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
```

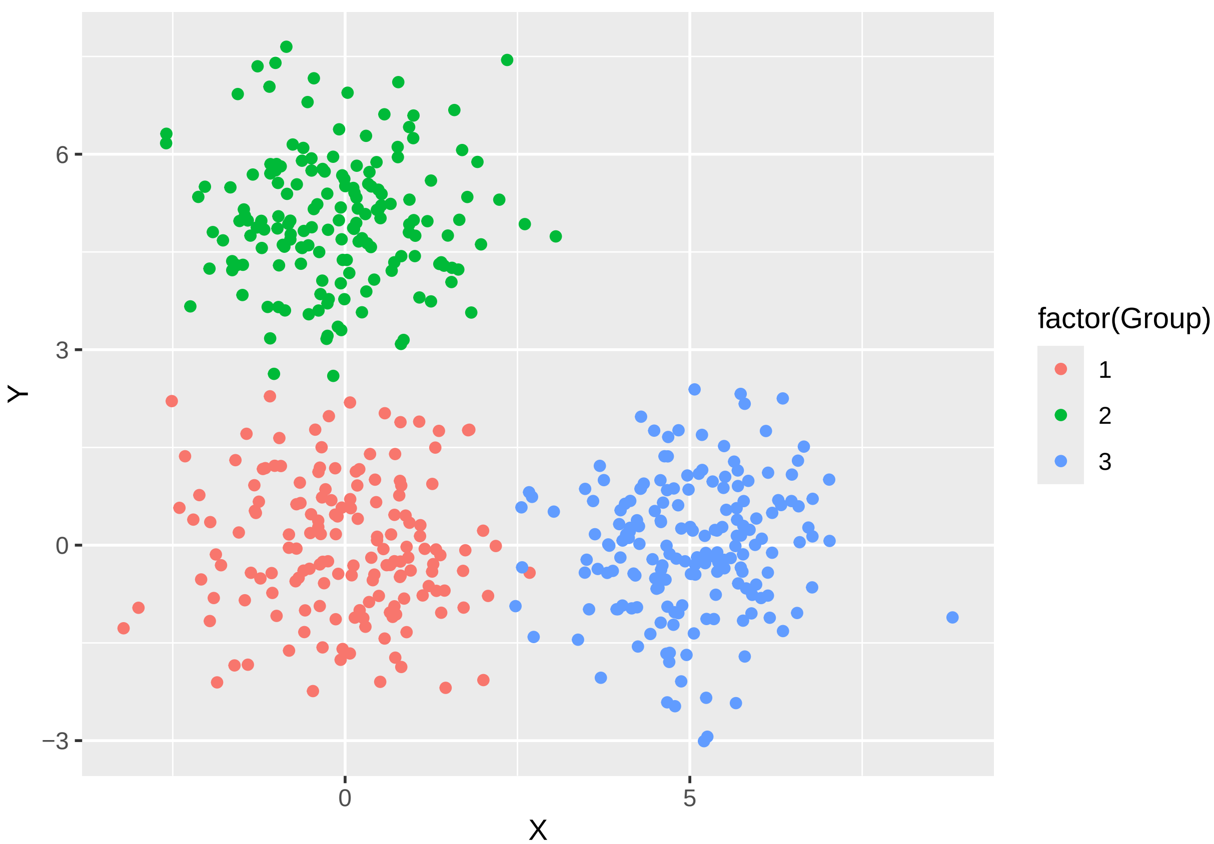
```
library(clue)

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) {
```

```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)
```

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

}
```

2d sample with 3 mixtures

```
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)))
```



```r
#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
```

correct guess for centers

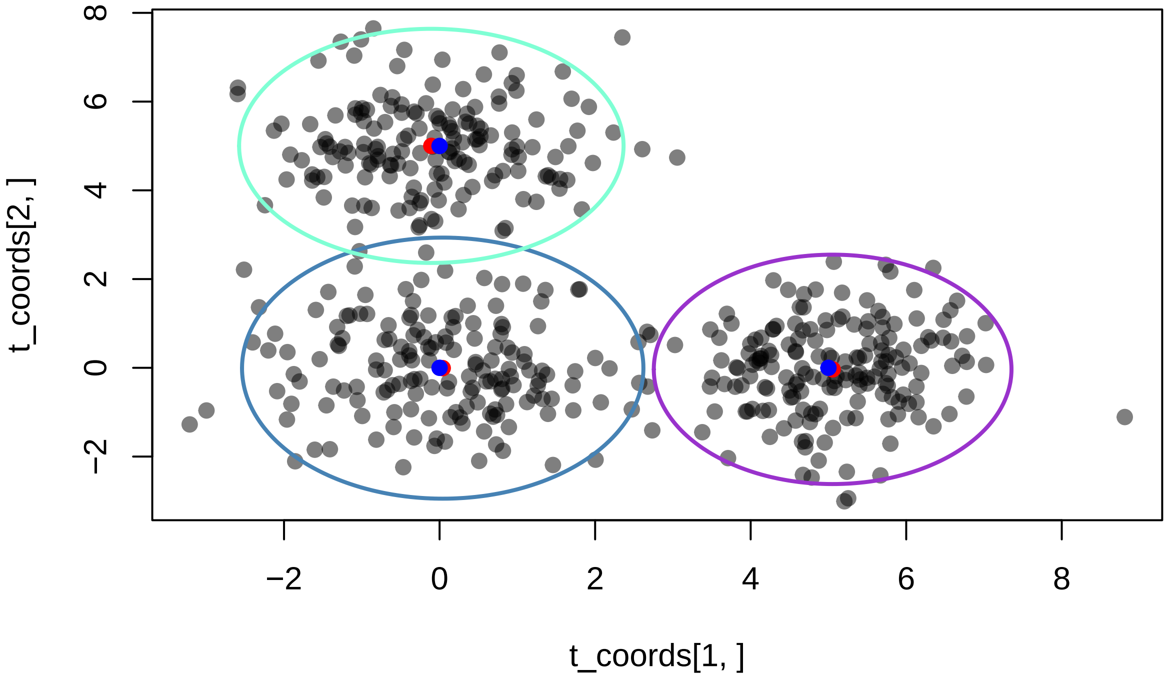
```
## $mu
## $mu[[1]]
##              [,1]
## [1,]  0.040387722
## [2,] -0.006874544
##
## $mu[[2]]
##            [,1]
## [1,] -0.1060505
## [2,]  5.0019716
##
## $mu[[3]]
##             [,1]
## [1,]  5.05384376
## [2,] -0.03447569
##
##
## $Sigma
## $Sigma[[1]]
##             [,1]        [,2]
## [1,]  1.42647543 -0.07499782
## [2,] -0.07499782  1.12917918
##
## $Sigma[[2]]
##             [,1]        [,2]
## [1,]  1.16063518 -0.01038747
## [2,] -0.01038747  1.02029944
##
## $Sigma[[3]]
##            [,1]       [,2]
## [1,] 0.92685715 0.09130935
## [2,] 0.09130935 1.07076205
##
##
## $alpha
## [1] 0.3243445 0.3273938 0.3482617
##
## $iter
## [1] 75
```

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

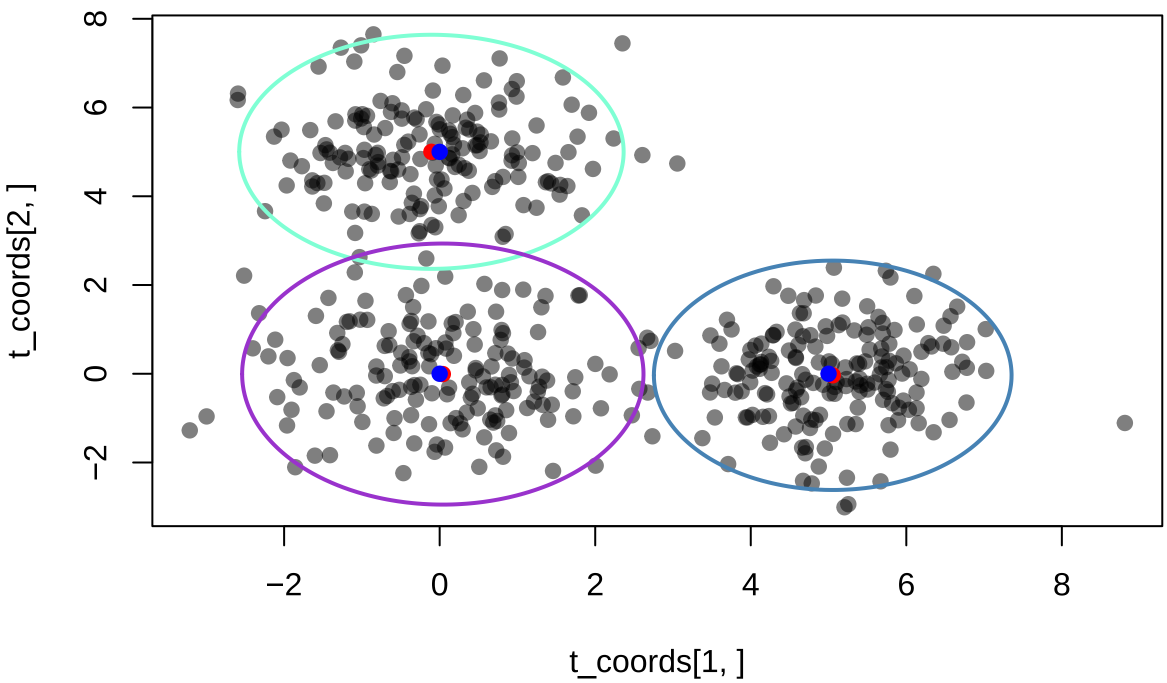
```
## $mu_error
```

```
## [1] 0.07032424
##
## $score
## [1] 0.6666667
##
## $sigma_error
## [1] 0.2616835
```

```
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)
```




```
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)
```



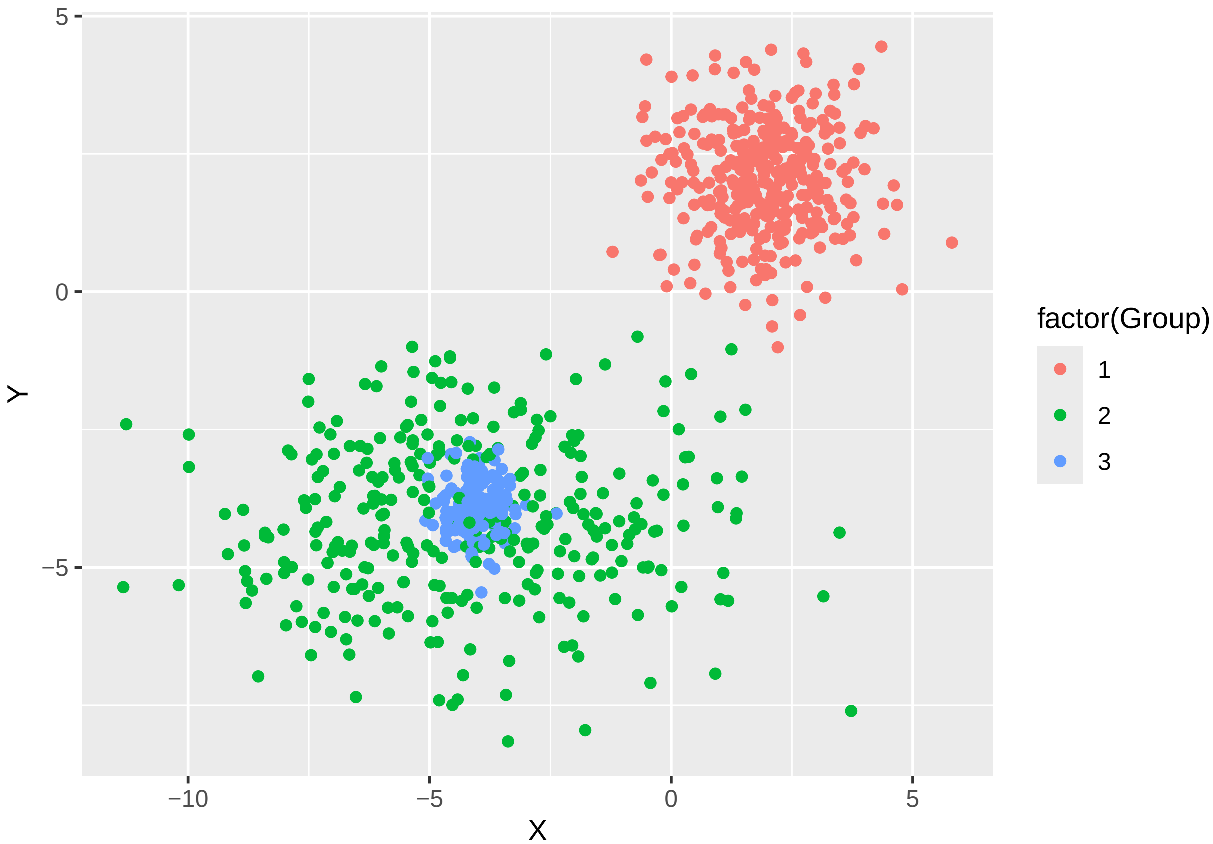
robust

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

```
## $mu_error
## [1] 0.07032471
##
## $score
## [1] 0.6666667
##
## $sigma_error
## [1] 0.2615778
```

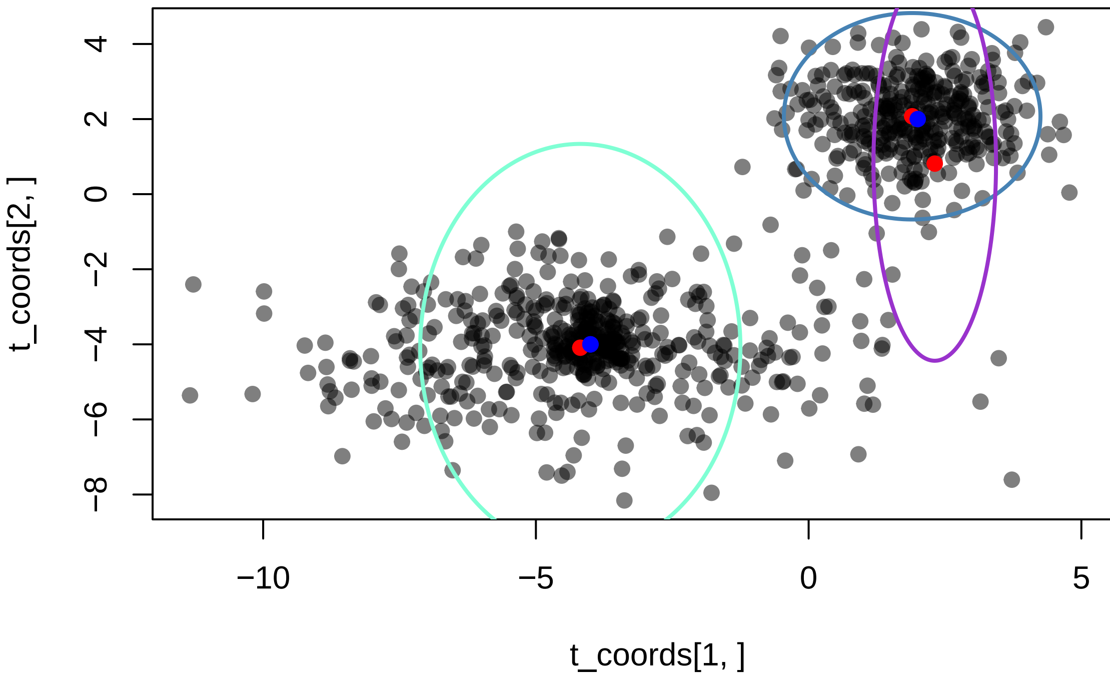
another 2d sample with 3 mixtures

```
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)
```

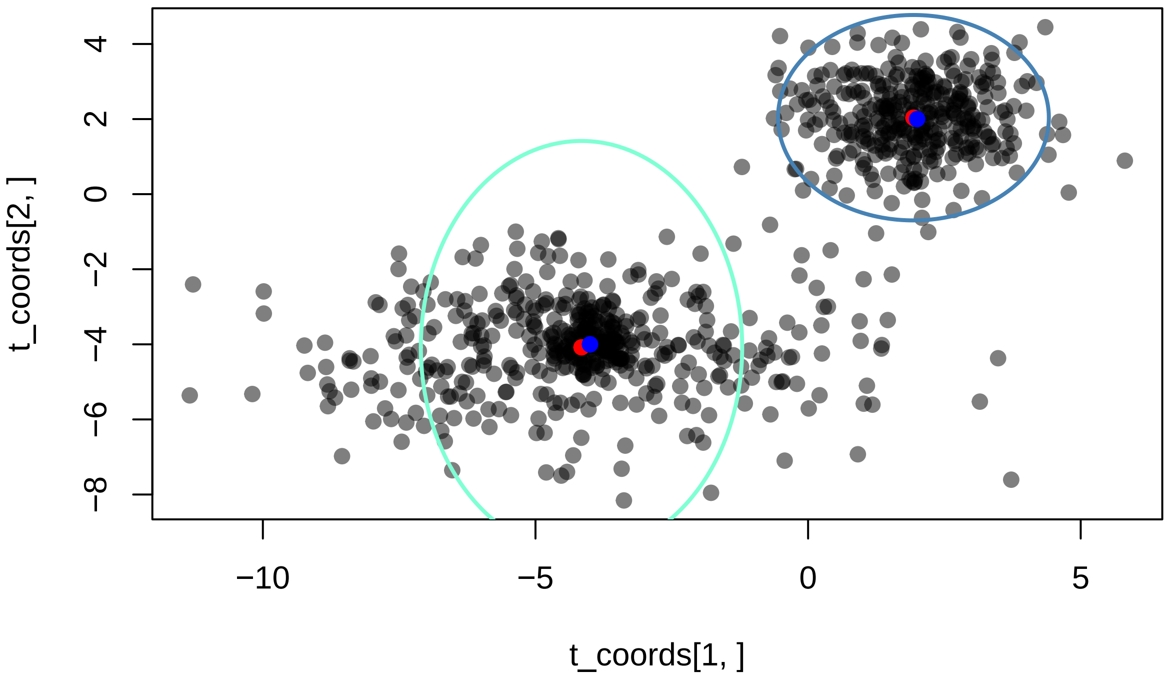


correct guess for centers

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

```
## $mu_error
## [1] 2.756958
##
## $score
## [1] 0
##
## $sigma_error
## [1] 1.964622
```

```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)
```



robust

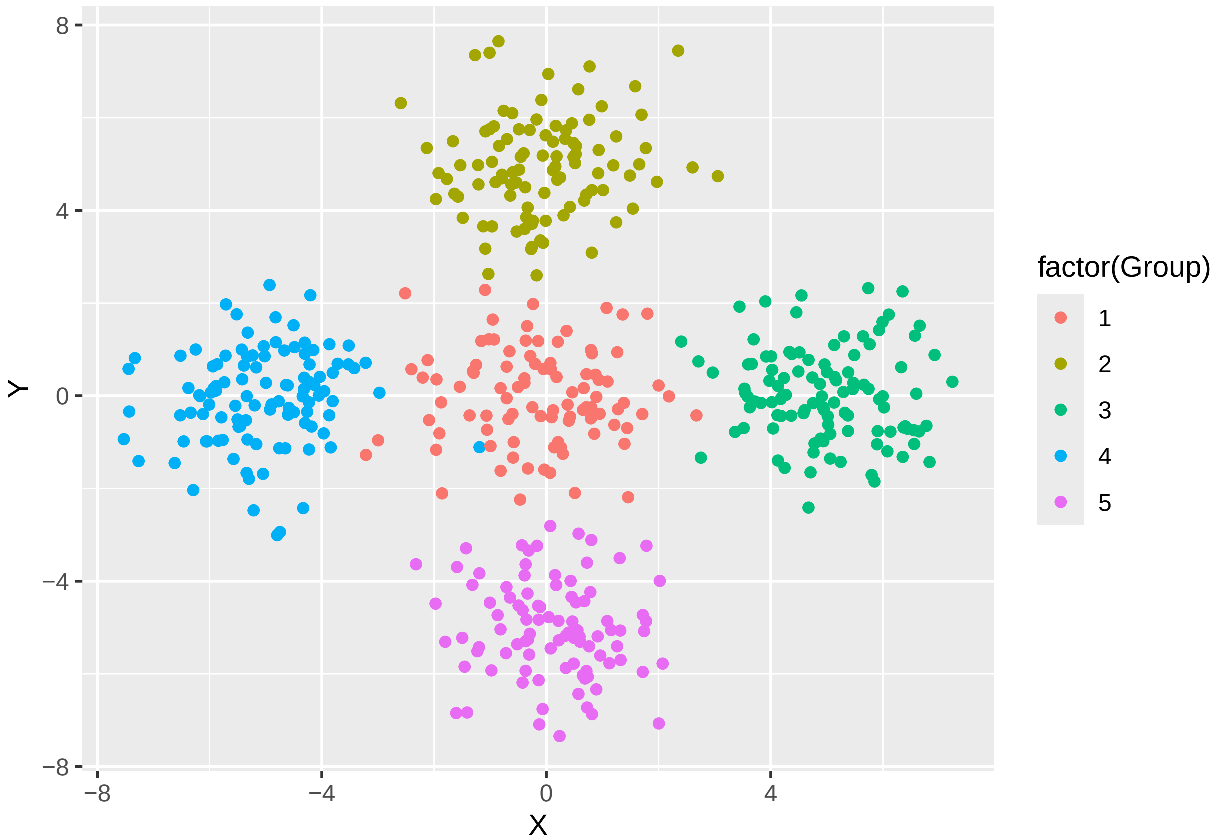
```
eval_EM(res = EMR_res, mu_list = mu_list, dim = dim, I2_list=I2_list)
```

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

```
## [1] -1
```

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))
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")
```

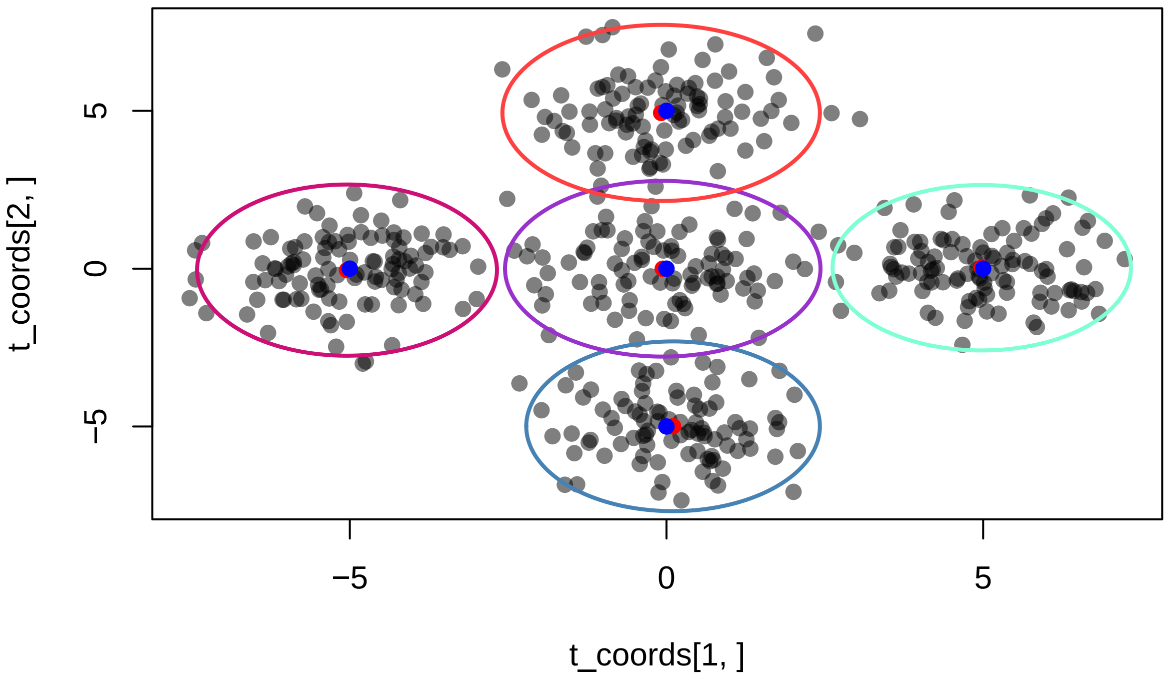
```
C = 5 #number of centers
Z <- sample(1:C, n, replace = T)
t_coords = t(coords)

EM_res <- EM(X=t_coords, C=C, Z=Z, tol=1e-10, m_iter=1e3)
EM_res
```

correct guess for centers

```
## $mu
## $mu[[1]]
##            [,1]
## [1,]  0.1040853
## [2,] -4.9932389
##
## $mu[[2]]
##            [,1]
## [1,] 4.97758128
## [2,] 0.02952239
##
## $mu[[3]]
##              [,1]
## [1,] -0.058758632
## [2,] -0.002494021
##
```

```
## $mu[[4]]
##             [,1]
## [1,] -0.08424581
## [2,]  4.93395801
##
## $mu[[5]]
##             [,1]
## [1,] -5.04431786
## [2,] -0.04598513
##
##
## $Sigma
## $Sigma[[1]]
##            [,1]       [,2]
## [1,]  0.9576152 -0.1230173
## [2,] -0.1230173  1.1455418
##
## $Sigma[[2]]
##             [,1]        [,2]
## [1,]  1.11826026 -0.07297194
## [2,] -0.07297194  0.95389476
##
## $Sigma[[3]]
##              [,1]         [,2]
## [1,]  1.292744967 -0.003791129
## [2,] -0.003791129  1.035152153
##
## $Sigma[[4]]
##           [,1]      [,2]
## [1,] 1.1455890 0.1220378
## [2,] 0.1220378 1.2014153
##
## $Sigma[[5]]
##           [,1]      [,2]
## [1,] 1.0788286 0.1456586
## [2,] 0.1456586 1.0837939
##
##
## $alpha
## [1] 0.1871386 0.2134916 0.1850623 0.1990016 0.2153059
##
## $iter
## [1] 145
```

```r
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)
```

```
## $mu_error
## [1] 0.07421941
##
## $score
## [1] 0.6
##
## $sigma_error
## [1] 0.2455497
```

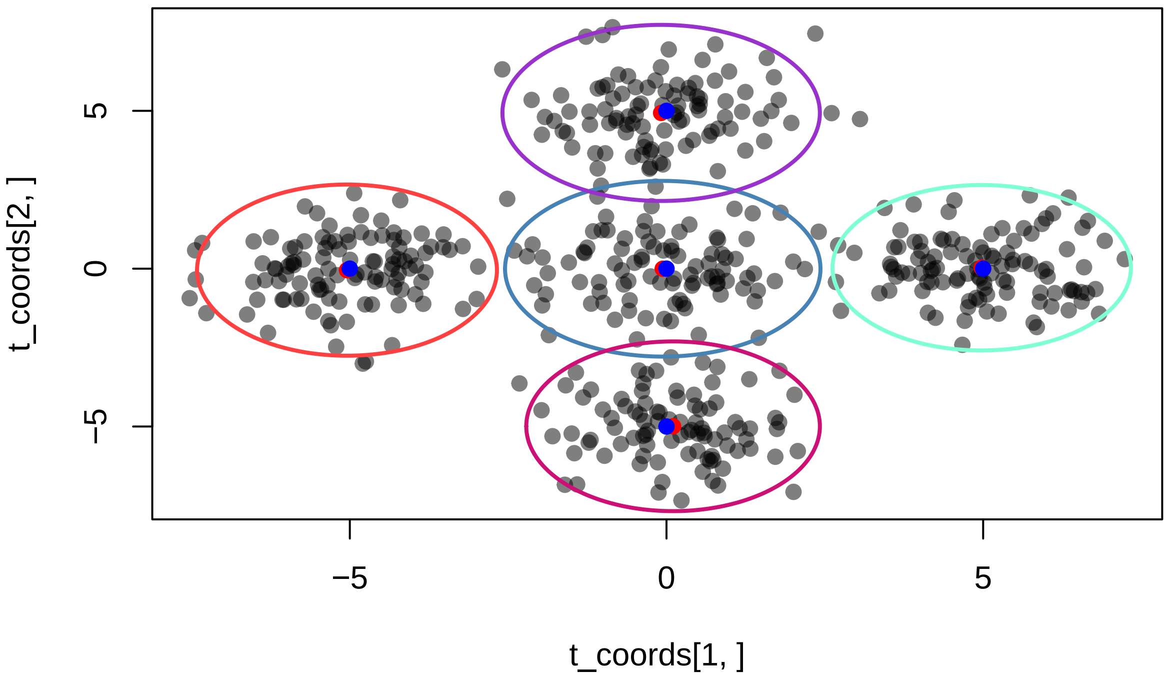
```
EMR_res <- EM_Robust(t_coords, n-1)
EMR_res
```

robust

```
## $mu
## $mu[[1]]
##                 [,1]
## [1,] -0.058760350
## [2,] -0.002494371
##
## $mu[[2]]
##            [,1]
## [1,] 4.977582
## [2,] 0.029522
##
## $mu[[3]]
##                [,1]
## [1,] -0.08424635
## [2,]  4.93396039
##
```

```
## $mu[[4]]
##              [,1]
## [1,] -5.04432055
## [2,] -0.04598501
##
## $mu[[5]]
##            [,1]
## [1,]  0.104086
## [2,] -4.993242
##
##
## $Sigma
## $Sigma[[1]]
##               [,1]         [,2]
## [1,]  1.292584413 -0.003786562
## [2,] -0.003786562  1.035030629
##
## $Sigma[[2]]
##             [,1]       [,2]
## [1,]  1.1181410 -0.0729633
## [2,] -0.0729633  0.9537991
##
## $Sigma[[3]]
##            [,1]      [,2]
## [1,] 1.1454736 0.1220268
## [2,] 0.1220268 1.2012839
##
## $Sigma[[4]]
##            [,1]      [,2]
## [1,] 1.0787088 0.1456441
## [2,] 0.1456441 1.0836862
##
## $Sigma[[5]]
##             [,1]       [,2]
## [1,]  0.9575191 -0.1230028
## [2,] -0.1230028  1.1454143
##
##
## $alpha
## [1] 0.1850623 0.2134917 0.1990016 0.2153059 0.1871385
##
## $iter
## [1] 130
##
## $outliers
## NULL
```

```r
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)
```




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

```
## $mu_error
## [1] 0.07421984
##
## $score
## [1] 0.6
##
## $sigma_error
## [1] 0.2454364
```

3d sample with 3 mixtures

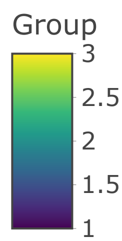
```r
library(plotly)
```

```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

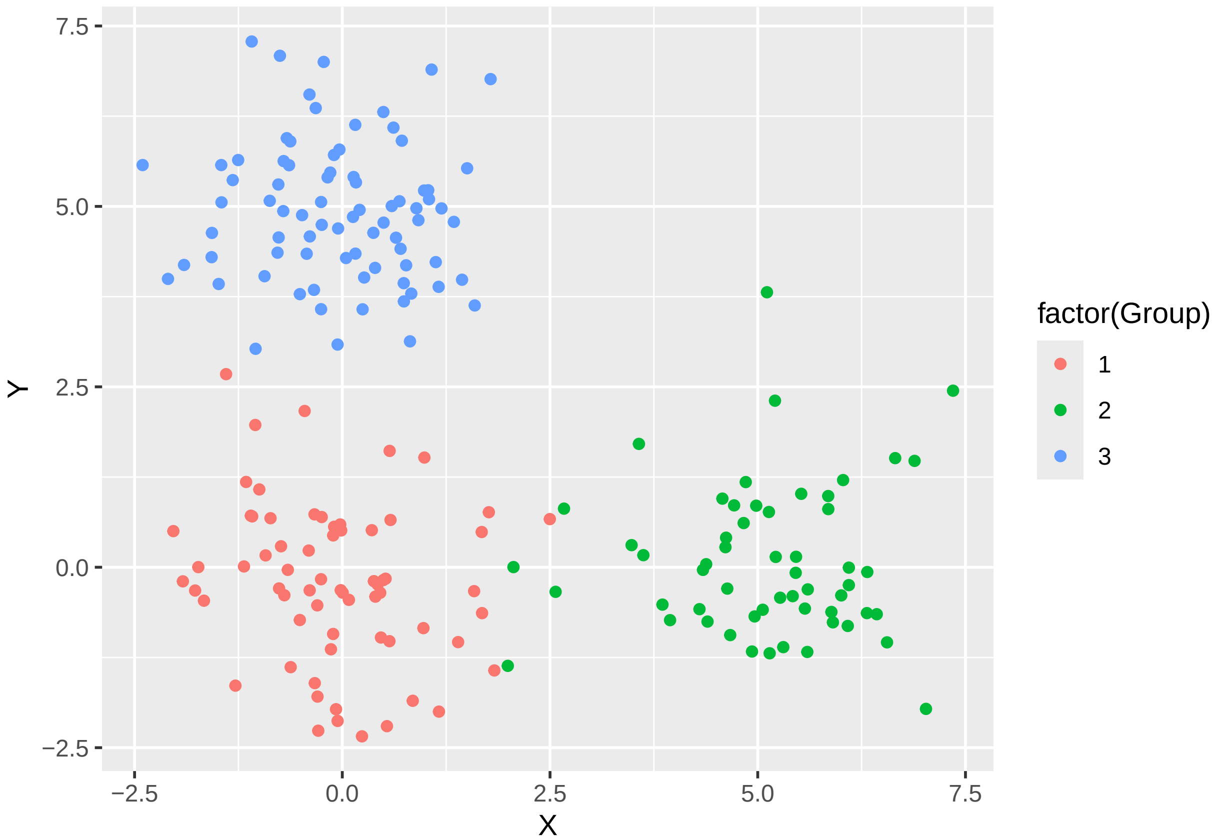
```r
set.seed(1)
n <- 200      # Number of data points
mu_list <- list(
  c(0, 0, 0),
  c(5, 0, 0),
```

```r
  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
```

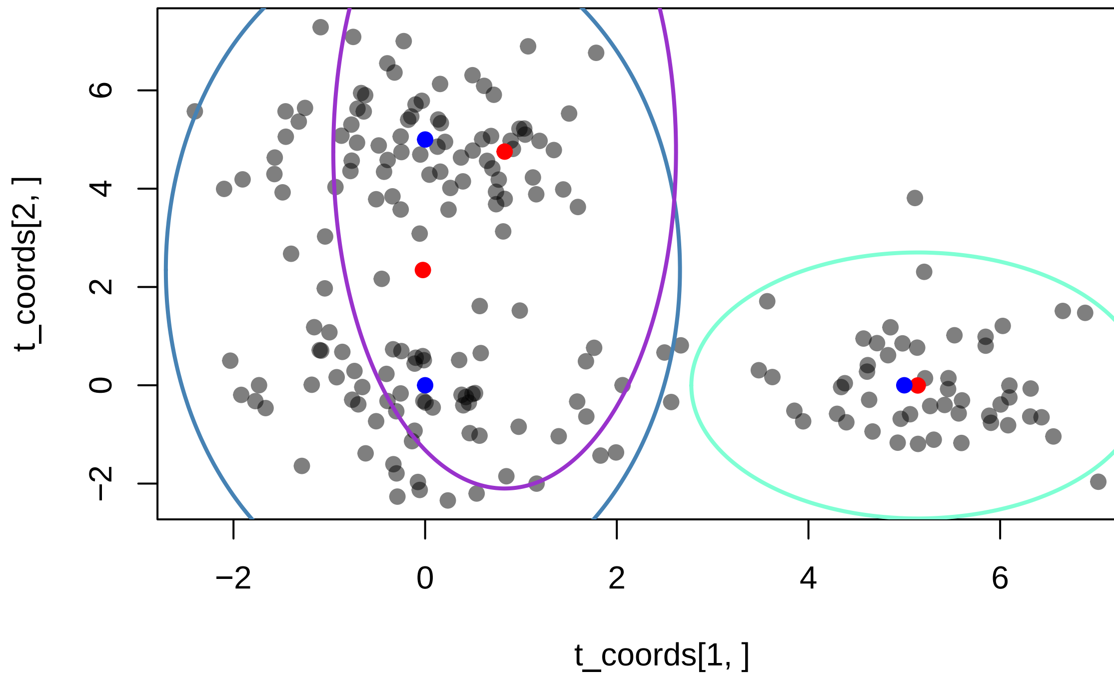
```
## file:///C:\Users\henry\AppData\Local\Temp\Rtmp4wpceu\file2a7841a373f9\widget2a78253a6fda.html screens
```




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




```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

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)
```

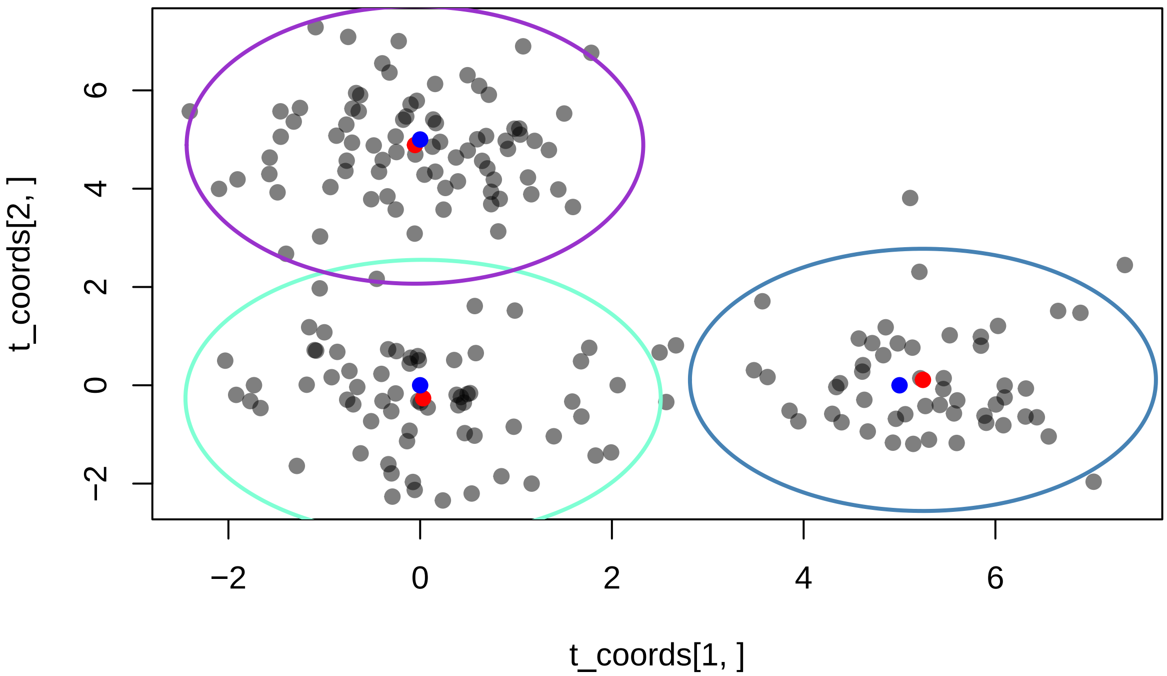


correct guess for centers

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

```
## $mu_error
## [1] 1.127488
##
## $score
## [1] 0
##
## $sigma_error
## [1] 4.672425
```

```
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

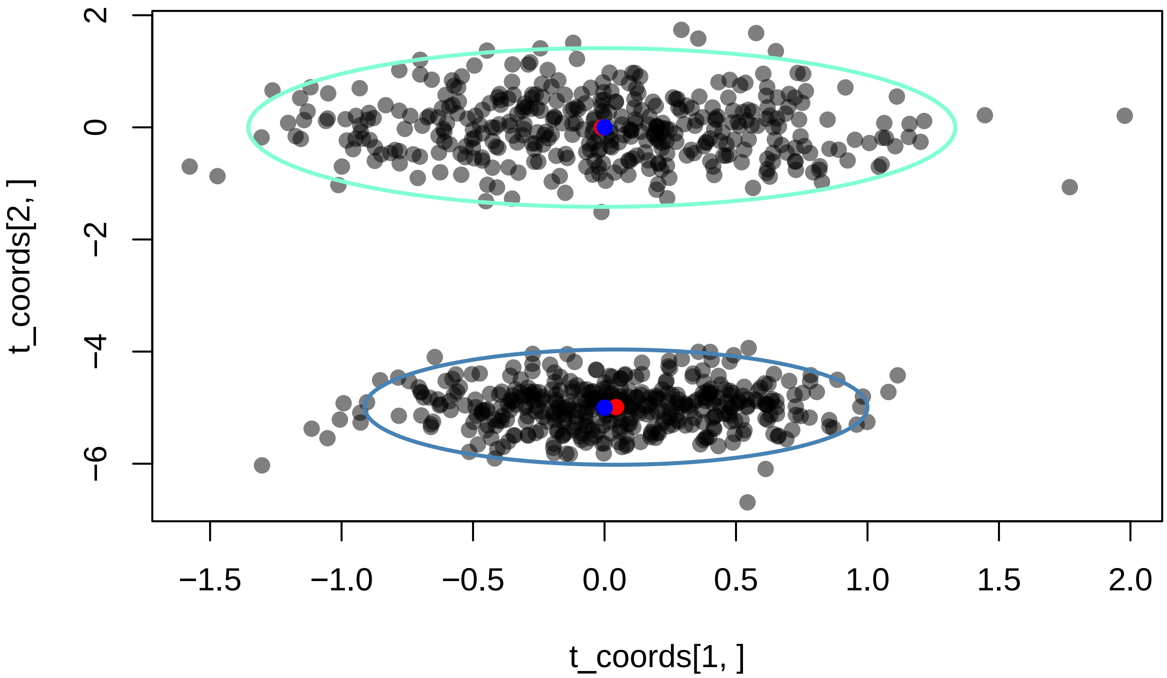
```
eval_EM(res = EMR_res, mu_list = mu_list, dim = dim, I2_list=I2_list)
```

```
## $mu_error
## [1] 0.2315216
##
## $score
## [1] 0
##
## $sigma_error
## [1] 0.3150871
```

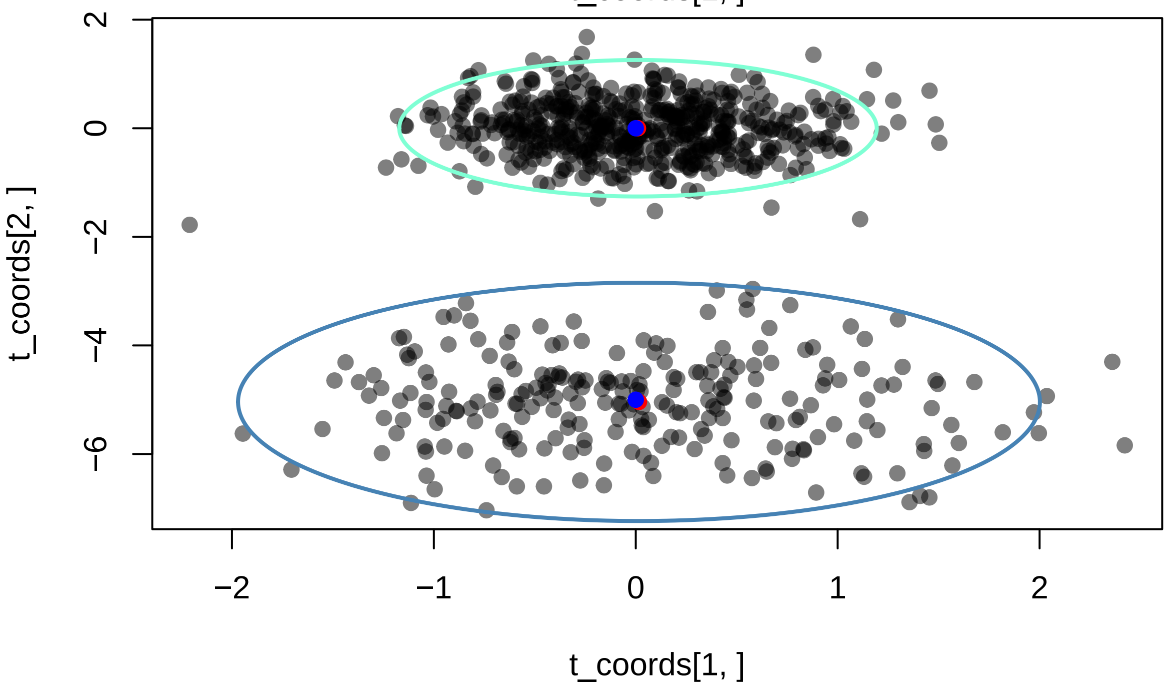
varing centers

```
set.seed(2)
m = 20 # NUMBER OF TESTS
n <- 800      # 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(0,5), c(5,0), c(-5,0), c(0,-5))
  mu_list = sample(mu_complete_list, size = k)
```

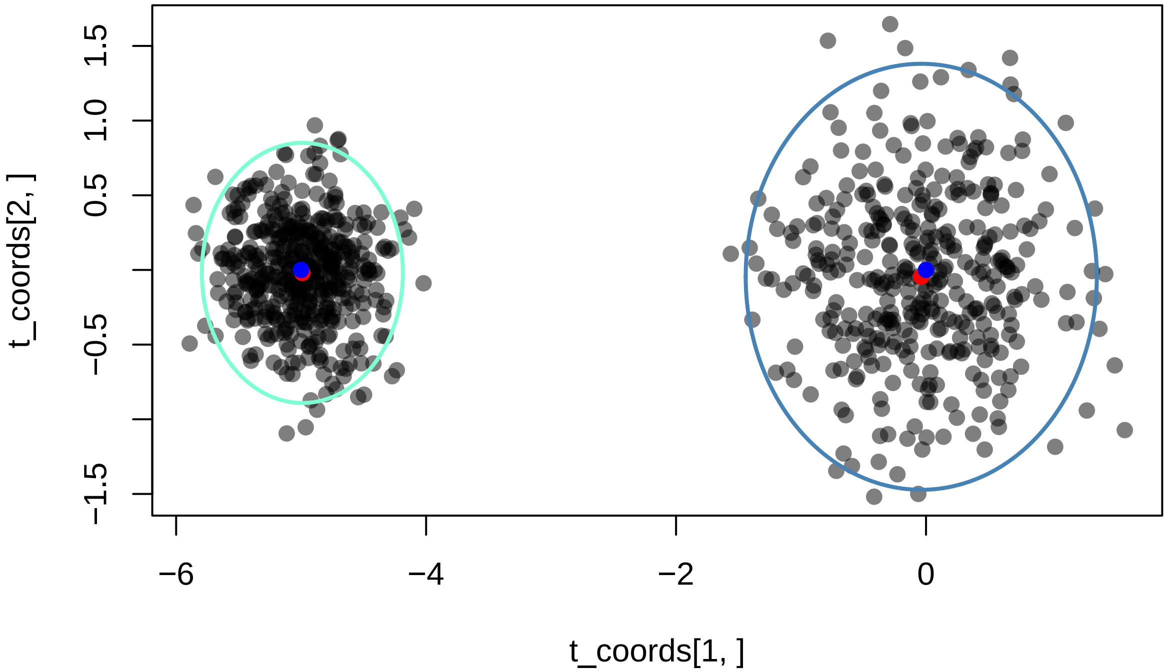
```
  prob = abs(rnorm(k, mean = 0, sd = 1)) + 1
  I2_variance = abs(rnorm(k, mean = 0, sd = 0.3))+0.1
  I2_list = lapply(I2_variance, function(x) diag(x,dim))

  #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=1e3) # 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
  }

}
```

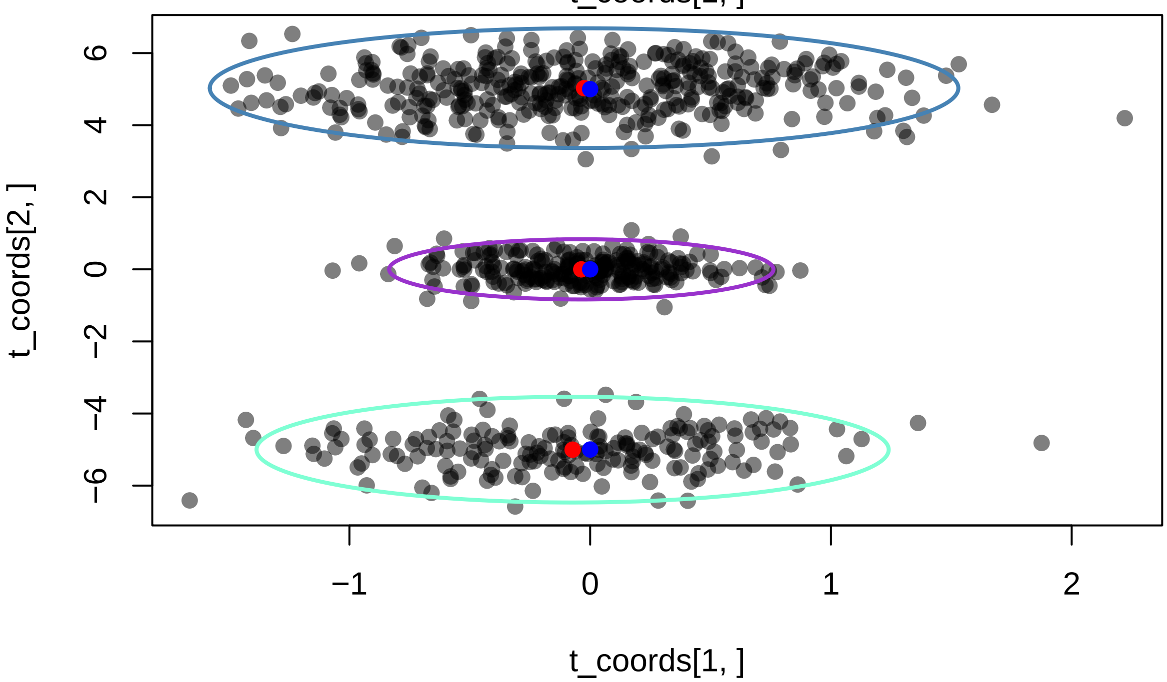


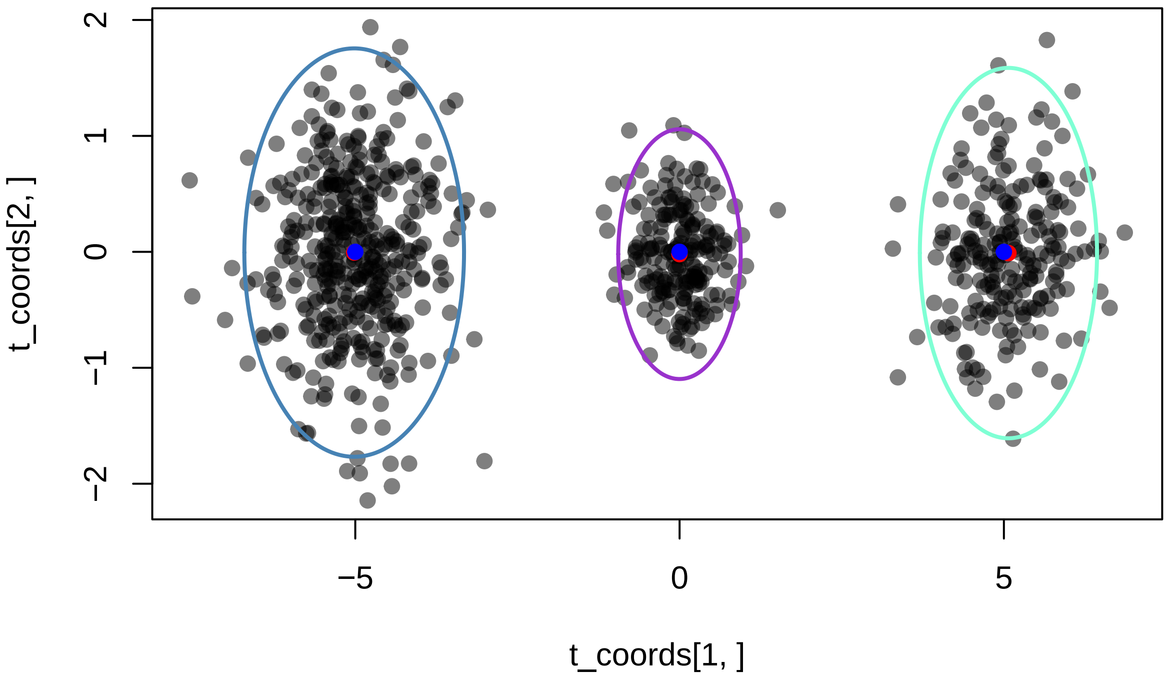
t_coords[2,]
2
0
−2
−4
−6
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
t_coords[1,]



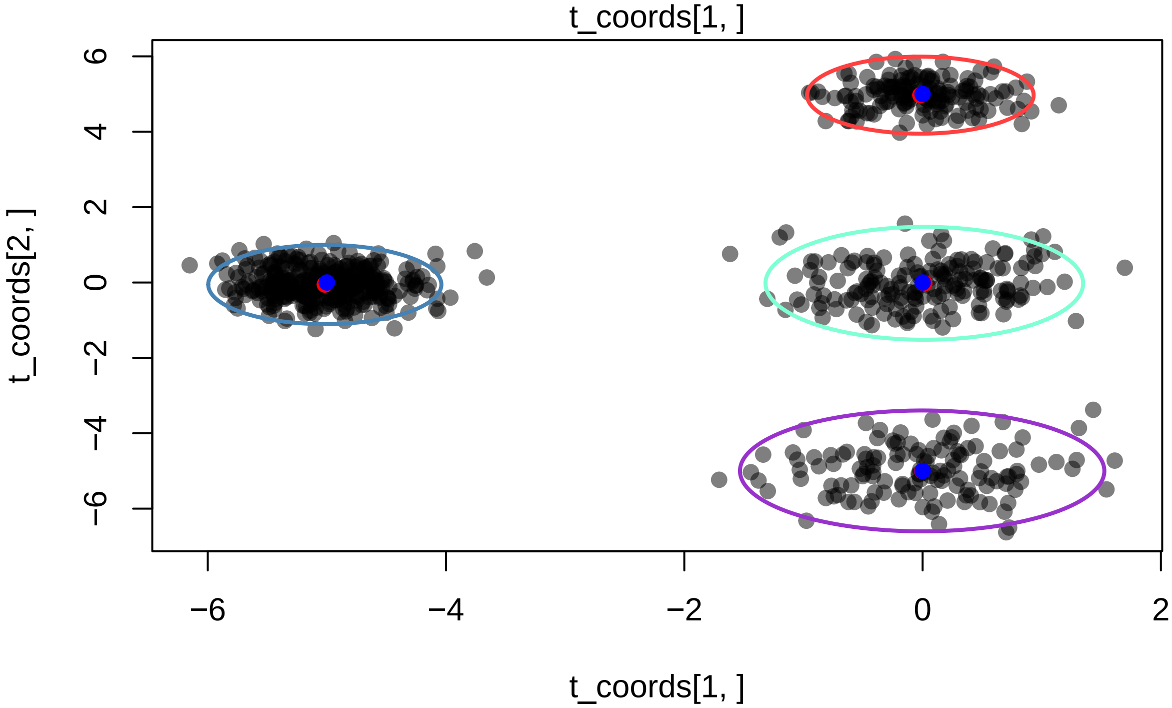
t_coords[2,]
2
0
−2
−4
−6
−2
−1
0
1
2
t_coords[1,]


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

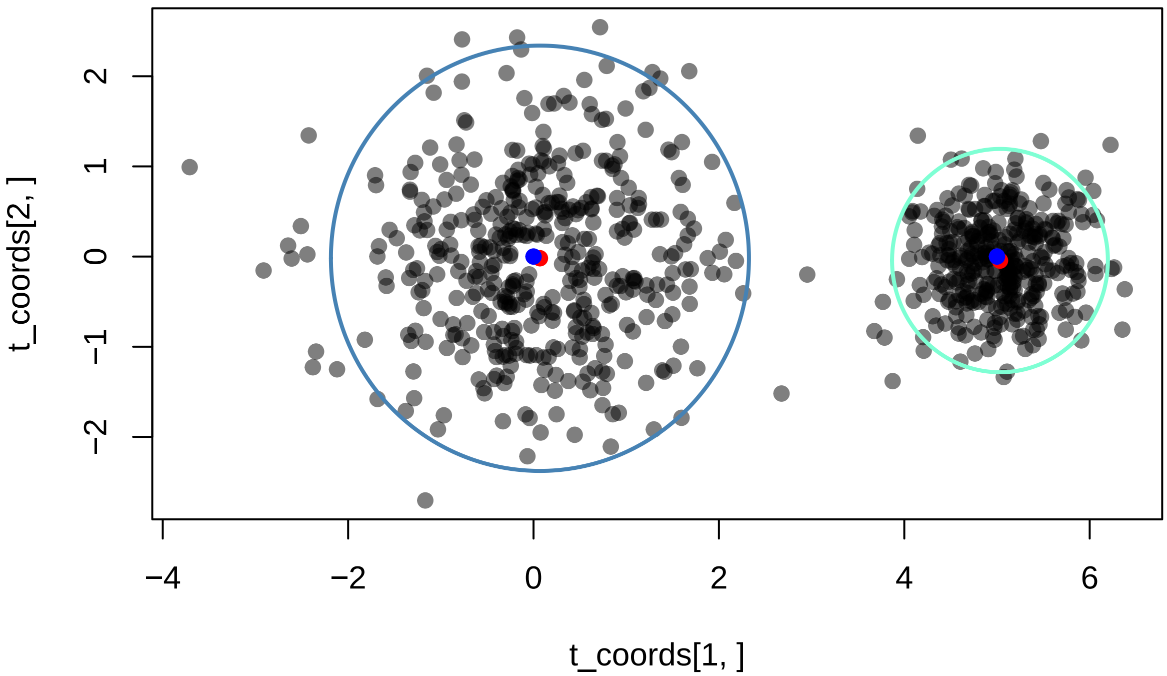

t_coords[2,]
t_coords[1,]
−1
0
1
2
6
4
2
0
−2
−4
−6

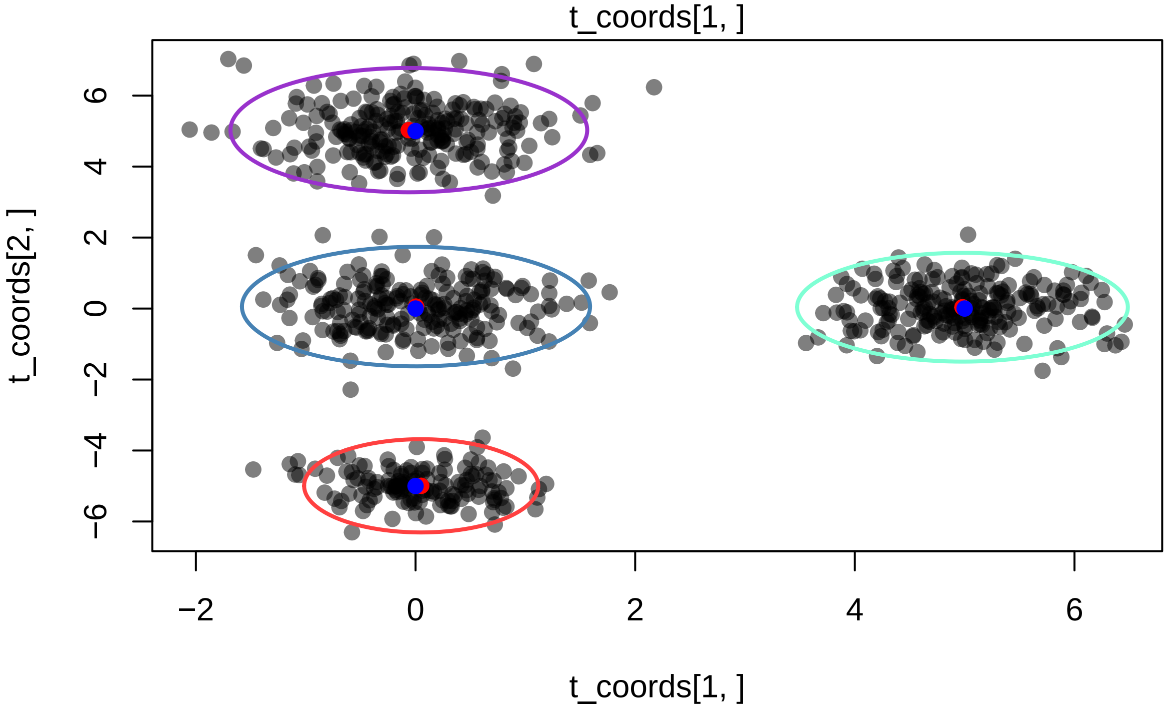


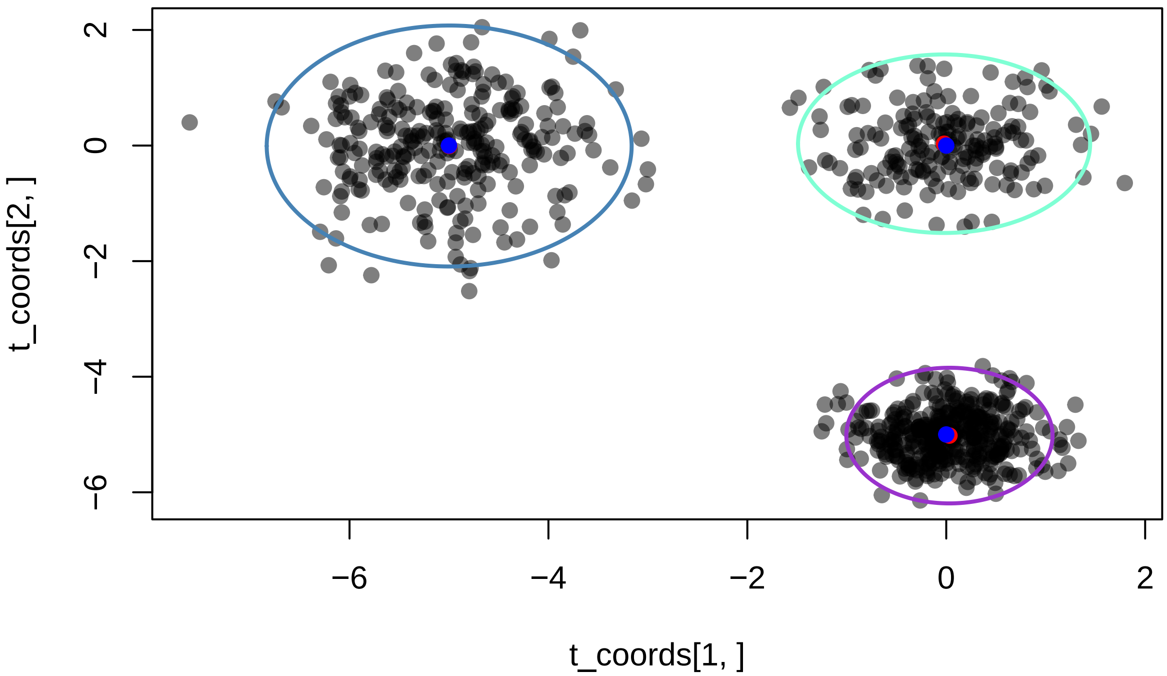
t_coords[2,]
t_coords[1,]
2
1
0
−1
−2
−5
0
5

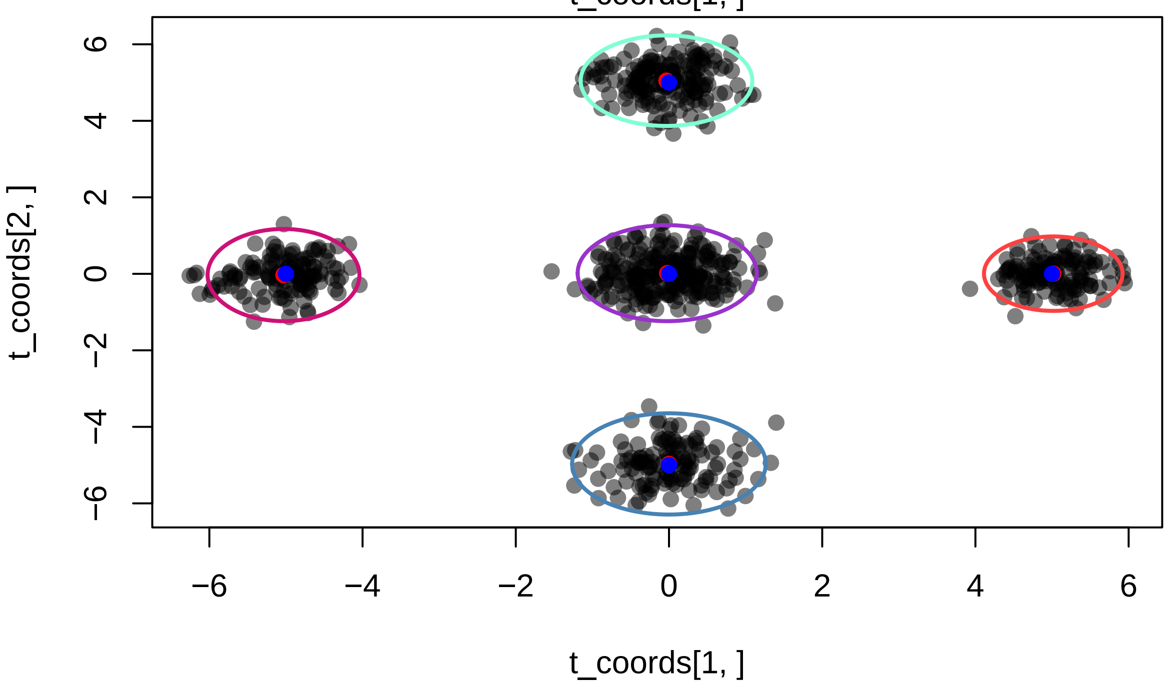


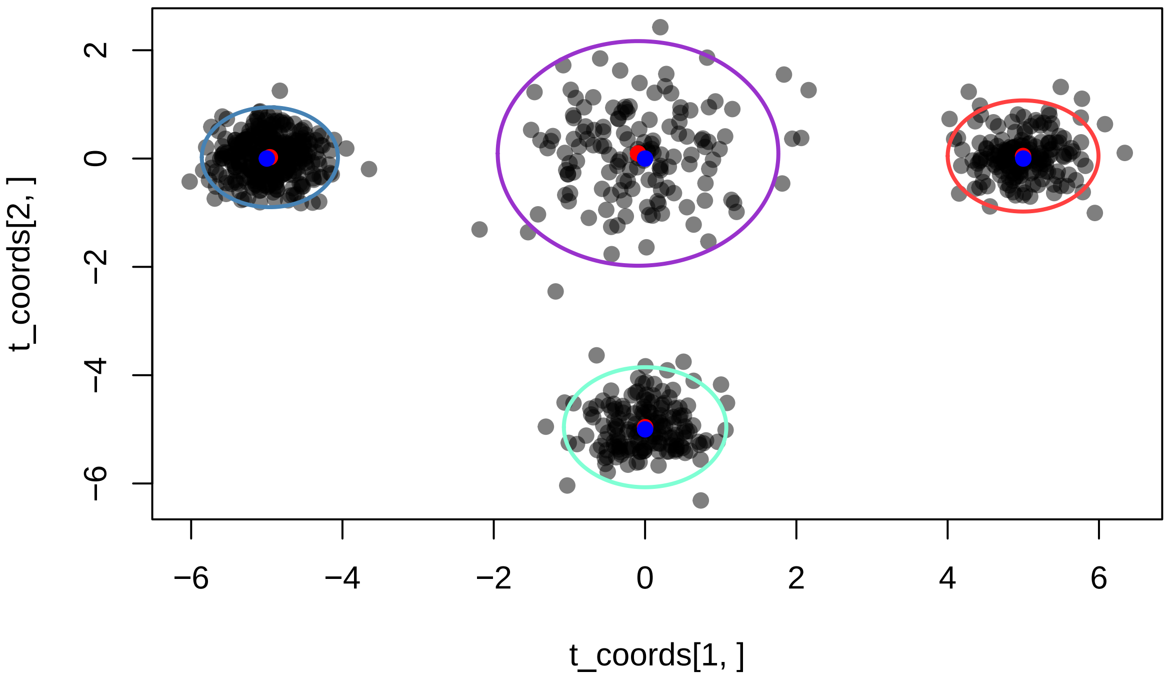
t_coords[2,]
t_coords[1,]
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2


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

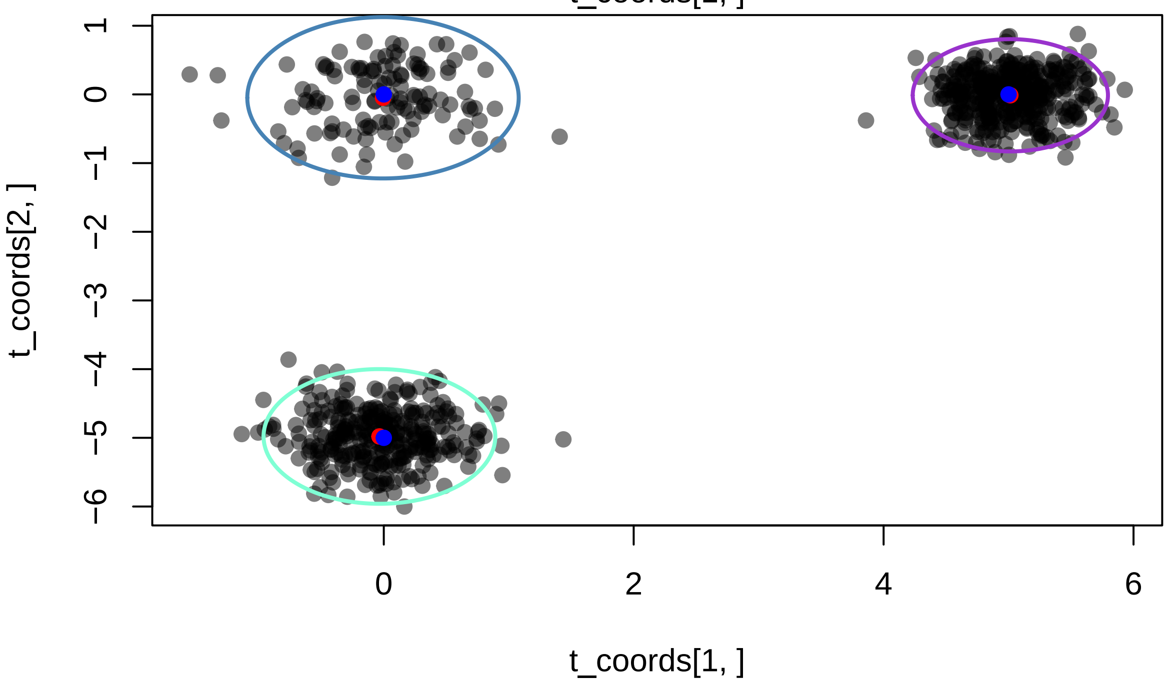

6
4
2
0
−2
−4
−6
t_coords[2,]
−2
0
2
4
6
t_coords[1,]


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

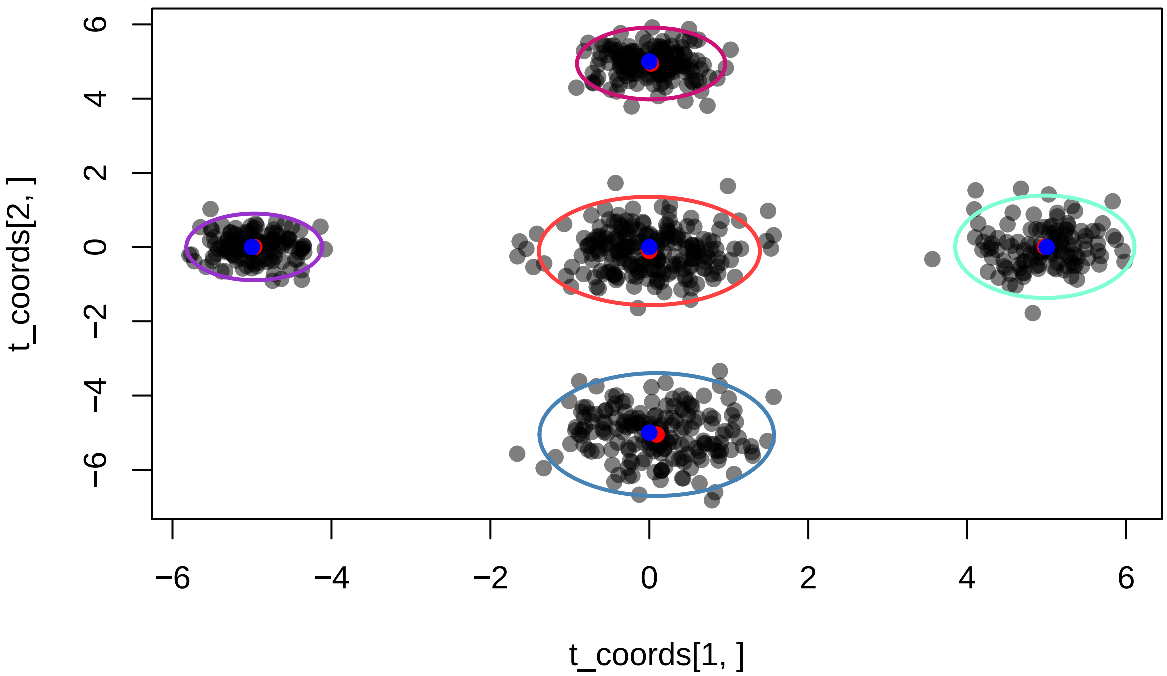

t_coords[2,]
t_coords[1,]
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

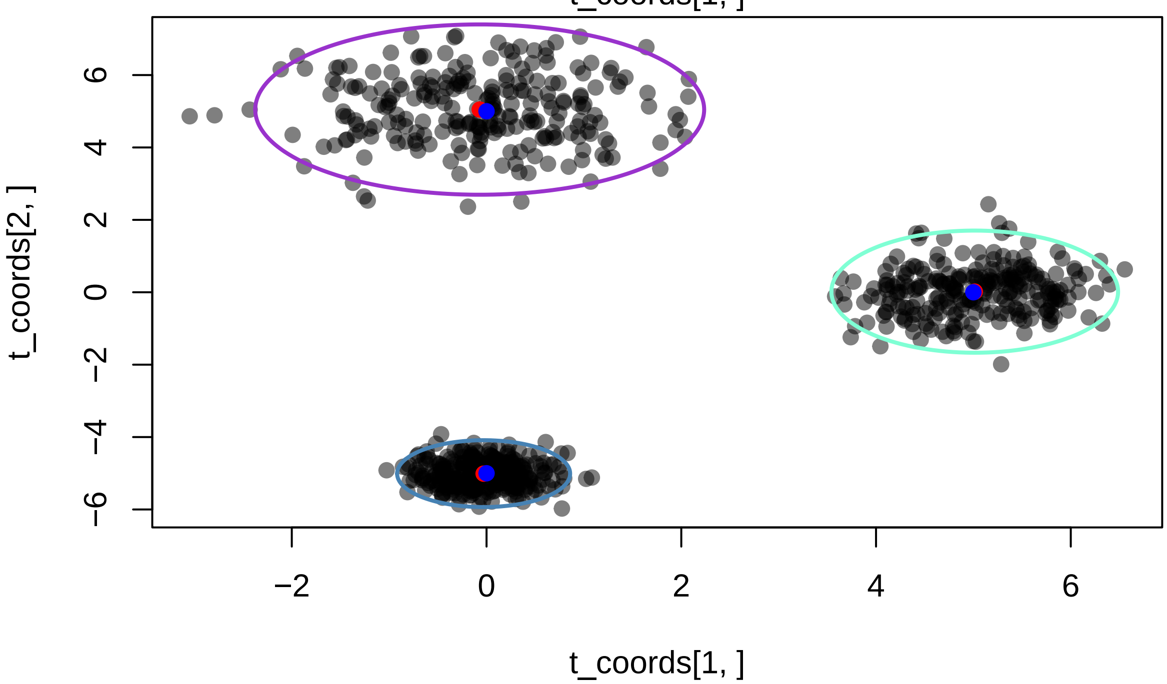


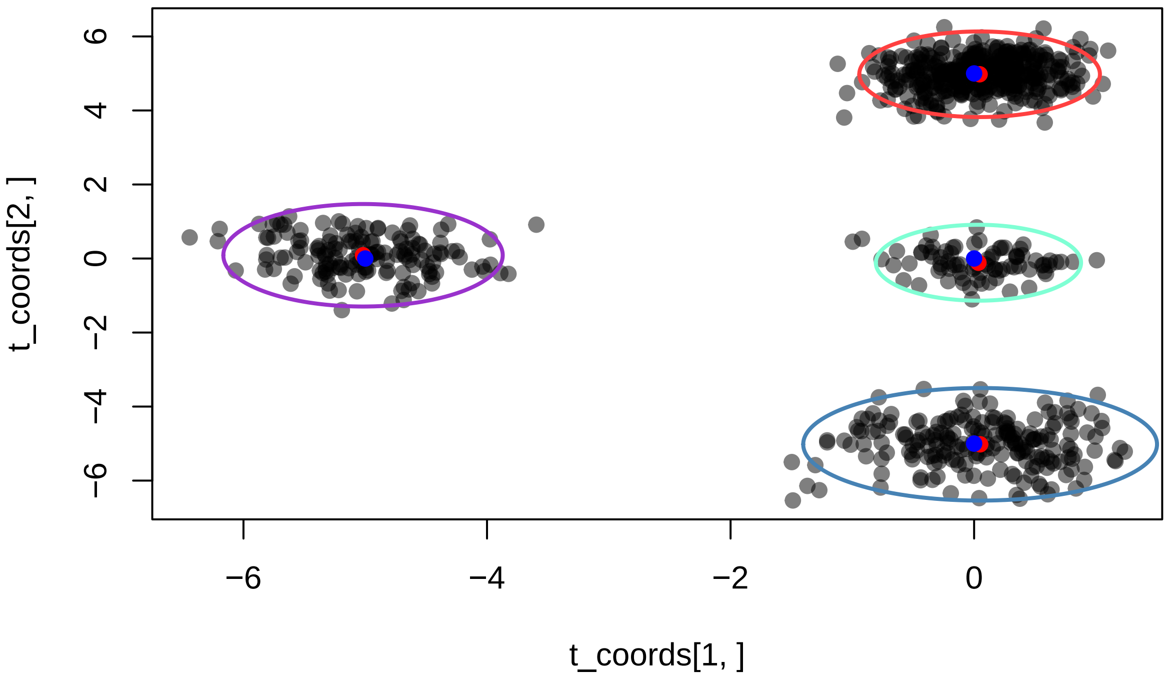
2
0
−2
−4
−6
t_coords[2,]
−6
−4
−2
0
2
4
6
t_coords[1,]



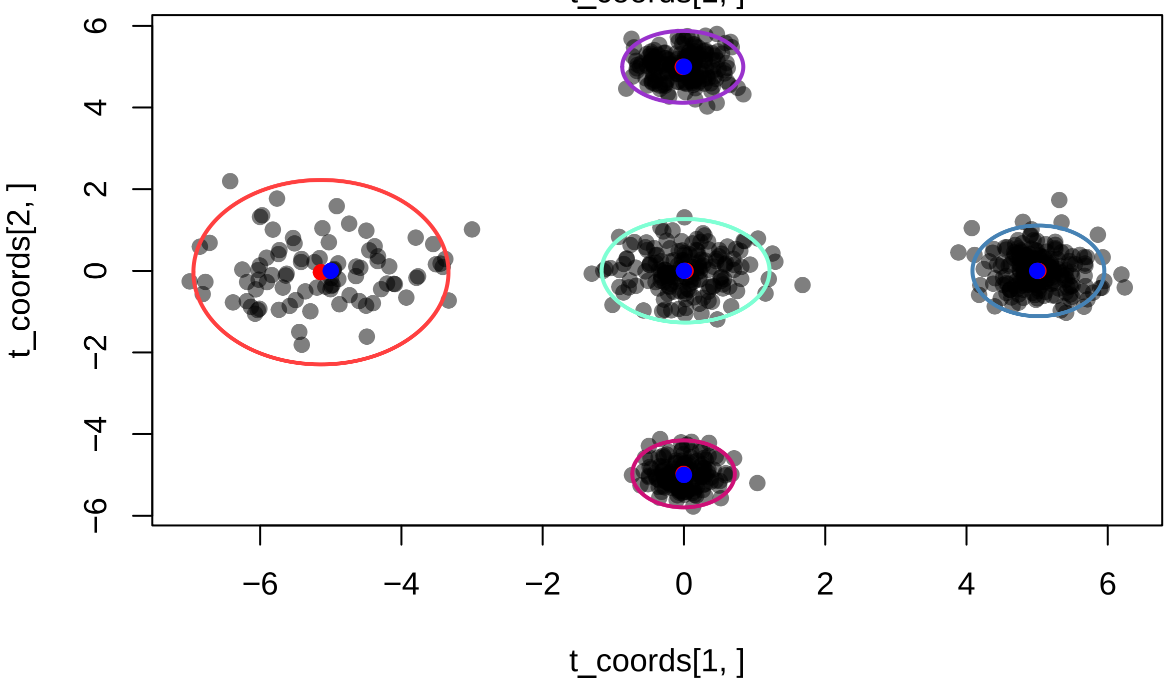
1
0
−1
−2
−3
−4
−5
−6
t_coords[2,]
0
2
4
6
t_coords[1,]


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

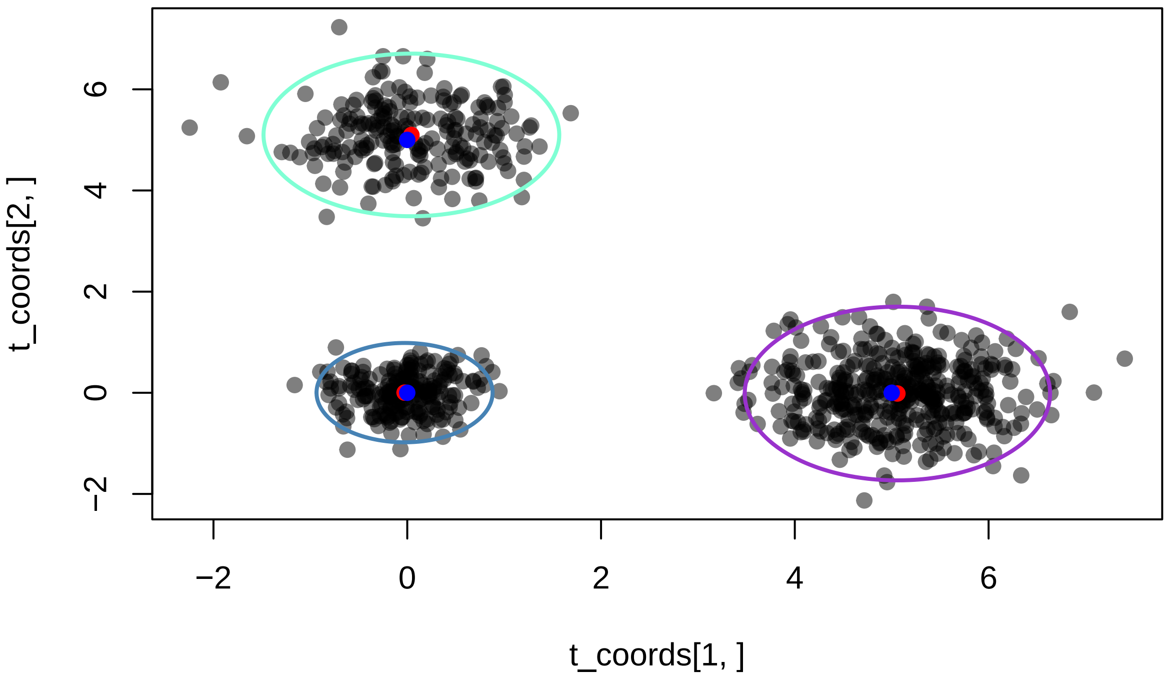

6
4
2
0
−2
−4
−6
t_coords[2,]
−2
0
2
4
6
t_coords[1,]

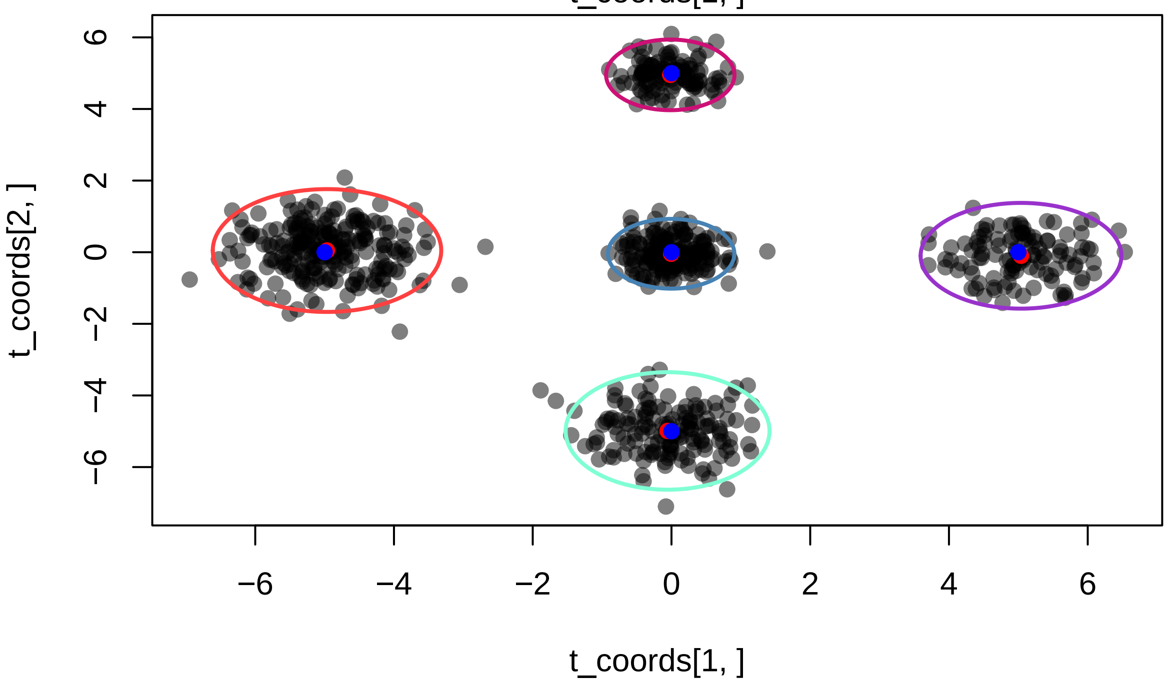


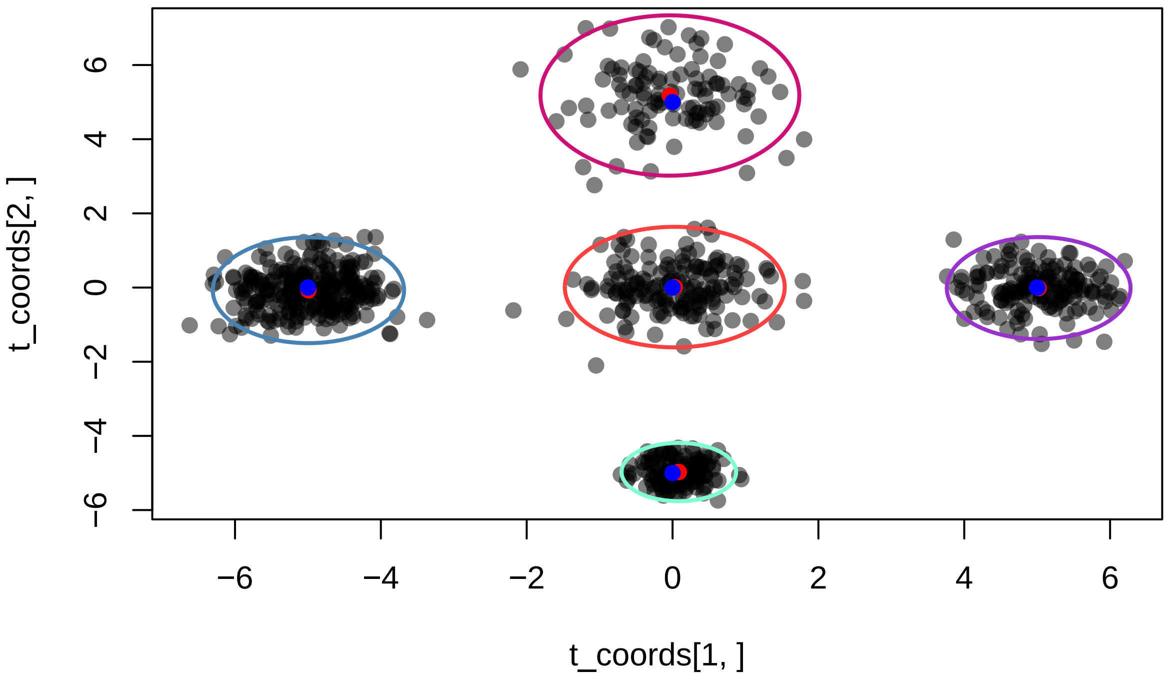
6
4
2
0
−2
−4
−6
t_coords[2,]
−6
−4
−2
0
t_coords[1,]

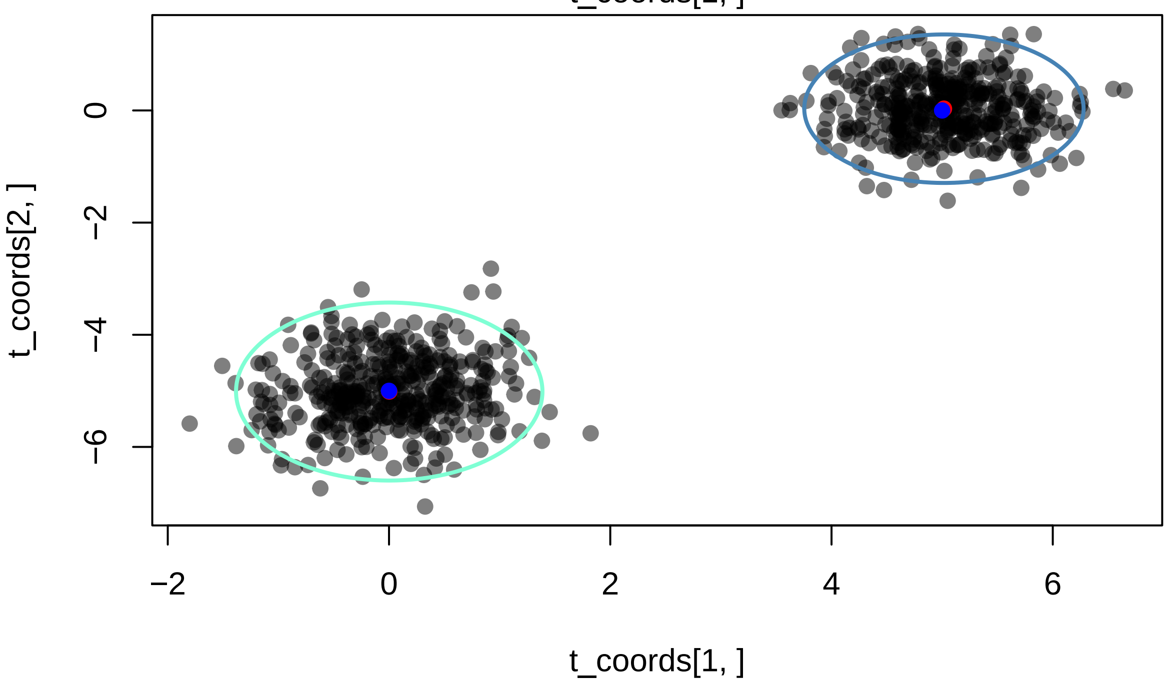


6
4
2
0
−2
−4
−6
t_coords[2,]
−6
−4
−2
0
2
4
6
t_coords[1,]


t_coords[2,]
t_coords[1,]
−2
0
2
4
6


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






```
all_results = data.frame(cbind(robust_mu_error_list, robust_score_list, robust_sigma_error_list, origina
```

```
library(knitr)

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

```
## [1] "times of predicting wrong clusters numbers:  0"
```

```
useful_results = subset(all_results, robust_score_list!=-1)
```

```
kable(useful_results, col.names = c("robust mean mu error", "robust center accuracy score", "robust mean
```

robust mean mu error	robust center accuracy score	robust mean sigma error	regular mean mu error	regular center accuracy score	regular mean sigma error
0.0277	1.0000	0.0221	0.0277	1.0000	0.0221
0.0269	1.0000	0.0831	0.0269	1.0000	0.0832
0.0409	1.0000	0.0154	0.0409	1.0000	0.0154
0.0493	1.0000	0.0421	0.0493	1.0000	0.0422
0.0361	1.0000	0.0532	0.0361	1.0000	0.0532
0.0304	1.0000	0.0580	0.0304	1.0000	0.0579
0.0634	1.0000	0.0534	0.0634	1.0000	0.0533
0.0539	1.0000	0.0586	0.0539	1.0000	0.0586
0.0280	1.0000	0.0725	0.0280	1.0000	0.0725
0.0363	1.0000	0.0337	0.0363	1.0000	0.0337
0.0666	0.7500	0.0389	0.0666	0.7500	0.0389
0.0362	1.0000	0.0212	0.0362	1.0000	0.0212
0.0647	0.6000	0.0413	0.0647	0.6000	0.0413
0.0460	1.0000	0.0484	0.0460	1.0000	0.0484
0.0779	0.7500	0.0492	0.0779	0.7500	0.0492
0.0454	0.8000	0.0690	0.0454	0.8000	0.0690
0.0652	0.6667	0.0427	0.0652	0.6667	0.0427
0.0632	0.8000	0.0418	0.0632	0.8000	0.0418
0.0805	0.8000	0.0728	0.0805	0.8000	0.0728
0.0239	1.0000	0.0509	0.0239	1.0000	0.0509

mean mu error is the mean euclidean distance between the true and predicted centers.
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) *mean sigma error* is the mean euclidean distance between the true and predicted covariance matrix.

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