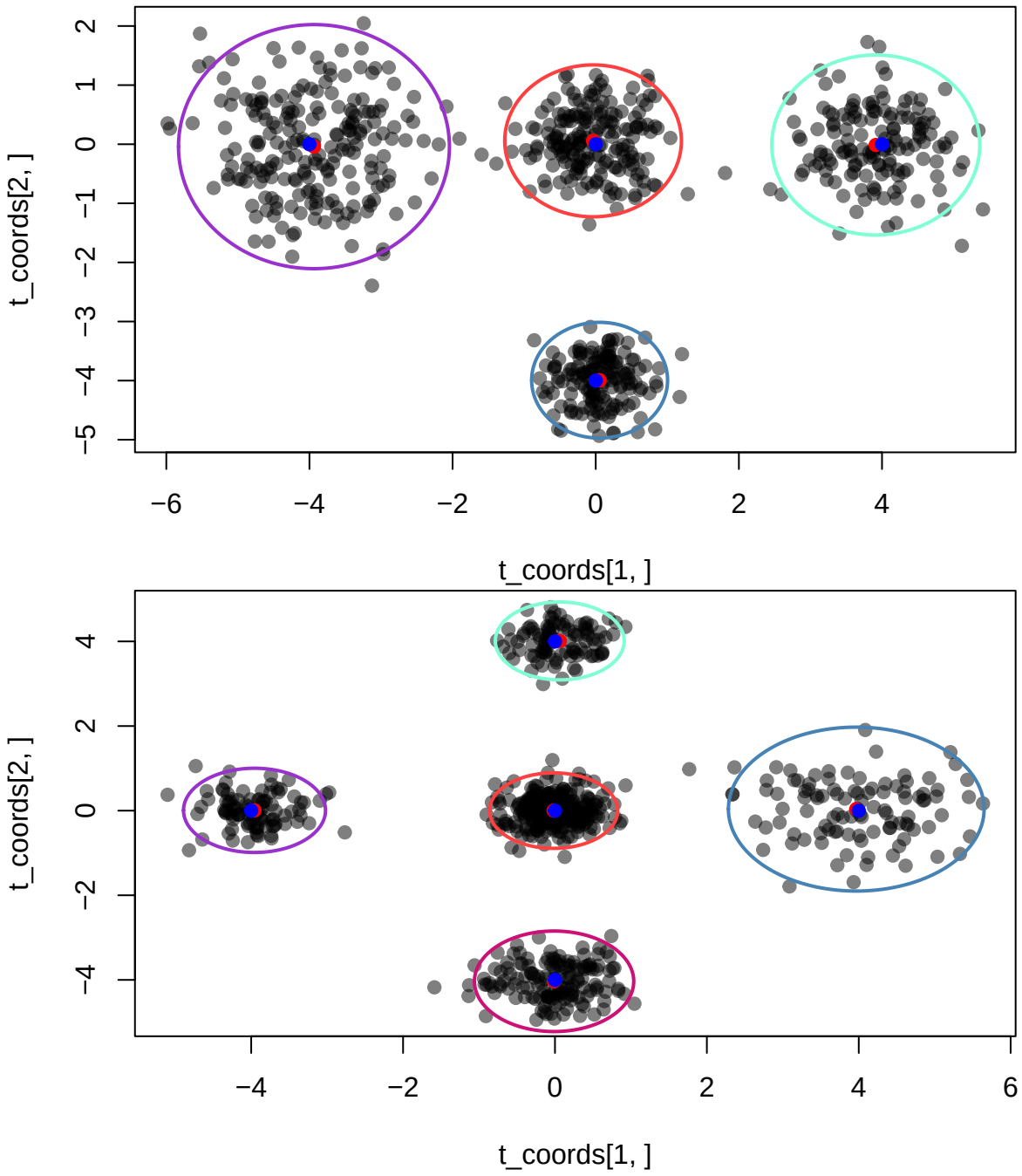

t_coords[2,]
2
0
−2
−4
−6
−4
−2
0
2
4
6
t_coords[1,]
t_coords[2,]
2
1
0
−1
−2
−4
−2
0
2
t_coords[1,]


t_coords[2,]
4
2
0
−2
−4
−4
−2
0
2
4
t_coords[1,]
t_coords[2,]
6
4
2
0
−2
−4
−5
−4
−3
−2
−1
0
1
t_coords[1,]


t_coords[2,]
6
4
2
0
−2
−4
−4
−2
0
2
4
6
t_coords[1,]


t_coords[2,]
6
4
2
0
−2
−4
−5
−4
−3
−2
−1
0
1
t_coords[1,]


t_coords[2,]
4
2
0
−2
−4
−2
0
2
4
6
t_coords[1,]
t_coords[2,]
0
−2
−4
−6
−1
0
1
2
3
4
5
t_coords[1,]




```
print(wrong_cluster_count)
```

```
## [1] 12  2  1  0
```

results

performance conclusion:

a. If robust is predicting correct number of clusters, it's accuracy is the same as original EM with correct cluster guess

b. takes less iterations but more time

c. When there are not that many data points, robust EM tend to get number of clusters wrong more easily.

d. When there are not that many data points, robust EM performs better in accuracy if it finds the correct number of scores