Project

2025-03-30

Normal Mixture model tests

helper functions

```r
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
```

```r
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) {
```

```r
    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)
```

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

}
```

2d sample with 3 mixtures

```r
set.seed(1)
n <- 500     # Number of data points
mu_list <- list(c(0,0), c(0,5), c(5,0))
dim = 2 # dim: # of dimension of the sample, mu = c(0,0) should be dim 2
k = 3 # k: number of components
prob = rep(1/k, k)
beta = 1
I2_list = lapply(1:3, function(x) diag(1, 2))

#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")
```

```r

C = 3 #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=1e3)
EM_res

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

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

correct guess for centers

```r

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)

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

robust

another 2d sample with 3 mixtures

```
set.seed(1)
n <- 800     # Number of data points
mu_list <- list(c(2,2), c(-4,-4), c(-4,-4))
dim = 2 # dim: # of dimension of the sample, mu = c(0,0) should be dim 2
k = 3 # k: number of components
prob = c(1,1,0.5)
I2_list = list(diag(c(1,1)), diag(c(6,2)), diag(c(1/5,1/5)))

#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 = 3 #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)
#EM_res

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

eval_EM(res = EM_res, mu_list = mu_list, dim = dim, I2_list=I2_list)
```

correct guess for centers

```
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)

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

robust

2d sample with 5 mixtures

```
set.seed(1)
n <- 500     # Number of data points
mu_list <- list(c(0,0), c(0,5), c(5,0), c(-5,0), c(0,-5))
```

```r
dim = 2
k = 5
prob = rep(1/k, k)
I2_list = lapply(1:k, function(x) diag(1,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")
```

```r
C = 5 #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=1e3)
EM_res

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

eval_EM(res = EM_res, mu_list = mu_list, dim = dim, I2_list=I2_list)
```

correct guess for centers

```r
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)

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

robust

3d sample with 3 mixtures

```r
library(plotly)
set.seed(1)
n <- 200      # Number of data points
mu_list <- list(
  c(0, 0, 0),
  c(5, 0, 0),
  c(0, 5, 0)
```

```
) # -> dim = 3, k = 3
dim = 3
k = 3
prob = rep(1/k, k)
I2_list = list(diag(c(1,1,1)), diag(c(1,1,1)), diag(c(1,1,1)))

#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 = data.frame(cbind(coords, group))
colnames(dat) = c("X", "Y", "Z", "Group")

# Plot the generated data
fig = plot_ly(dat, x = ~X, y = ~Y, z = ~Z,
             type = "scatter3d", mode = "markers",
              color = ~Group, size = 1)
fig

ggplot(data = dat, aes(x=X, y=Y, colour = factor(Group))) +
  geom_point()
```

```
C = 3 #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=1e3)
#EM_res

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

eval_EM(res = EM_res, mu_list = mu_list, dim = dim, I2_list=I2_list)
```

correct guess for centers

```
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)

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

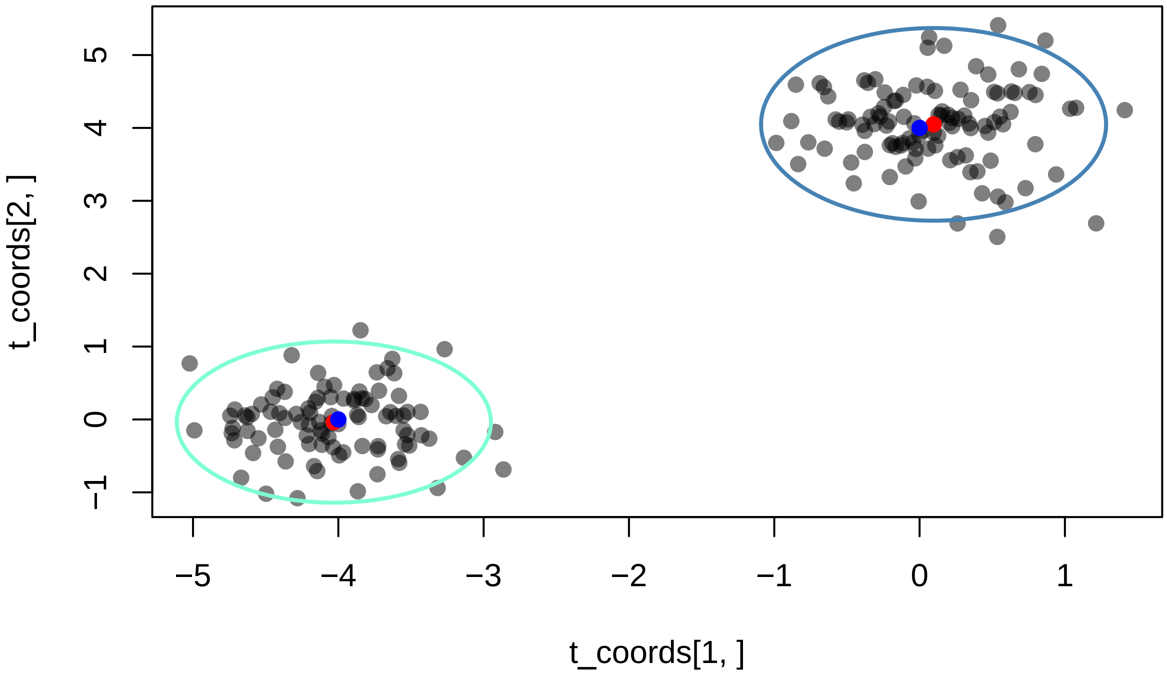
robust

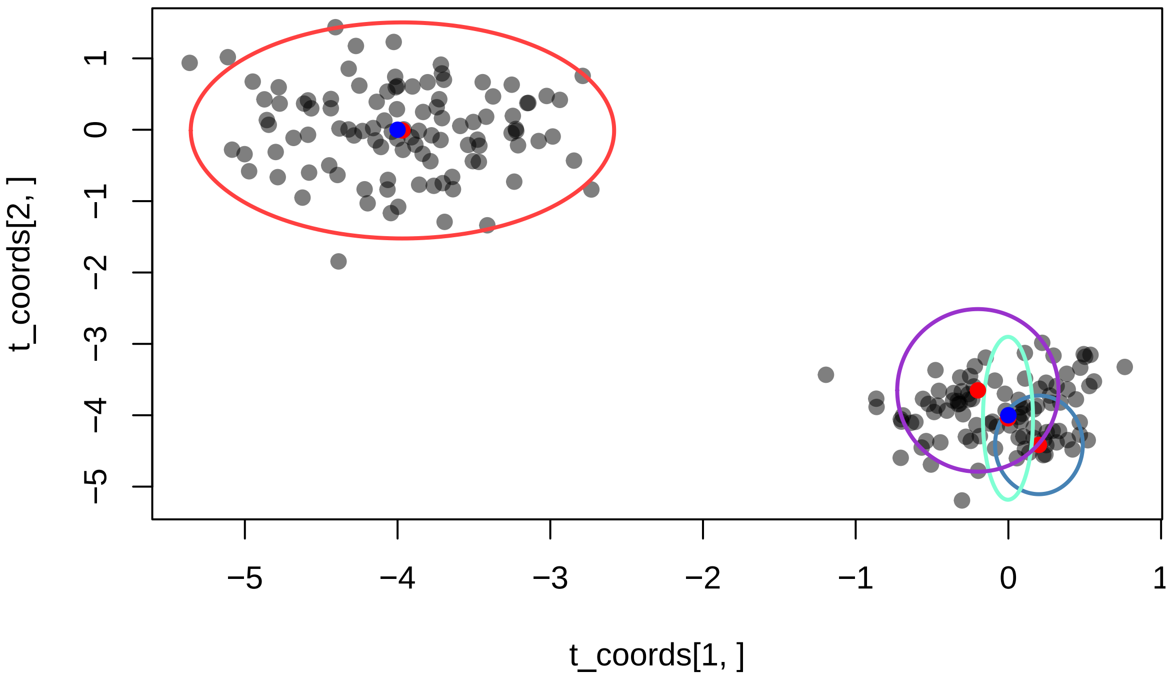
varing centers

```
set.seed(3)
m = 40 # NUMBER OF TESTS
```

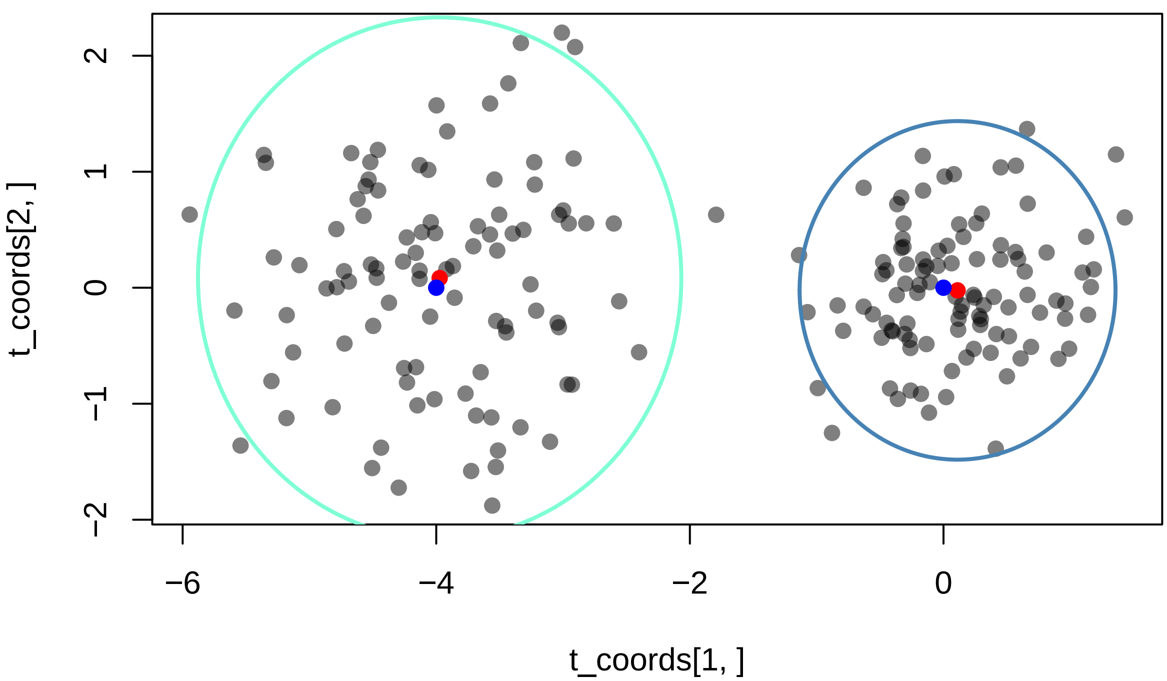
```r
n <- 200     # Number of data points
dim = 2

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)
```

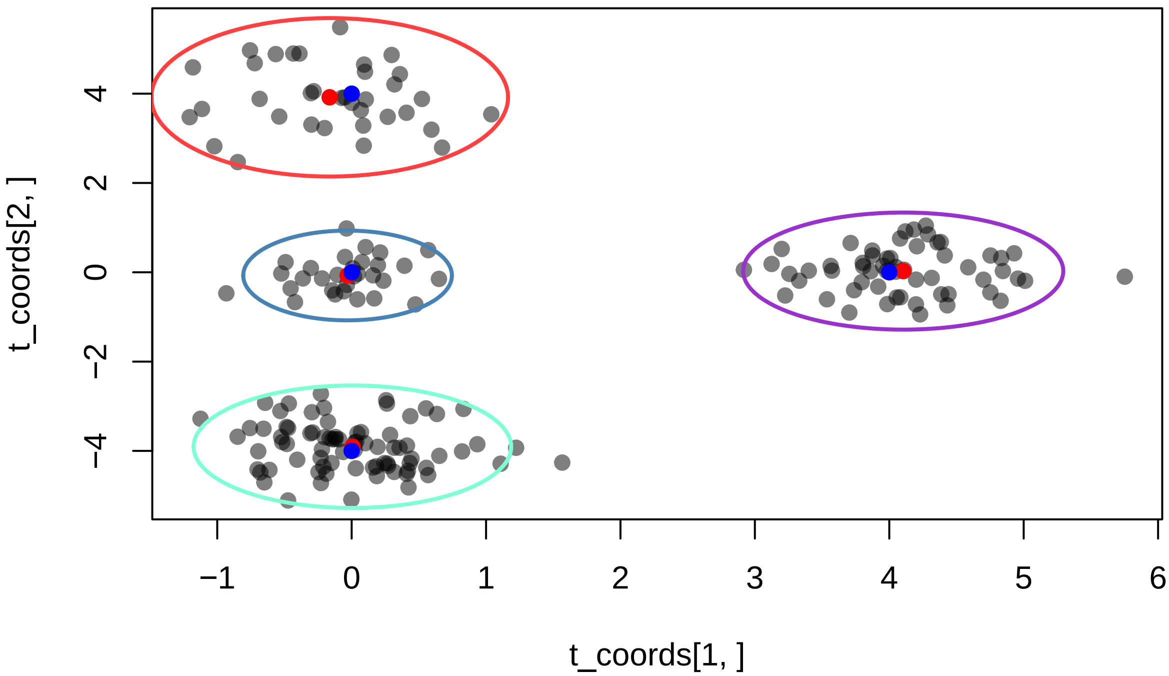
```
  robust_results = eval_EM(res = EMR_res, mu_list = mu_list, dim = dim, I2_list=I2_list)

  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
  }

}
```



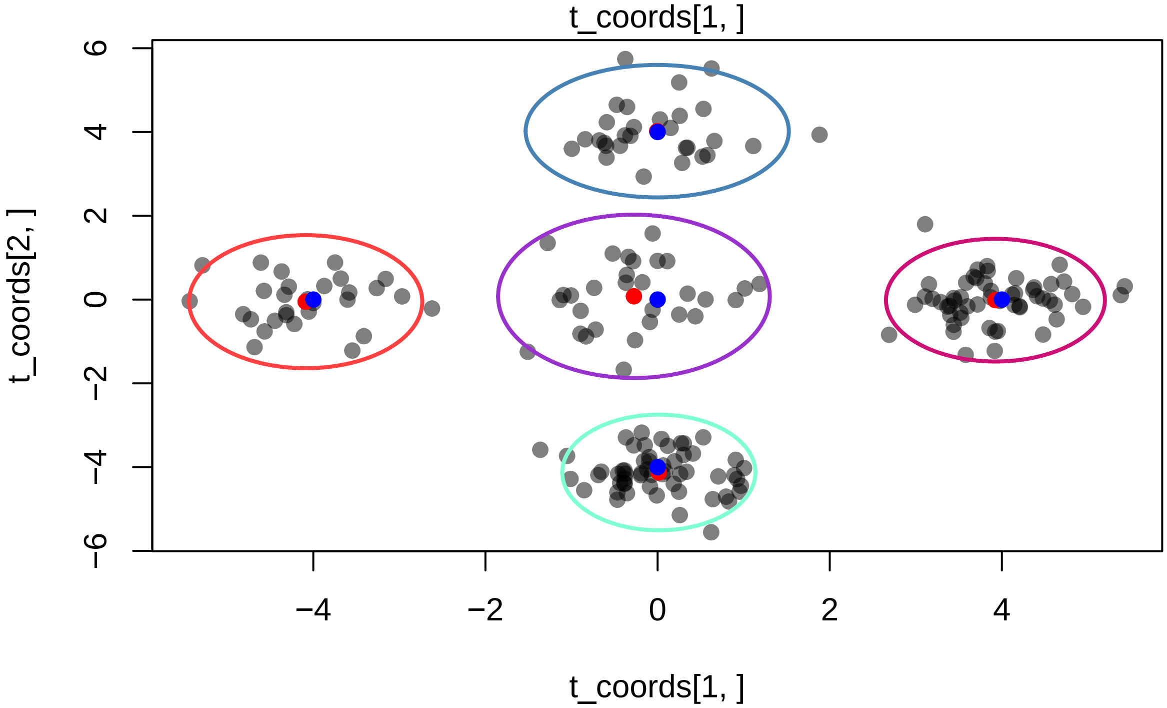



```
## predicted wrong number of clusters
```

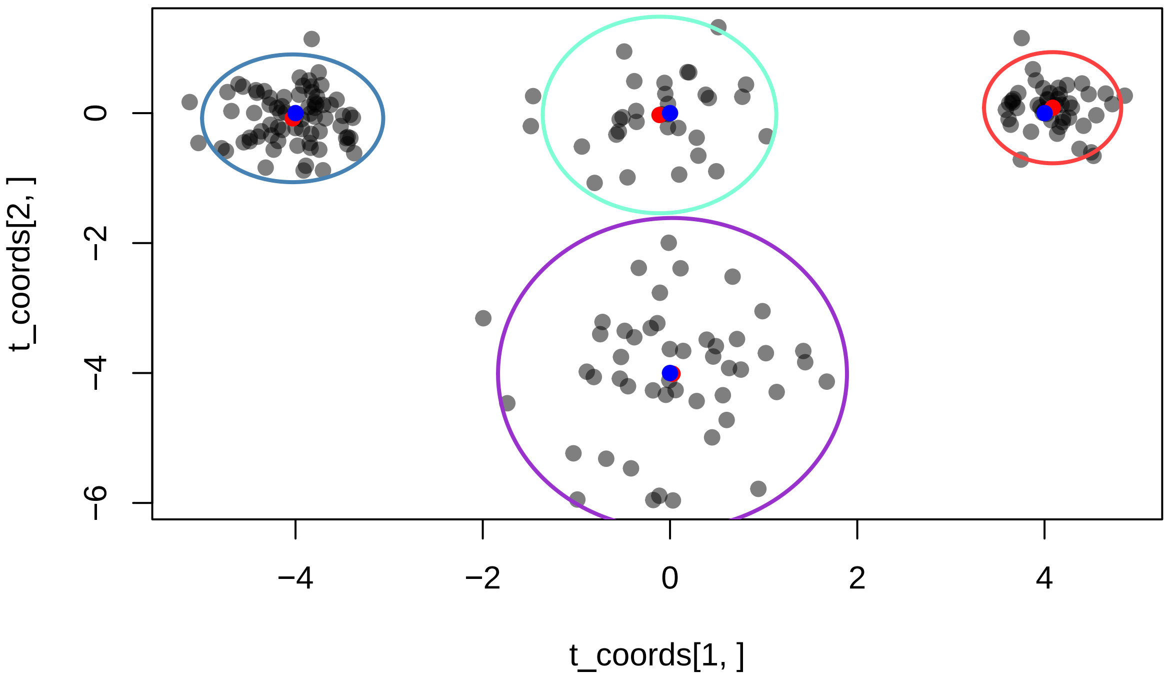




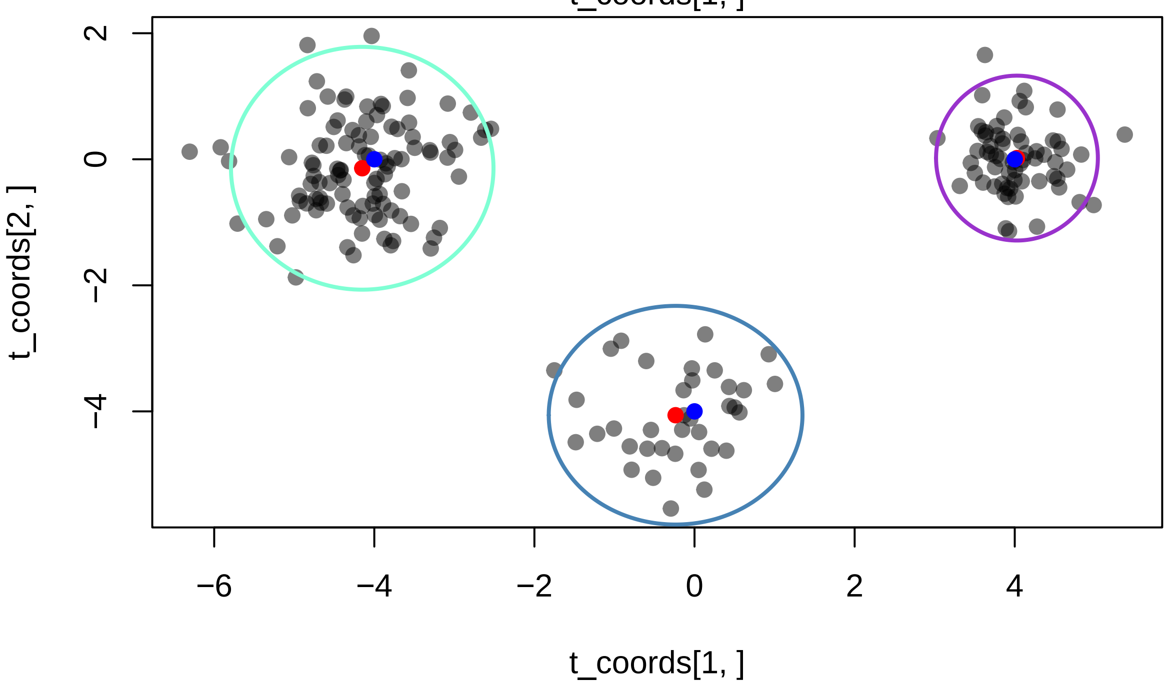
t_coords[2,]
t_coords[1,]



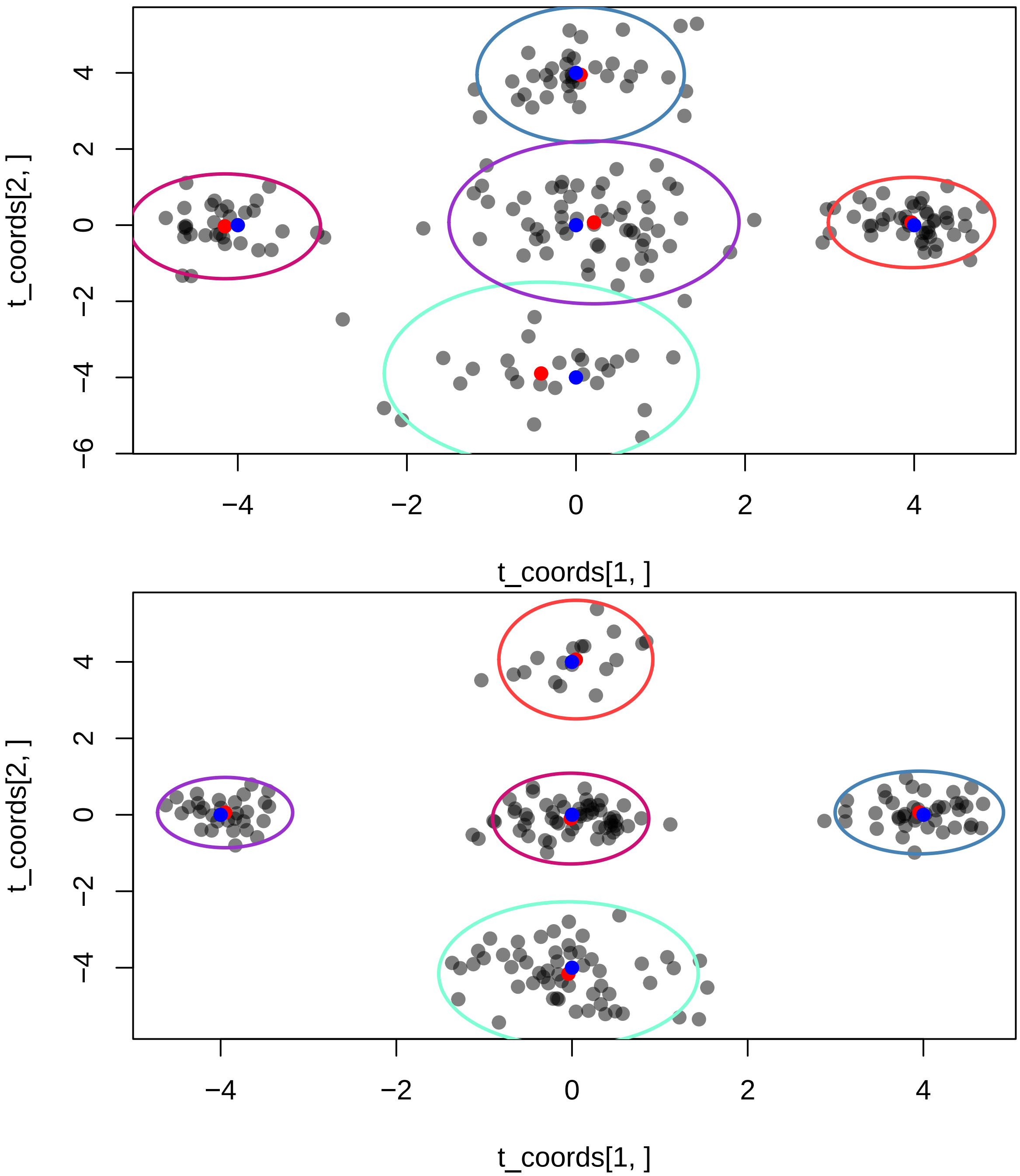
t_coords[2,]
t_coords[1,]

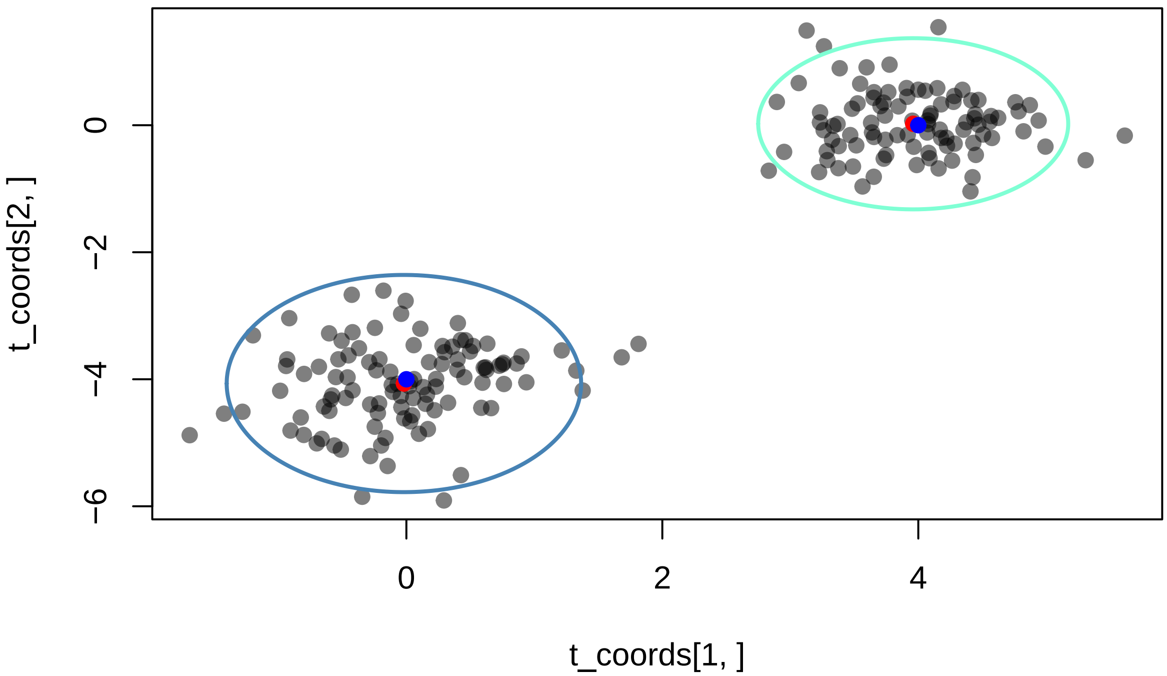


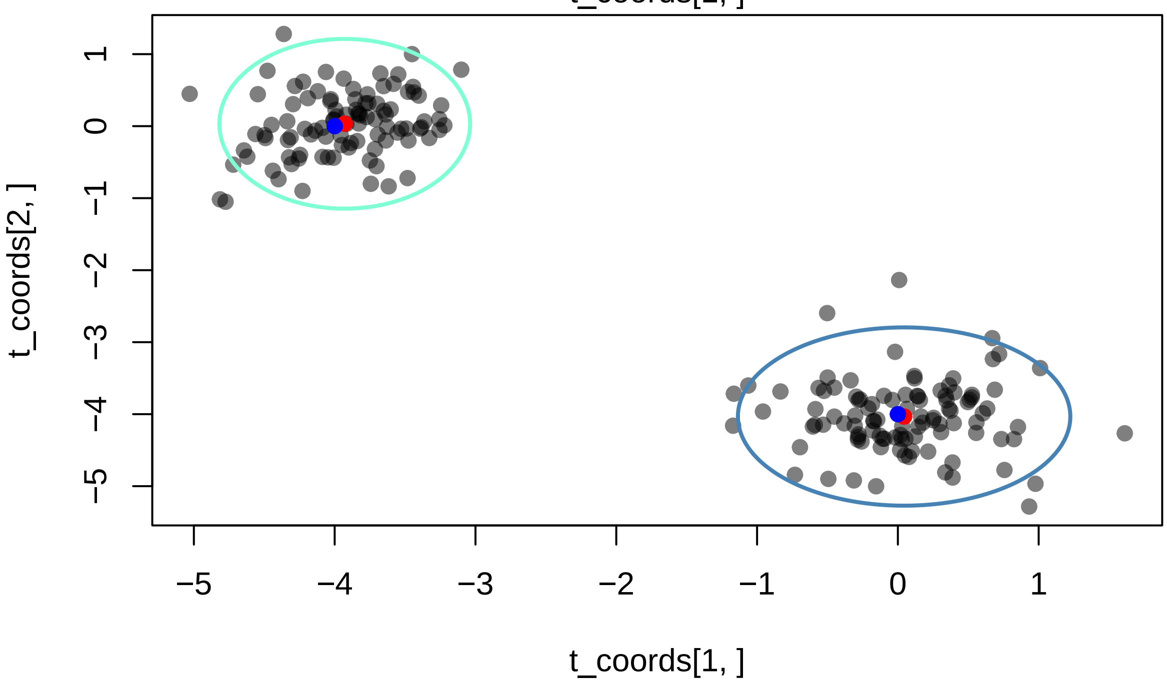
t_coords[2,]
0
−2
−4
−6
−4
−2
0
2
4
t_coords[1,]

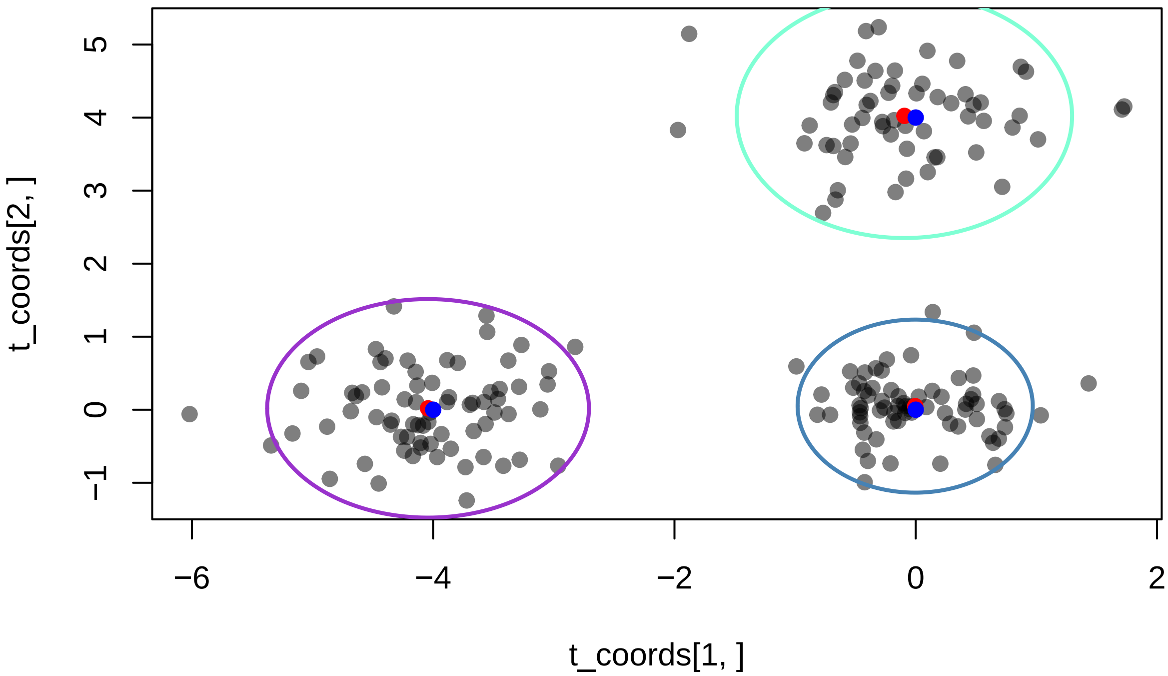


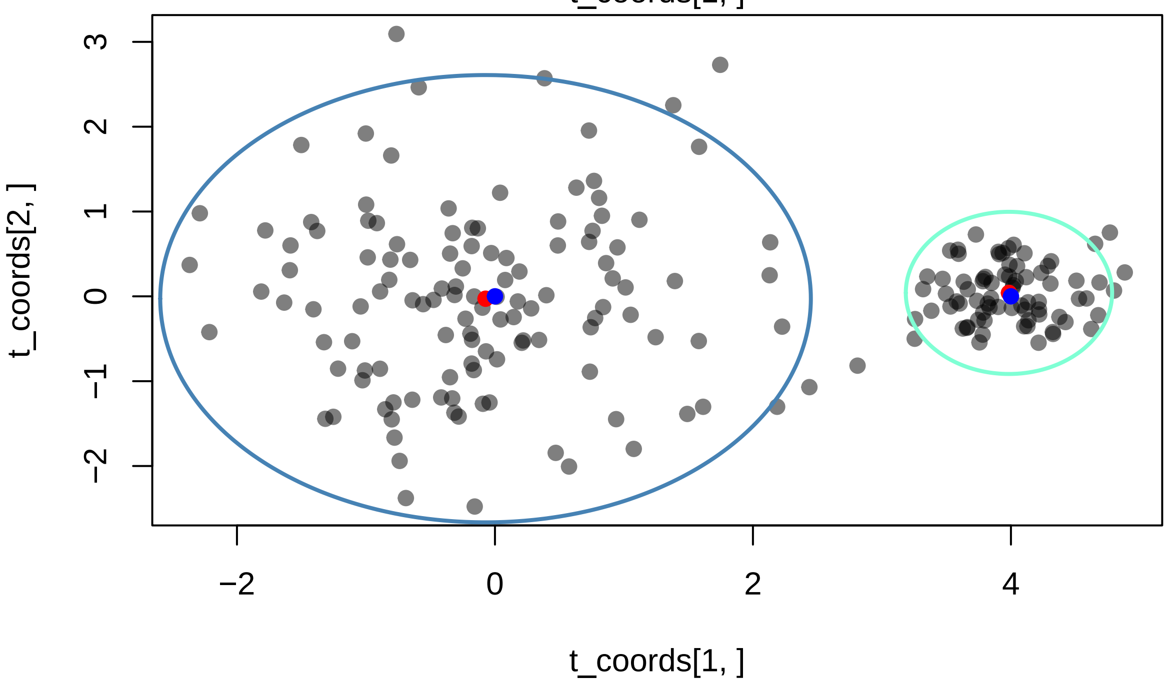
t_coords[2,]
2
0
−2
−4
−6
−4
−2
0
2
4
t_coords[1,]

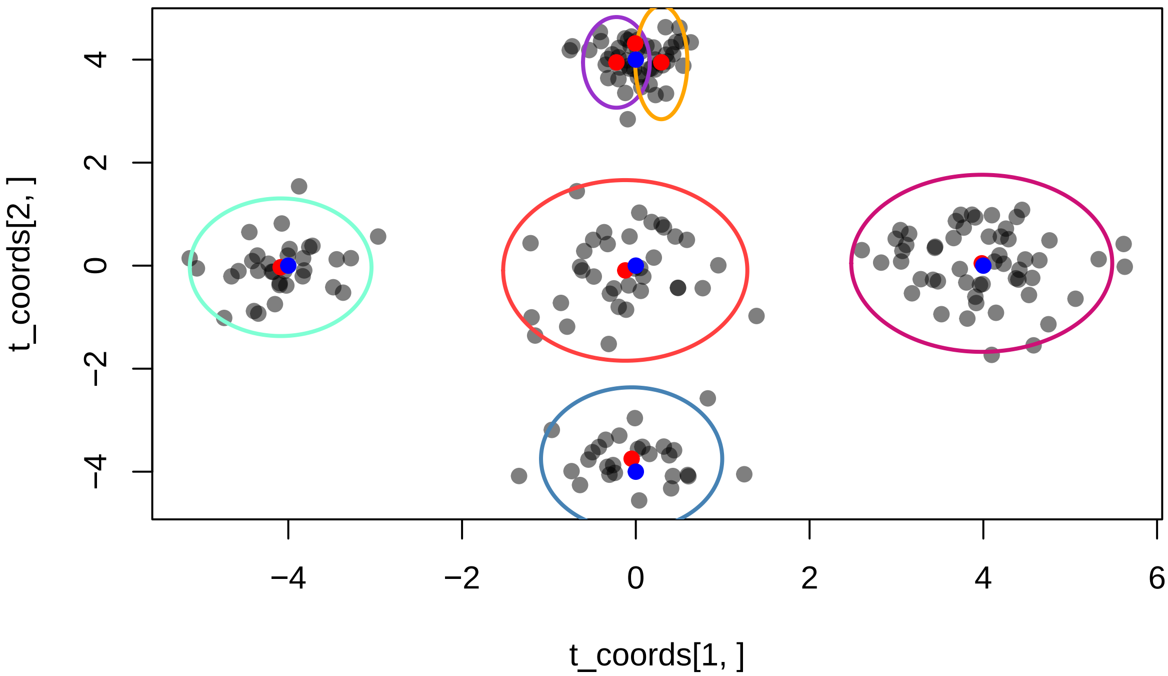

t_coords[2,]
t_coords[1,]
−6
−4
−2
0
2
4
t_coords[2,]
t_coords[1,]
−4
−2
0
2
4


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

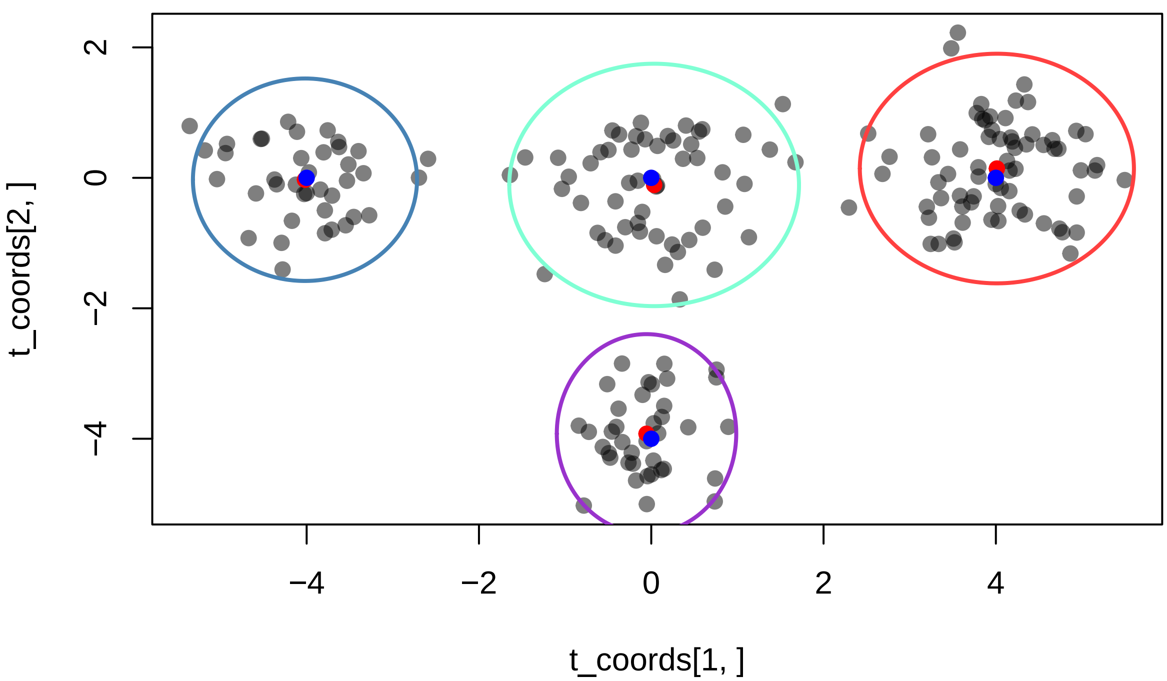

1
0
−1
−2
−3
−4
−5
t_coords[2,]
−5
−4
−3
−2
−1
0
1
t_coords[1,]

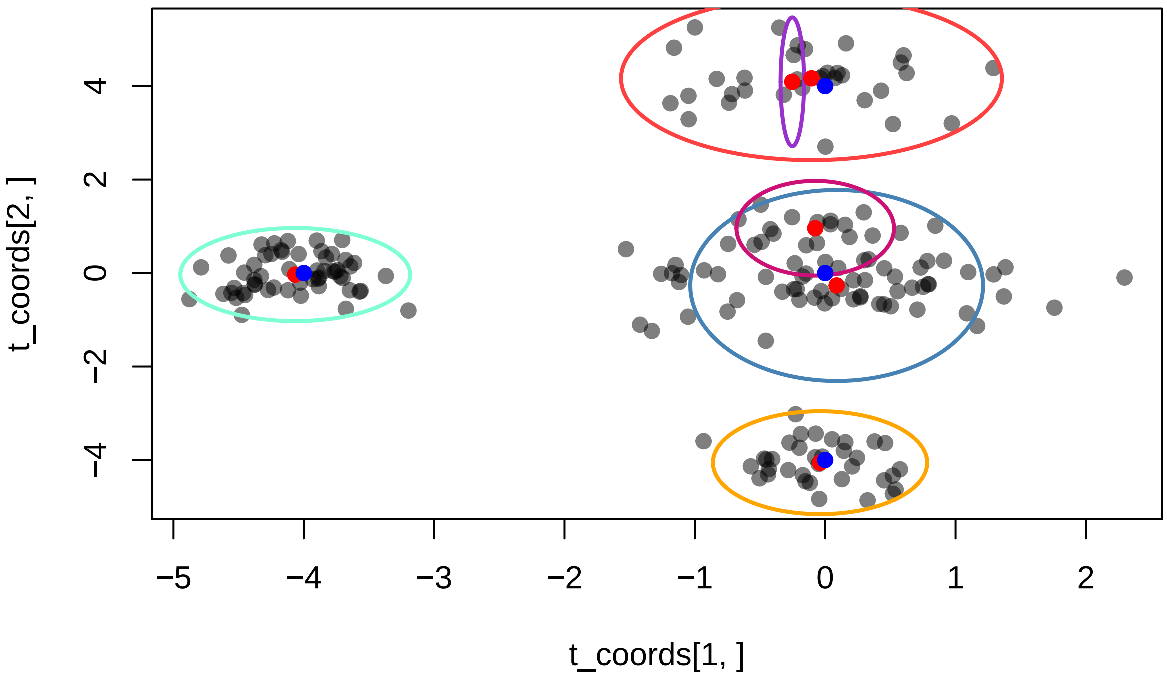

t_coords[2,]
t_coords[1,]
−6
−4
−2
0
2
−1
0
1
2
3
4
5


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

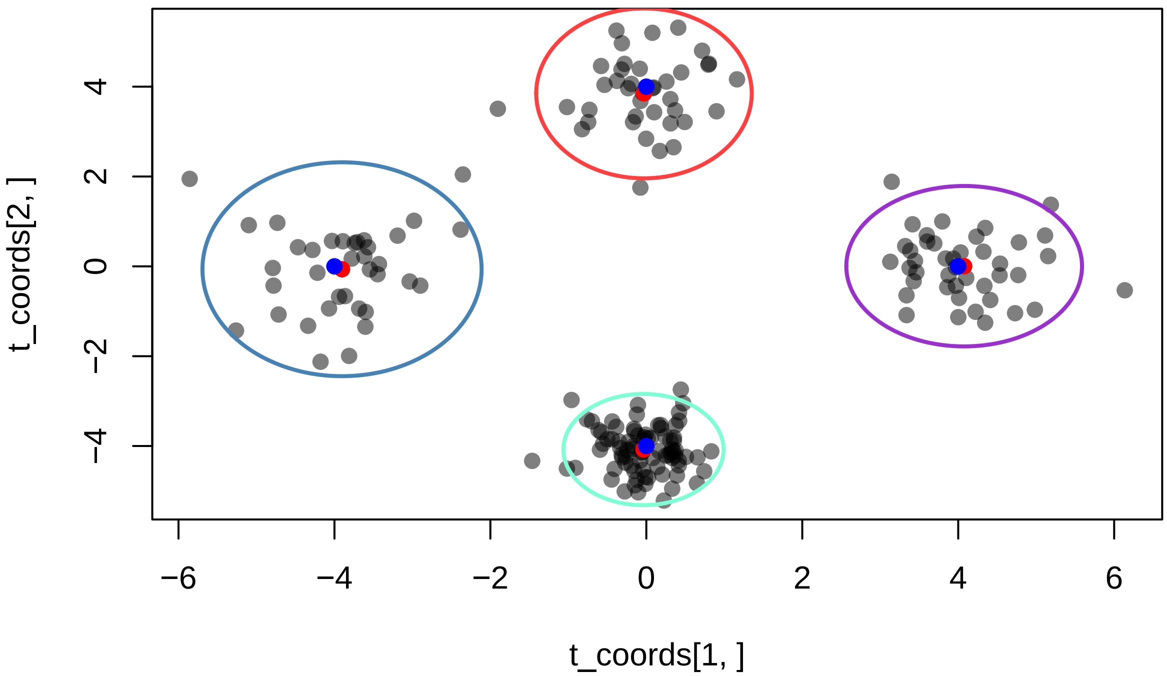


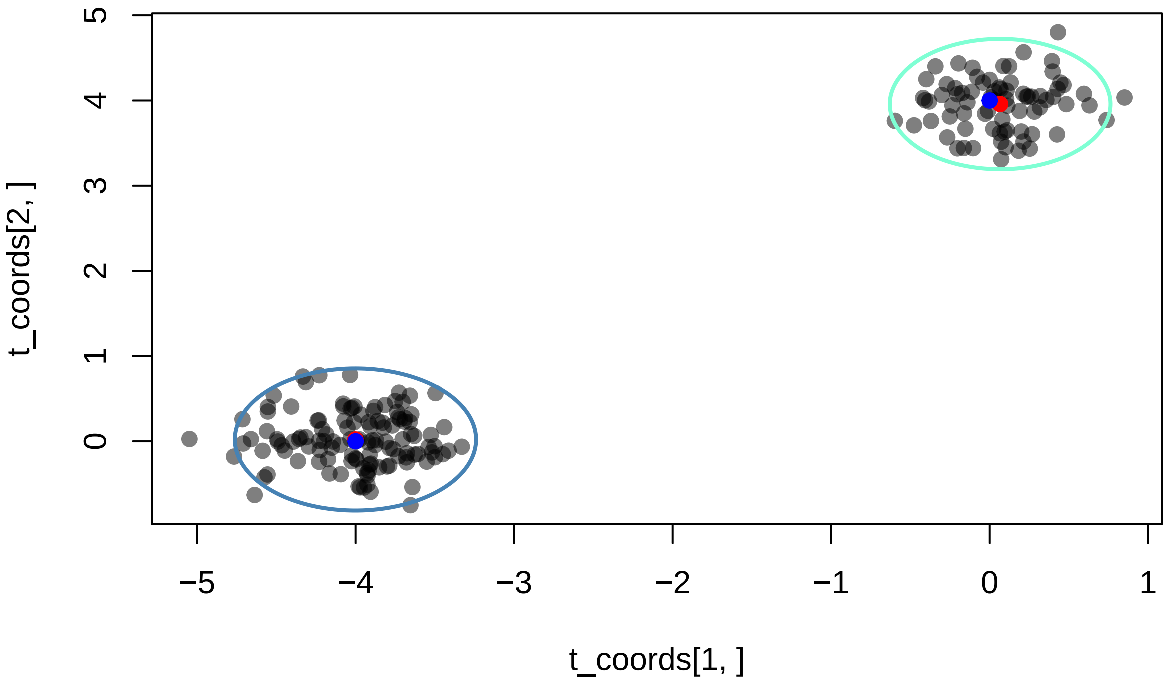

```
## predicted wrong number of clusters
```

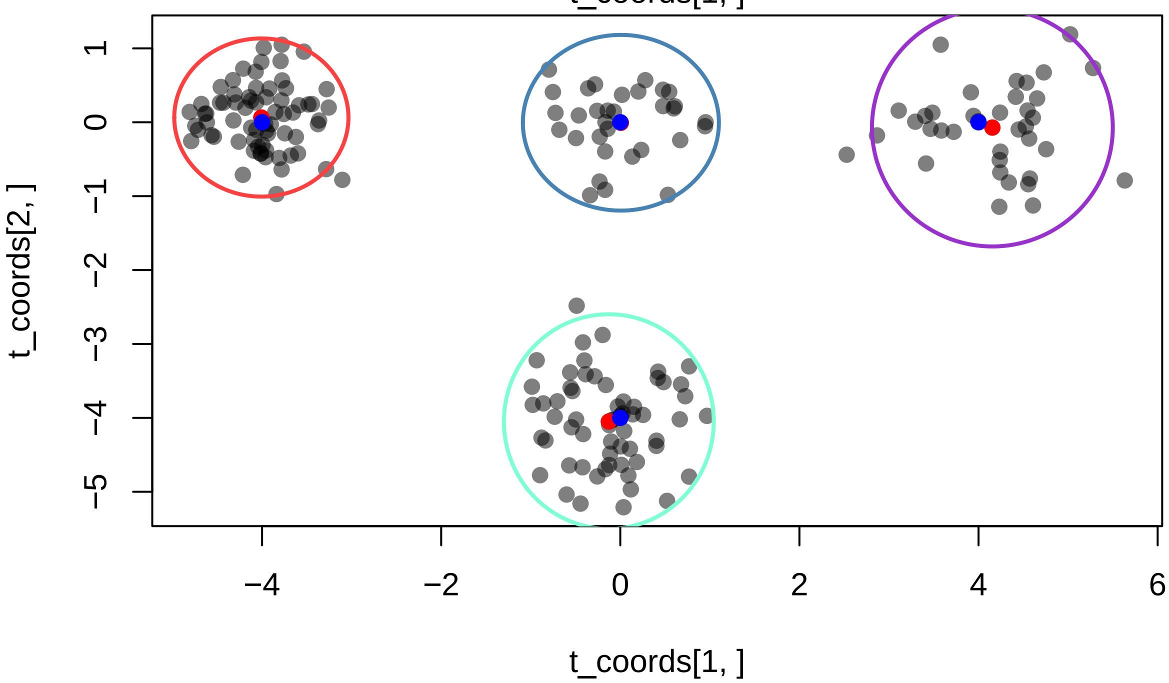


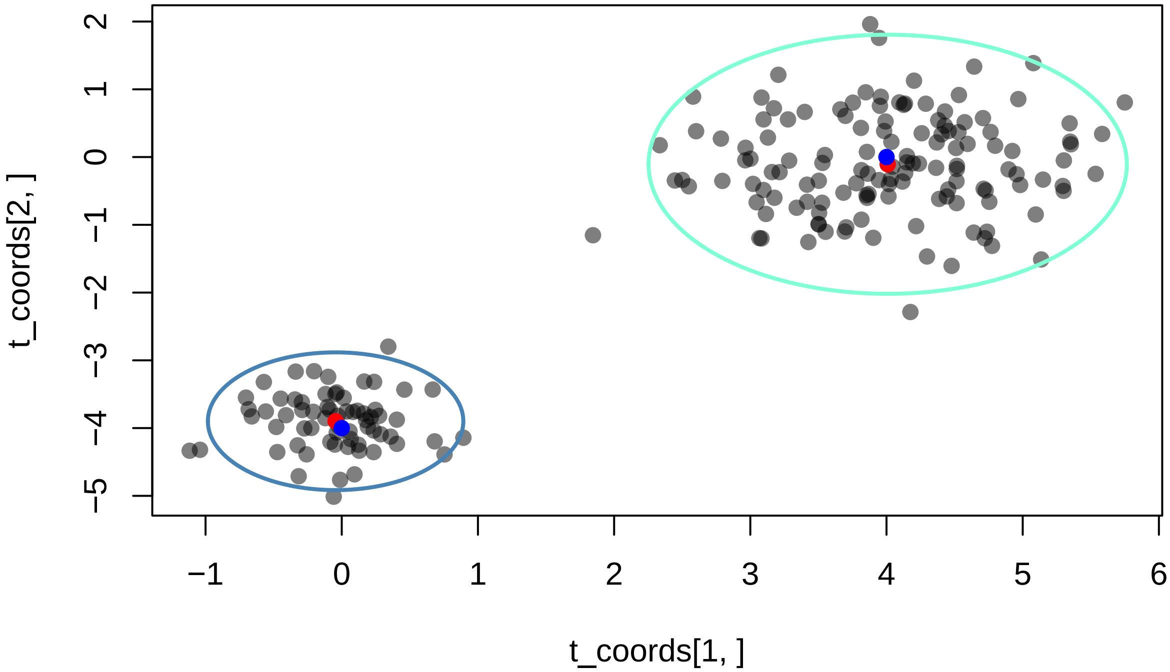


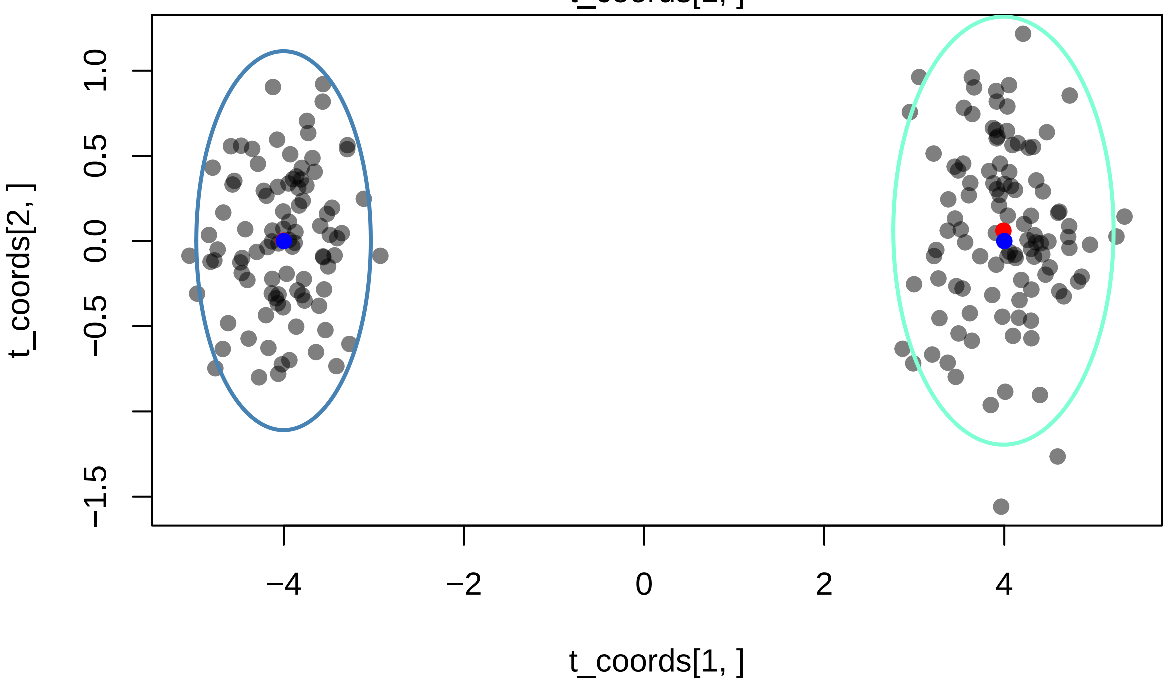
predicted wrong number of clusters

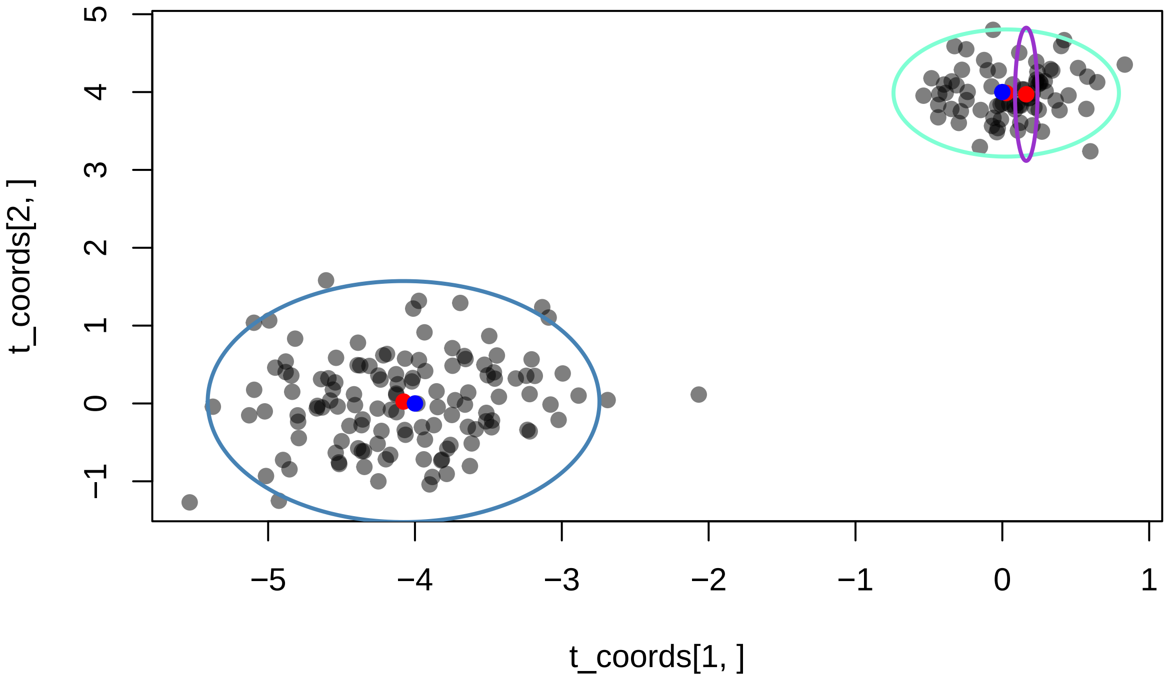



t_coords[2,]
t_coords[1,]
5
4
3
2
1
0
−5
−4
−3
−2
−1
0
1

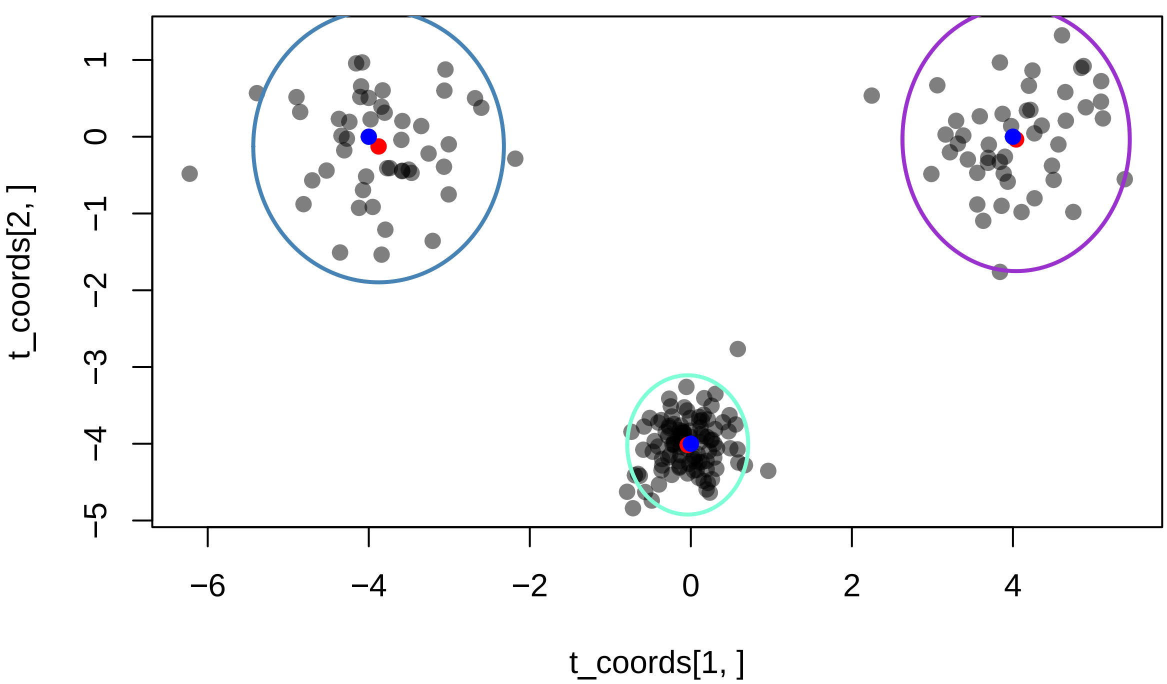

t_coords[2,]
t_coords[1,]
1
0
−1
−2
−3
−4
−5
−4
−2
0
2
4
6

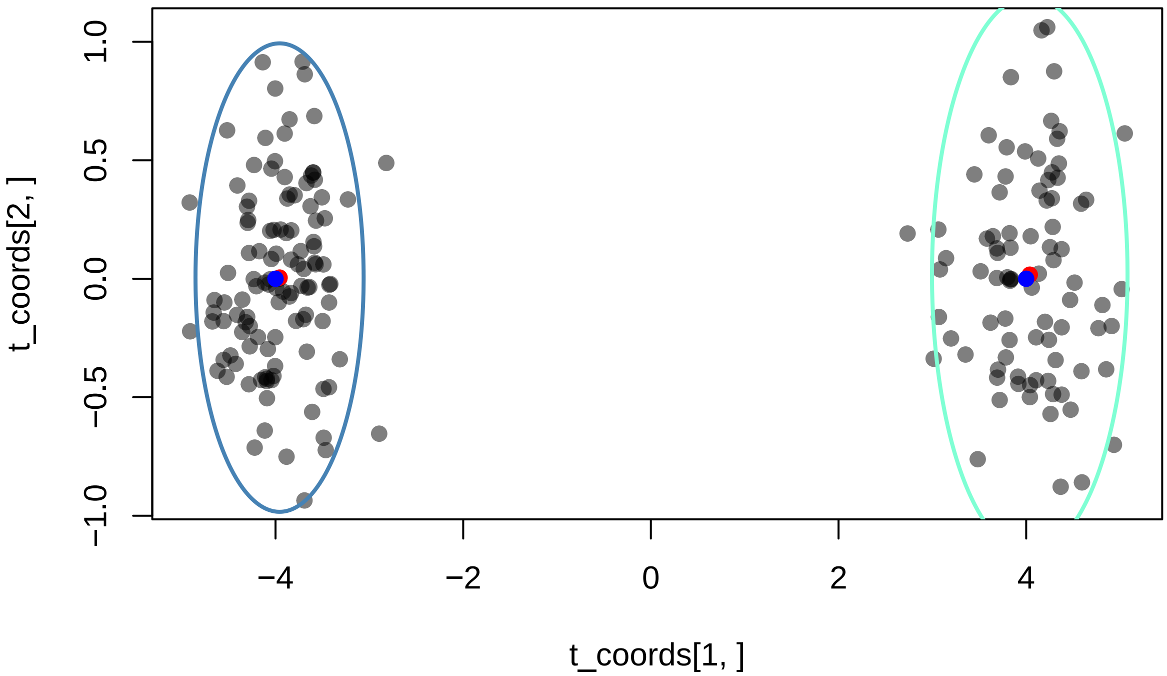

t_coords[2,]
t_coords[1,]
2
1
0
−1
−2
−3
−4
−5
−1
0
1
2
3
4
5
6


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

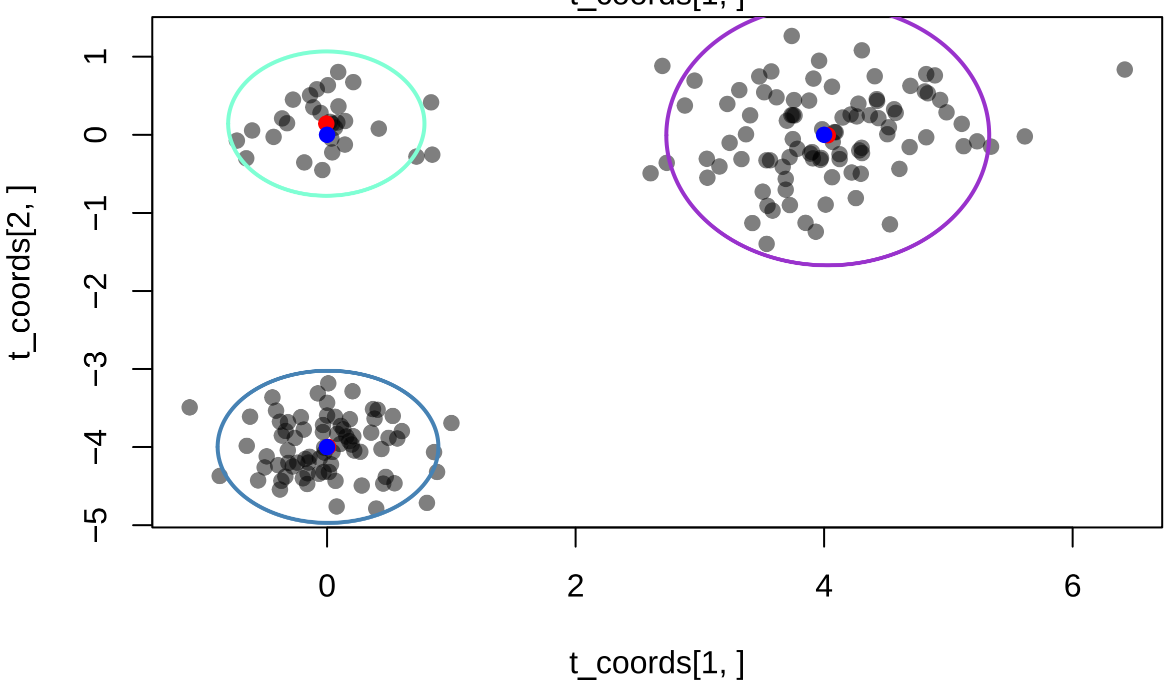


predicted wrong number of clusters

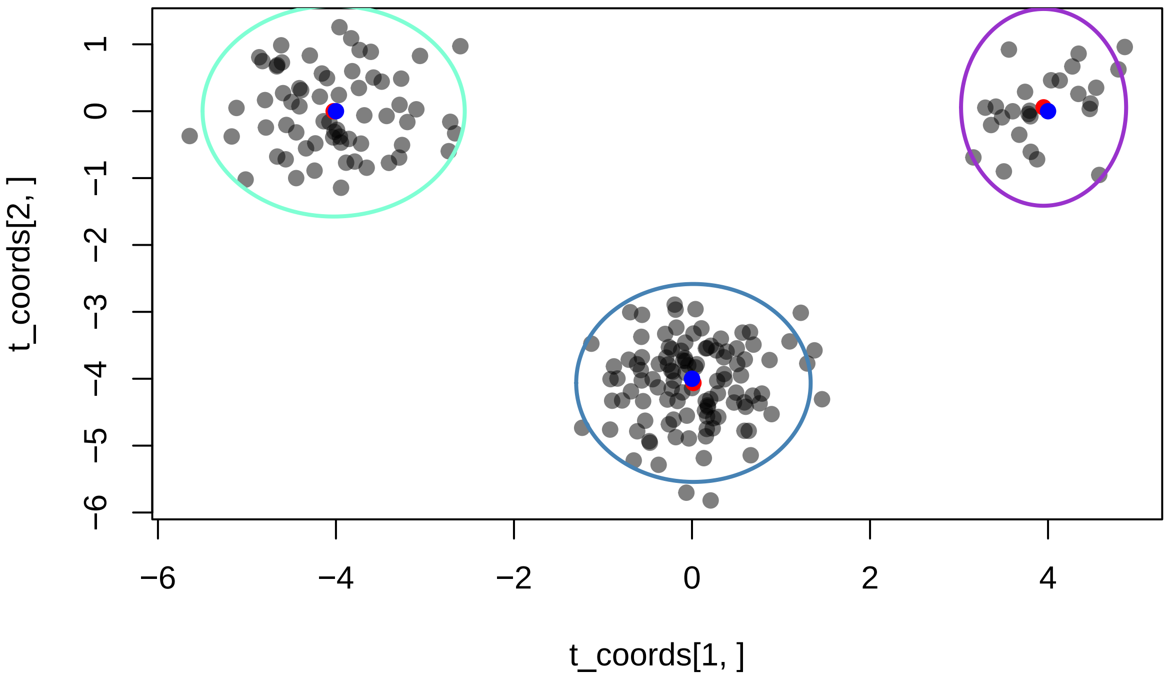


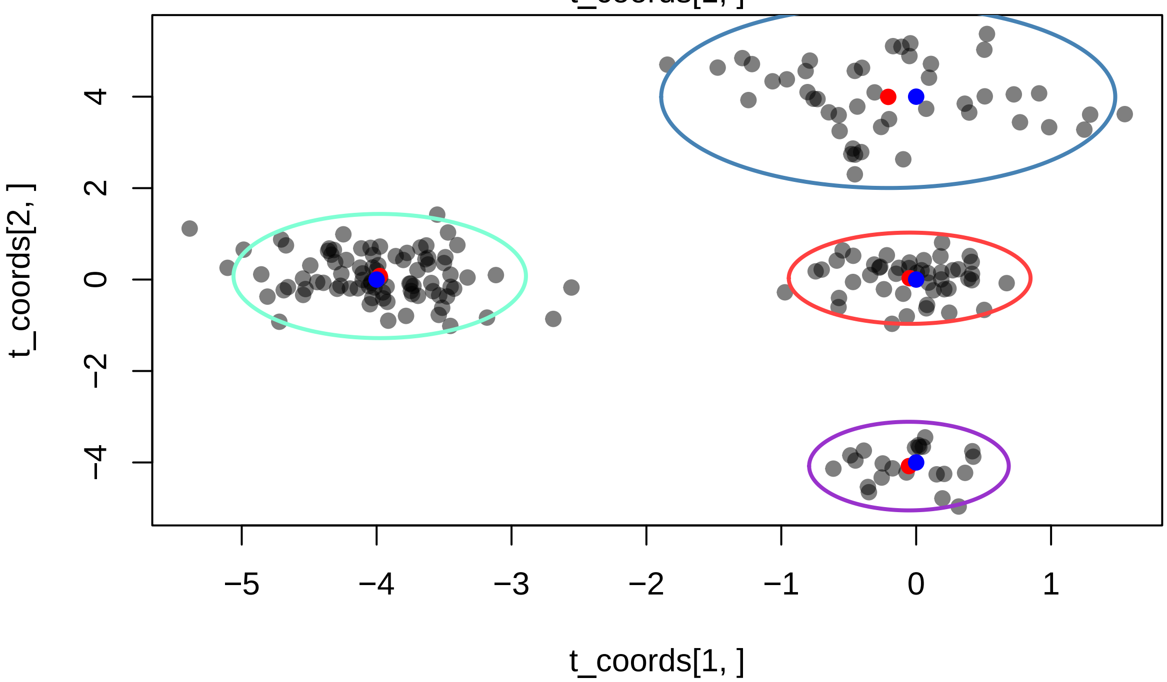


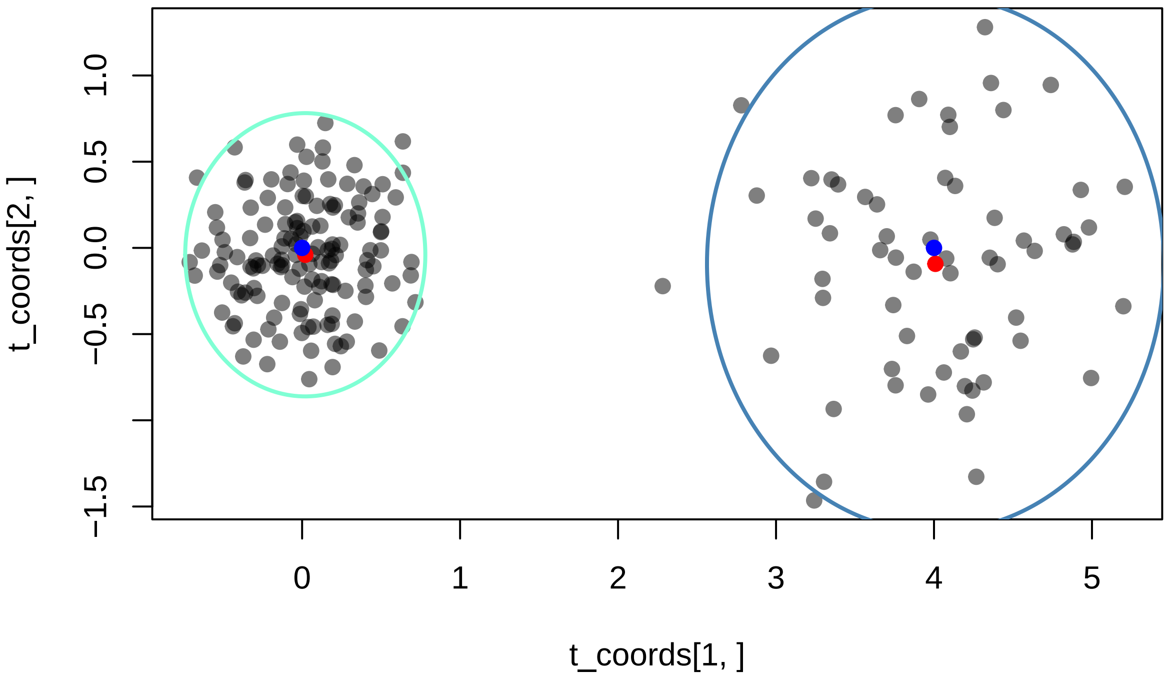
1.0
0.5
0.0
−0.5
−1.0
t_coords[2,]
−4
−2
0
2
4
t_coords[1,]

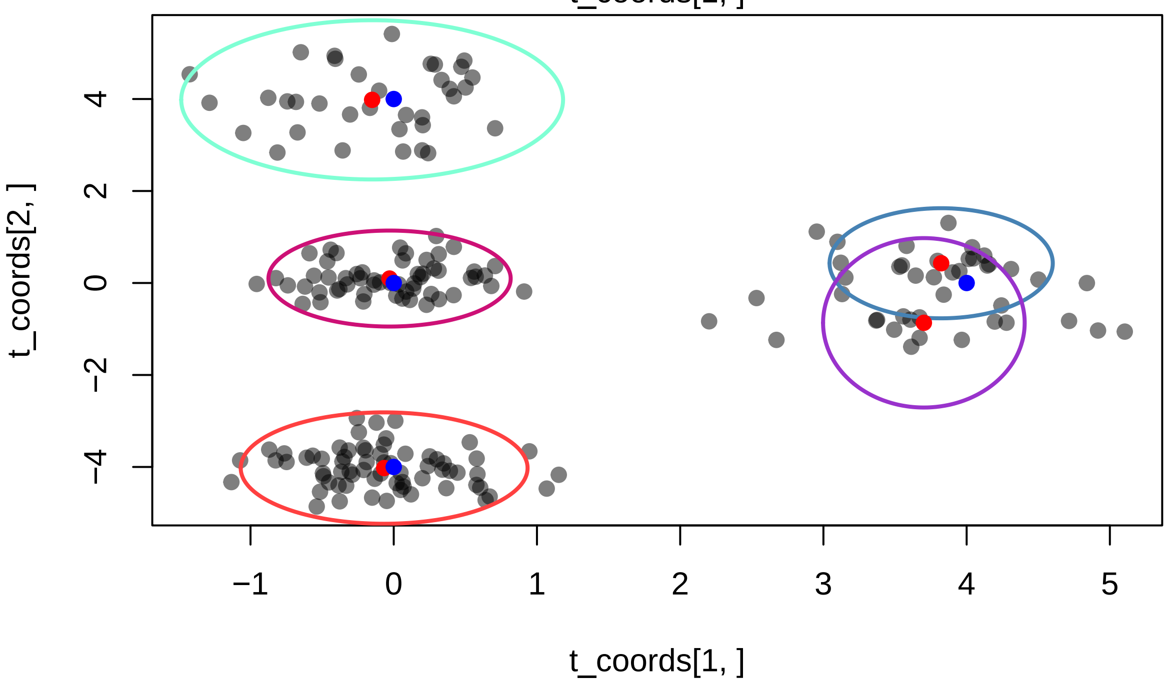


1
0
−1
−2
−3
−4
−5
t_coords[2,]
0
2
4
6
t_coords[1,]

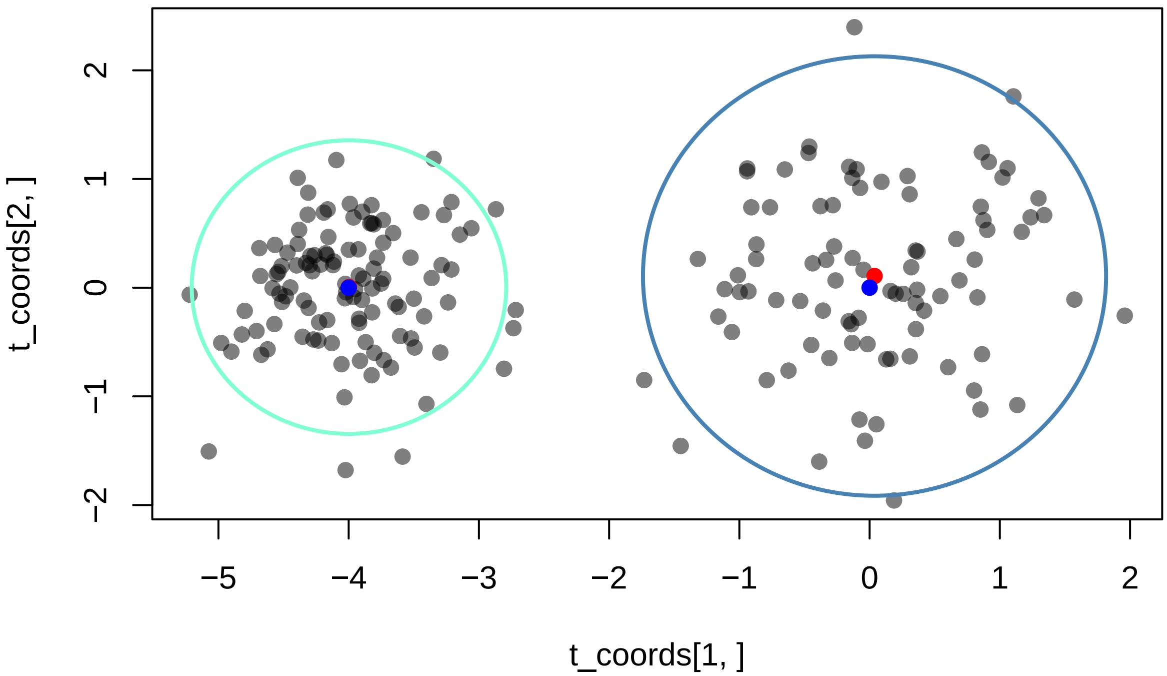

t_coords[2,]
1
0
−1
−2
−3
−4
−5
−6
−6
−4
−2
0
2
4
t_coords[1,]

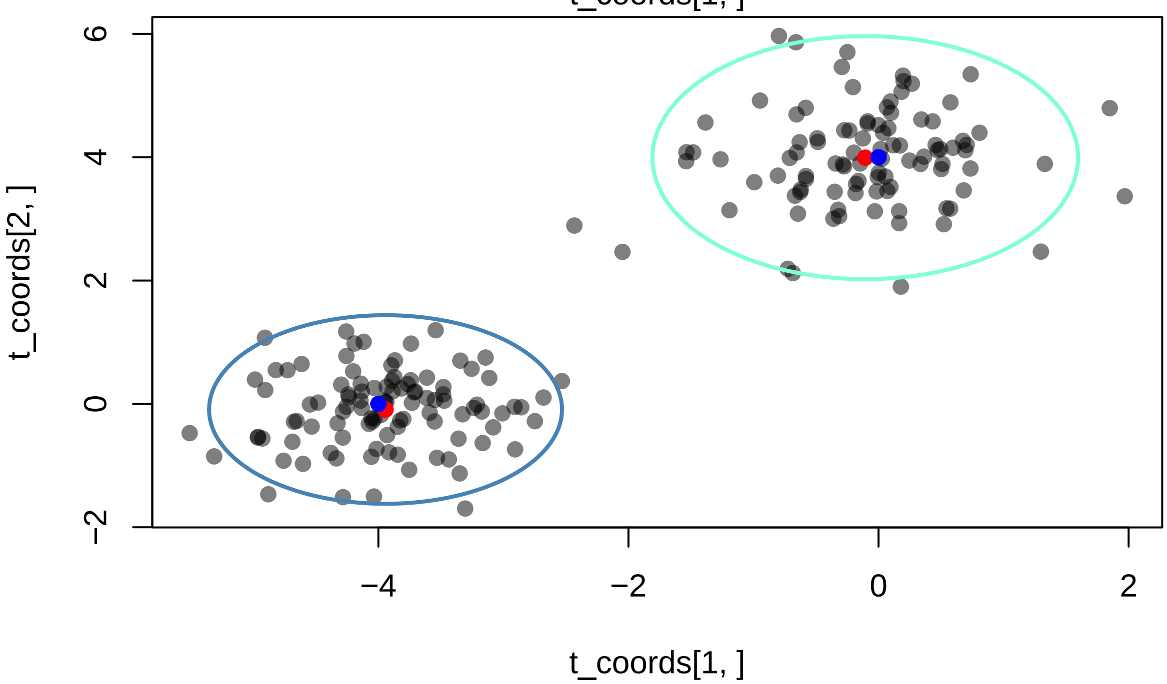

t_coords[2,]
4
2
0
−2
−4
−5
−4
−3
−2
−1
0
1
t_coords[1,]

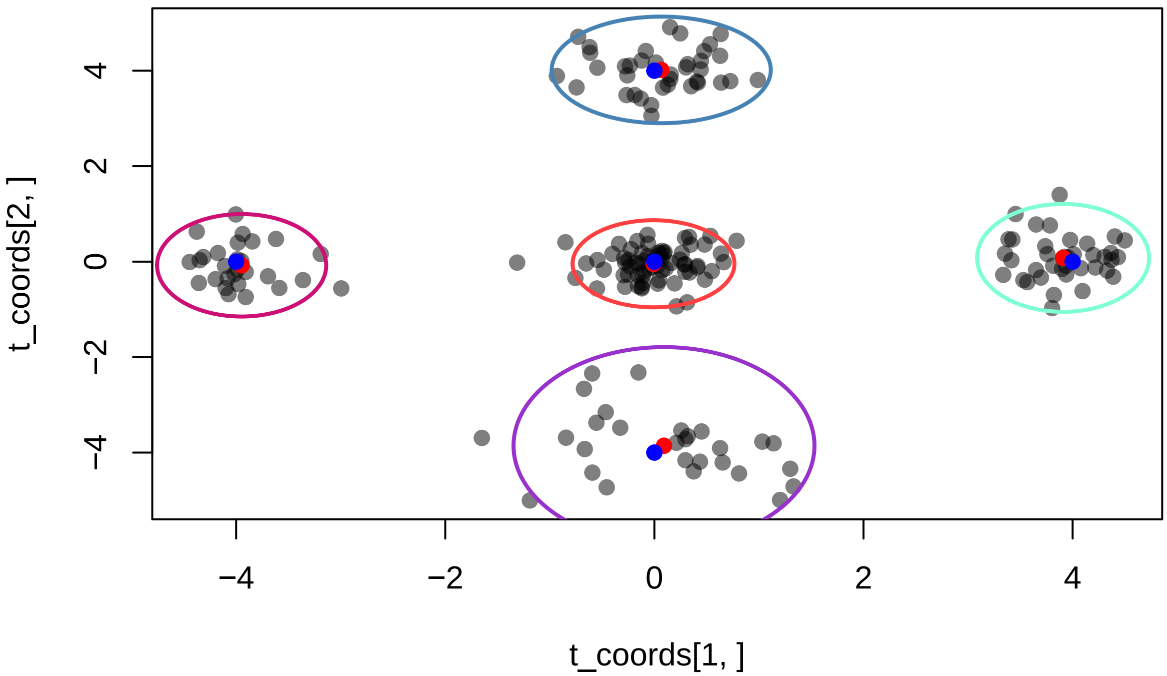


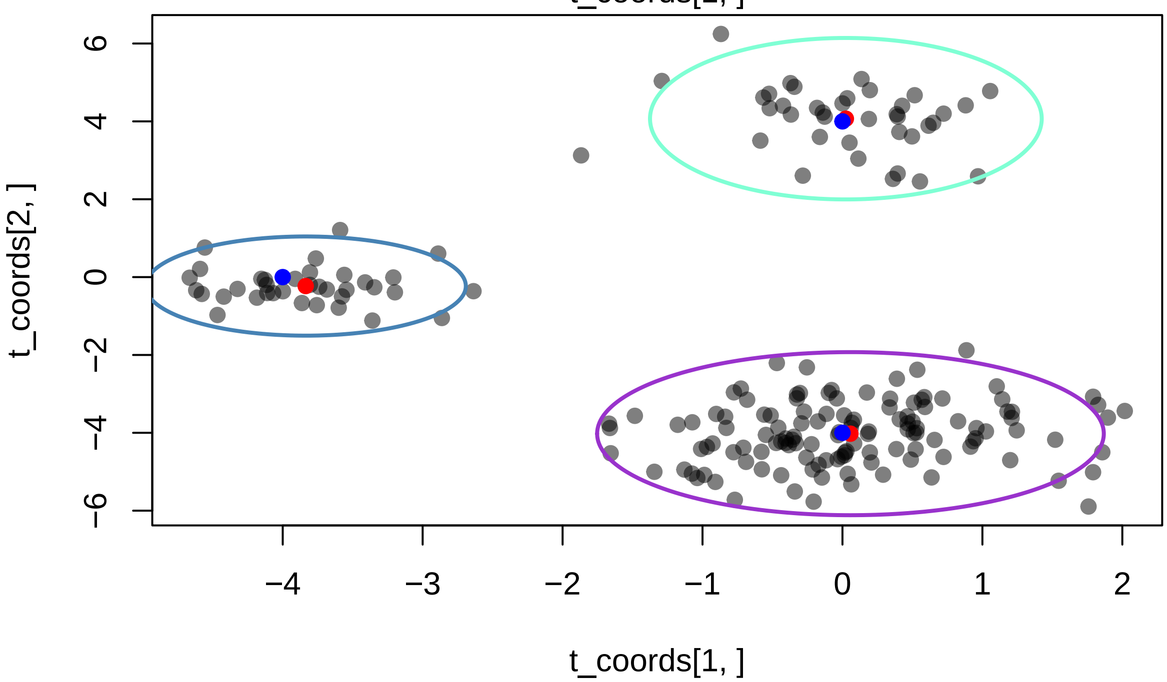


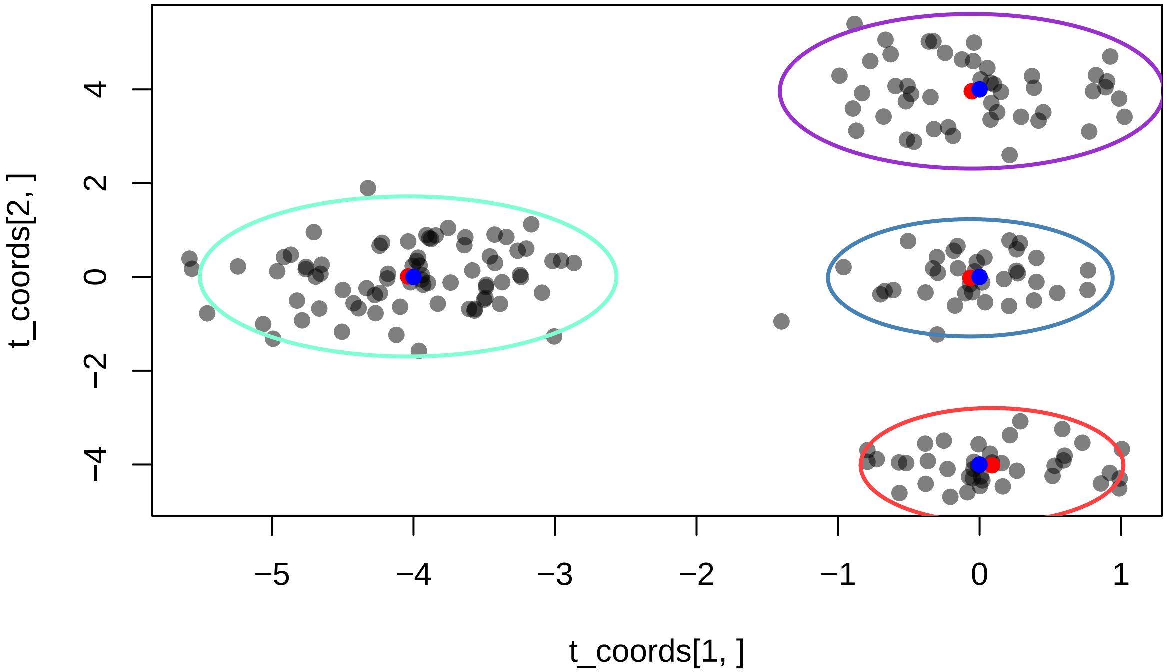

```
## predicted wrong number of clusters
```

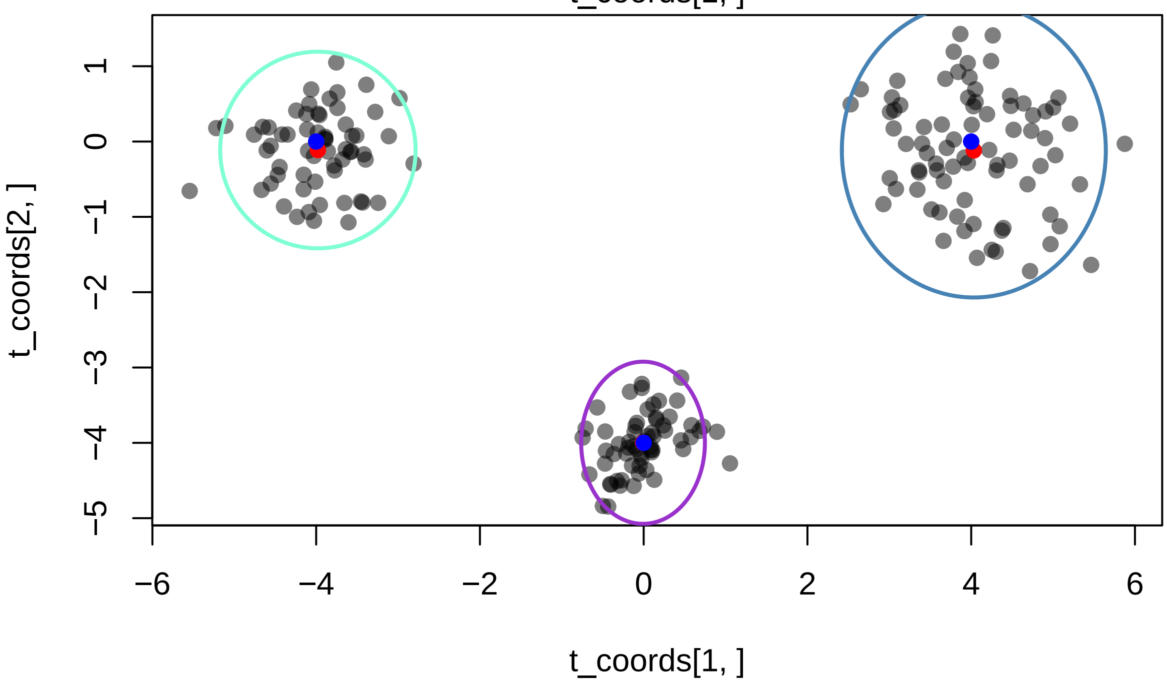

t_coords[2,]
t_coords[1,]
−5
−4
−3
−2
−1
0
1
2
−2
−1
0
1
2

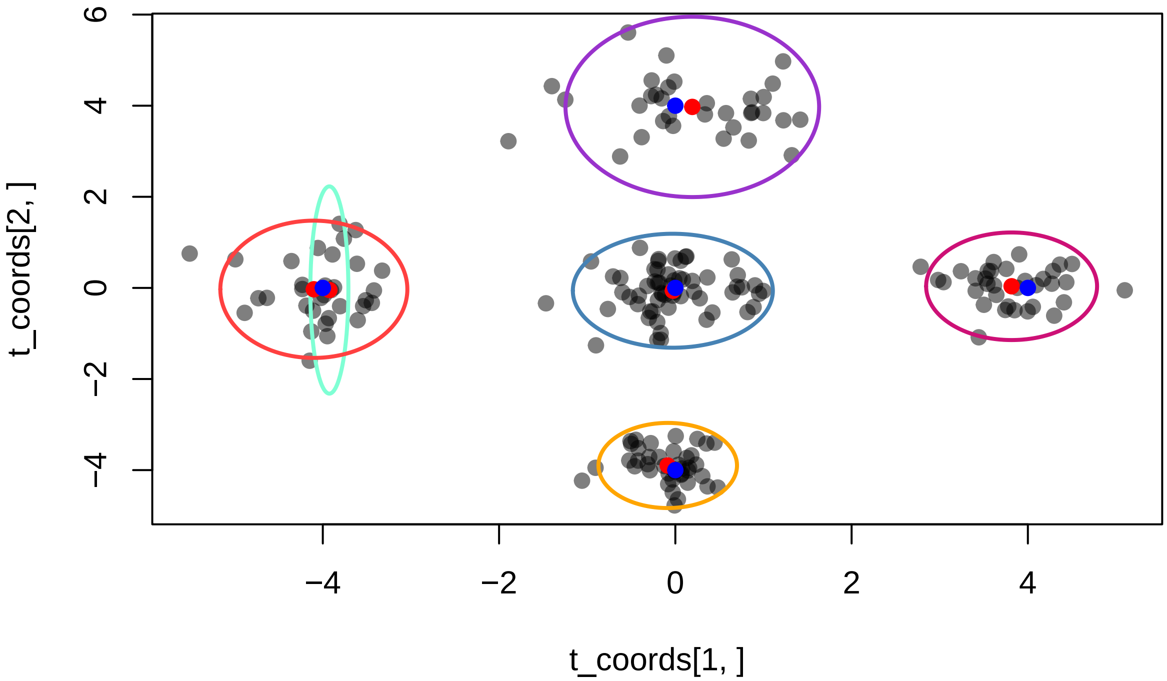

t_coords[2,]
t_coords[1,]
−4
−2
0
2
−2
0
2
4
6


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

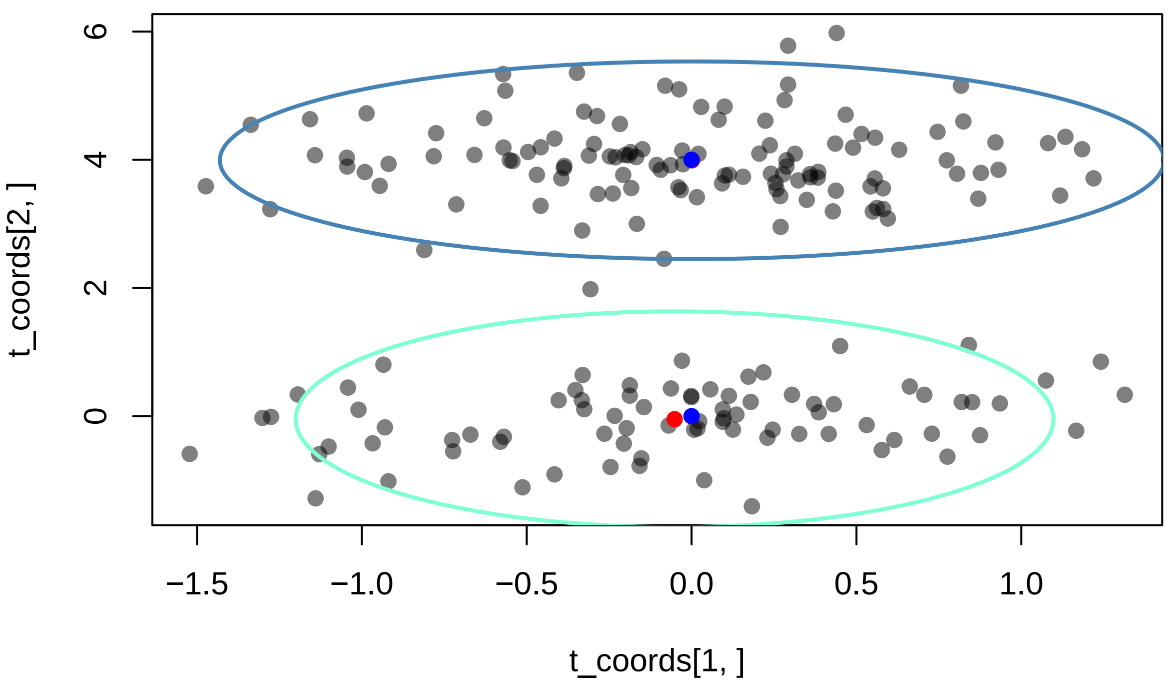

t_coords[2,]
t_coords[1,]
6
4
2
0
−2
−4
−6
−4
−3
−2
−1
0
1
2

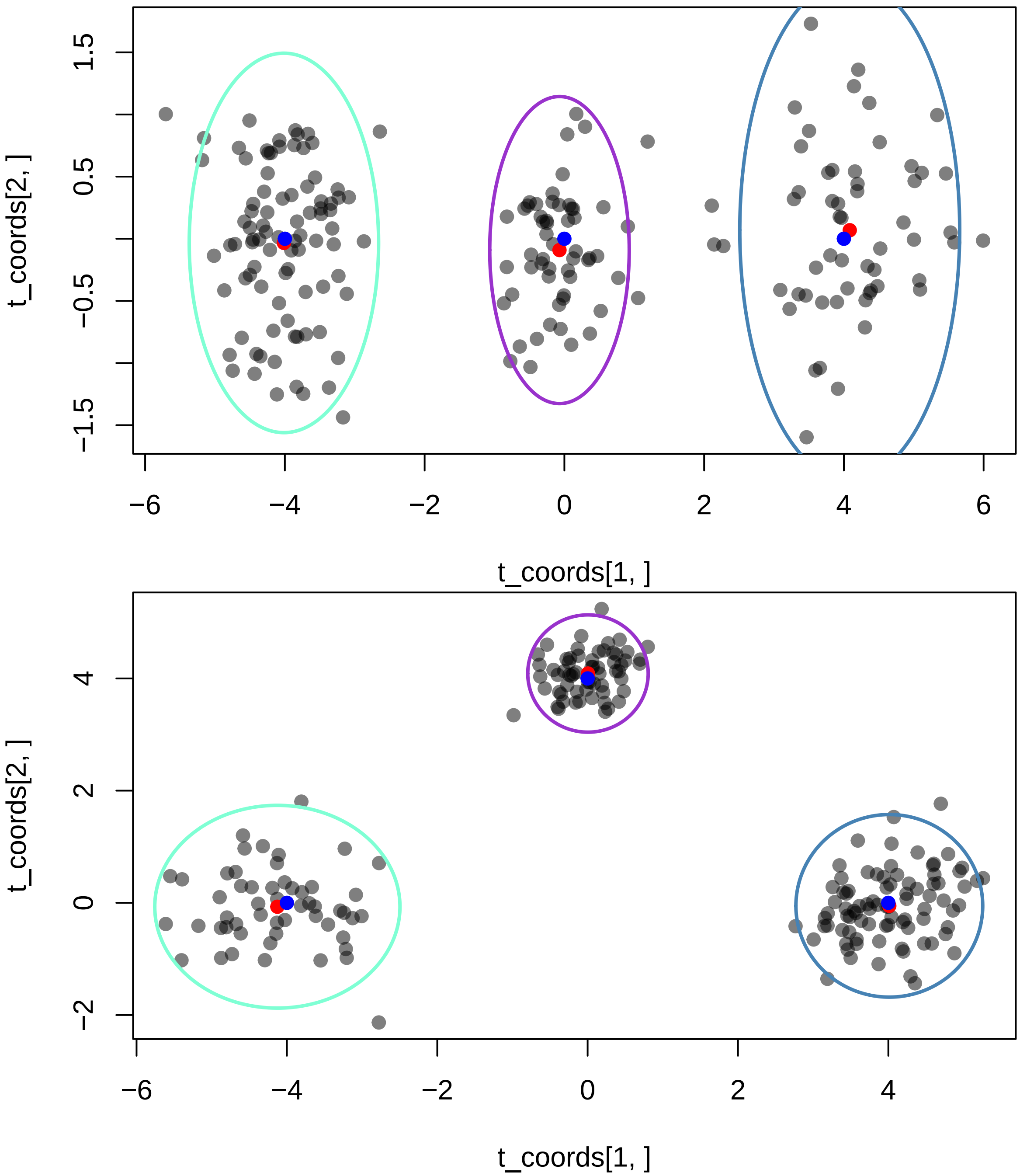

t_coords[2,]
t_coords[1,]
4
2
0
−2
−4
−5
−4
−3
−2
−1
0
1


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

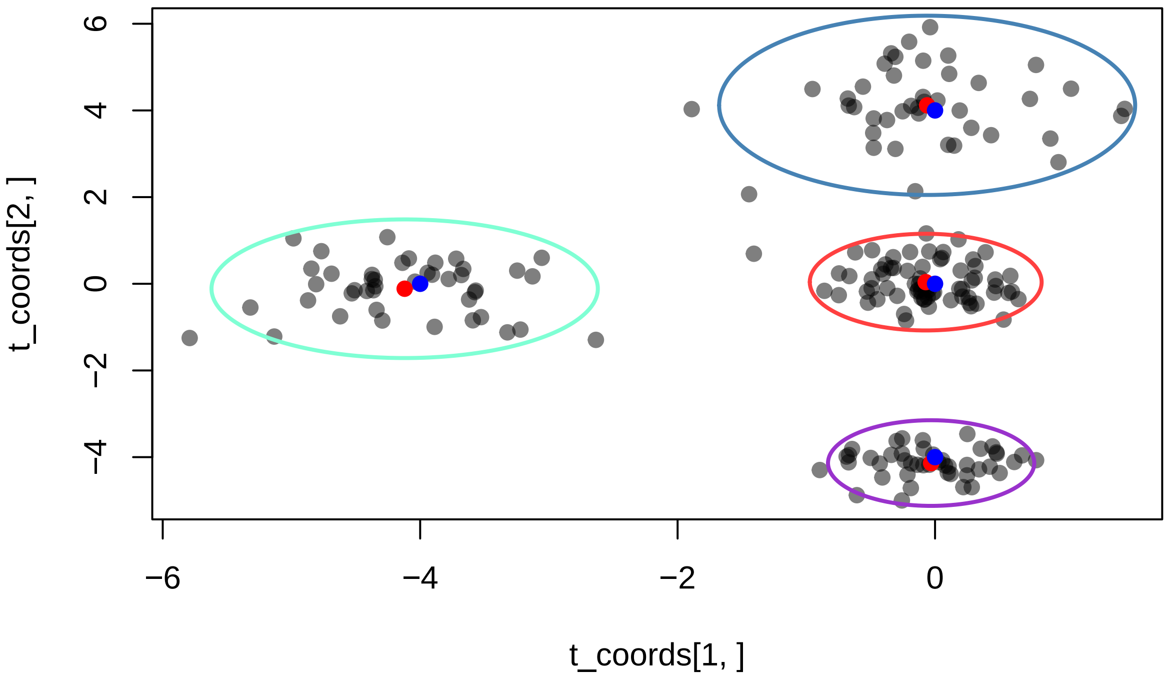



```
## predicted wrong number of clusters
```





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




```r
all_results = data.frame(cbind(robust_mu_error_list, original_mu_error_list, robust_score_list, original
```

```r
library(knitr)

print(paste("times of predicting wrong clusters numbers: ", wrong_ks))
```

```
## [1] "times of predicting wrong clusters numbers:  6"
```

```r
useful_results = subset(all_results, robust_score_list!=-1)
col_means <- colMeans(useful_results, na.rm = TRUE)
useful_results_with_mean <- rbind(useful_results, col_means)
rownames(useful_results_with_mean)[nrow(useful_results_with_mean)] <- "Mean"


kable(useful_results_with_mean, col.names = c("robust mean mu error", "regular mean mu error","robust ce
```

	robust mean mu error	regular mean mu error	robust center accuracy score	regular center accuracy score	robust mean sigma error	regular mean sigma error
1	0.0777	0.0777	0.5000	0.5000	0.0232	0.0232
3	0.1003	0.1003	0.5000	0.5000	0.1223	0.1223
4	0.1154	0.1154	0.5000	0.5000	0.0792	0.0792
5	0.1225	1.2432	0.4000	0.2000	0.1102	1.5031
6	0.0874	1.6717	0.5000	0.5000	0.1137	1.6665
7	0.1617	0.1617	0.3333	0.3333	0.0947	0.0947
8	0.1918	0.1918	0.4000	0.4000	0.1879	0.1879
9	0.0981	0.0981	0.8000	0.8000	0.1198	0.1198
10	0.0579	0.0579	1.0000	1.0000	0.0885	0.0885
11	0.0669	0.0669	1.0000	1.0000	0.0661	0.0661
12	0.0641	0.0641	1.0000	1.0000	0.0761	0.0762
13	0.0608	0.0608	1.0000	1.0000	0.1061	0.1062
15	0.0965	0.0965	0.5000	0.5000	0.1217	0.1218
17	0.1079	0.1079	0.5000	0.5000	0.1712	0.1713
18	0.0497	0.0497	1.0000	1.0000	0.0238	0.0238

	robust mean mu error	regular mean mu error	robust center accuracy score	regular center accuracy score	robust mean sigma error	regular mean sigma error
19	0.0954	0.0954	0.5000	0.5000	0.0641	0.0641
20	0.1087	0.1087	0.0000	0.0000	0.0799	0.0798
21	0.0328	0.0328	1.0000	1.0000	0.0456	0.0457
23	0.0900	0.0900	0.6667	0.6667	0.1230	0.1231
24	0.0425	0.0425	1.0000	1.0000	0.0453	0.0453
25	0.0606	0.0606	0.6667	0.6667	0.0673	0.0673
26	0.0556	0.0556	1.0000	1.0000	0.1243	0.1243
27	0.1101	0.1101	0.7500	0.7500	0.0794	0.0793
28	0.0687	0.0687	1.0000	1.0000	0.0322	0.0322
30	0.0600	0.0600	0.5000	0.5000	0.1063	0.1063
31	0.1069	0.1069	0.0000	0.0000	0.0946	0.0947
32	0.0996	0.0996	0.6000	0.6000	0.0927	0.0927
33	0.1392	0.1392	0.6667	0.6667	0.1841	0.1841
34	0.0662	1.7806	1.0000	0.5000	0.0955	3.0003
35	0.0816	0.0816	0.3333	0.3333	0.0960	0.0960
37	0.0388	0.0388	1.0000	1.0000	0.1282	0.1282
38	0.0865	0.0865	0.3333	0.3333	0.1192	0.1192
39	0.0958	0.0958	0.6667	0.6667	0.0995	0.0995
40	0.1301	0.8837	0.2500	0.0000	0.0883	3.6629
Mean	0.0891	0.2412	0.6431	0.6152	0.0962	0.3734

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

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