machine learning(732A99) lab1 Block2

Anubhav Dikshit(anudi287), Lennart Schilling(lensc874), Thijs Quast(thiqu264)

04 December 2018

Contents

Contributions

During the lab, Lenart focused on assignment 2 using loops, Thijs focused on assignment 1 and Anuhav focused on assignment 2 using matrix. All codes and analysis was indepedently done and is also reflected in the individual reports.

Assignment 1

1. Ensemble Methods

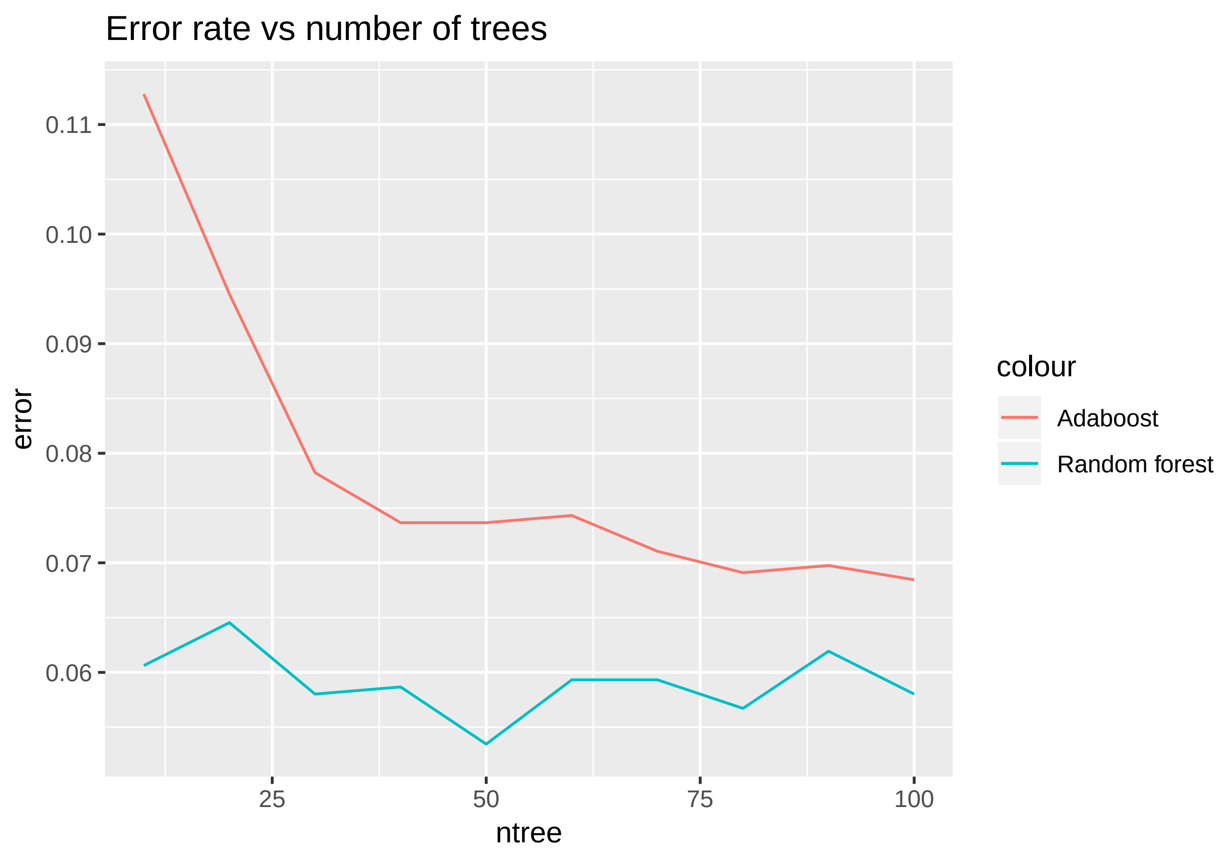
```
# Loading packages and importing files ####
sp <- read.csv2("spambase.data", header = FALSE, sep = ",", stringsAsFactors = FALSE)
num_sp <- data.frame(data.matrix(sp))
num_sp$V58 <- factor(num_sp$V58)
```

```
# shuffling data and dividing into train and test ####
n <- dim(num_sp)[1]
ncol <- dim(num_sp)[2]
set.seed(1234567890)
id <- sample(1:n, floor(n*(2/3)))
train <- num_sp[id,]
test <- num_sp[-id,]
```

```
# Adaboost
ntree <- c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
error <- c()

for (i in seq(from = 10, to = 100, by = 10)){
bb <- blackboost(V58 ~., data = train, control = boost_control(mstop = i), family = AdaExp())
bb_predict <- predict(bb, newdata = test, type = c("class"))
confusion_bb <- table(test$V58, bb_predict)
miss_class_bb <- (confusion_bb[1,2] + confusion_bb[2,1])/nrow(test)
error[(i/10)] <- miss_class_bb
}

error_df <- data.frame(cbind(ntree, error))
```

```
# Random forest ####
ntree_rf <- c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
error_rf <- c()

for (i in seq(from = 10, to = 100, by = 10)){
rf <- randomForest(V58 ~., data = train, ntree= 10)
rf_predict <- predict(rf, newdata = test, type = c("class"))
confusion_rf <- table(test$V58, rf_predict)
miss_class_rf <- (confusion_rf[1,2] + confusion_rf[2,1])/nrow(test)
error_rf[i/10] <- miss_class_rf
}

error_df_rf <- data.frame(cbind(ntree_rf, error_rf))
```

```
df <- cbind(error_df, error_df_rf)
df <- df[, -3]

plot_final <- ggplot(df, aes(ntree)) +
  geom_line(aes(y=error, color = "Adaboost")) +
  geom_line(aes(y=error_rf, color = "Random forest"))

plot_final <- plot_final + ggtitle("Error rate vs number of trees")
plot_final
```



The error rate for the AdaBoost model are clearly going down when the number of trees increases. Finally the model arrives at an error rate below 7% when 100 trees are included in the model. For the randomforest the pattern is less obvious, the error rate seems to go up and down as the number of trees in the model increases. 50 trees result in the lowest error rate. This error rate is also lower than the error rate produced by the best Adaboost model (100 trees). Therefore, for this spam classification, a randomforest with 50 trees seems to be most suitable.

2. Mixture Models

Using loops

To compare the results for K = 2,3,4, the em-function provides a graphical analysis for every iteration. The function includes comments which explain what I did at which step to create the EM algorithm. The function will be finally run with K = 2,3,4.

```
em_loop = function(K) {
# Initializing data
set.seed(1234567890)
max_it = 100 # max number of EM iterations
min_change = 0.1 # min change in log likelihood between two consecutive EM iterations
N = 1000 # number of training points
D = 10 # number of dimensions
x = matrix(nrow=N, ncol = D) # training data
true_pi = vector(length = K) # true mixing coefficients
true_mu = matrix(nrow = K, ncol = D) # true conditional distributions
```

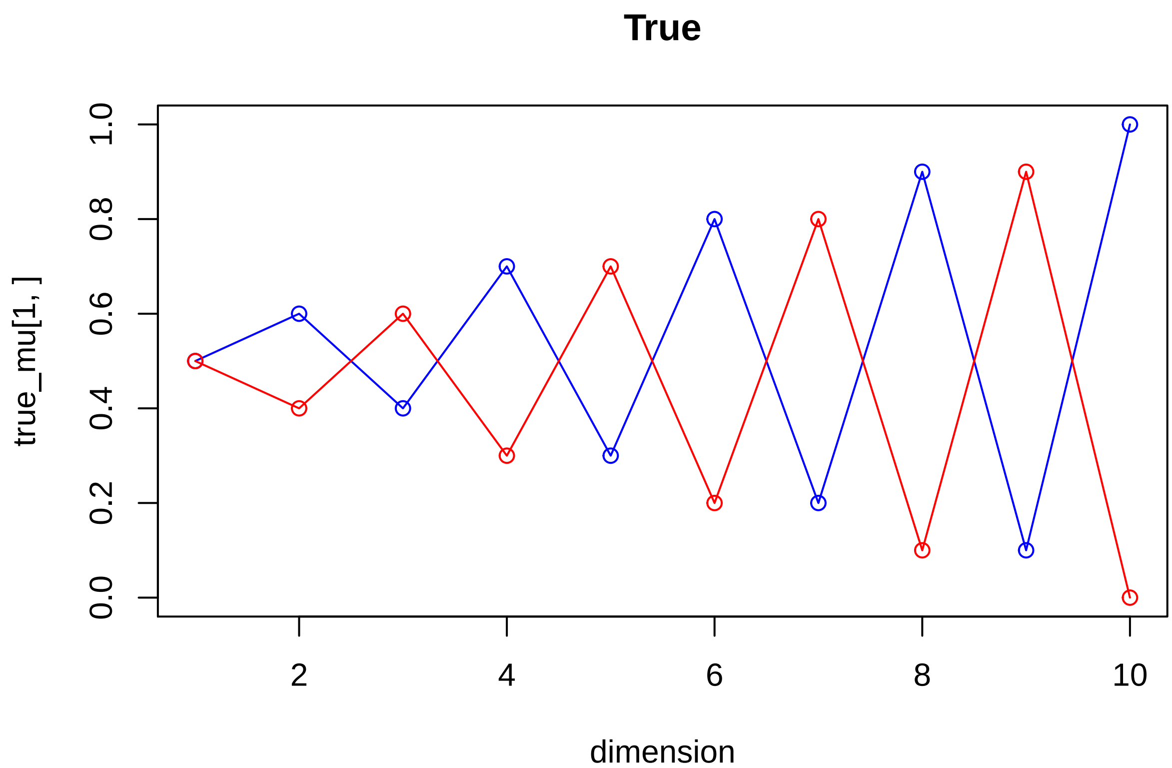
```r
true_pi = c(rep(1/K, K))
if (K == 2) {
true_mu[1,] = c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,] = c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
plot(true_mu[1,], type = "o", xlab = "dimension", col = "blue",
ylim = c(0,1), main = "True")
points(true_mu[2,], type="o", xlab = "dimension", col = "red",
main = "True")
} else if (K == 3) {
true_mu[1,] = c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,] = c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
true_mu[3,] = c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
plot(true_mu[1,], type = "o", xlab = "dimension", col = "blue", ylim=c(0,1),
main = "True")
points(true_mu[2,], type = "o", xlab = "dimension", col = "red",
main = "True")
points(true_mu[3,], type = "o", xlab = "dimension", col = "green",
main = "True")
} else {
true_mu[1,] = c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,] = c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
true_mu[3,] = c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
true_mu[4,] = c(0.3,0.5,0.5,0.7,0.5,0.5,0.5,0.5,0.4,0.5)
plot(true_mu[1,], type = "o", xlab = "dimension", col = "blue",
ylim = c(0,1), main = "True")
points(true_mu[2,], type = "o", xlab = "dimension", col = "red",
main = "True")
points(true_mu[3,], type = "o", xlab = "dimension", col = "green",
main = "True")
points(true_mu[4,], type = "o", xlab = "dimension", col = "yellow",
main = "True")
}
z = matrix(nrow = N, ncol = K) # fractional component assignments
pi = vector(length = K) # mixing coefficients
mu = matrix(nrow = K, ncol = D) # conditional distributions
llik = vector(length = max_it) # log likelihood of the EM iterations
# Producing the training data
for(n in 1:N) {
k = sample(1:K, 1, prob=true_pi)
for(d in 1:D) {
x[n,d] = rbinom(1, 1, true_mu[k,d])
}
}
# Random initialization of the paramters
pi = runif(K, 0.49, 0.51)
pi = pi / sum(pi)
for(k in 1:K) {
mu[k,] = runif(D, 0.49, 0.51)
}
#EM algorithm
for(it in 1:max_it) {
# Plotting mu
# Defining plot title
```

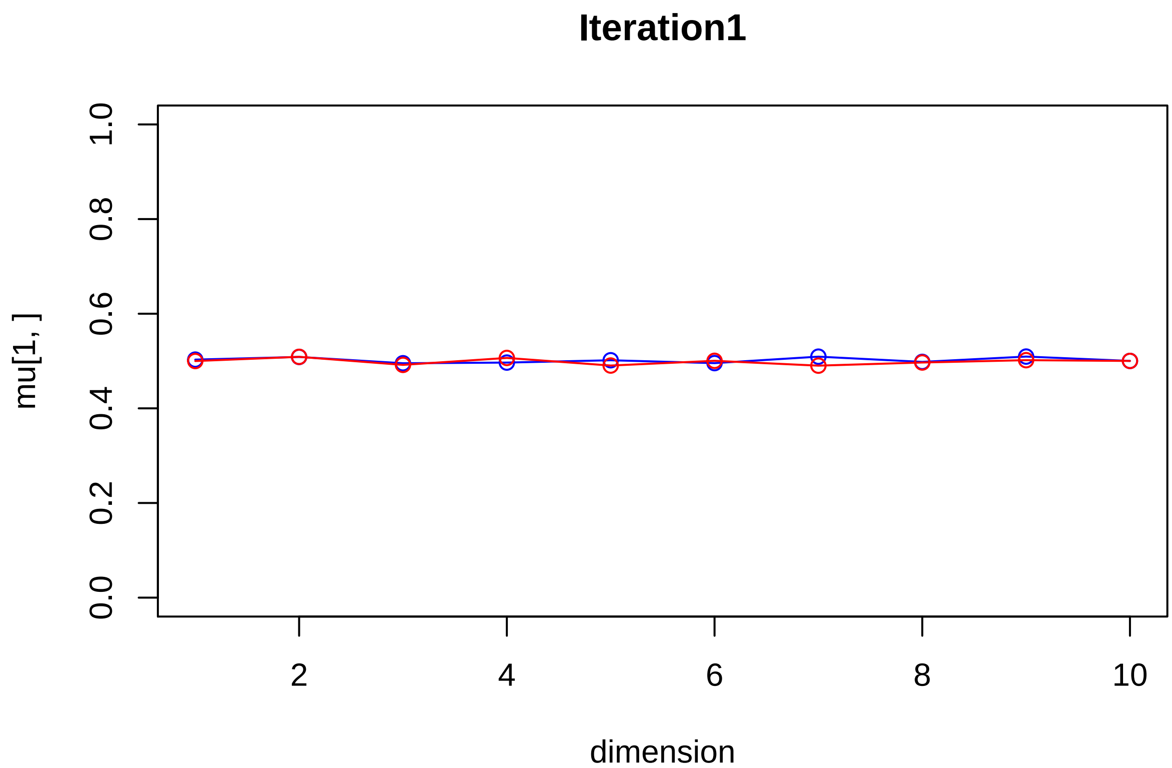
```r
title = paste0("Iteration", it)
if (K == 2) {
plot(mu[1,], type = "o", xlab = "dimension", col = "blue", ylim = c(0,1), main = title)
points(mu[2,], type = "o", xlab = "dimension", col = "red", main = title)
} else if (K == 3) {
plot(mu[1,], type = "o", xlab = "dimension", col = "blue", ylim = c(0,1), main = title)
points(mu[2,], type = "o", xlab = "dimension", col = "red", main = title)
points(mu[3,], type = "o", xlab = "dimension", col = "green", main = title)
} else {
plot(mu[1,], type = "o", xlab = "dimension", col = "blue", ylim = c(0,1), main = title)
points(mu[2,], type = "o", xlab = "dimension", col = "red", main = title)
points(mu[3,], type = "o", xlab = "dimension", col = "green", main = title)
points(mu[4,], type = "o", xlab = "dimension", col = "yellow", main = title)
}
Sys.sleep(0.5)
# E-step: Computation of the fractional component assignments
for (n in 1:N) {
# Creating empty matrix (column 1:K = p_x_given_k; column K+1 = p(x|all k)
p_x = matrix(data = c(rep(1,K), 0), nrow = 1, ncol = K+1)
# Calculating p(x|k) and p(x|all k)
for (k in 1:K) {
# Calculating p(x|k)
for (d in 1:D) {
p_x[1,k] = p_x[1,k] * (mu[k,d]^x[n,d]) * (1-mu[k,d])^(1-x[n,d])
}
p_x[1,k] = p_x[1,k] * pi[k] # weighting with pi[k]
# Calculating p(x|all k) (denominator)
p_x[1,K+1] = p_x[1,K+1] + p_x[1,k]
}
#Calculating z for n and all k
for (k in 1:K) {
z[n,k] = p_x[1,k] / p_x[1,K+1]
}
}
#Log likelihood computation
for (n in 1:N) {
for (k in 1:K) {
log_term = 0
for (d in 1:D) {
log_term = log_term + x[n,d] * log(mu[k,d]) + (1-x[n,d]) * log(1-mu[k,d])
}
llik[it] = llik[it] + z[n,k] * (log(pi[k]) + log_term)
}
}
cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
flush.console()
# Stop if the log likelihood has not changed significantly
if (it != 1) {
if (abs(llik[it] - llik[it-1]) < min_change) {
break
}
}
#M-step: ML parameter estimation from the data and fractional component assignments
```

```r
# Updating pi
for (k in 1:K) {
pi[k] = sum(z[,k])/N
}
#Updating mu
for (k in 1:K) {
mu[k,] = 0
for (n in 1:N) {
    mu[k,] = mu[k,] + x[n,] * z[n,k]
}
mu[k,] = mu[k,] / sum(z[,k])
}
}
#Printing pi, mu and development of log likelihood at the end
return(list(
pi = pi,
mu = mu,
logLikelihoodDevelopment = plot(llik[1:it],
type = "o",
main = "Development of the log likelihood",
xlab = "iteration",
ylab = "log likelihood")
))
}
```

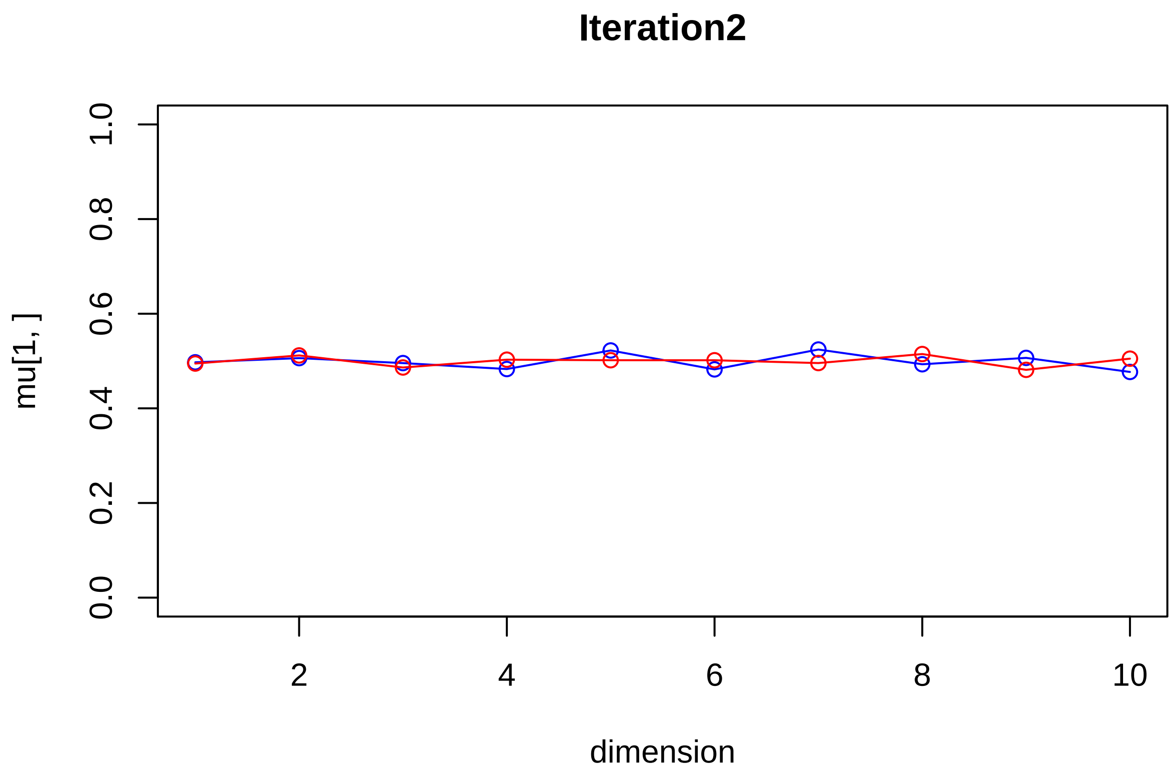
2. K=2

```r
em_loop(2)
```

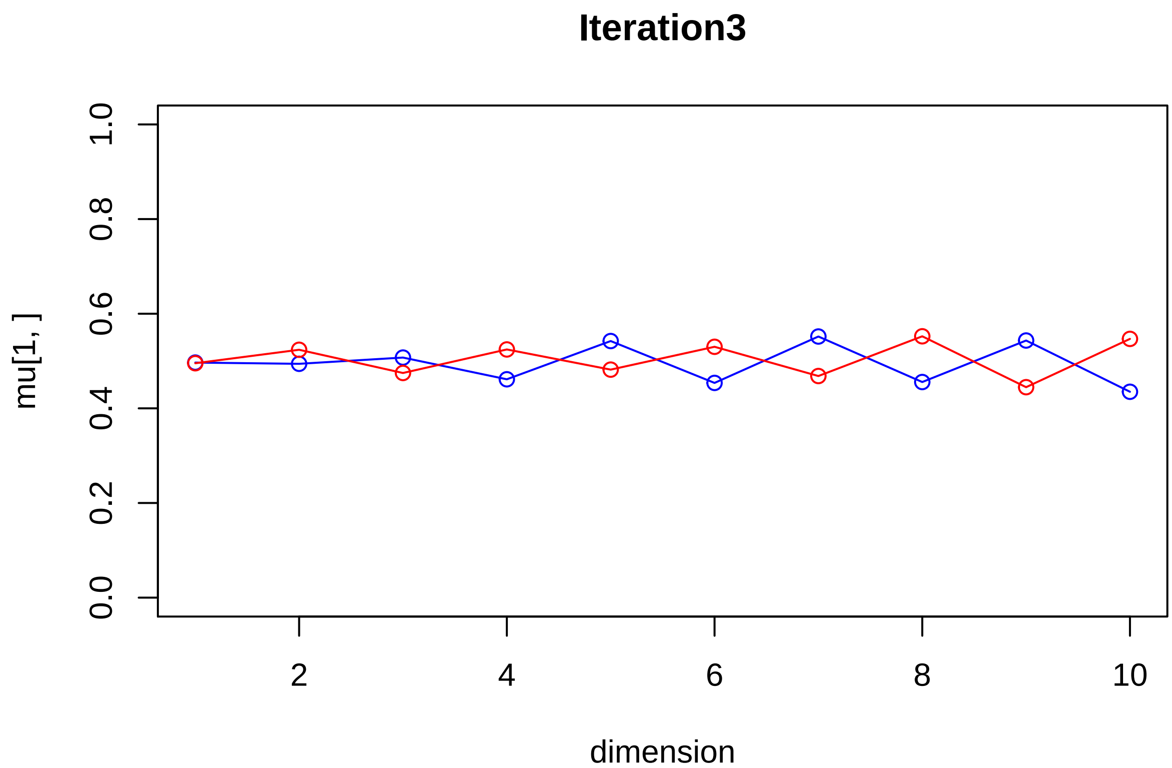

True
1.0
0.8
0.6
0.4
0.2
0.0
true_mu[1,]
2
4
6
8
10
dimension



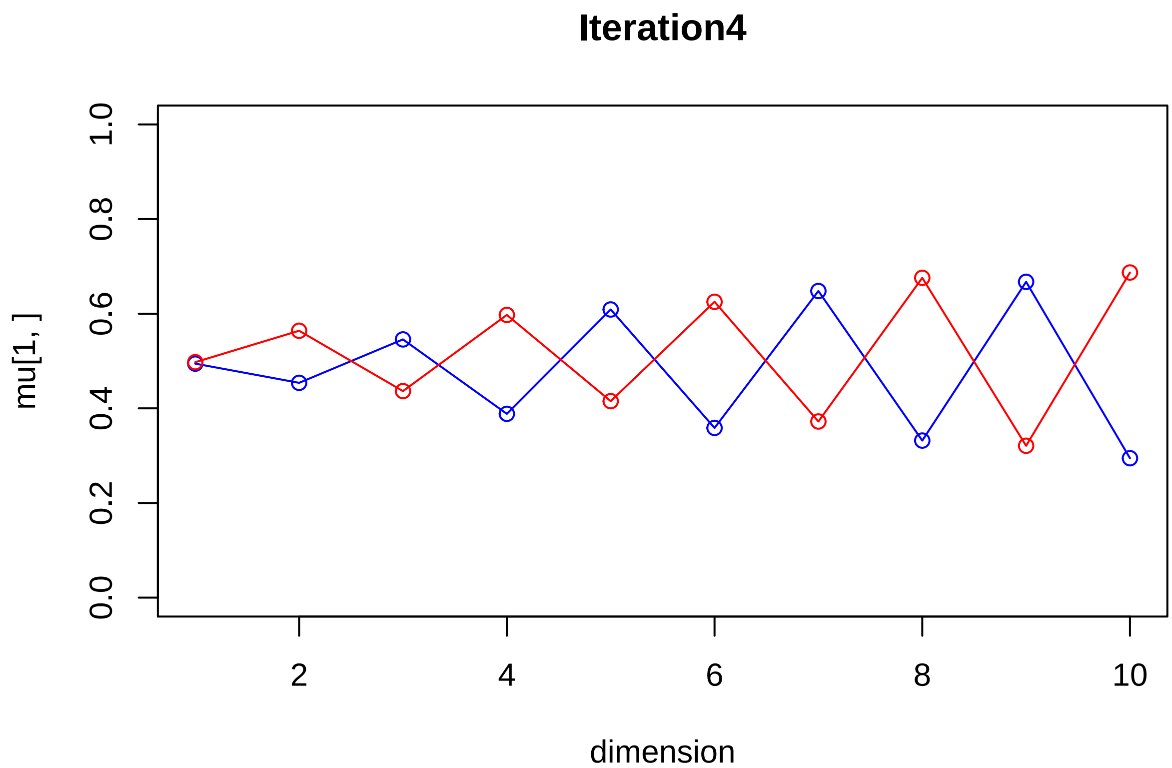

```
## iteration:  1 log likelihood:  -7623.897
```



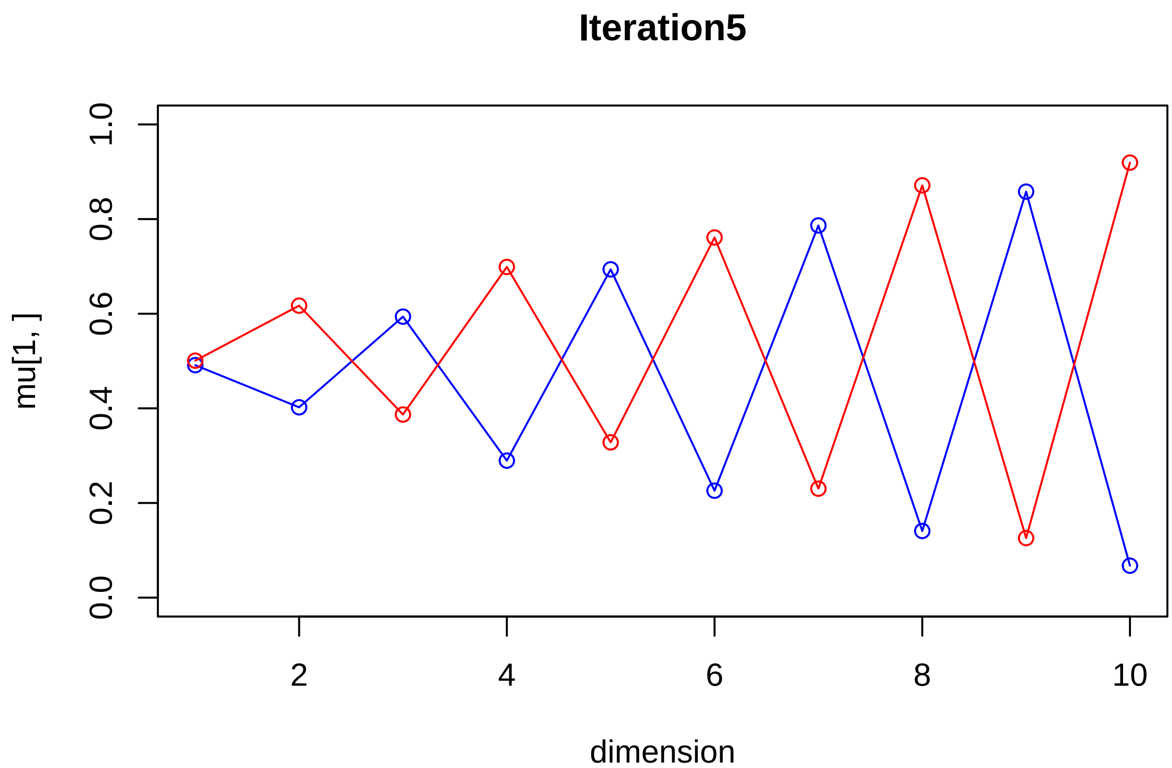

```
## iteration:  2 log likelihood:  -7610.745
```



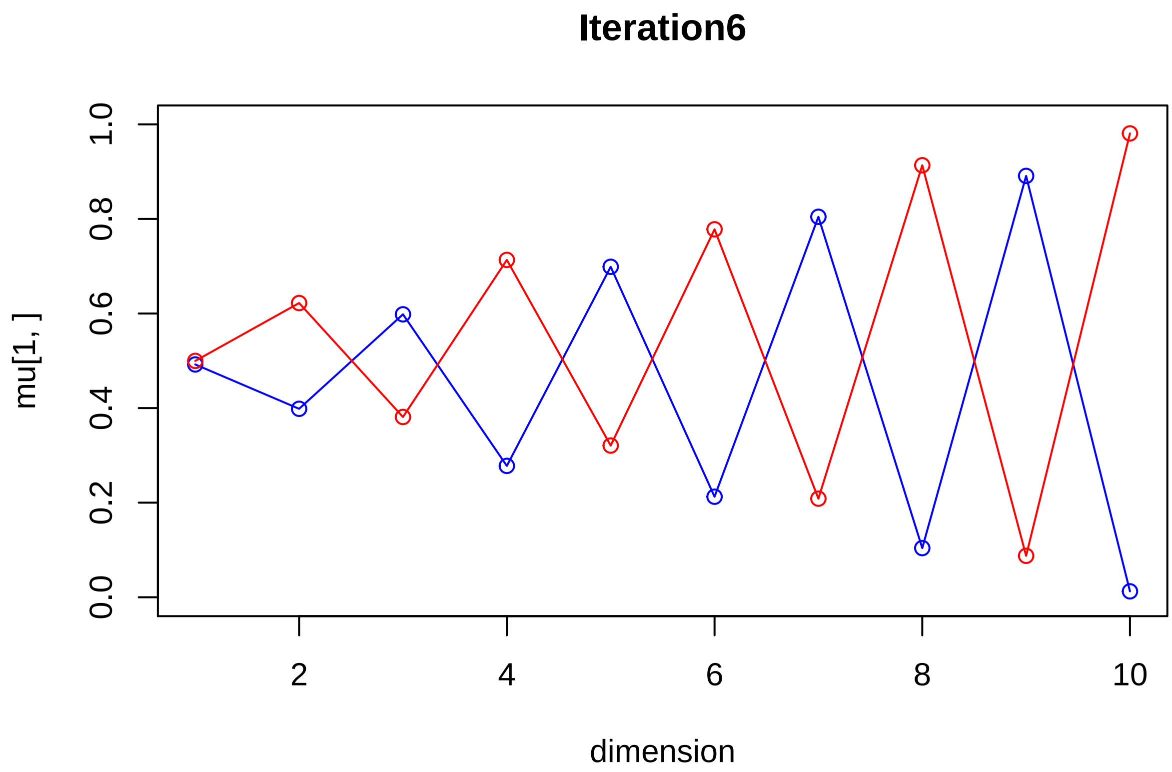

```
## iteration:  3 log likelihood:  -7463.445
```



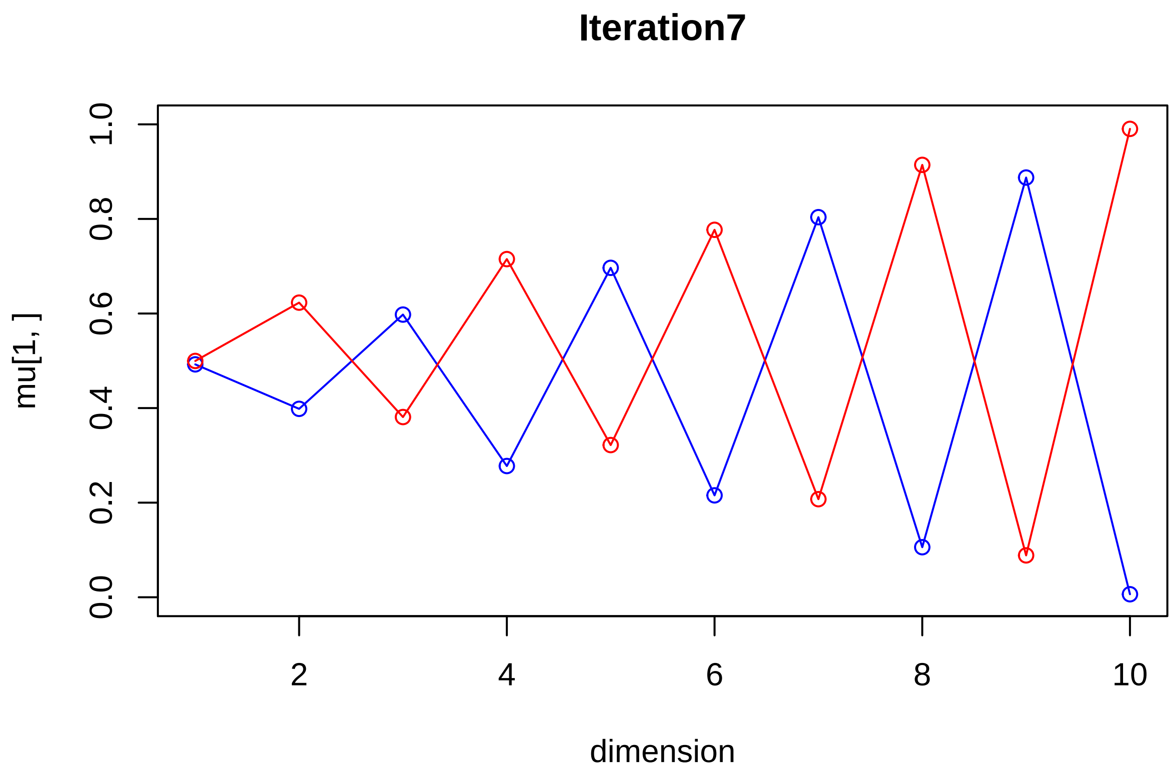

```
## iteration:  4 log likelihood:  -6575.121
```



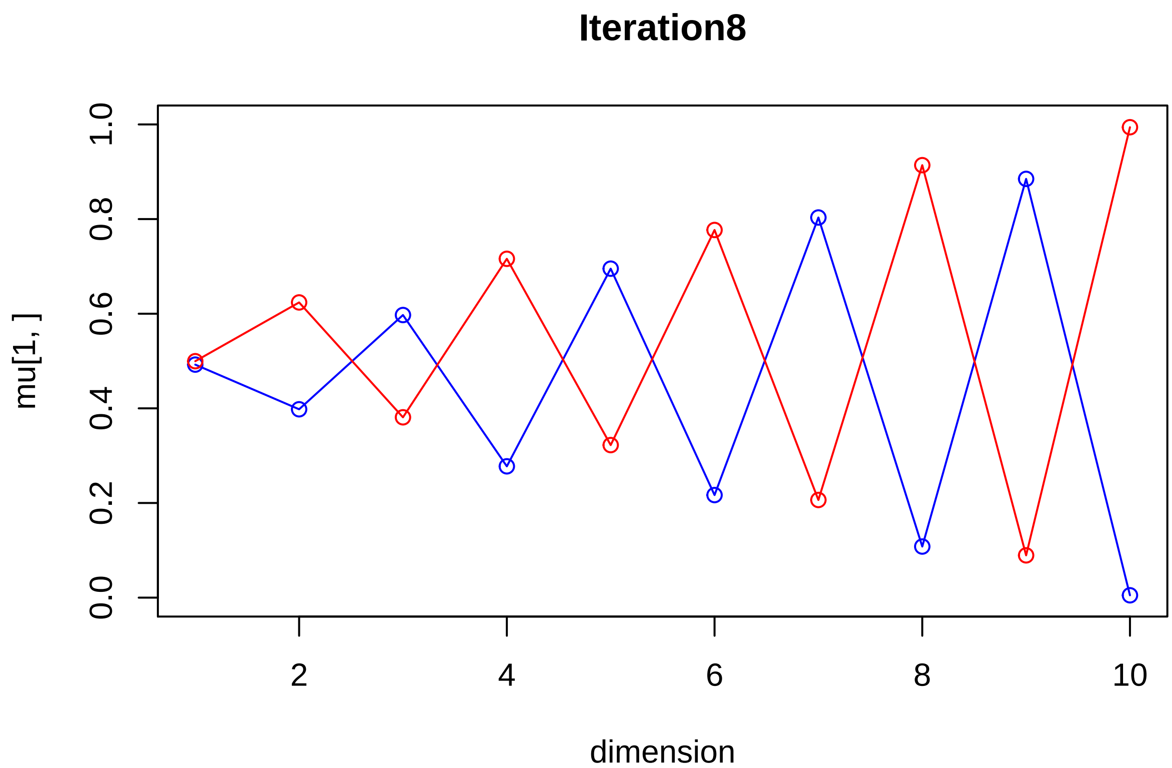

```
## iteration:  5 log likelihood:  -5731.559
```



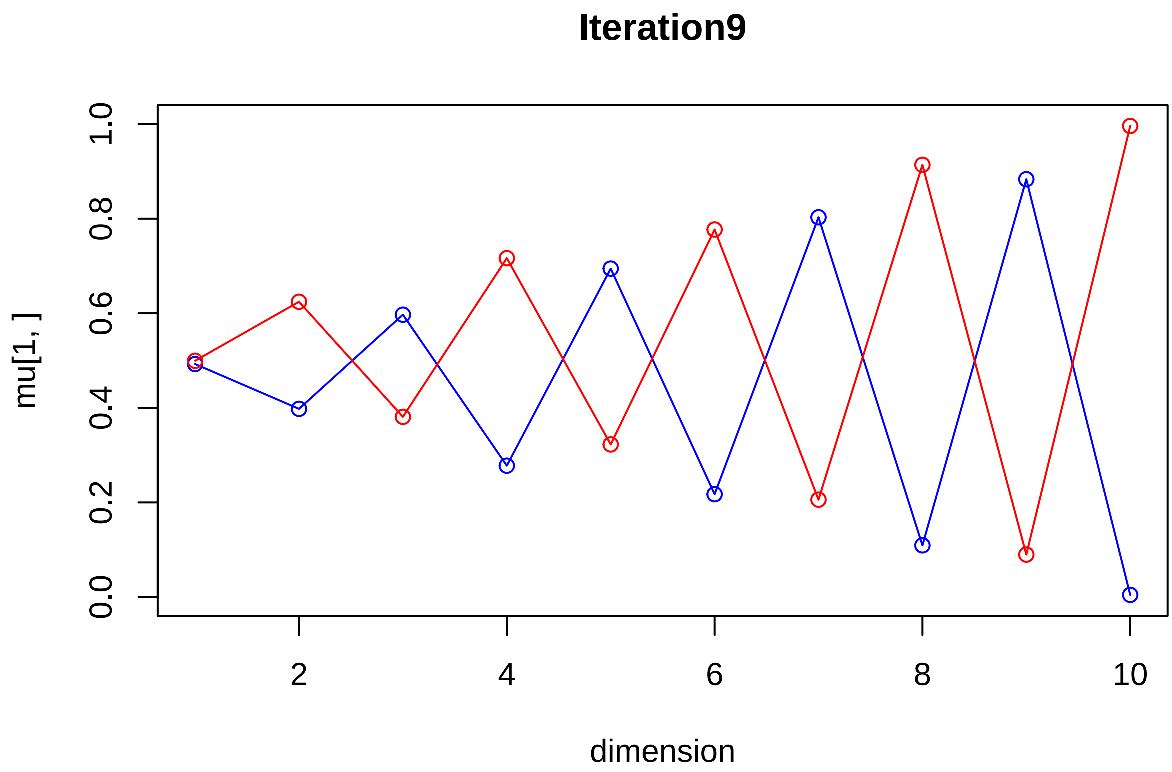

```
## iteration:  6 log likelihood:  -5656.174
```



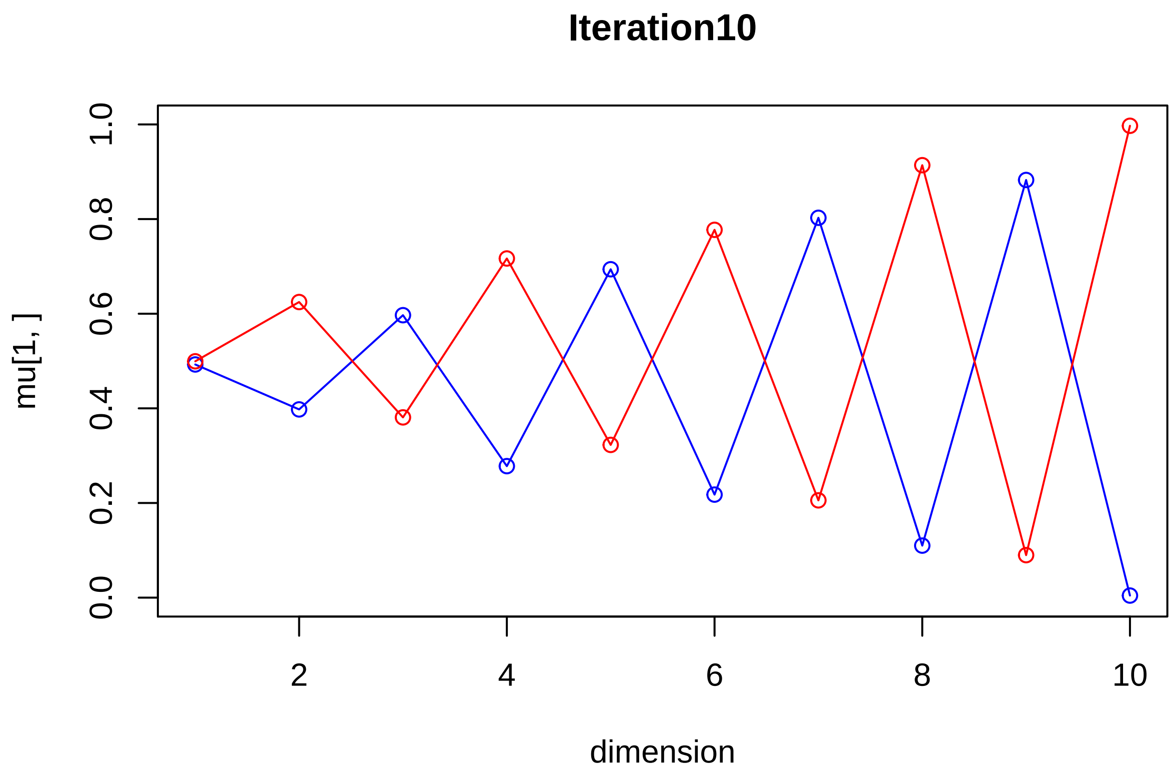

```
## iteration:  7 log likelihood:  -5648.904
```



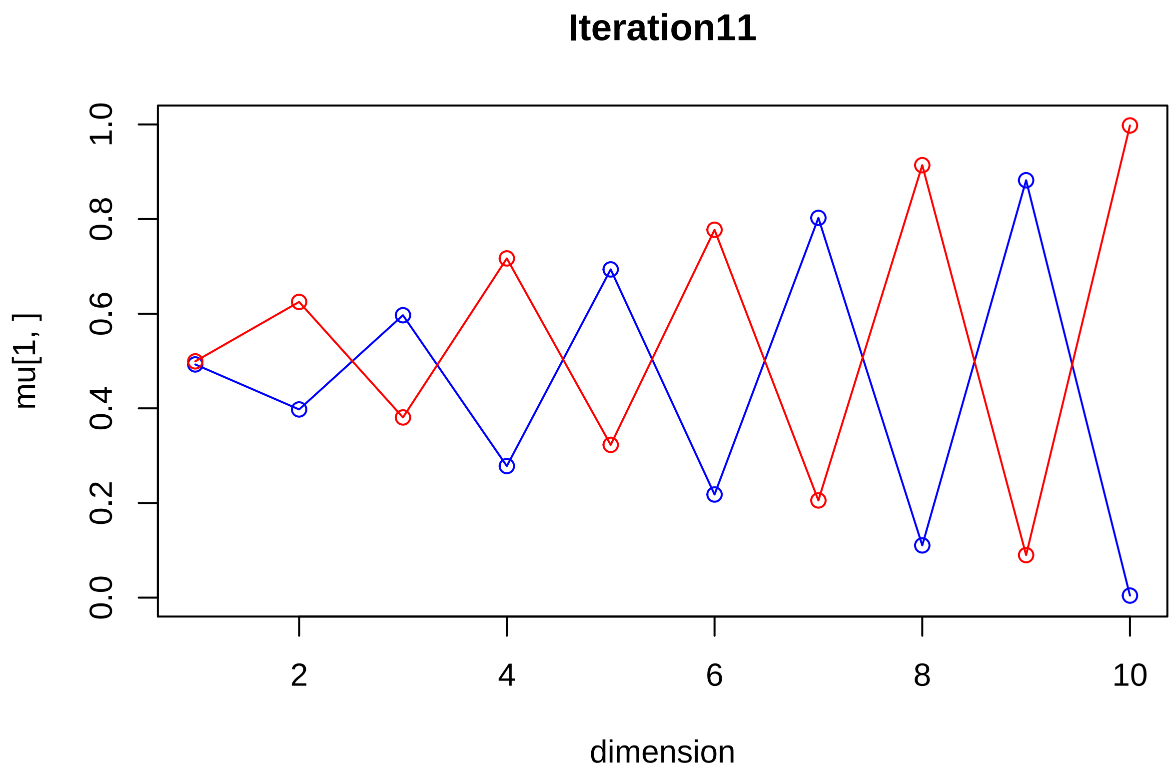

```
## iteration:  8 log likelihood:  -5646.139
```



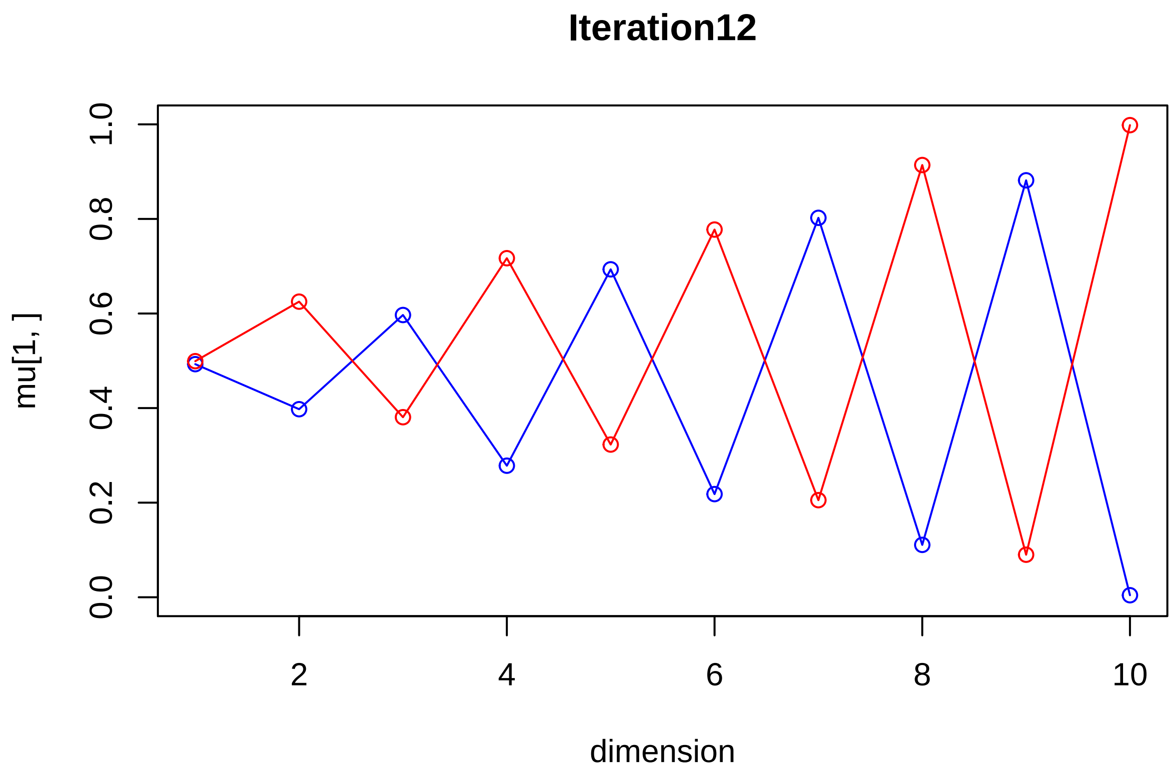

```
## iteration:  9 log likelihood:  -5644.608
```



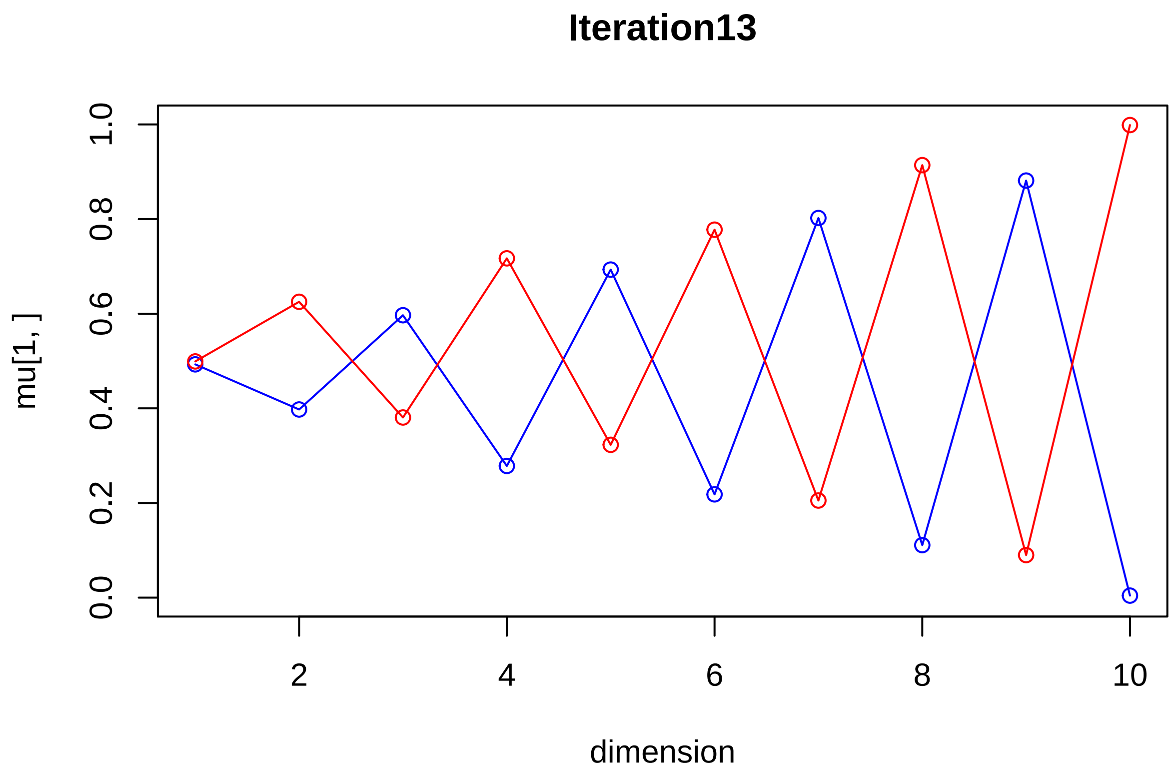

```
## iteration:  10 log likelihood:  -5643.615
```



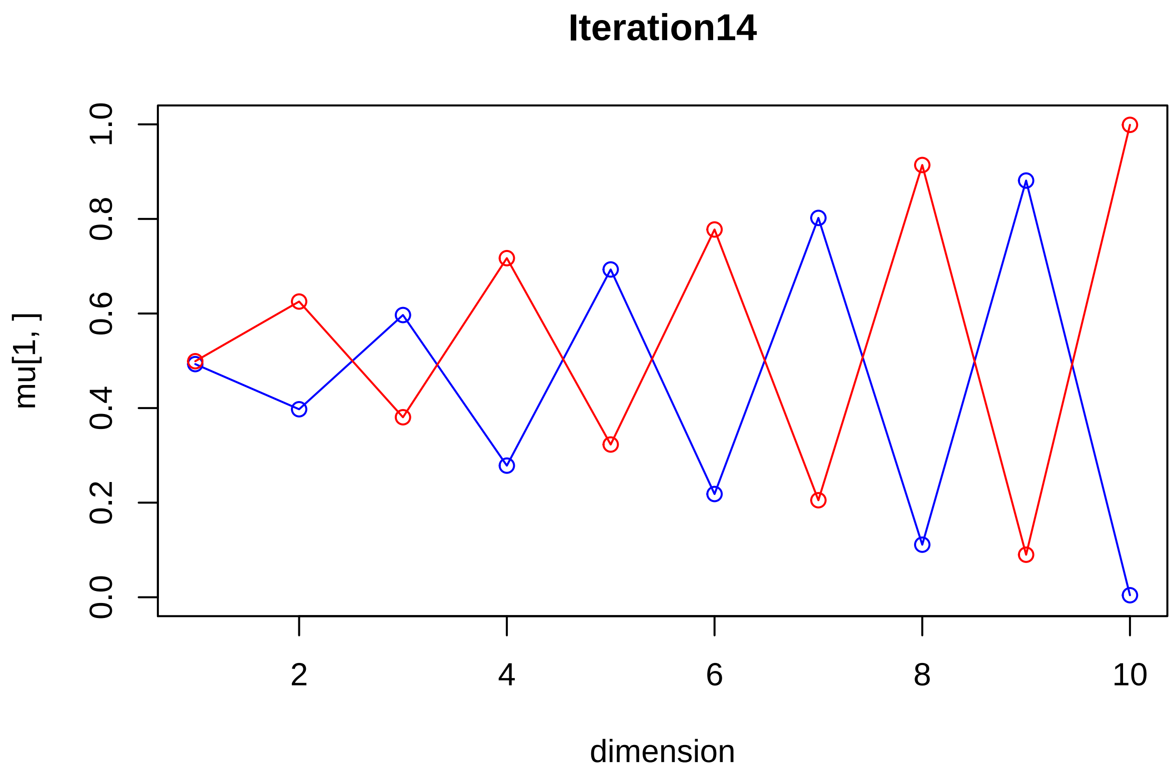

```
## iteration:  11 log likelihood:  -5642.913
```



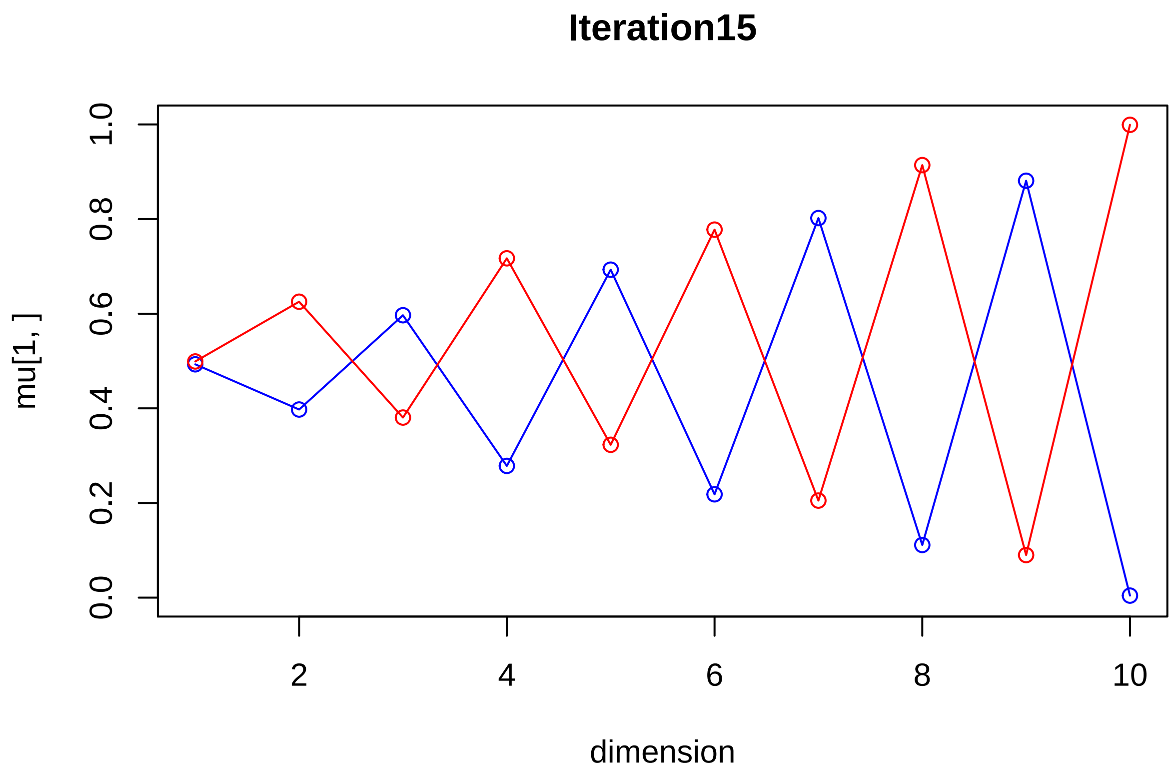

```
## iteration:  12 log likelihood:  -5642.386
```



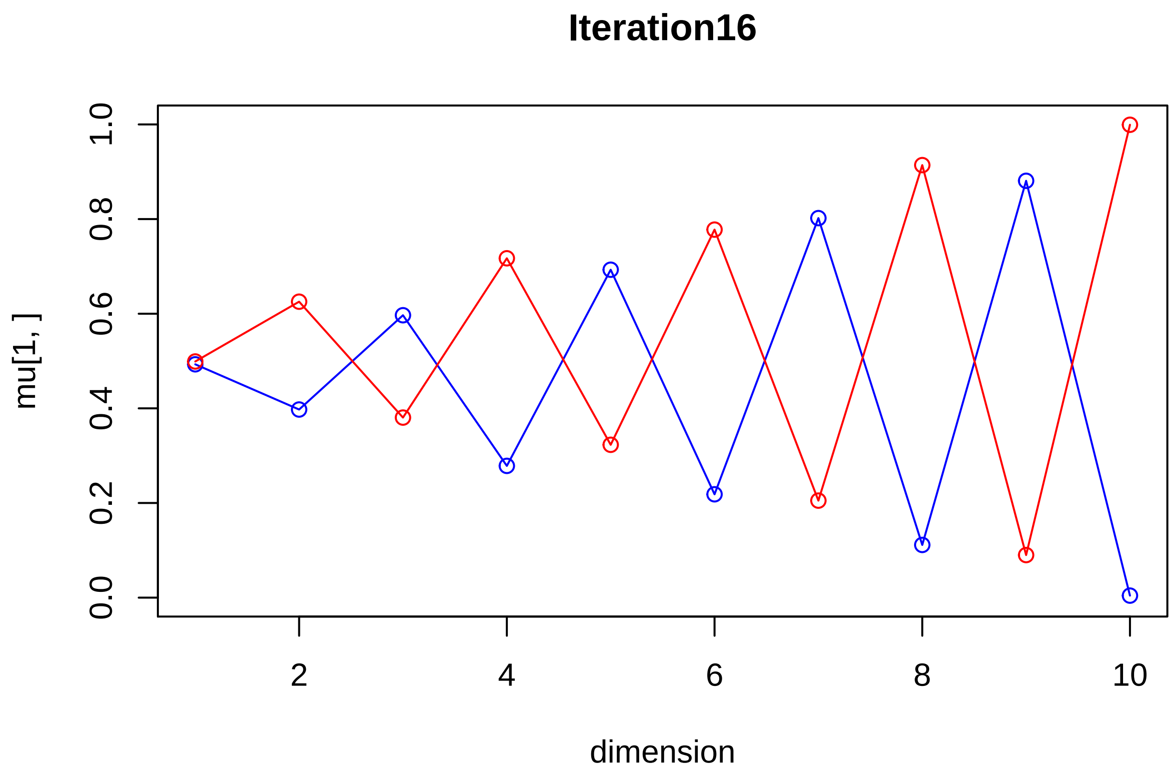

```
## iteration:  13 log likelihood:  -5641.977
```



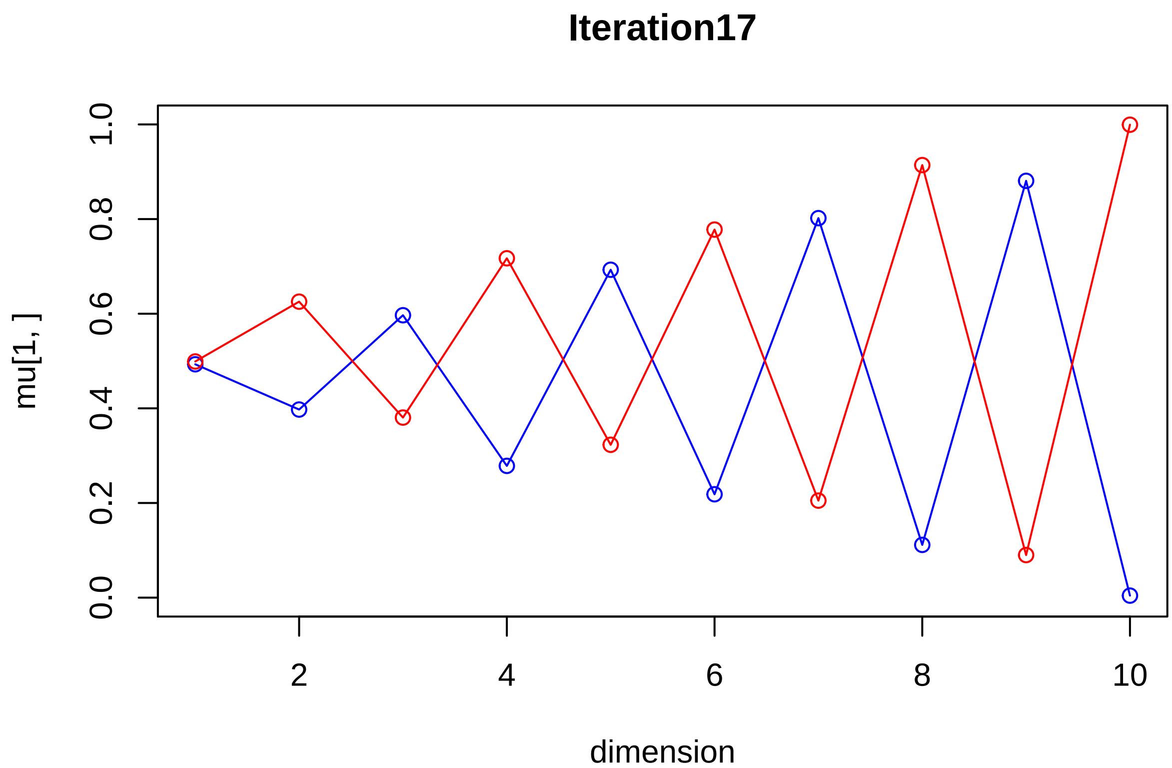

```
## iteration:  14 log likelihood:  -5641.649
```



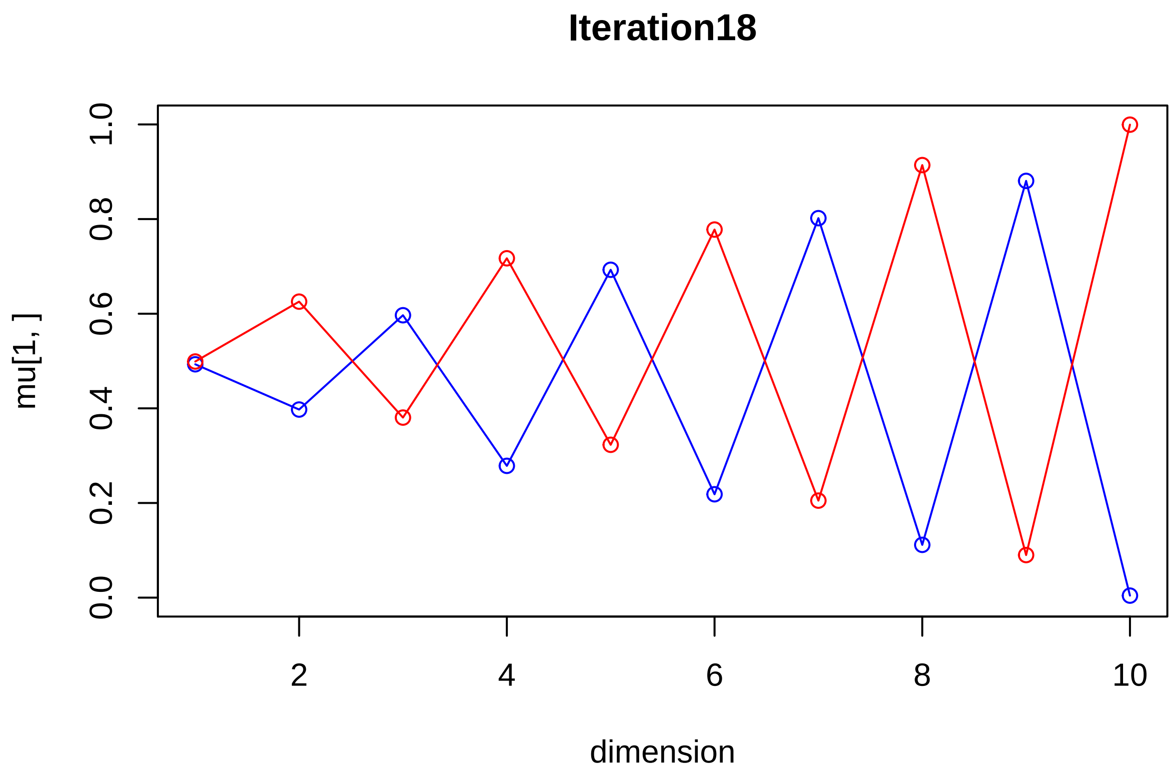

```
## iteration:  15 log likelihood:  -5641.382
```



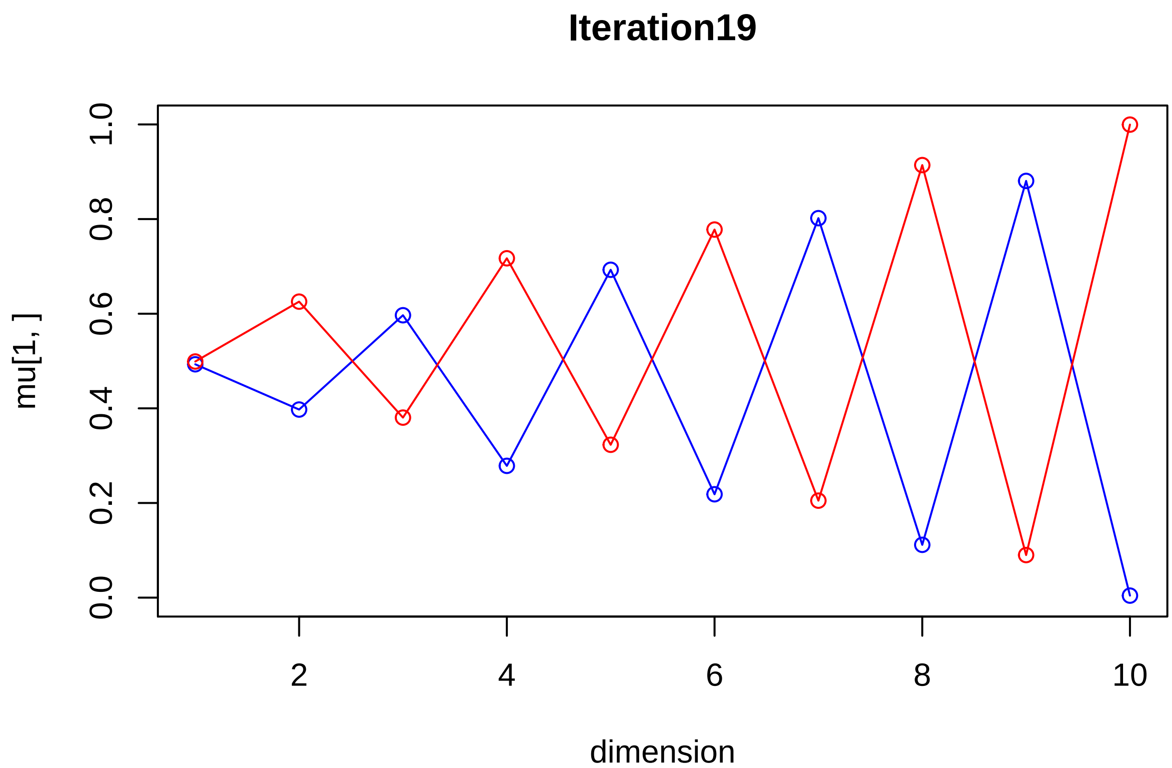

```
## iteration:  16 log likelihood:  -5641.161
```



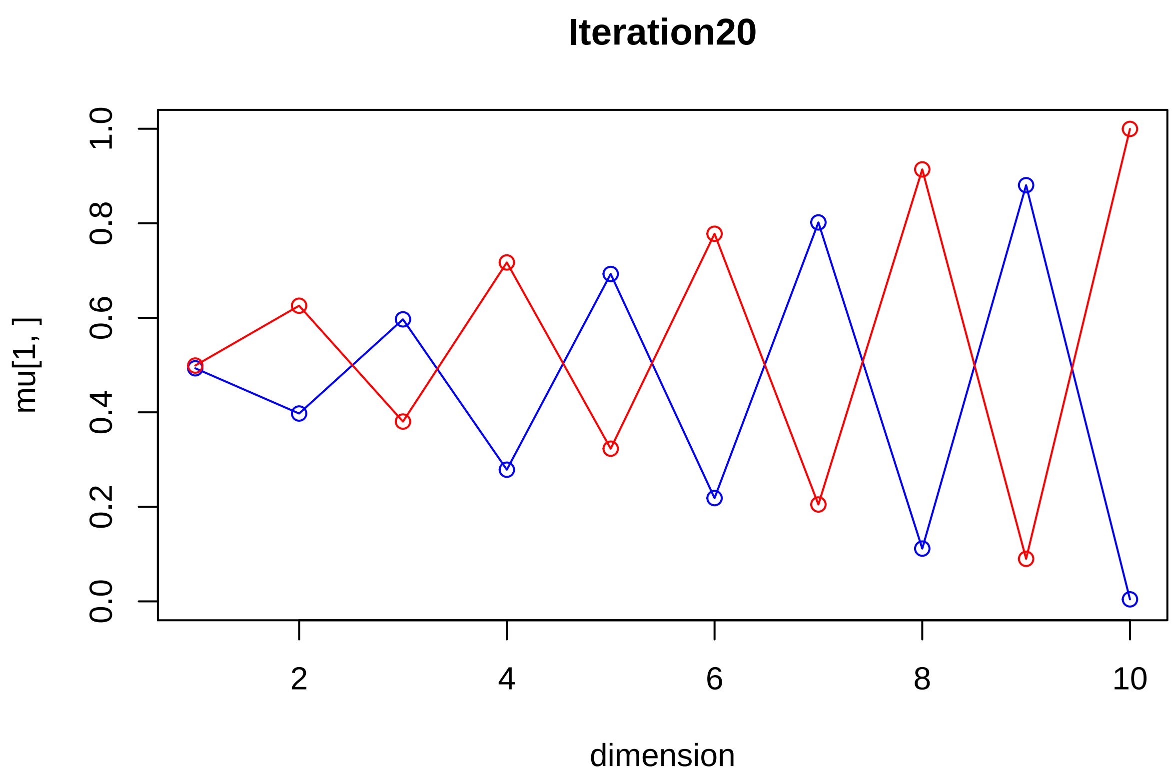

```
## iteration:  17 log likelihood:  -5640.975
```



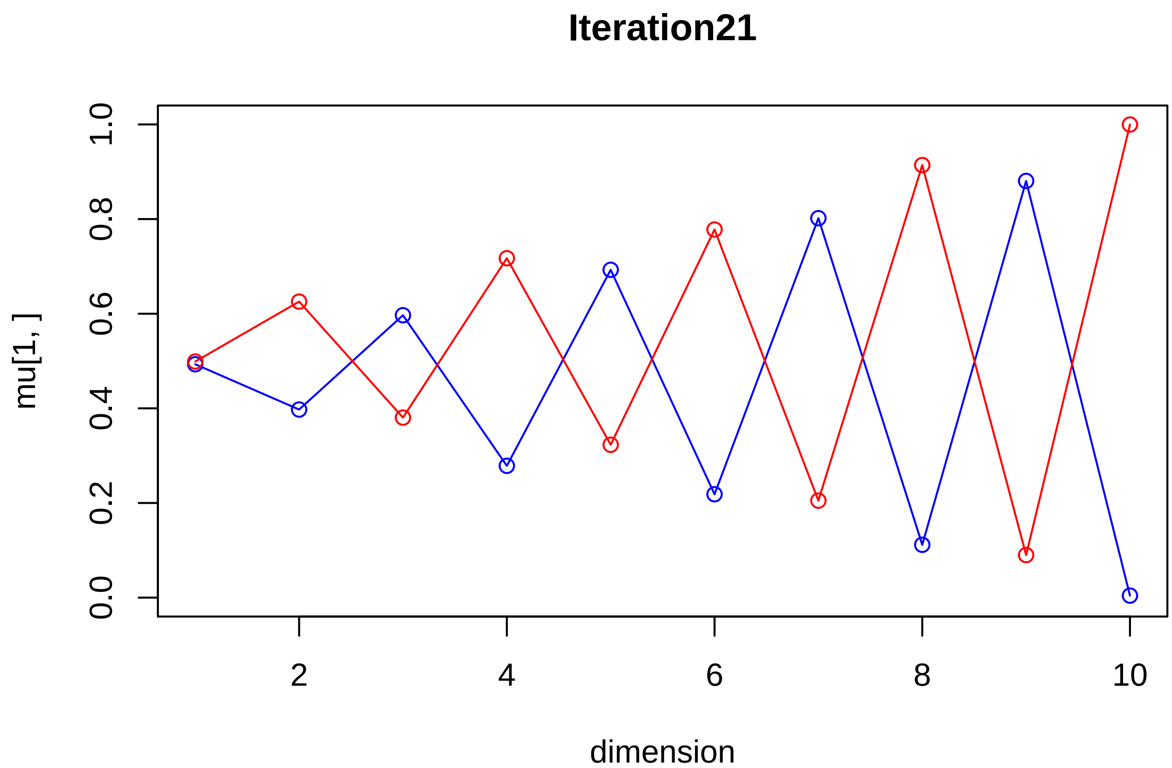

```
## iteration:  18 log likelihood:  -5640.819
```



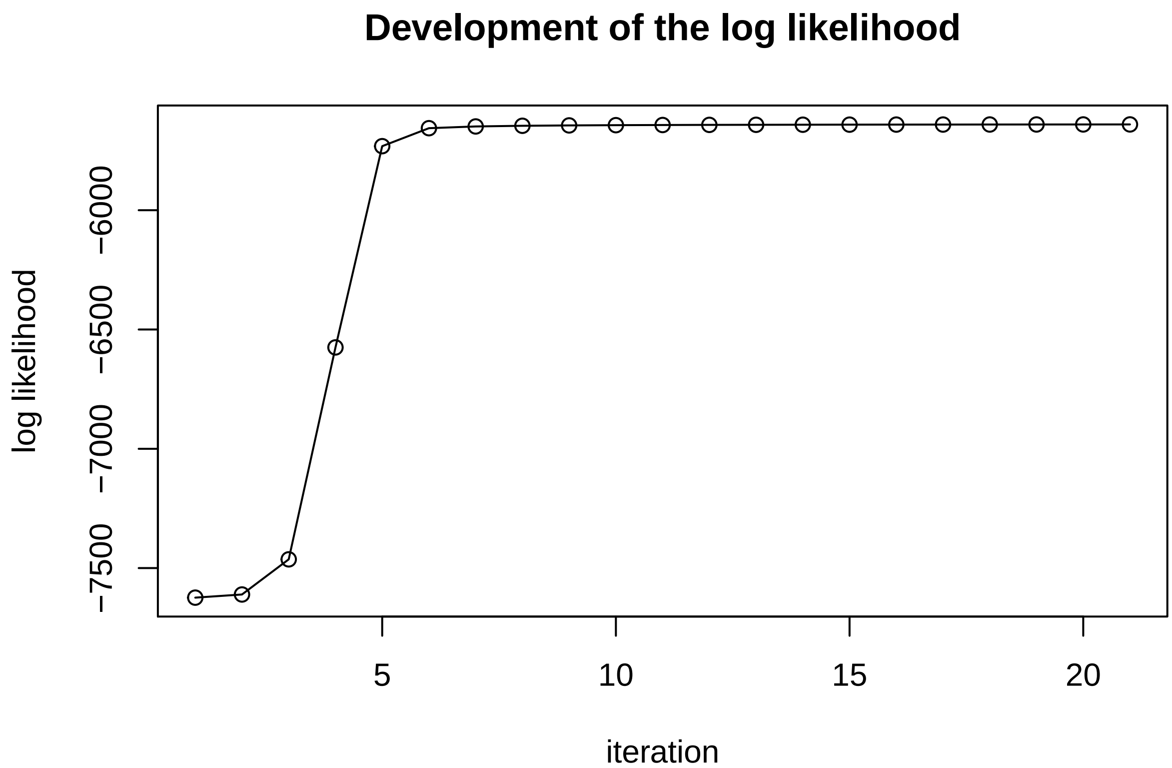

```
## iteration:  19 log likelihood:  -5640.685
```




```
## iteration:  20 log likelihood:  -5640.571
```



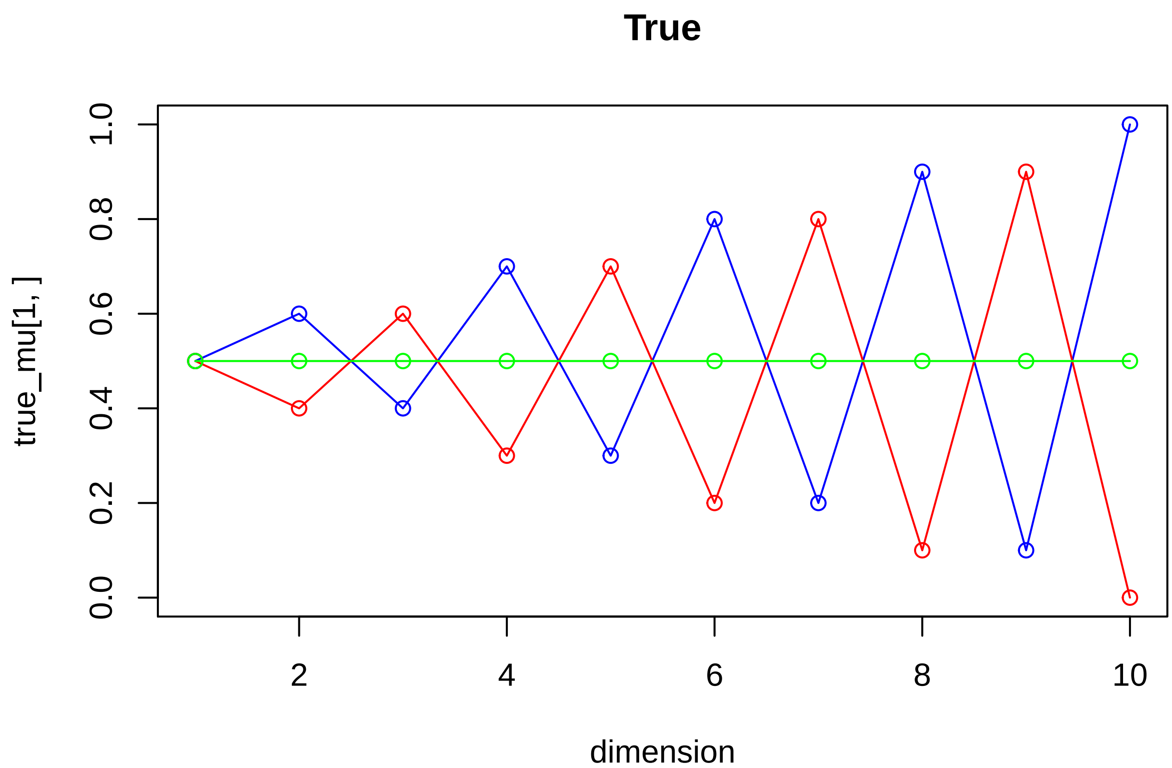

```
## iteration:  21 log likelihood:  -5640.473
```

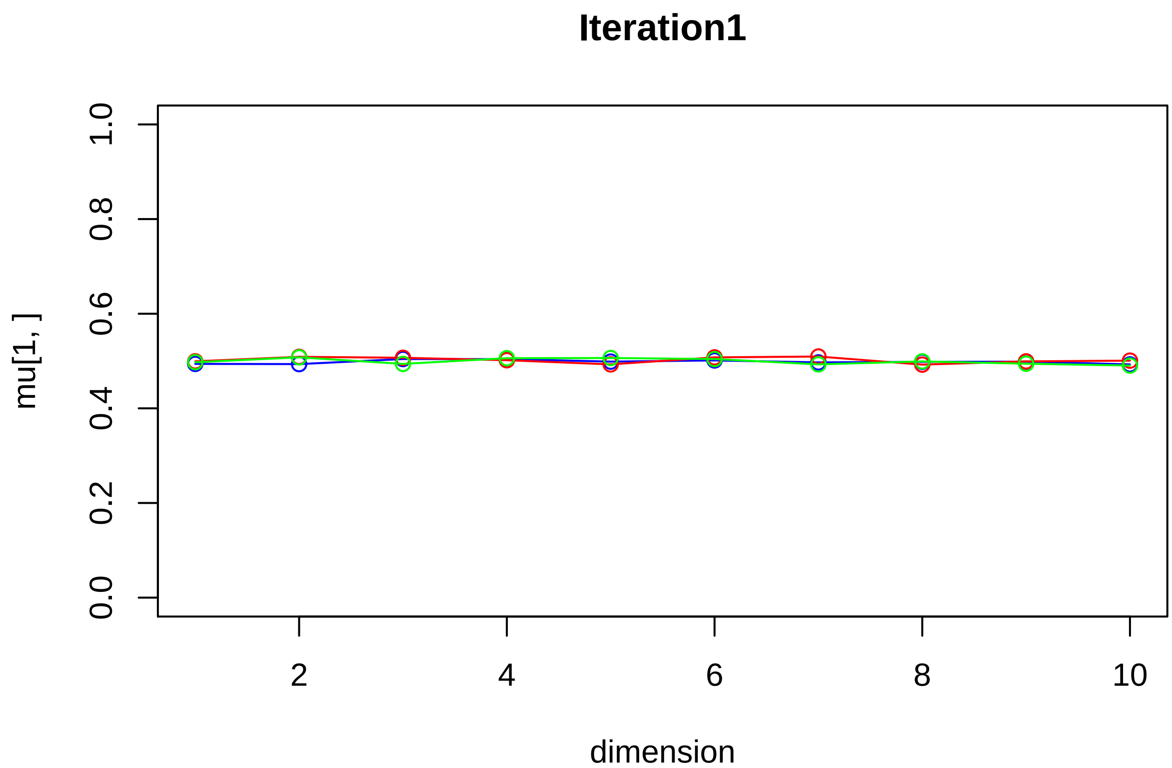



```
## $pi
## [1] 0.5110531 0.4889469
##
## $mu
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4931735 0.3974606 0.5967811 0.2785480 0.6927917 0.2184957 0.8018491
## [2,] 0.4989543 0.6255823 0.3804363 0.7171478 0.3230343 0.7778699 0.2049559
##           [,8]       [,9]       [,10]
## [1,] 0.1116477 0.88054439 0.004290353
## [2,] 0.9140913 0.08997919 0.999714736
##
## $logLikelihoodDevelopment
## NULL
```

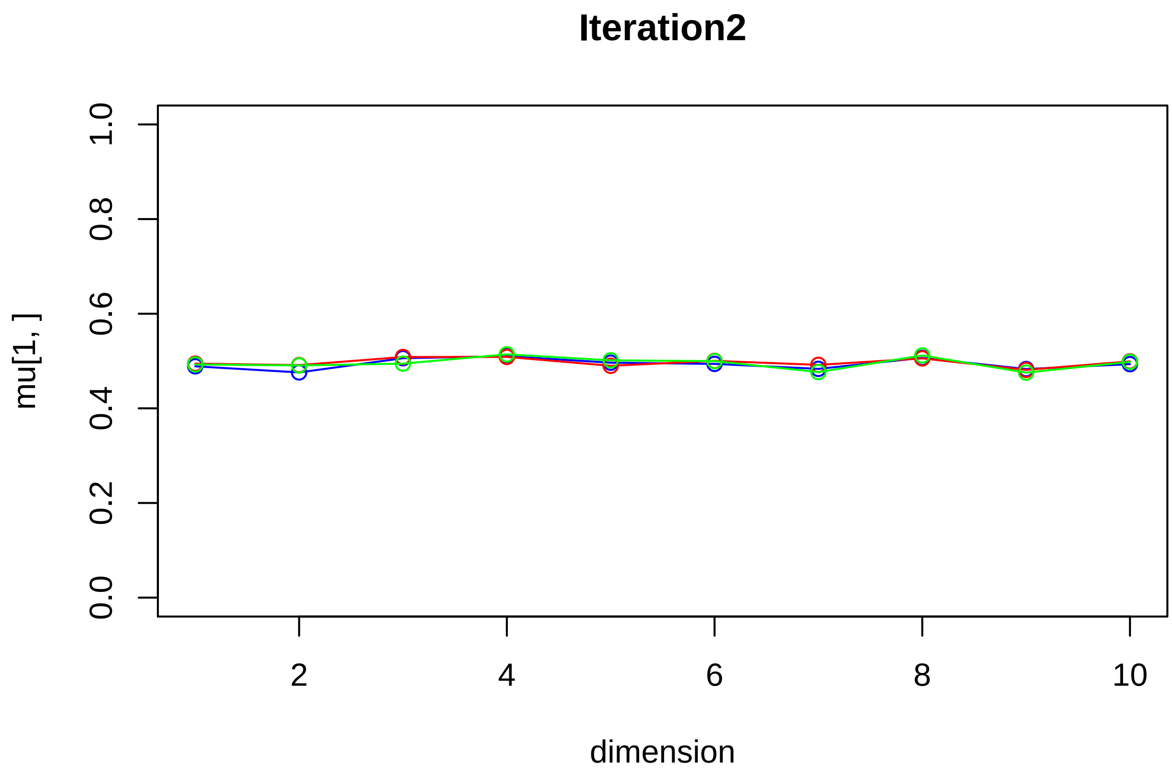
3. K=3

```
em_loop(3)
```

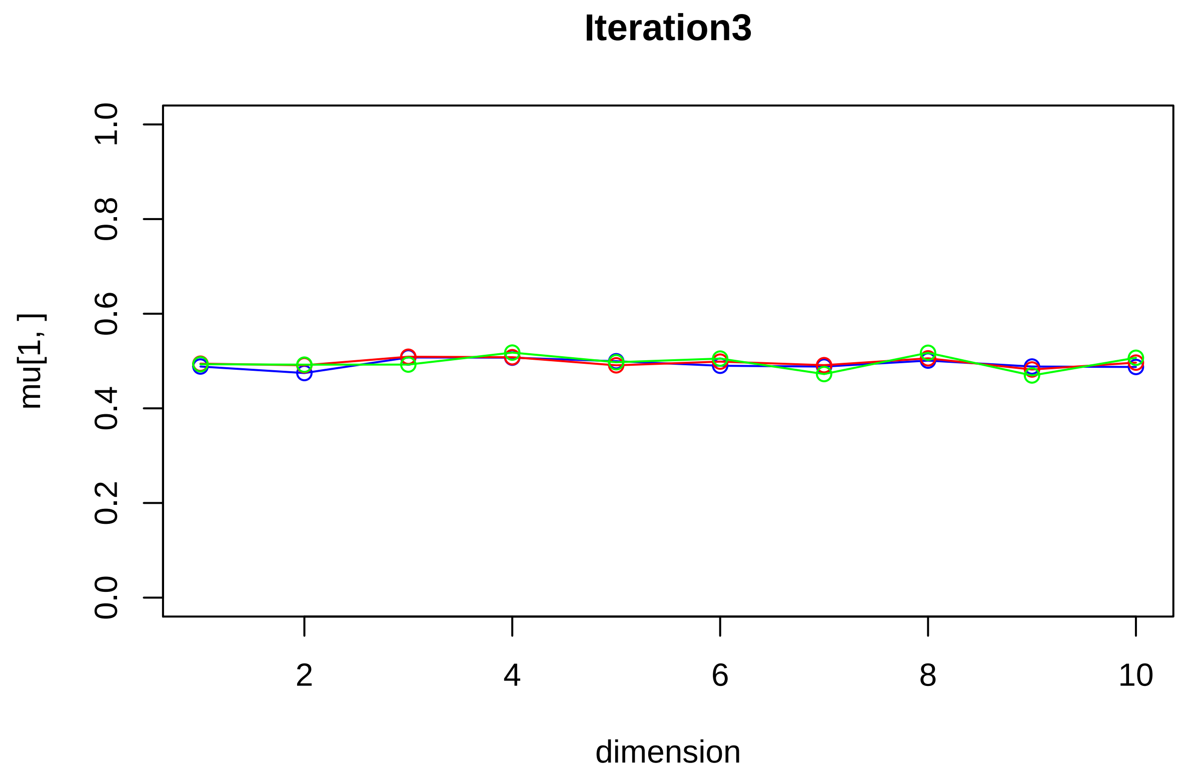

True
true_mu[1,]
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
dimension



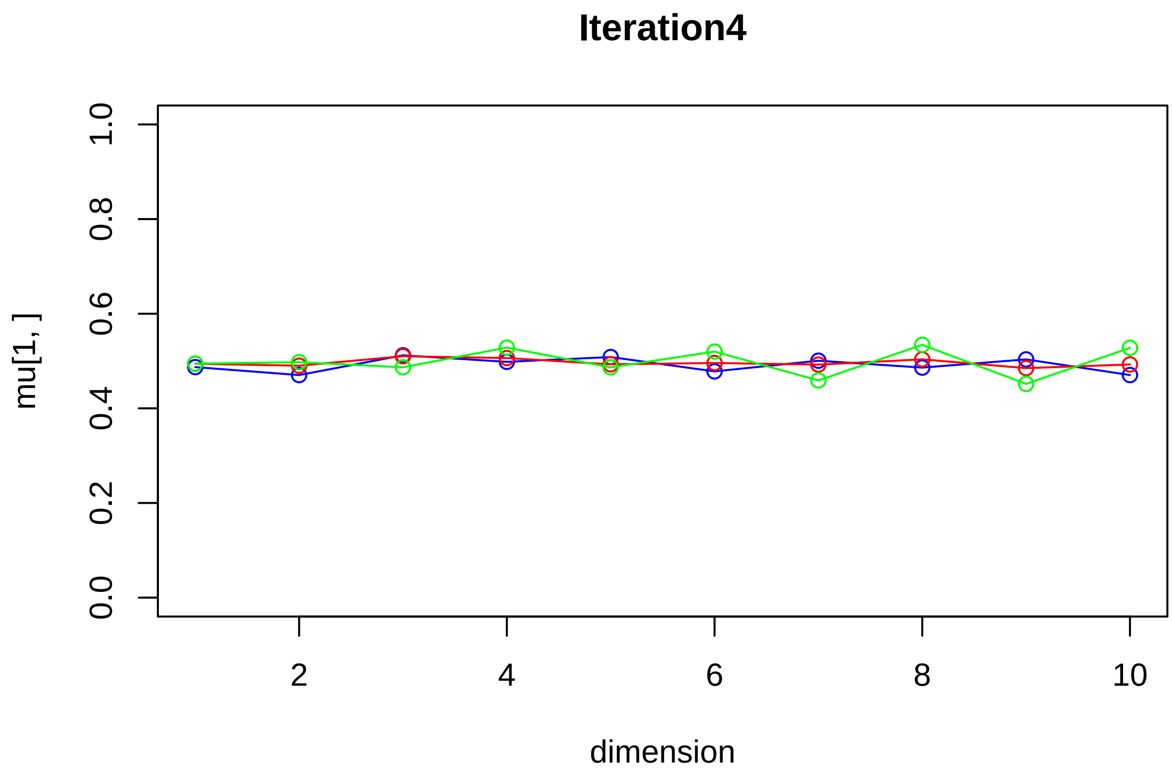

```
## iteration:  1 log likelihood:  -8029.723
```



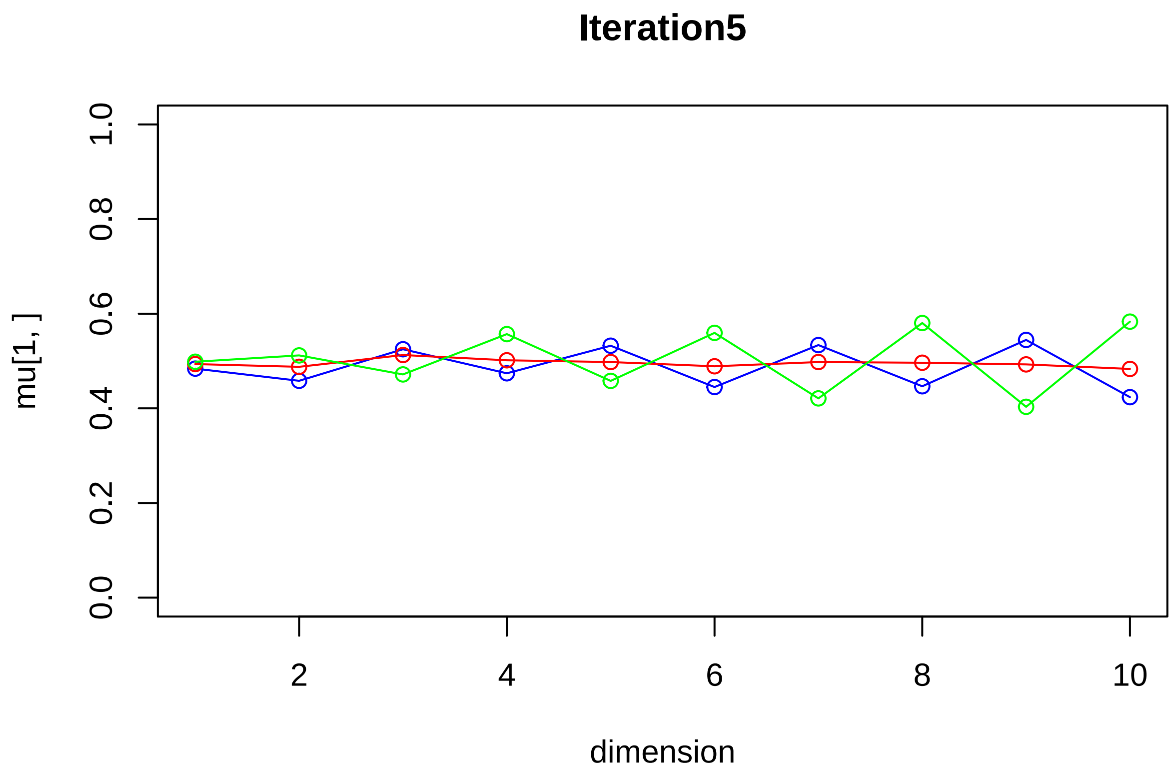

```
## iteration:  2 log likelihood:  -8027.183
```



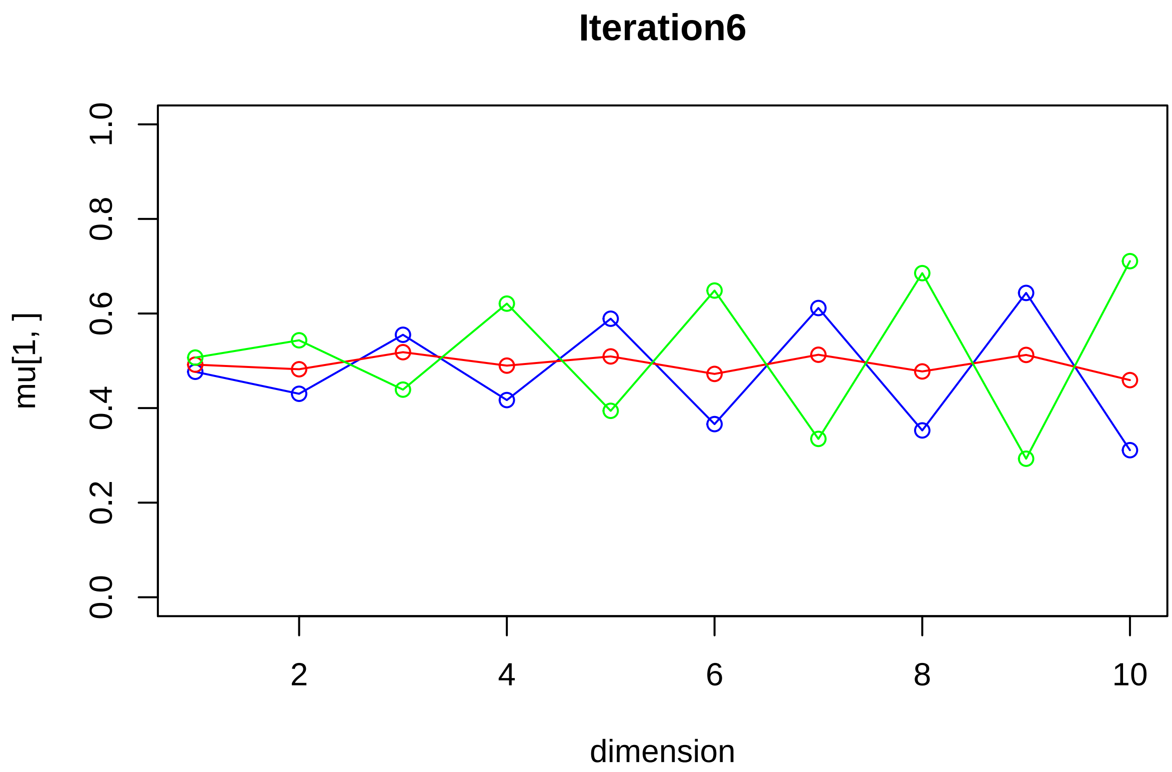

```
## iteration:  3 log likelihood:  -8024.696
```



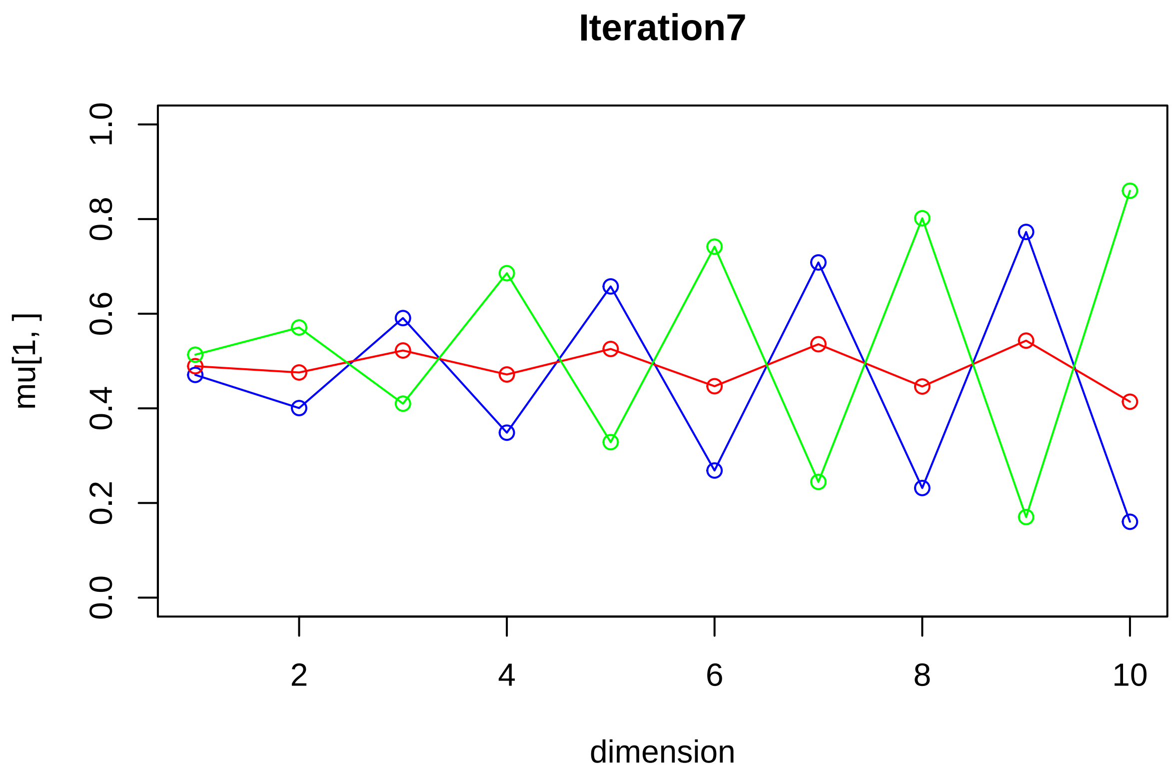

```
## iteration:  4 log likelihood:  -8005.631
```



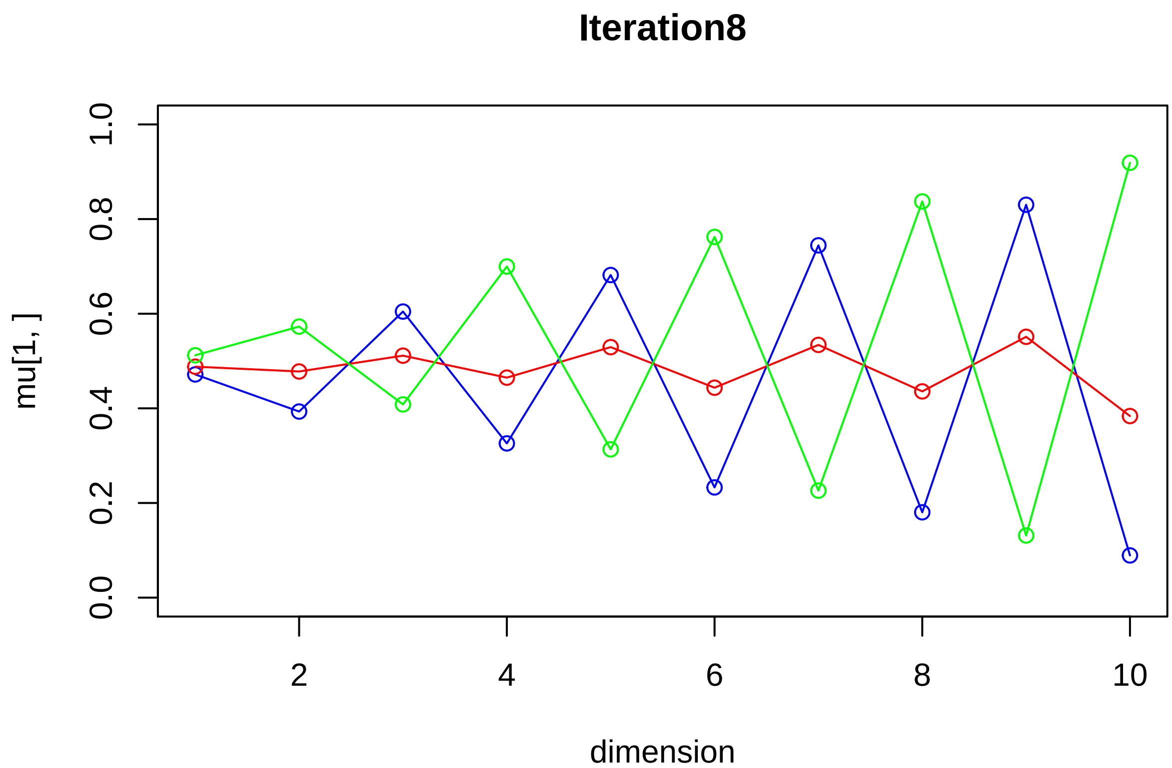

```
## iteration:  5 log likelihood:  -7877.606
```



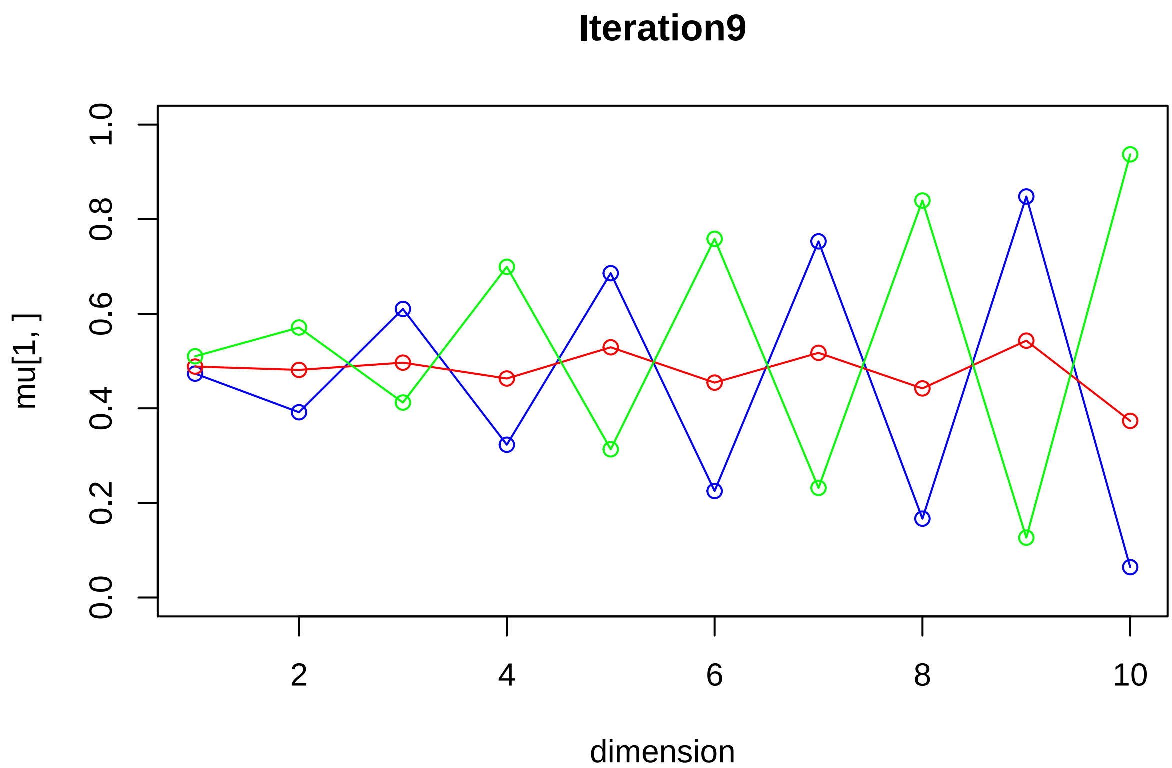

```
## iteration:  6 log likelihood:  -7403.513
```



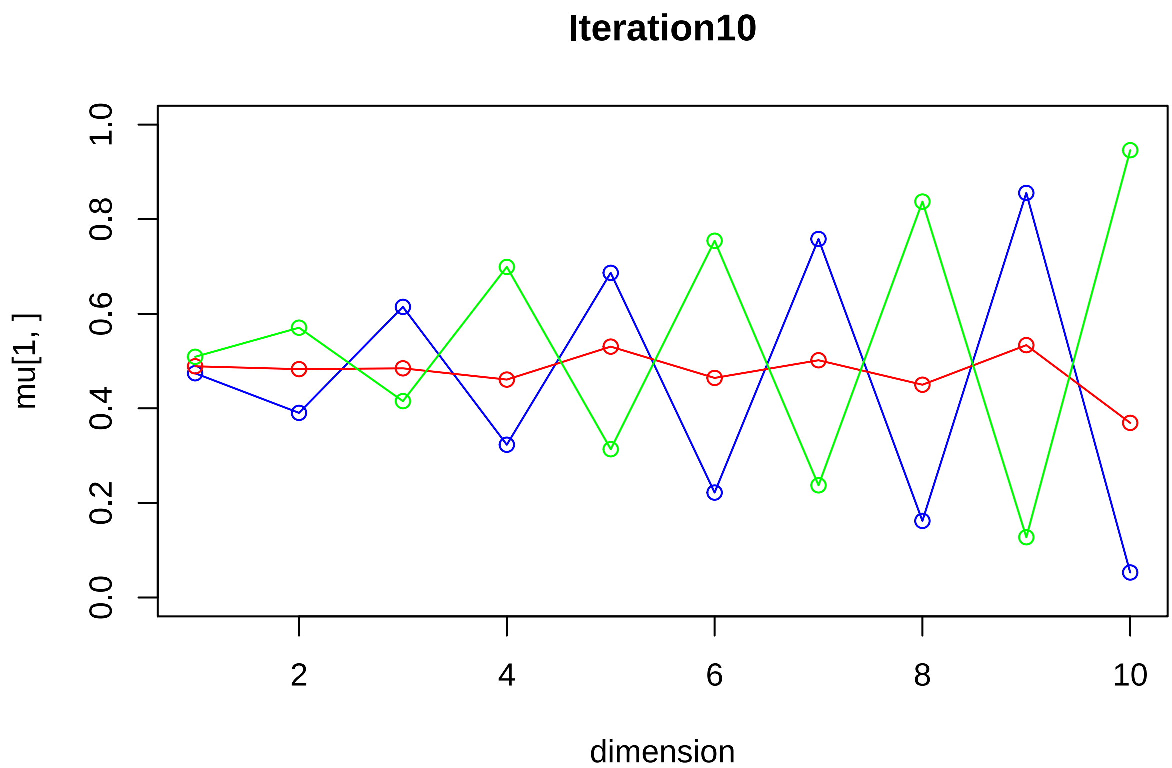

```
## iteration:  7 log likelihood:  -6936.919
```



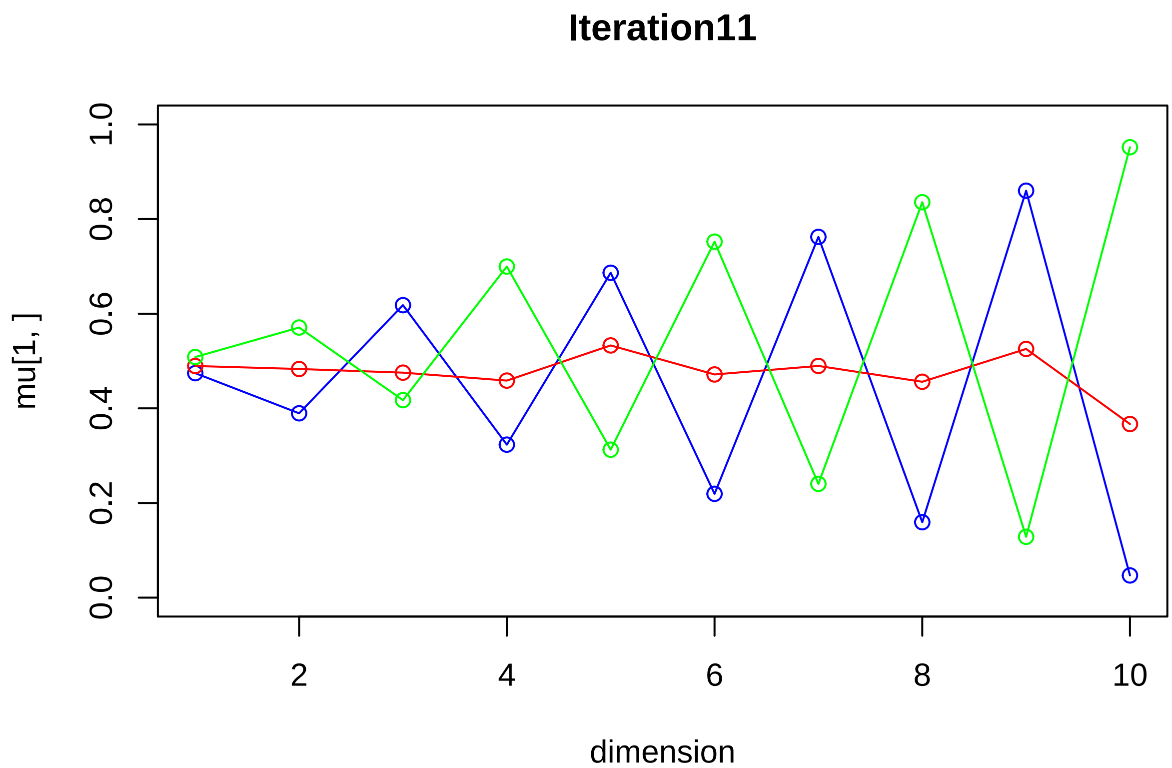

```
## iteration:  8 log likelihood:  -6818.582
```



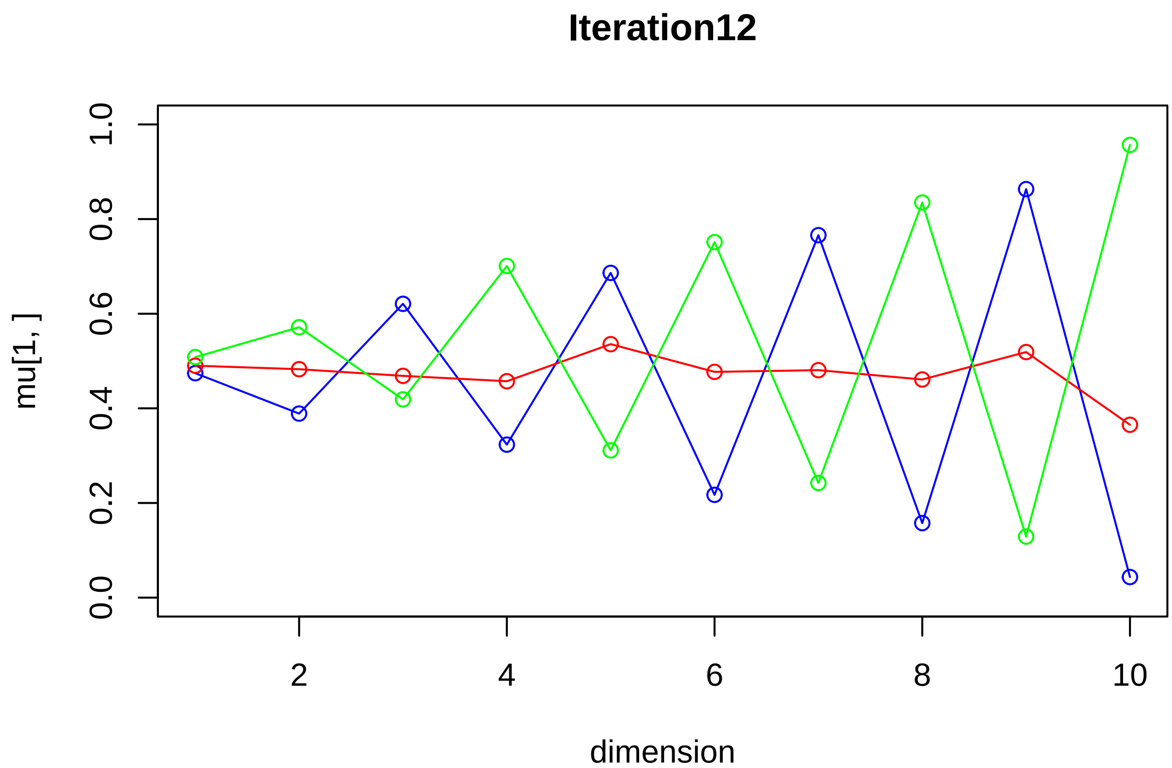

```
## iteration:  9 log likelihood:  -6791.377
```



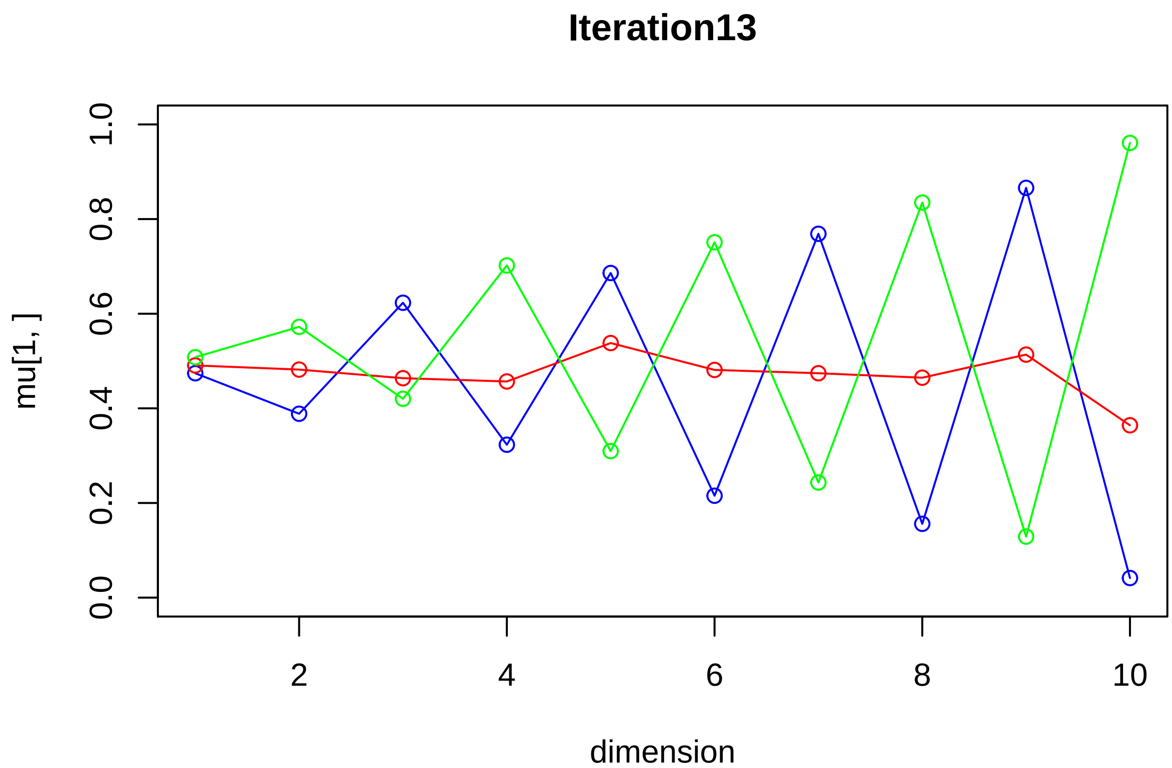

```
## iteration:  10 log likelihood:  -6780.713
```



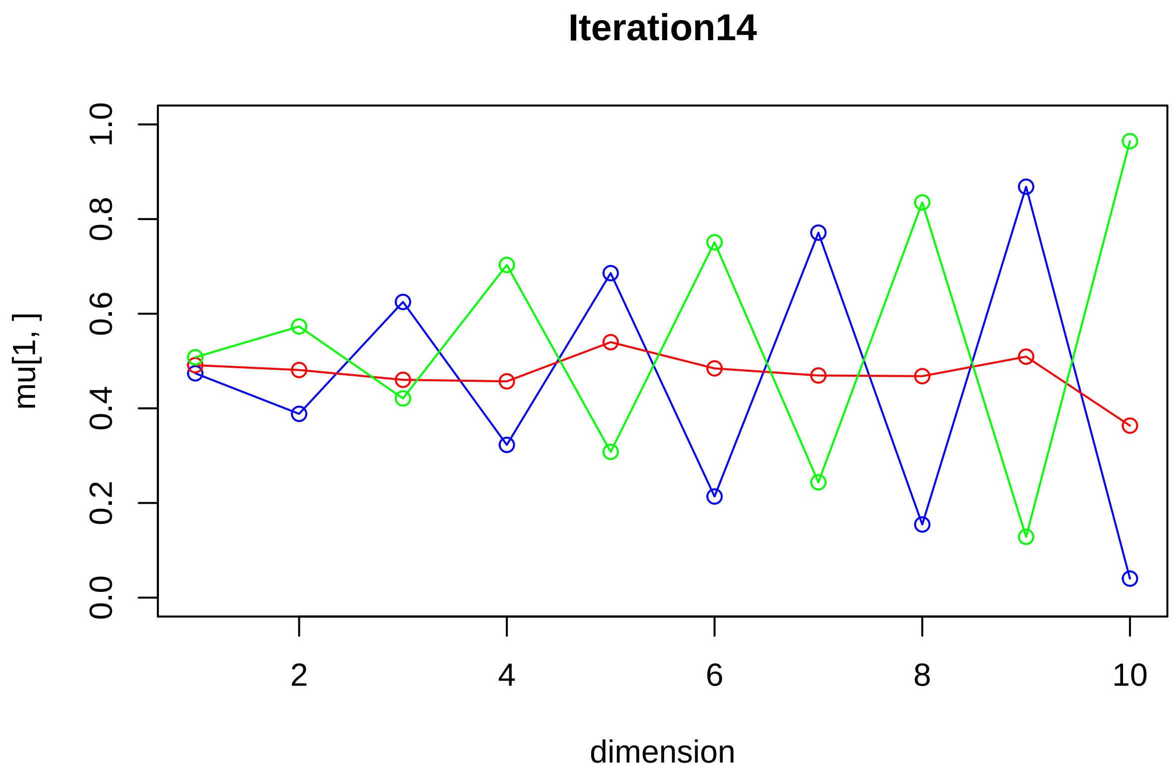

```
## iteration:  11 log likelihood:  -6774.958
```



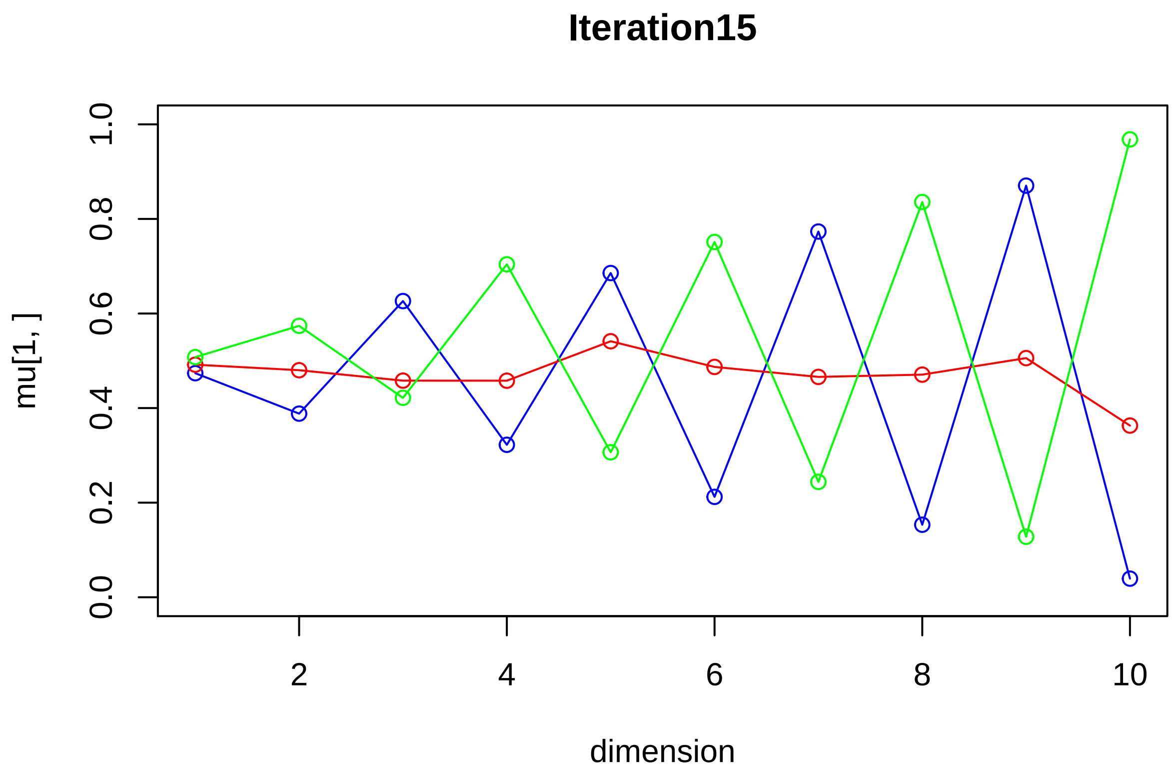

```
## iteration:  12 log likelihood:  -6771.261
```



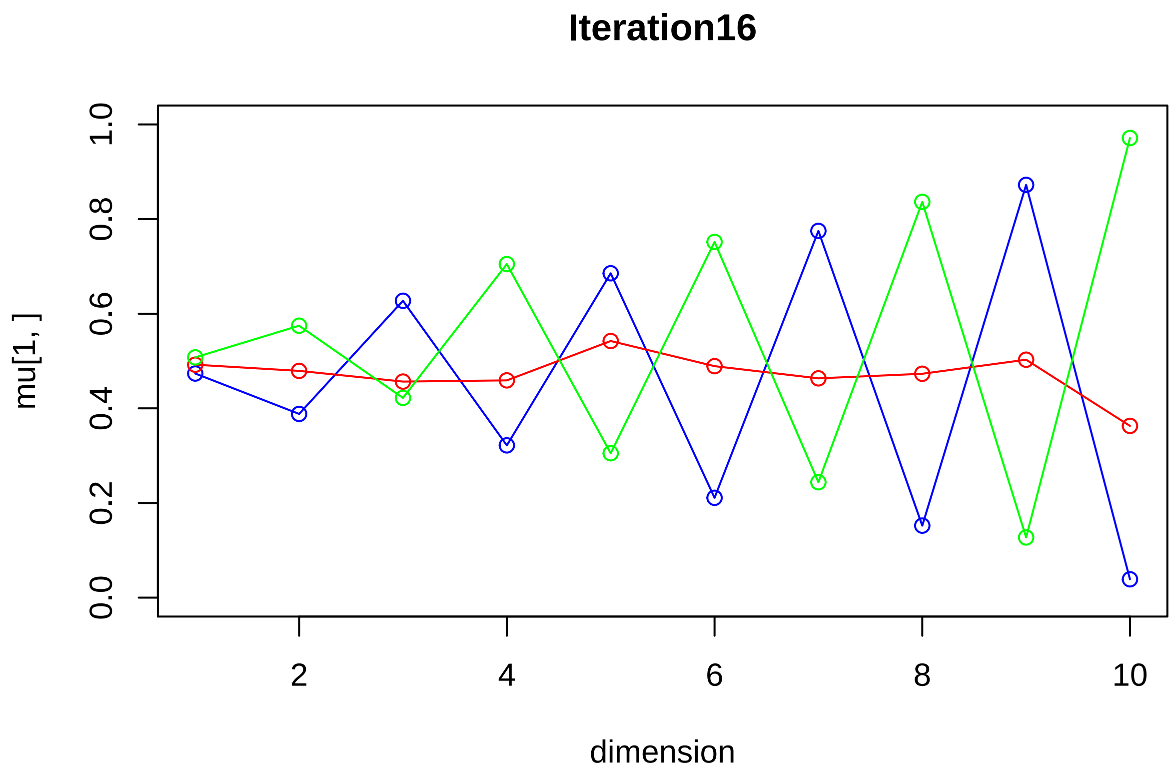

```
## iteration:  13 log likelihood:  -6768.606
```



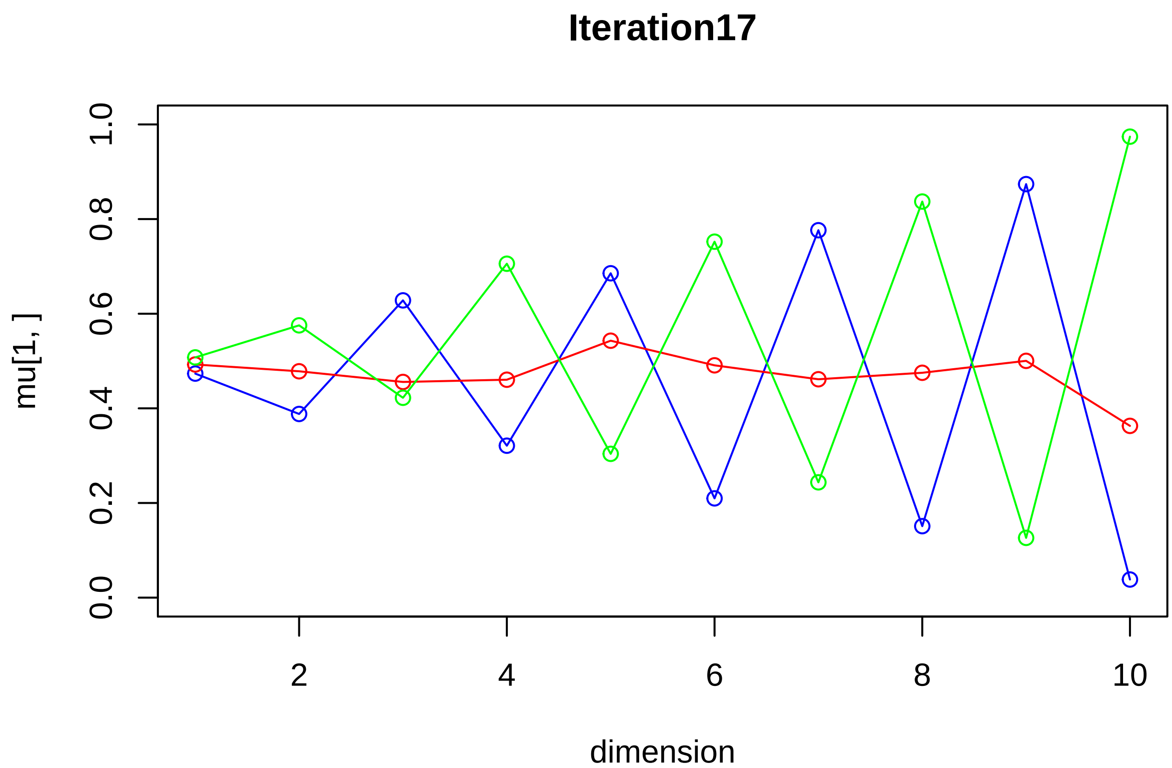

```
## iteration:  14 log likelihood:  -6766.535
```



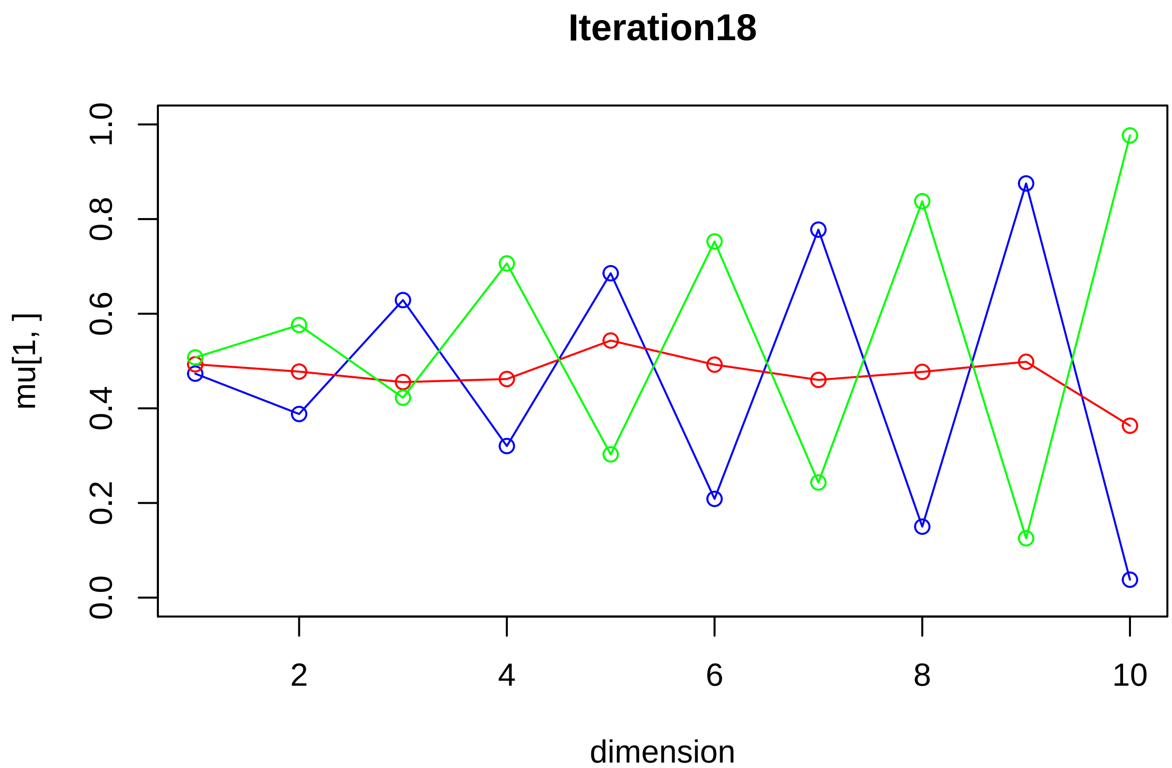

```
## iteration:  15 log likelihood:  -6764.815
```



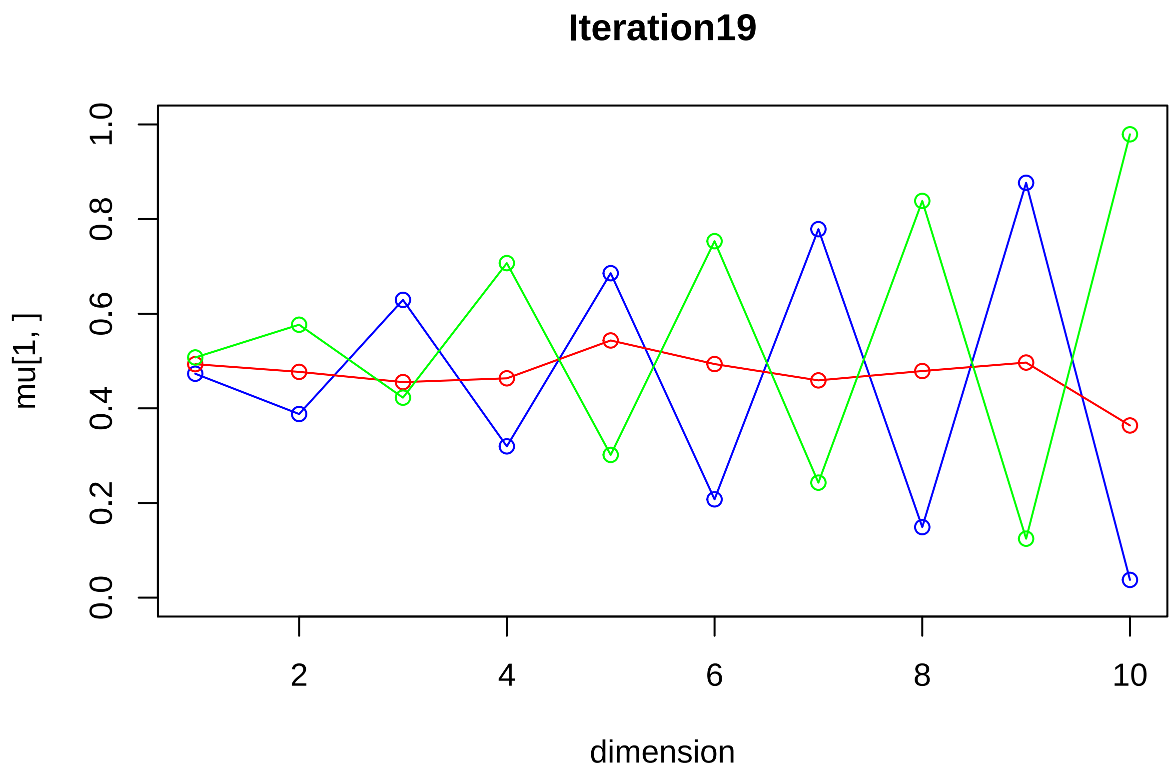

```
## iteration:  16 log likelihood:  -6763.316
```



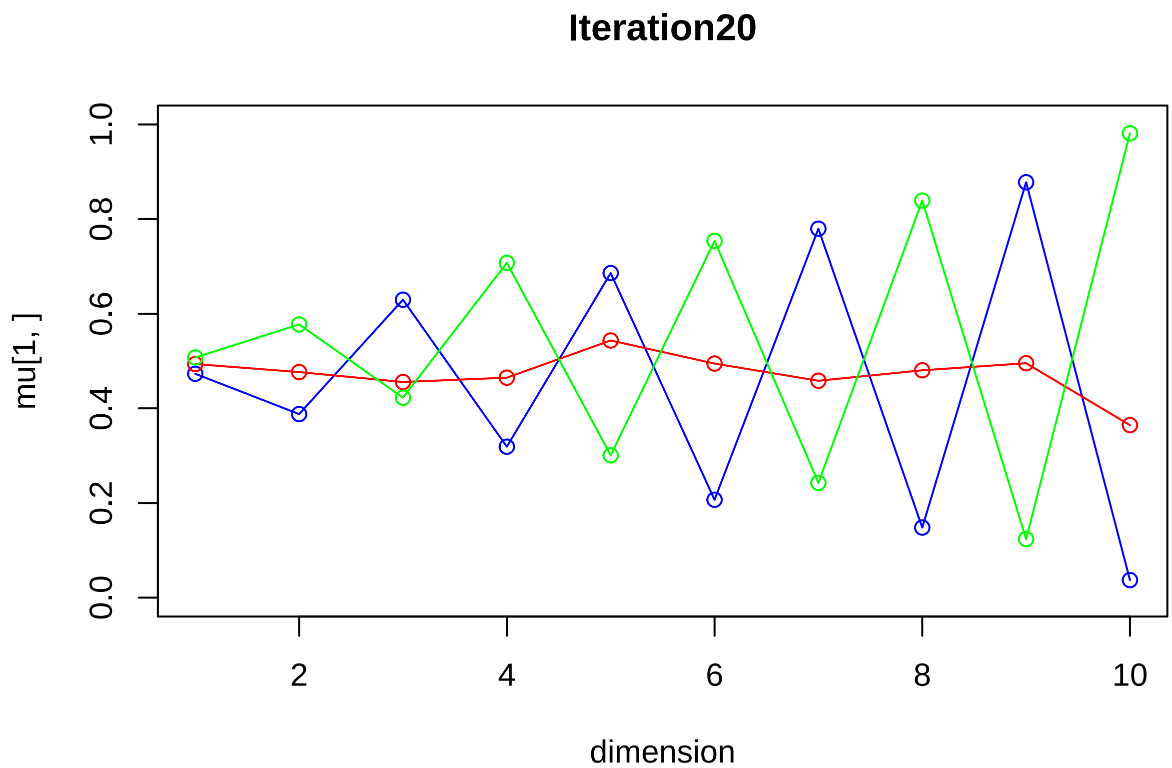

```
## iteration:  17 log likelihood:  -6761.967
```



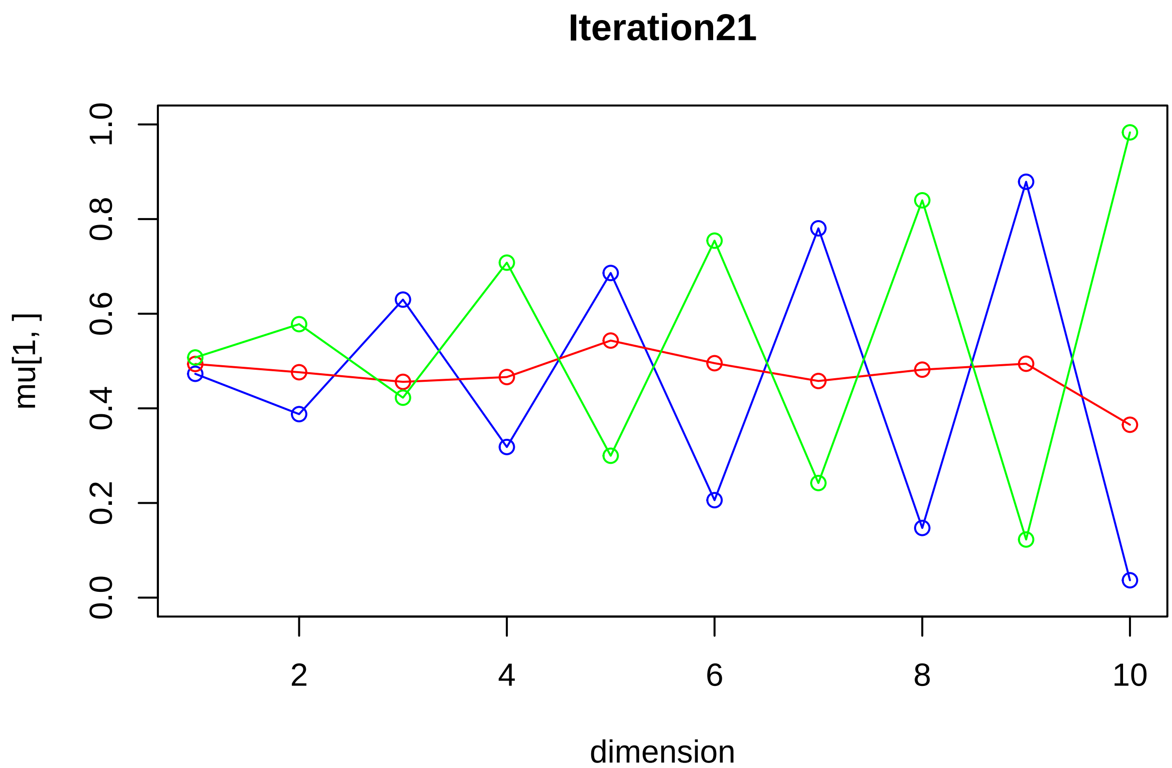

```
## iteration:  18 log likelihood:  -6760.727
```



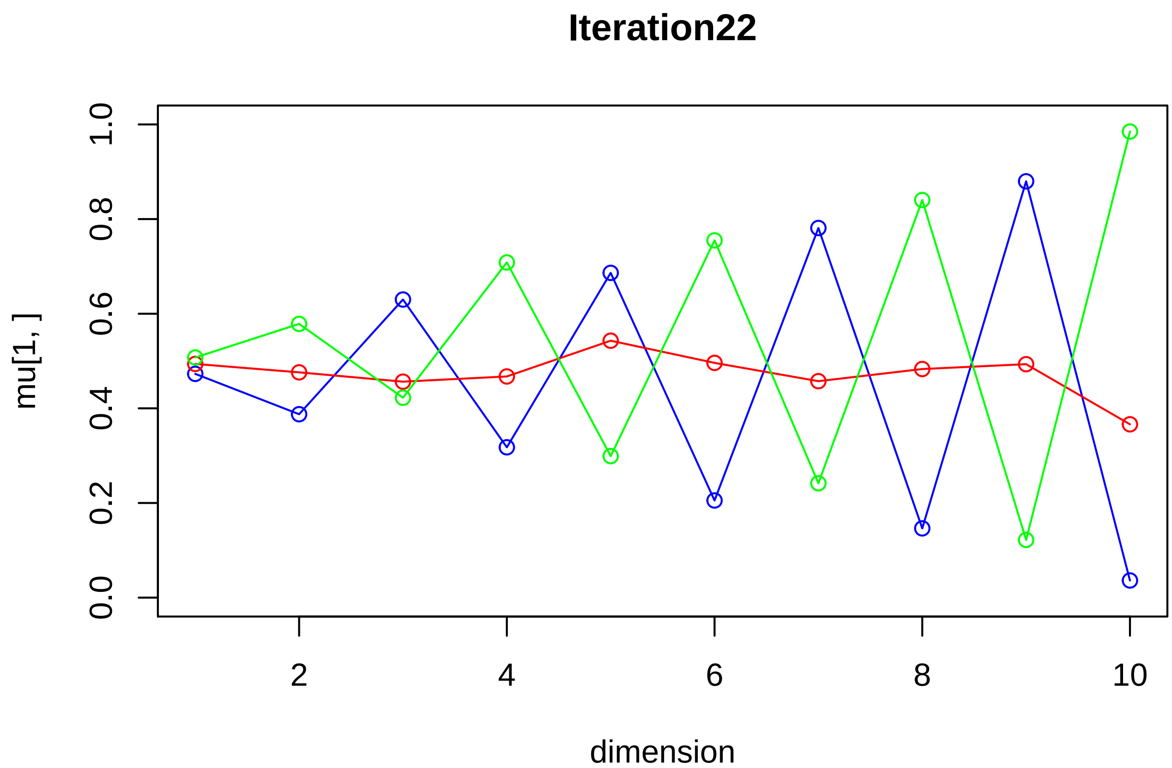

```
## iteration:  19 log likelihood:  -6759.572
```



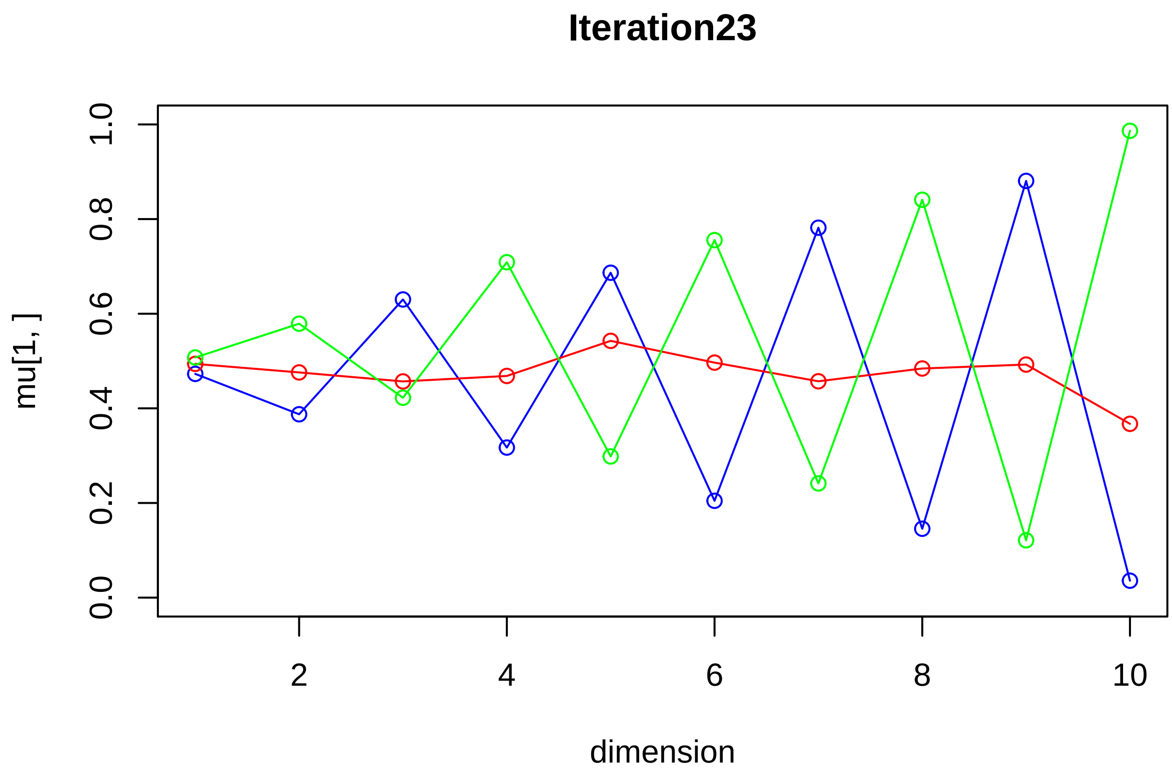

```
## iteration:  20 log likelihood:  -6758.491
```



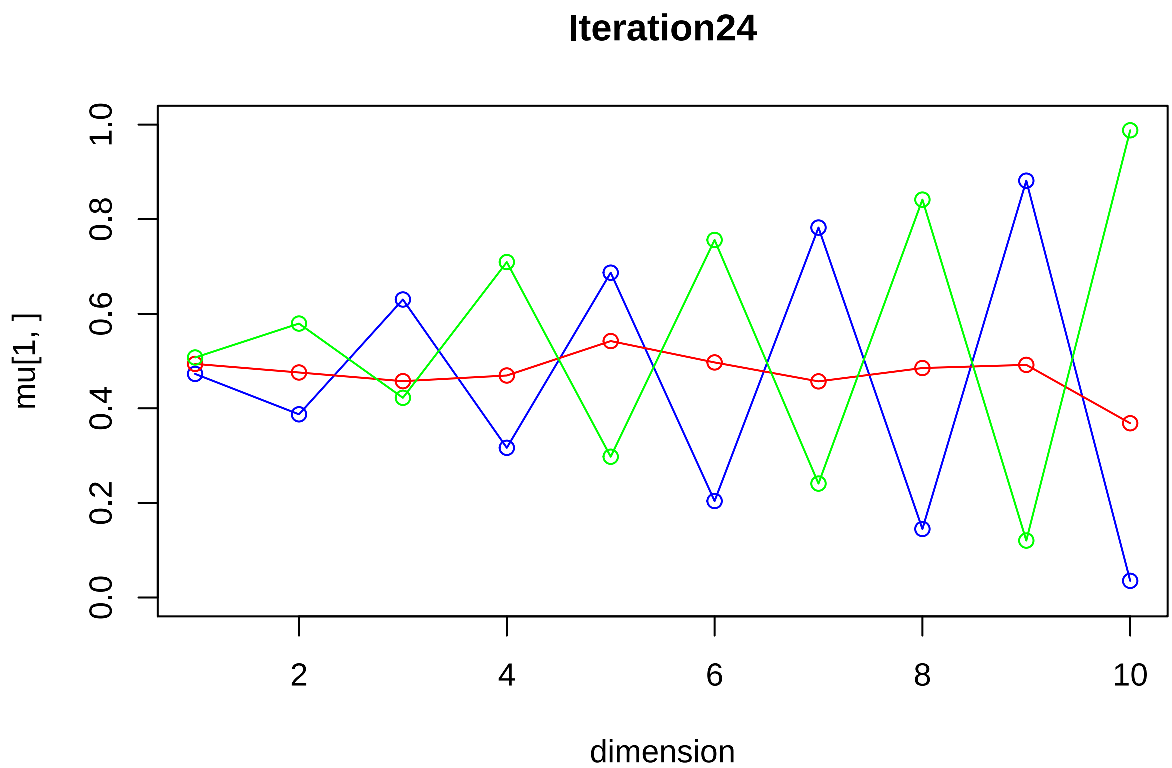

```
## iteration:  21 log likelihood:  -6757.475
```



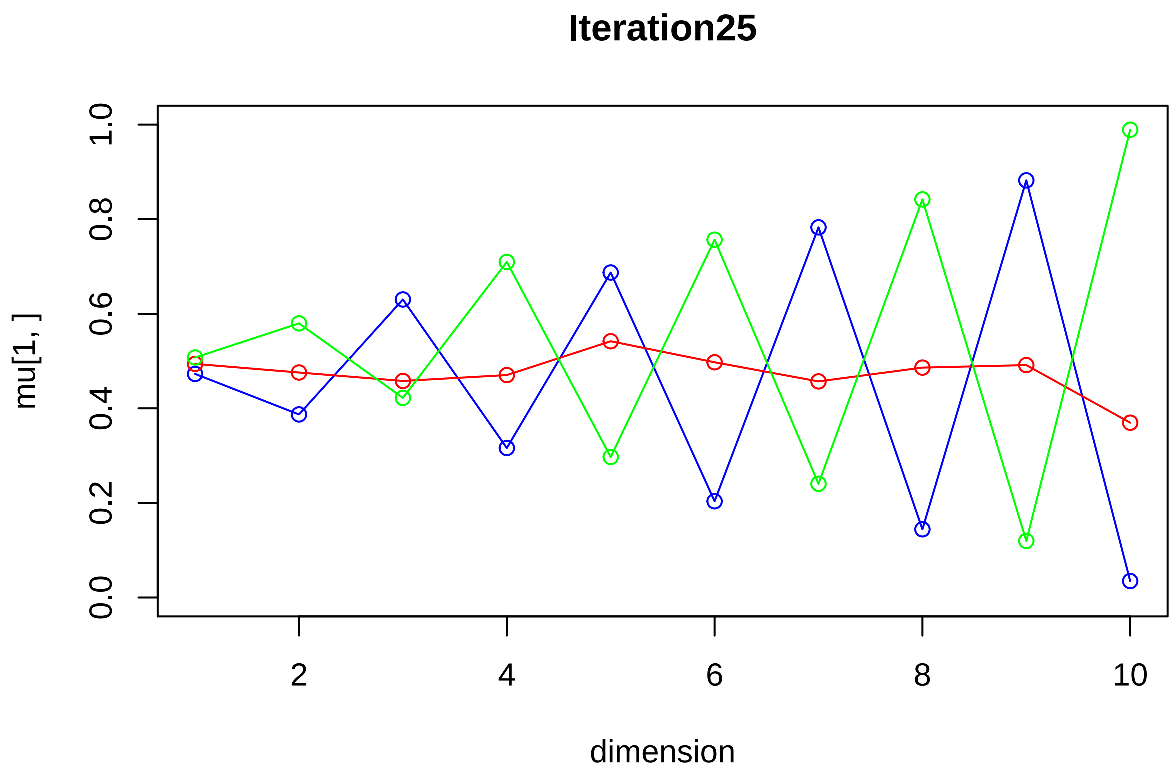

```
## iteration:  22 log likelihood:  -6756.521
```



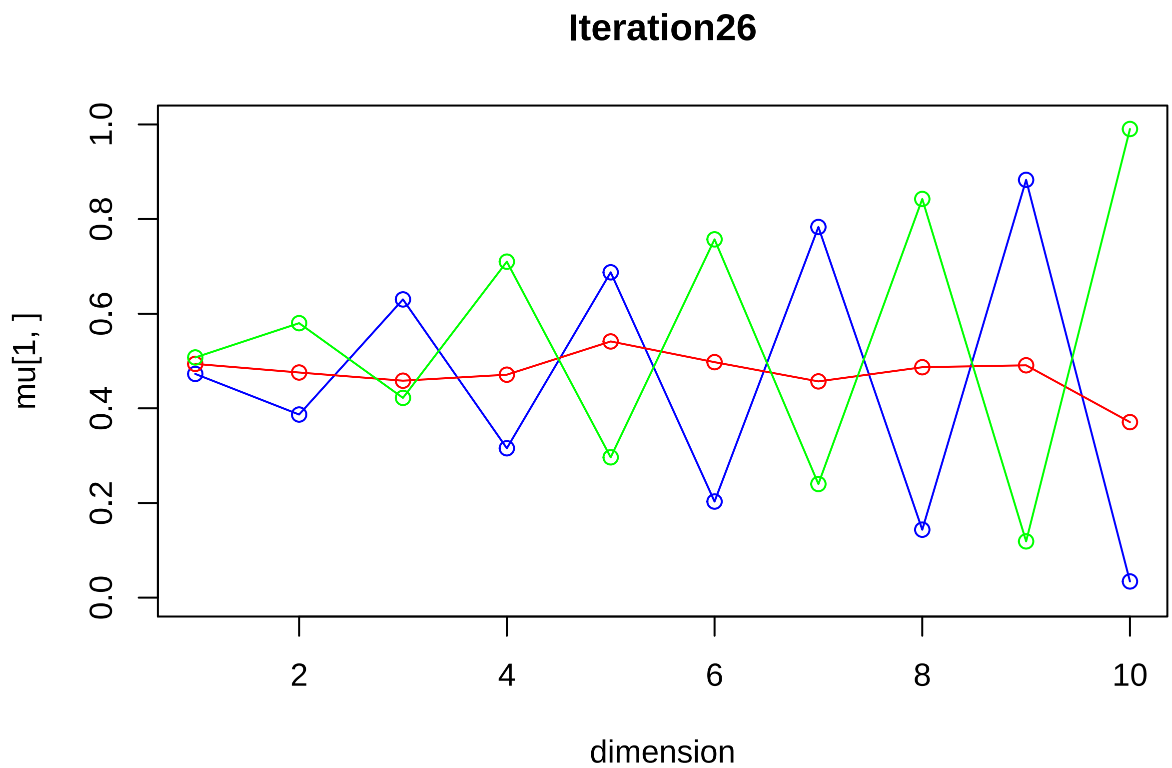

```
## iteration:  23 log likelihood:  -6755.625
```



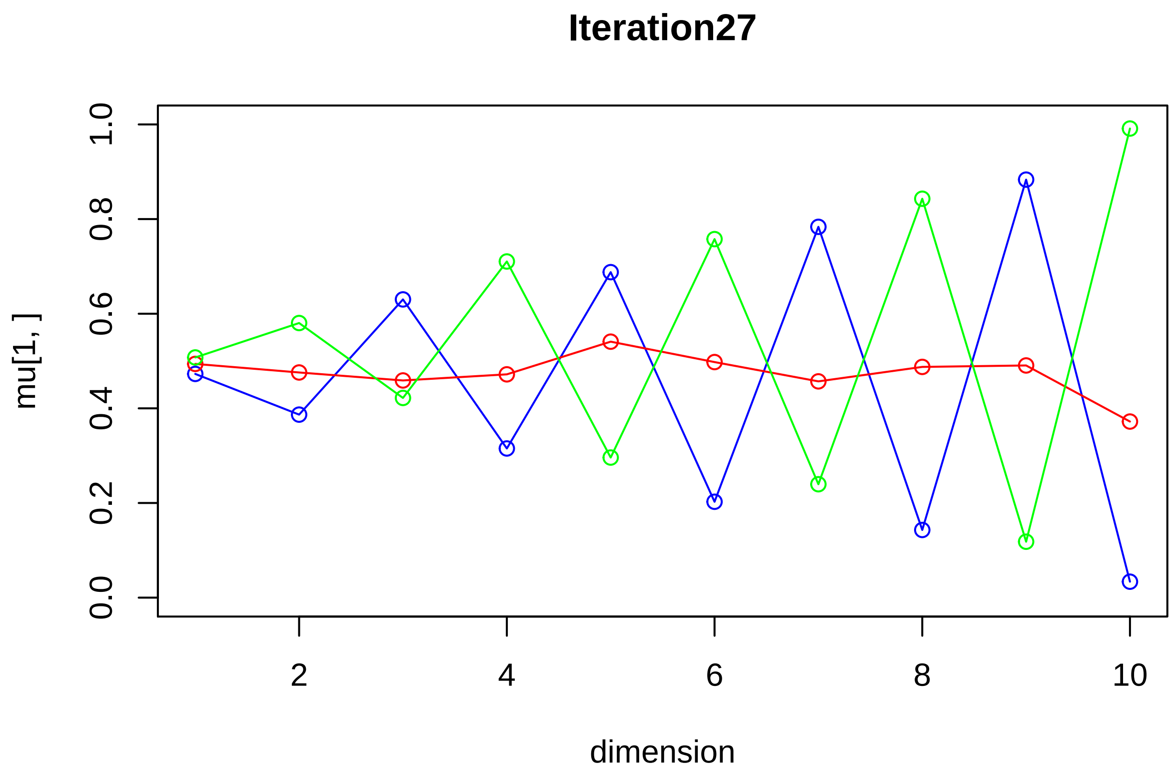

```
## iteration:  24 log likelihood:  -6754.784
```



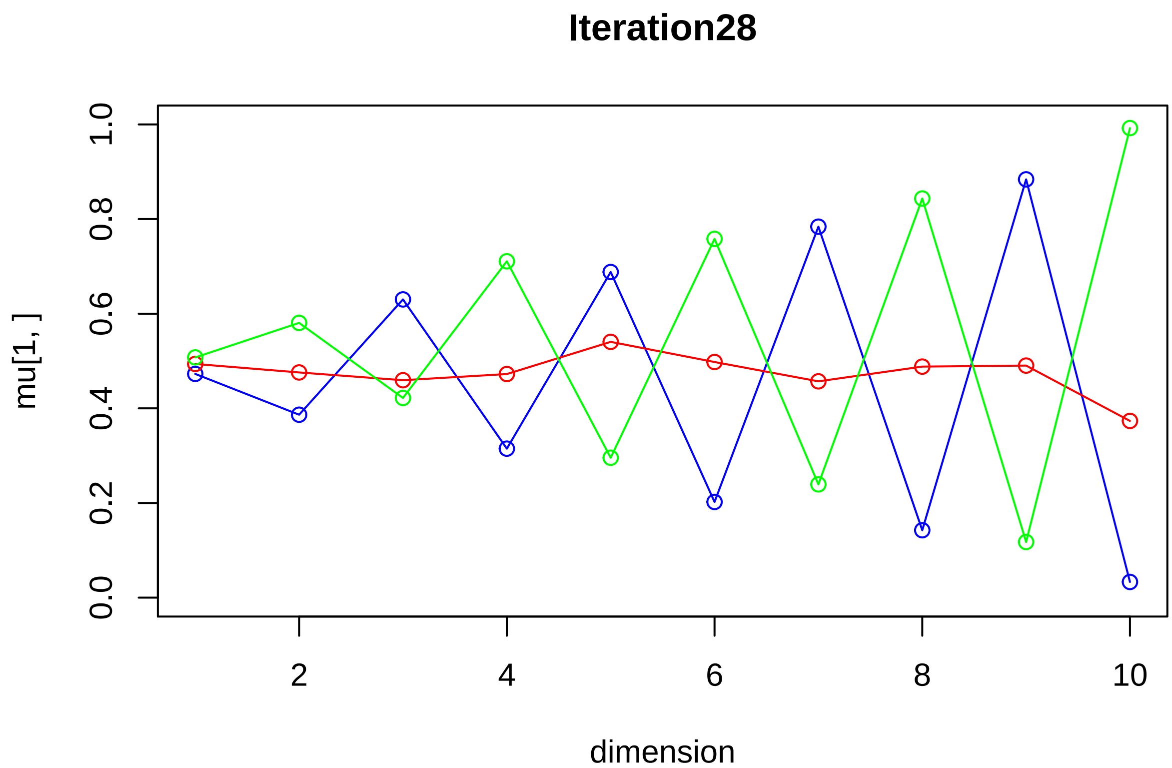

```
## iteration:  25 log likelihood:  -6753.996
```



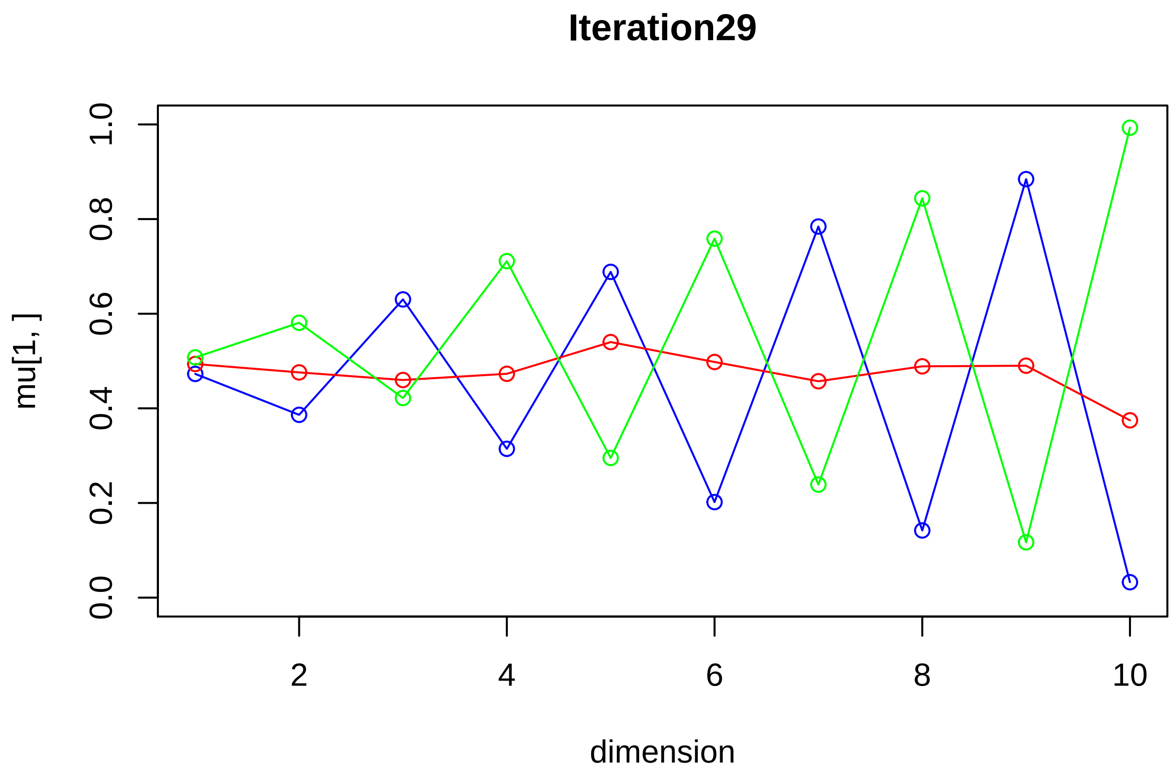

```
## iteration:  26 log likelihood:  -6753.26
```



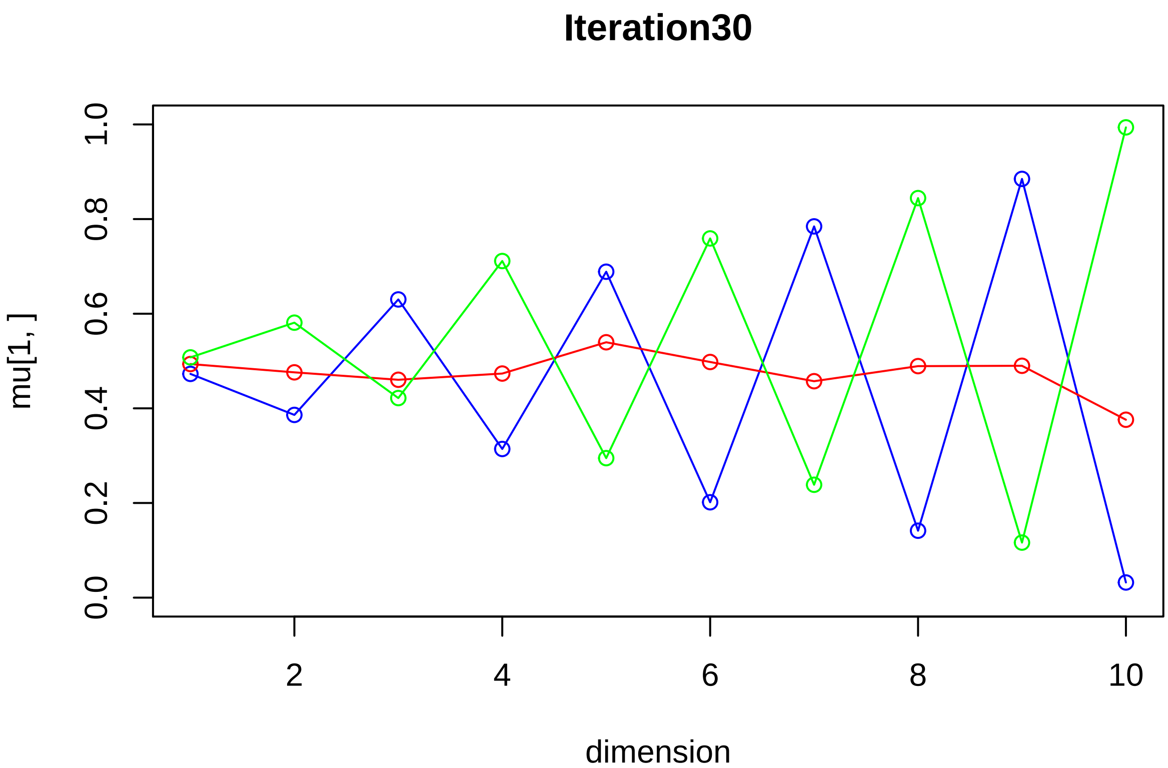

```
## iteration:  27 log likelihood:  -6752.571
```



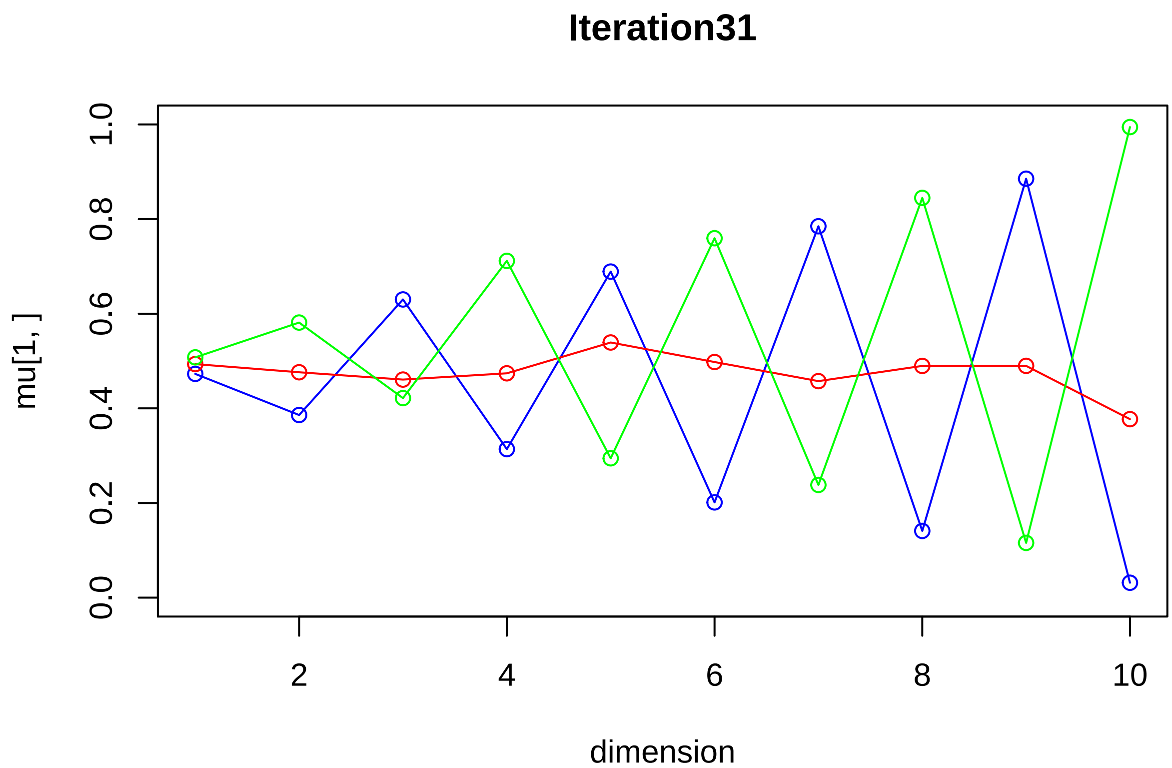

```
## iteration:  28 log likelihood:  -6751.928
```



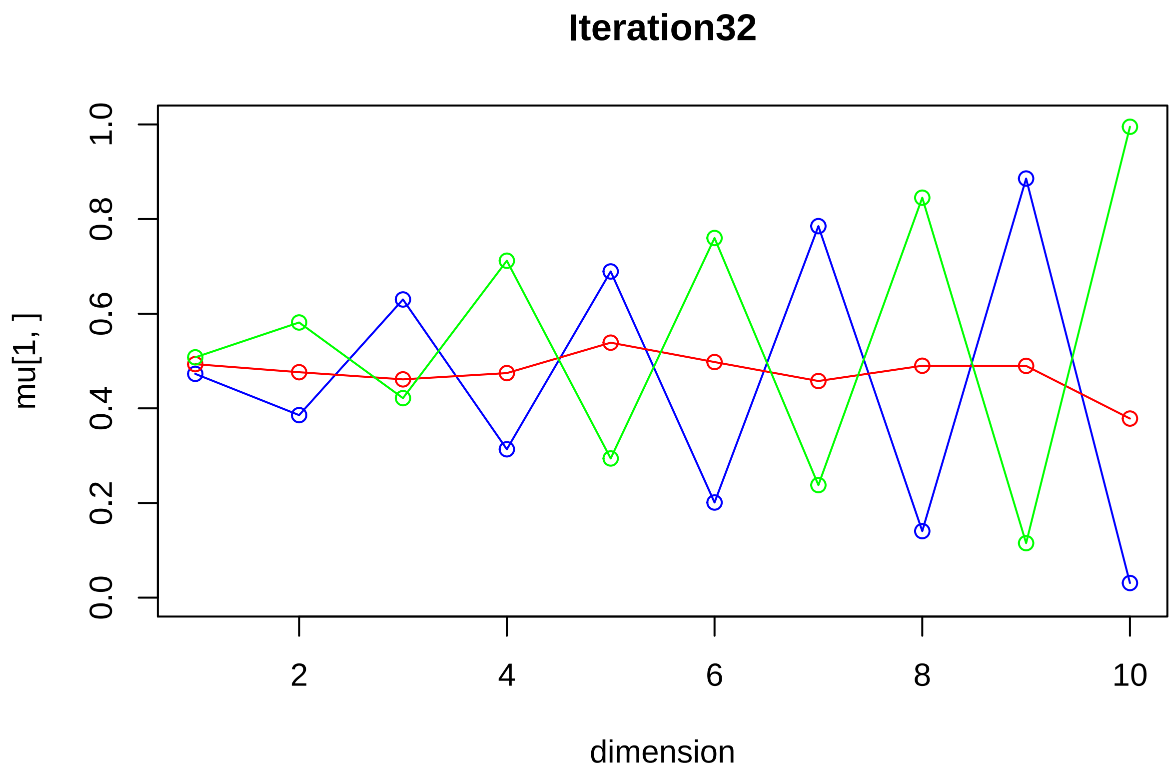

```
## iteration:  29 log likelihood:  -6751.328
```



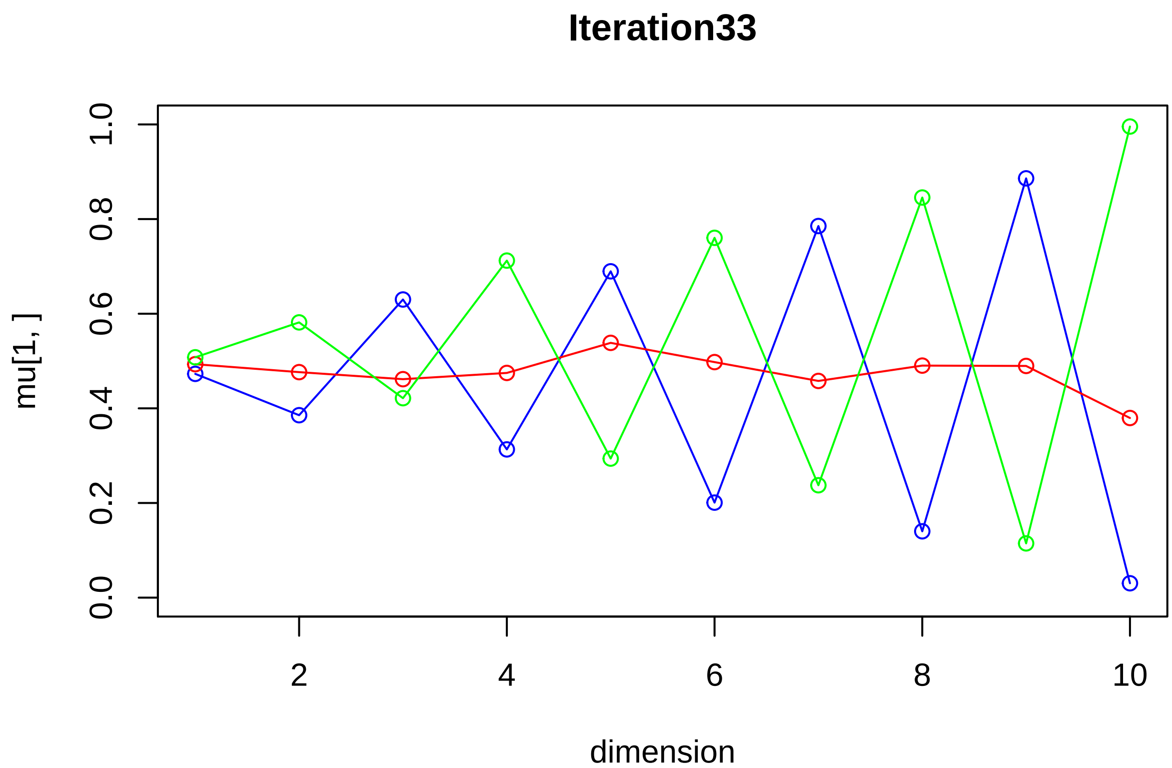

```
## iteration:  30 log likelihood:  -6750.768
```



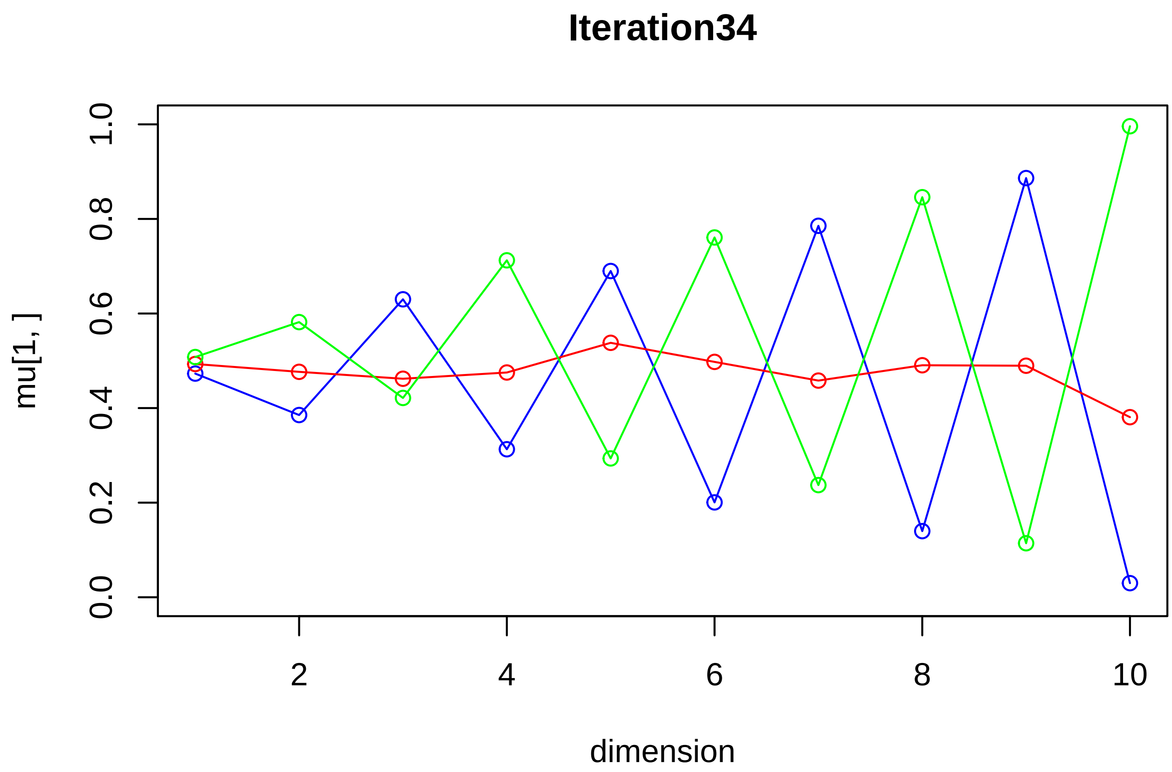

```
## iteration:  31 log likelihood:  -6750.246
```



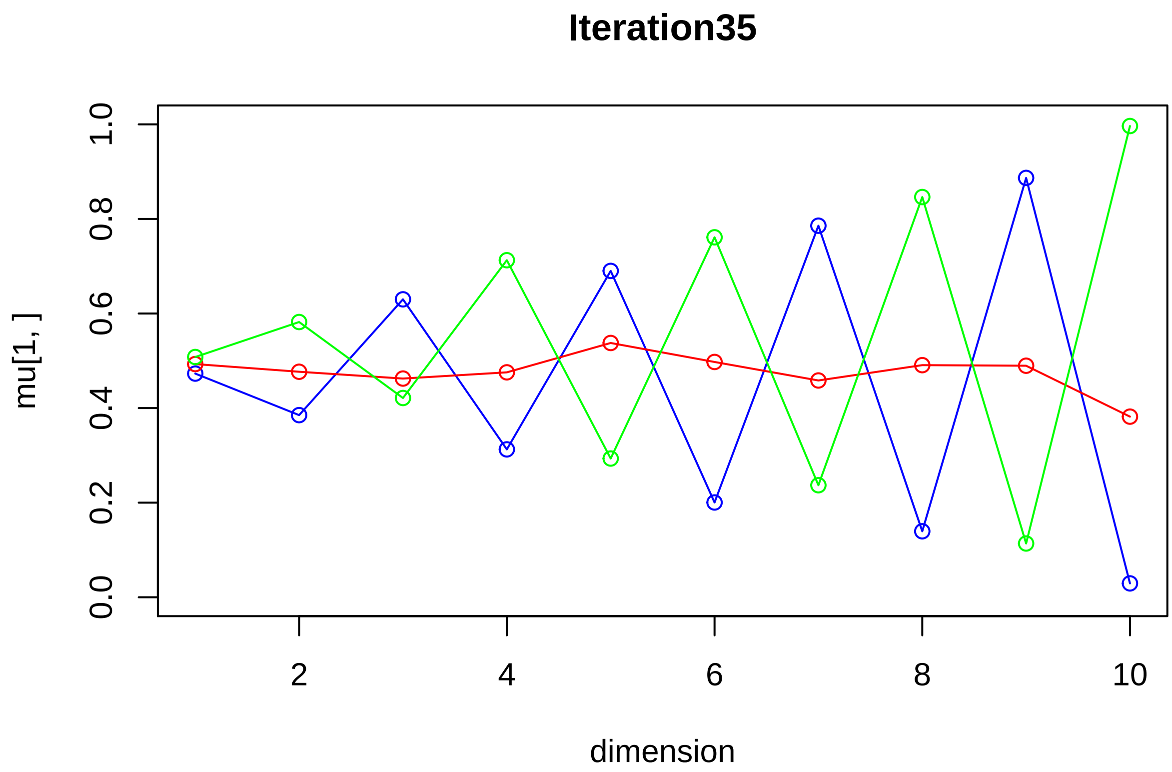

```
## iteration:  32 log likelihood:  -6749.758
```



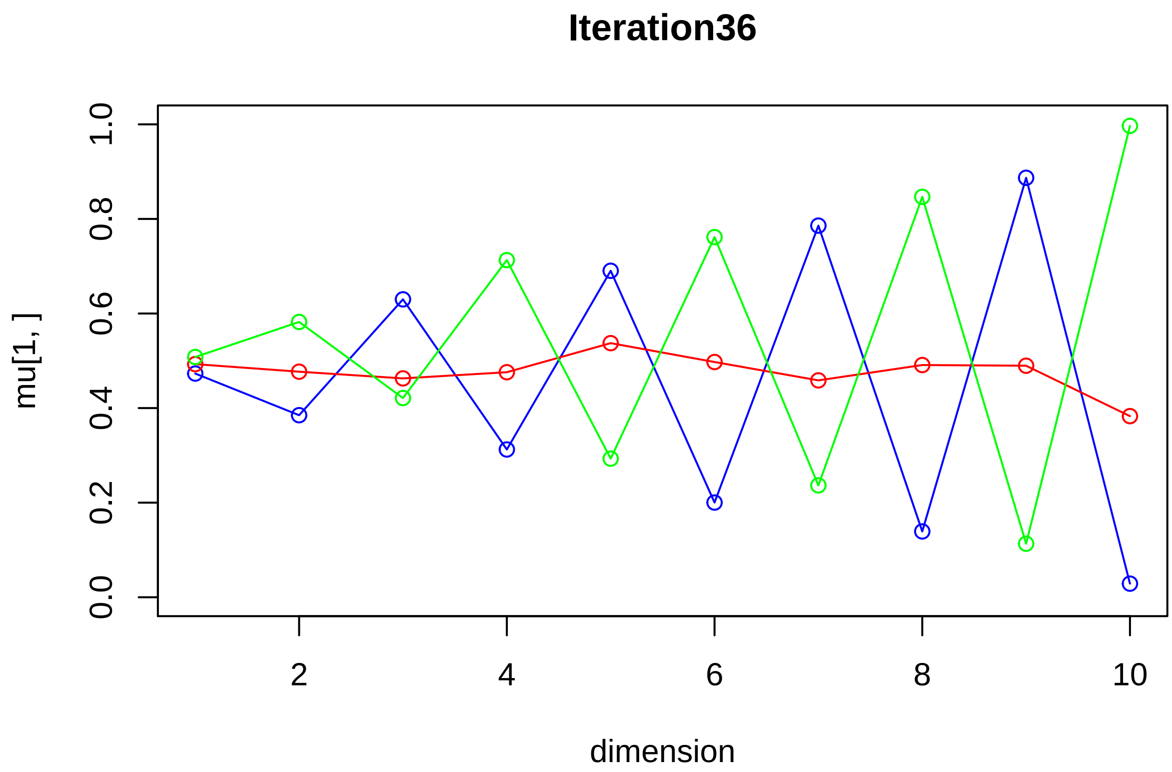

```
## iteration:  33 log likelihood:  -6749.304
```



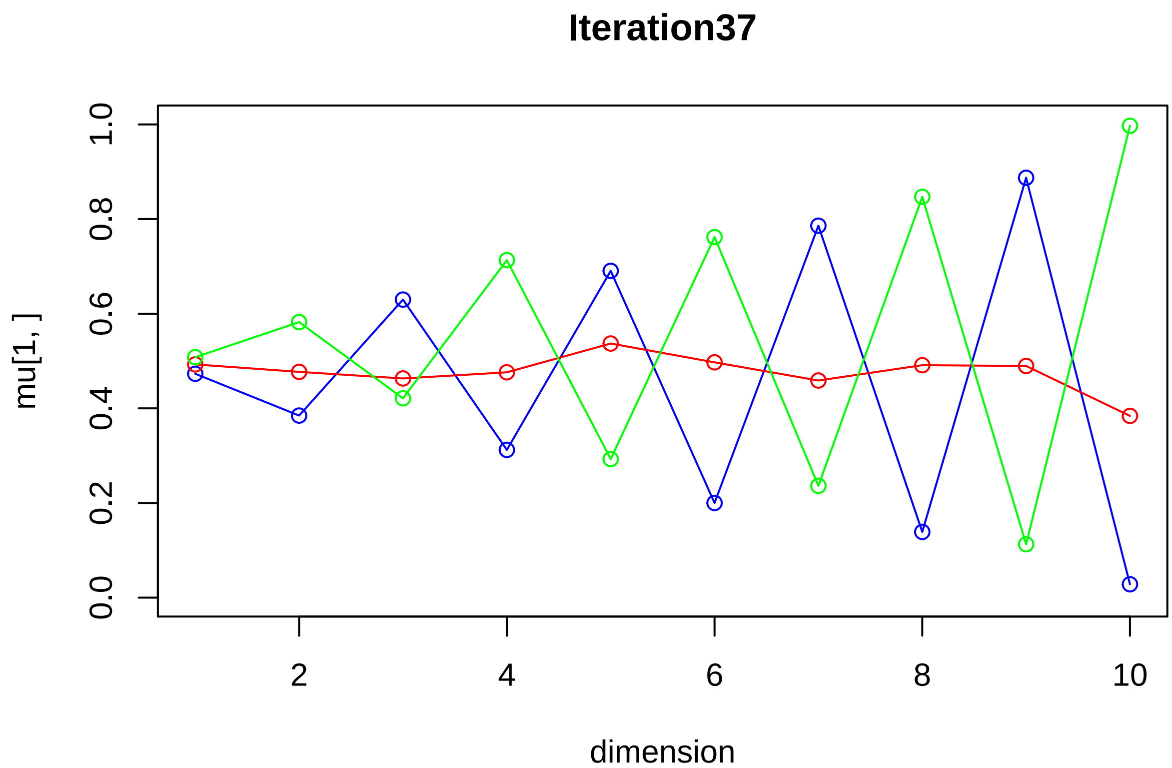

```
## iteration:  34 log likelihood:  -6748.88
```



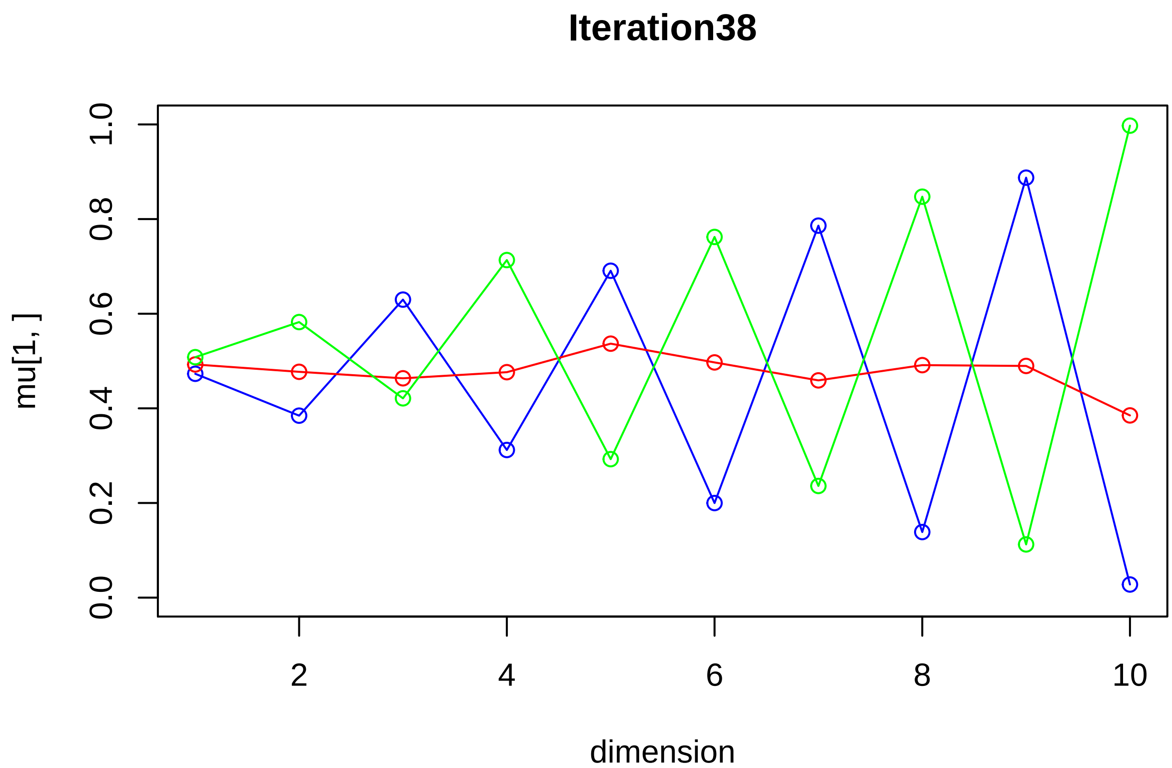

```
## iteration:  35 log likelihood:  -6748.484
```



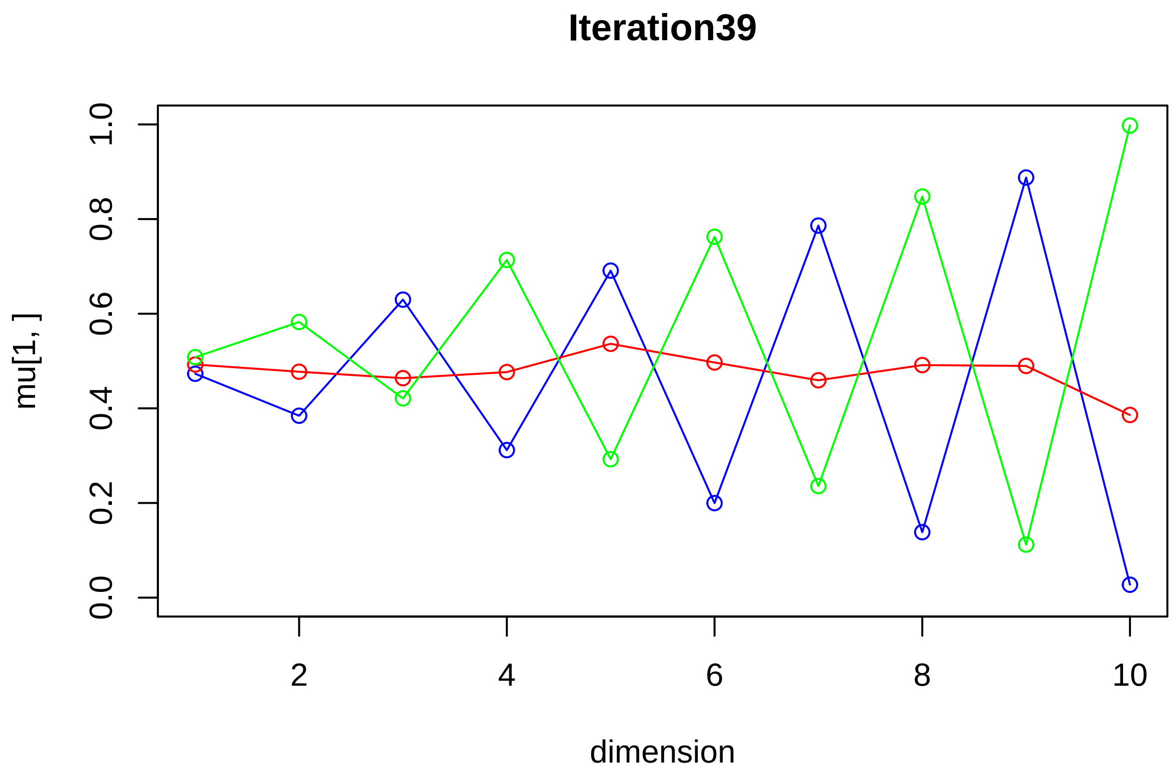

```
## iteration:  36 log likelihood:  -6748.114
```



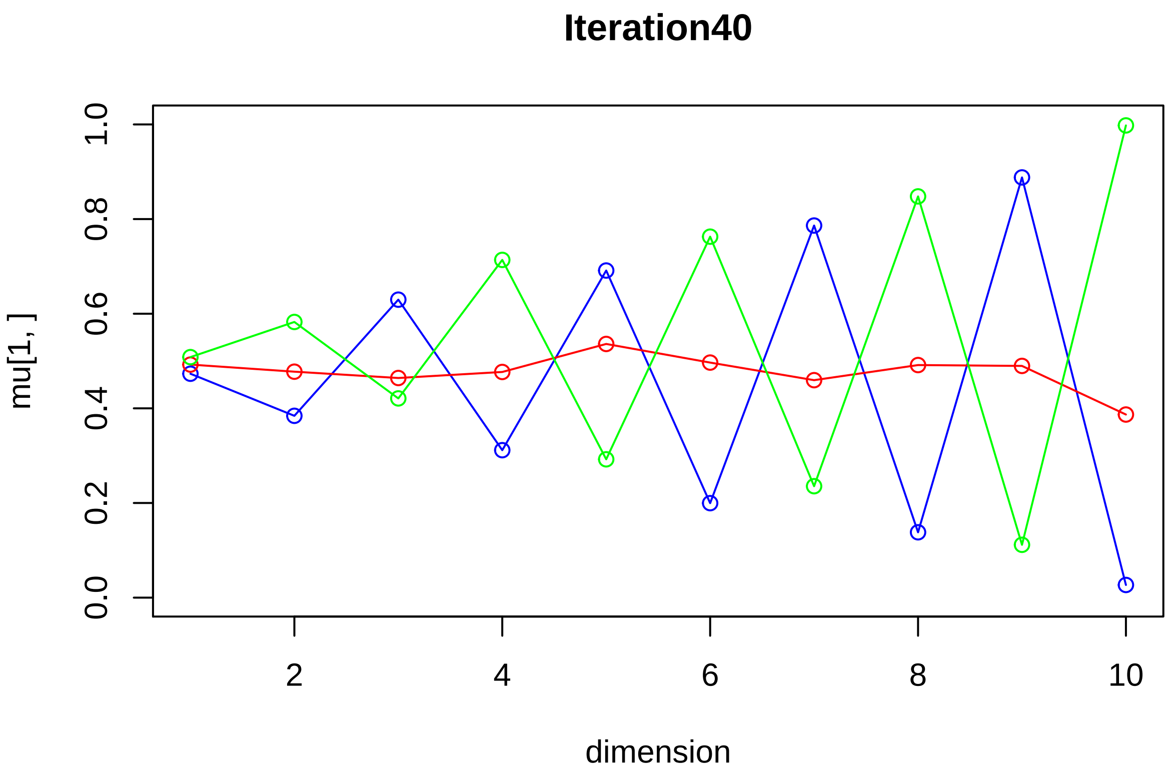

```
## iteration:  37 log likelihood:  -6747.767
```



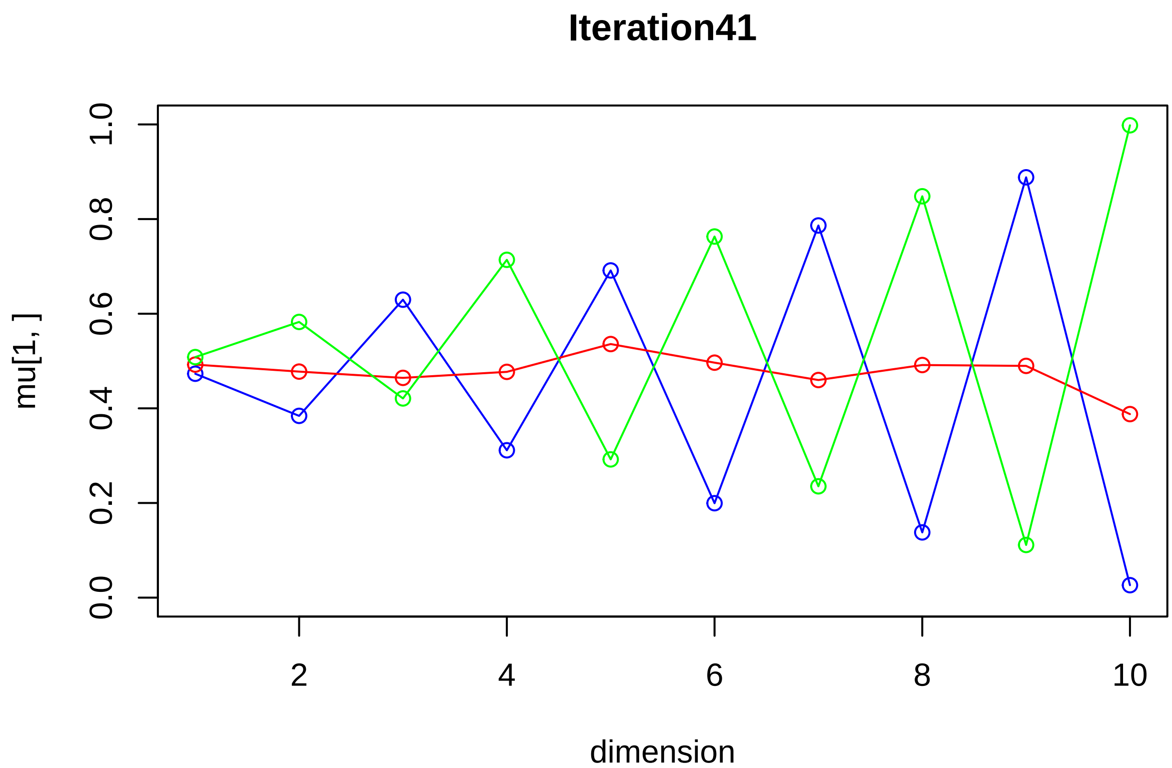

```
## iteration:  38 log likelihood:  -6747.444
```



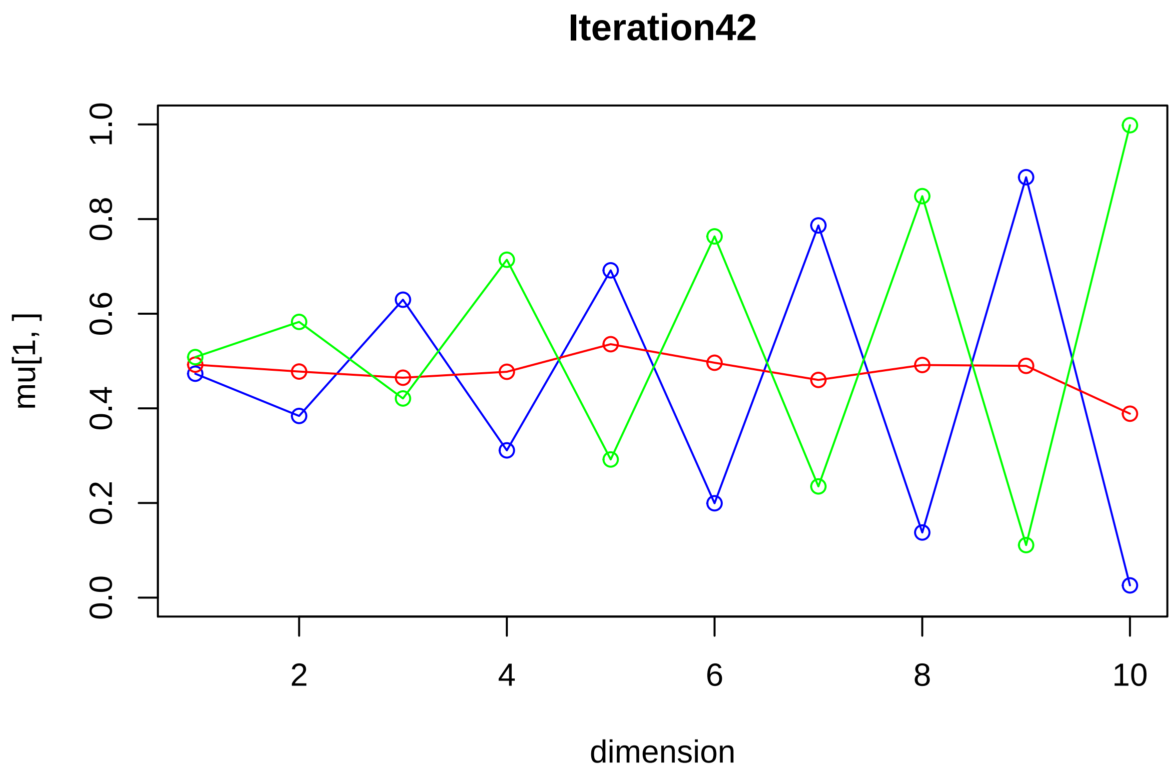

```
## iteration:  39 log likelihood:  -6747.14
```



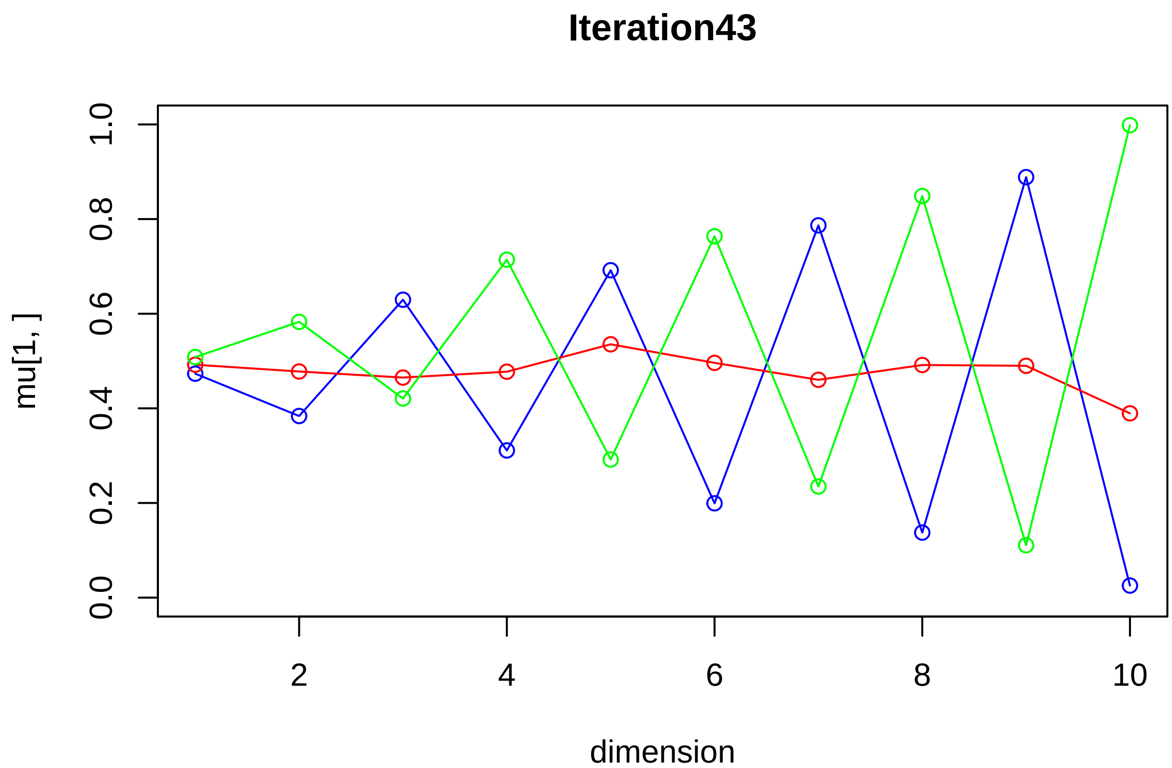

```
## iteration:  40 log likelihood:  -6746.856
```



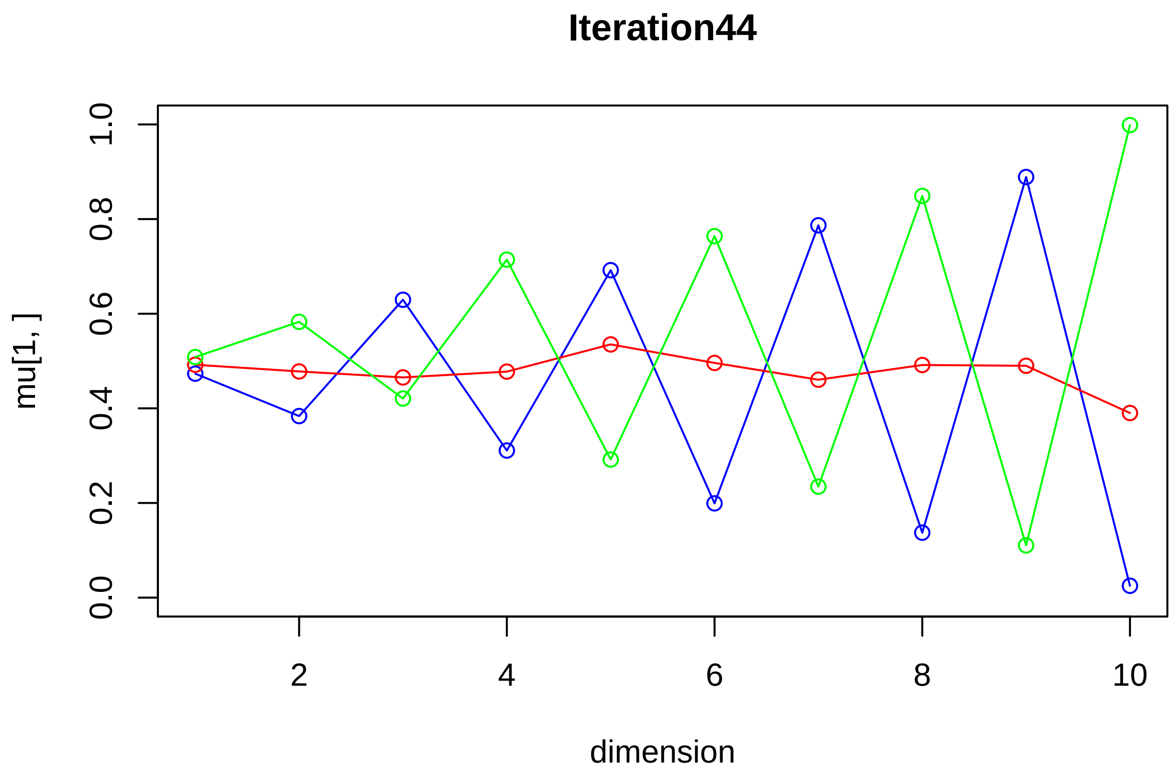

```
## iteration:  41 log likelihood:  -6746.589
```



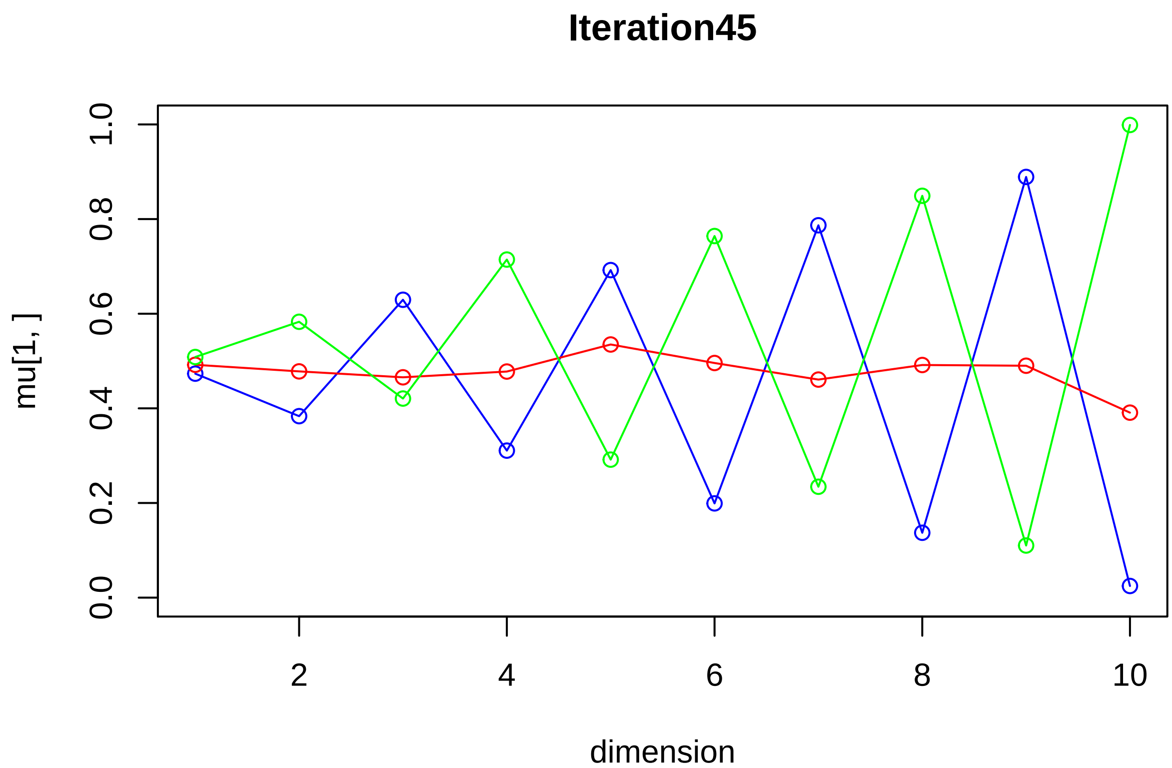

```
## iteration:  42 log likelihood:  -6746.338
```



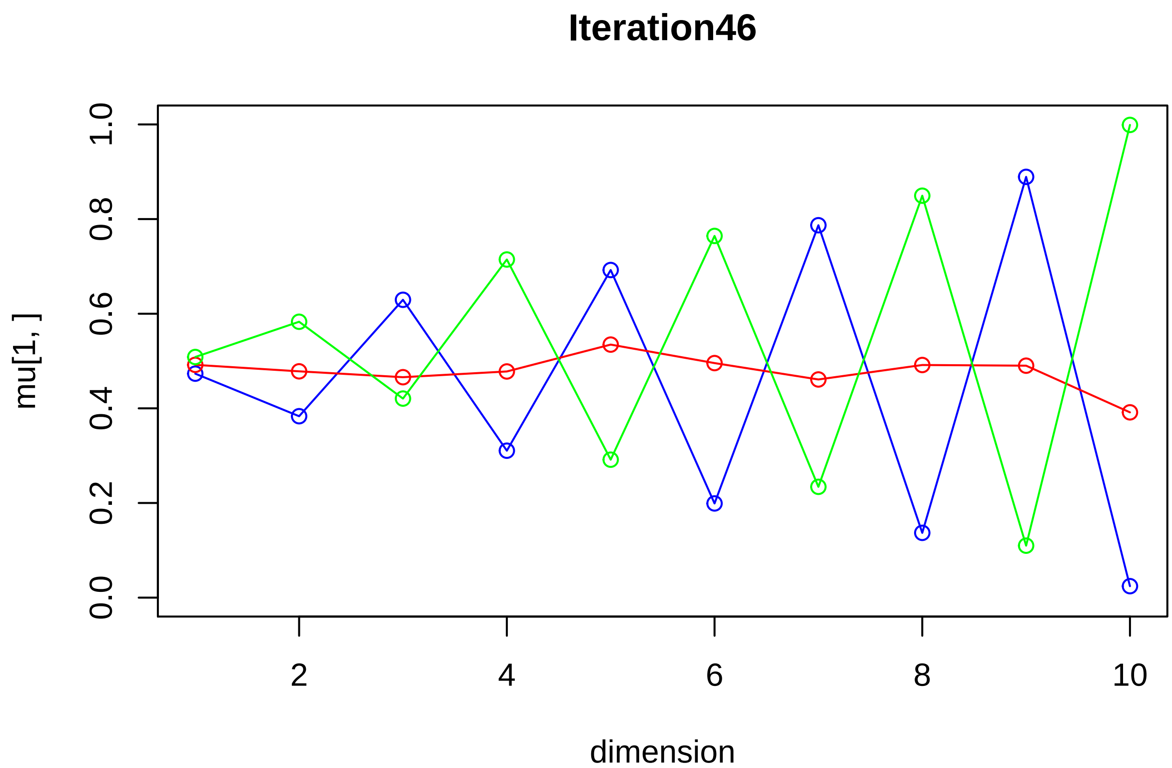

```
## iteration:  43 log likelihood:  -6746.102
```



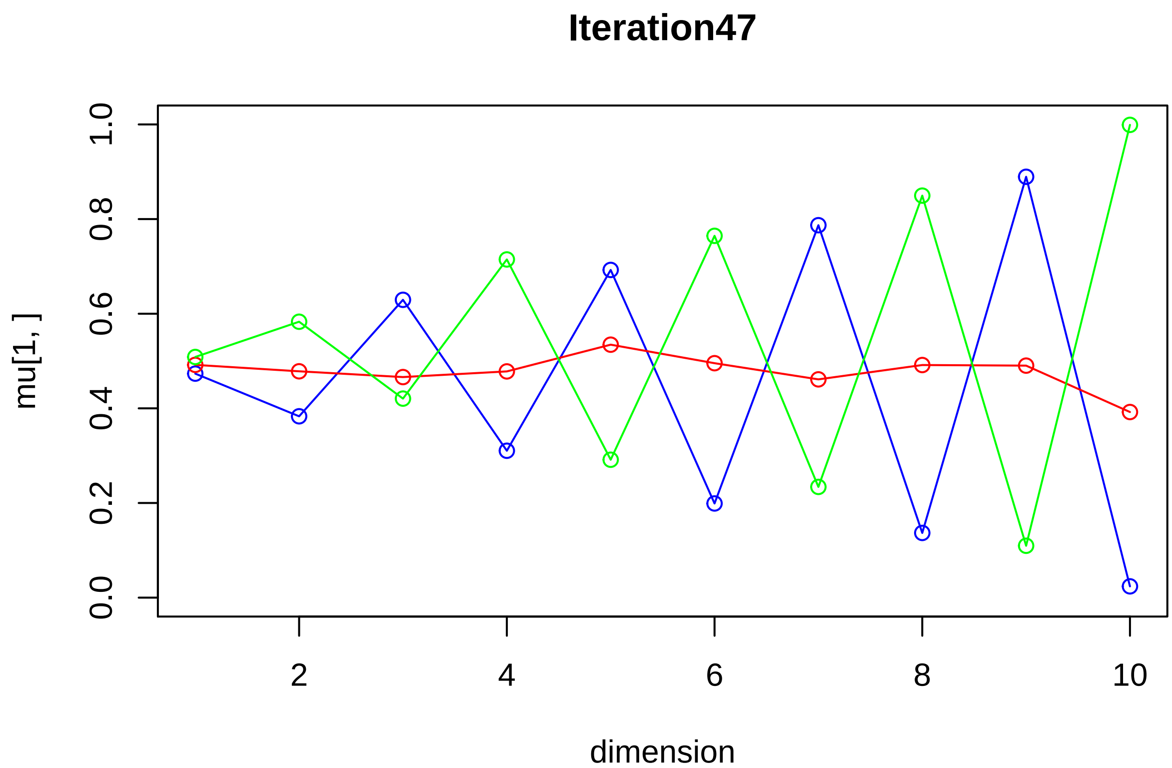

```
## iteration:  44 log likelihood:  -6745.88
```



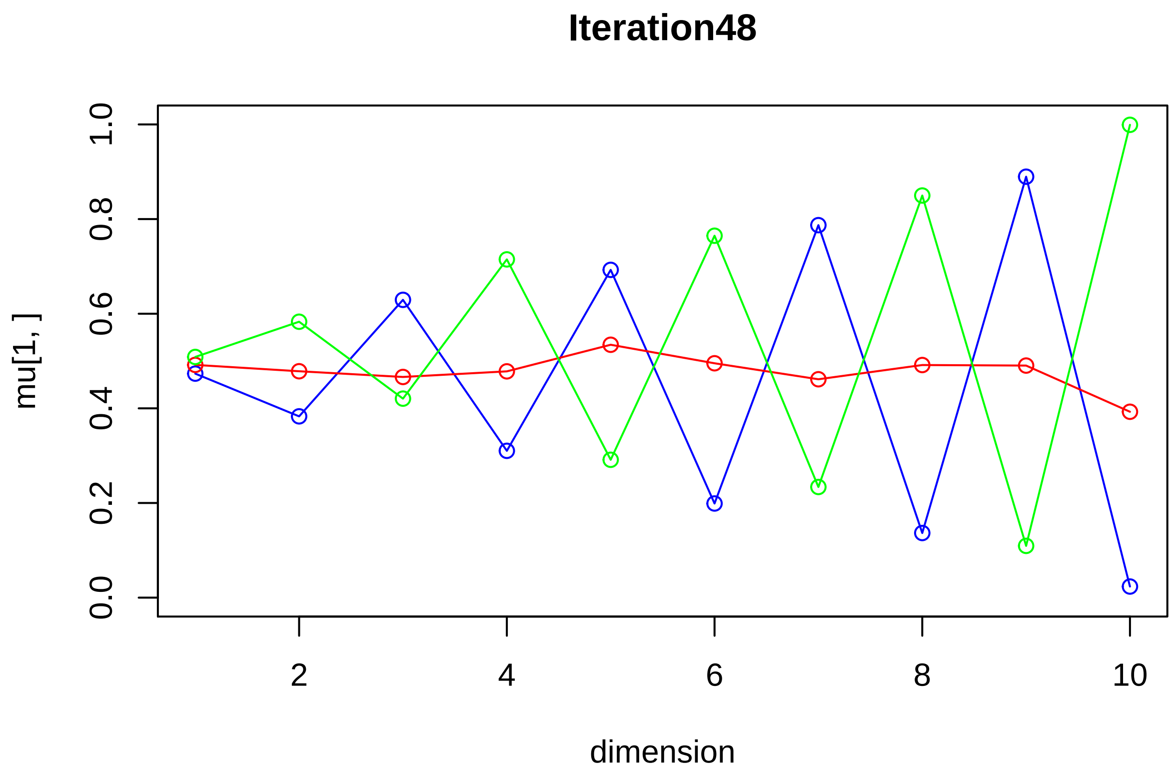

```
## iteration:  45 log likelihood:  -6745.67
```



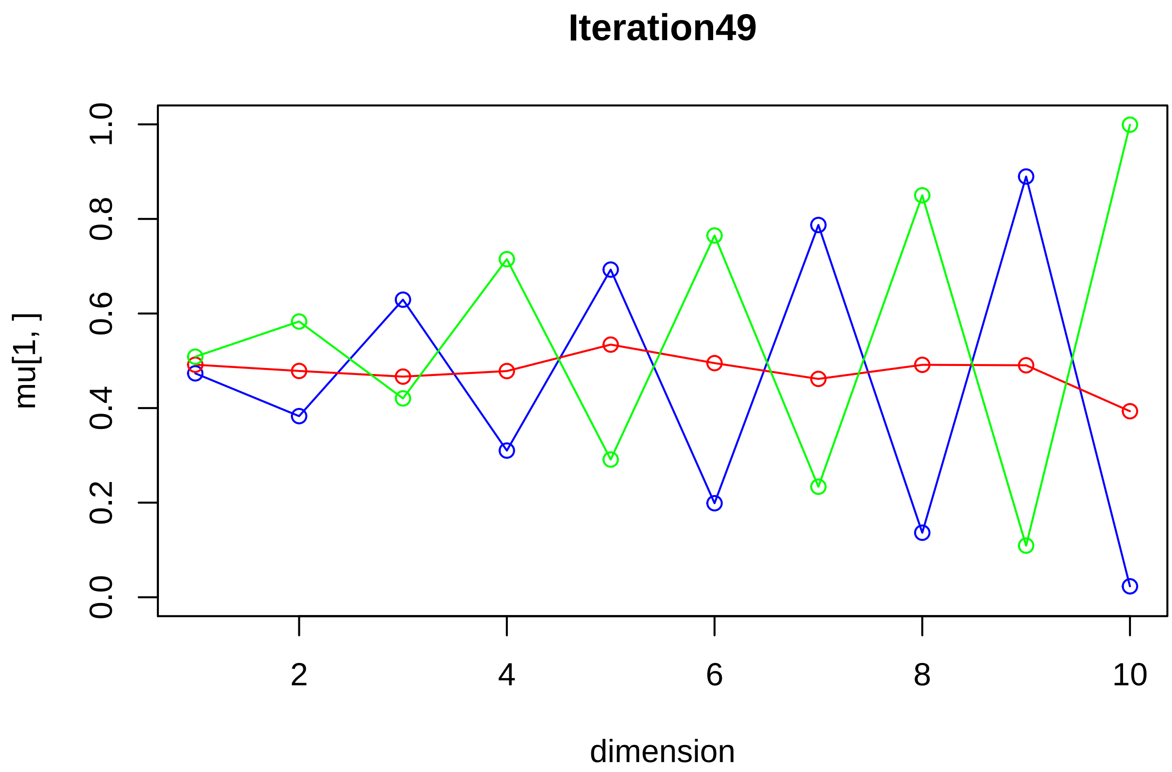

```
## iteration:  46 log likelihood:  -6745.472
```



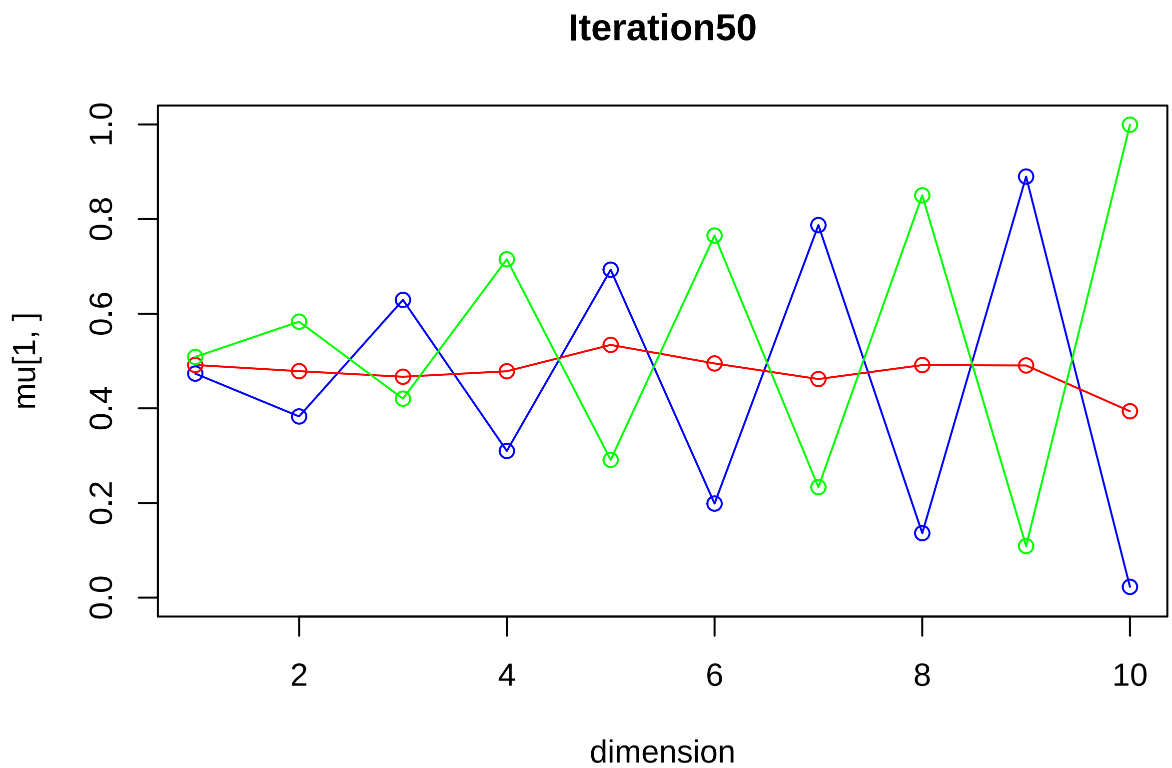

```
## iteration:  47 log likelihood:  -6745.285
```



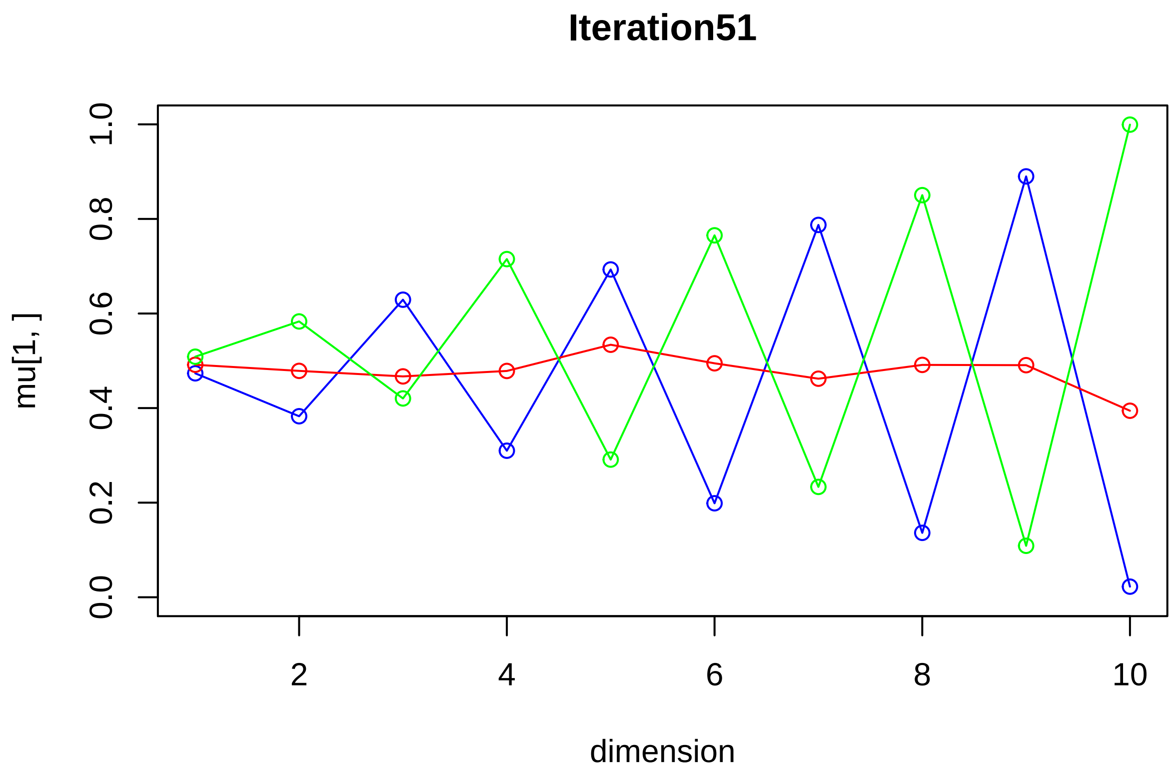

```
## iteration:  48 log likelihood:  -6745.108
```



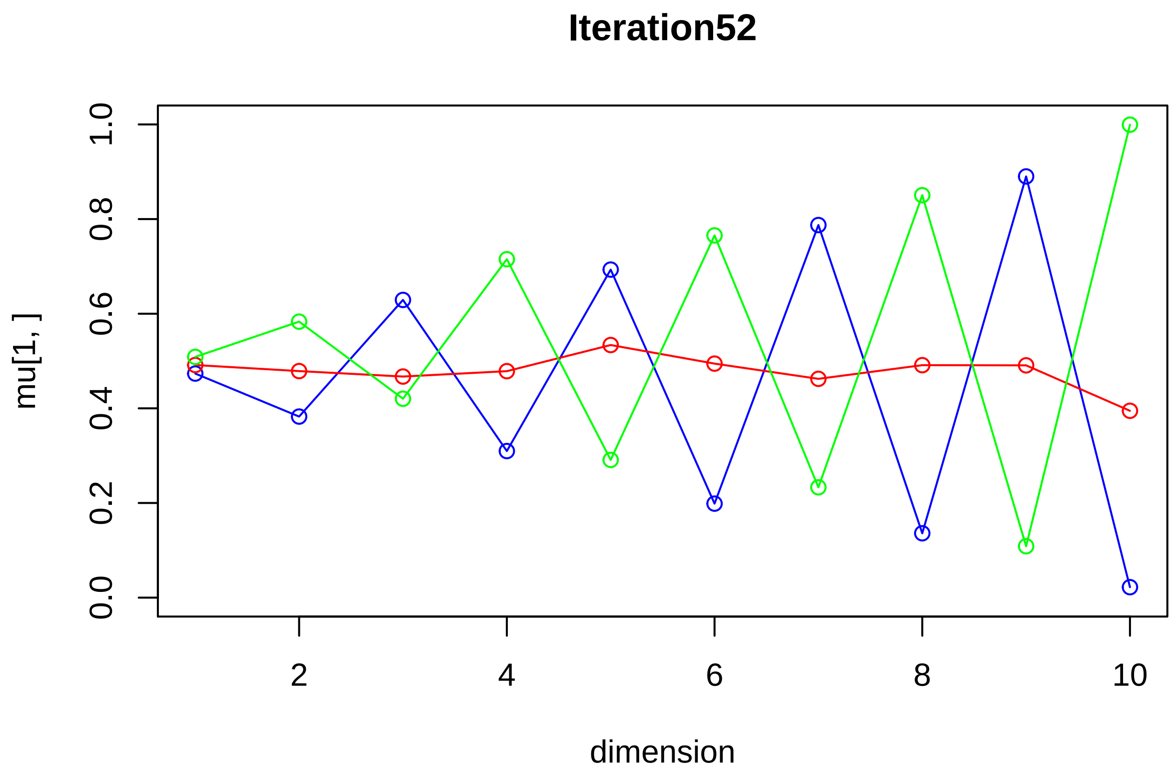

```
## iteration:  49 log likelihood:  -6744.939
```



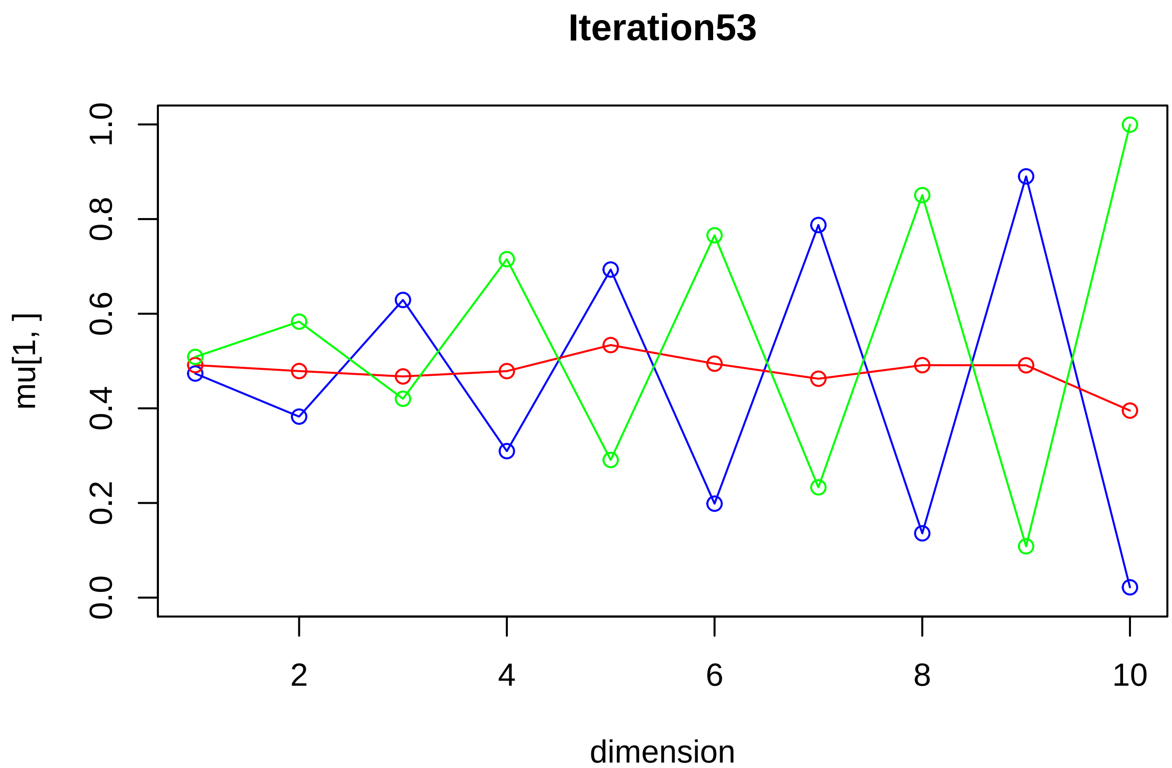

```
## iteration:  50 log likelihood:  -6744.78
```



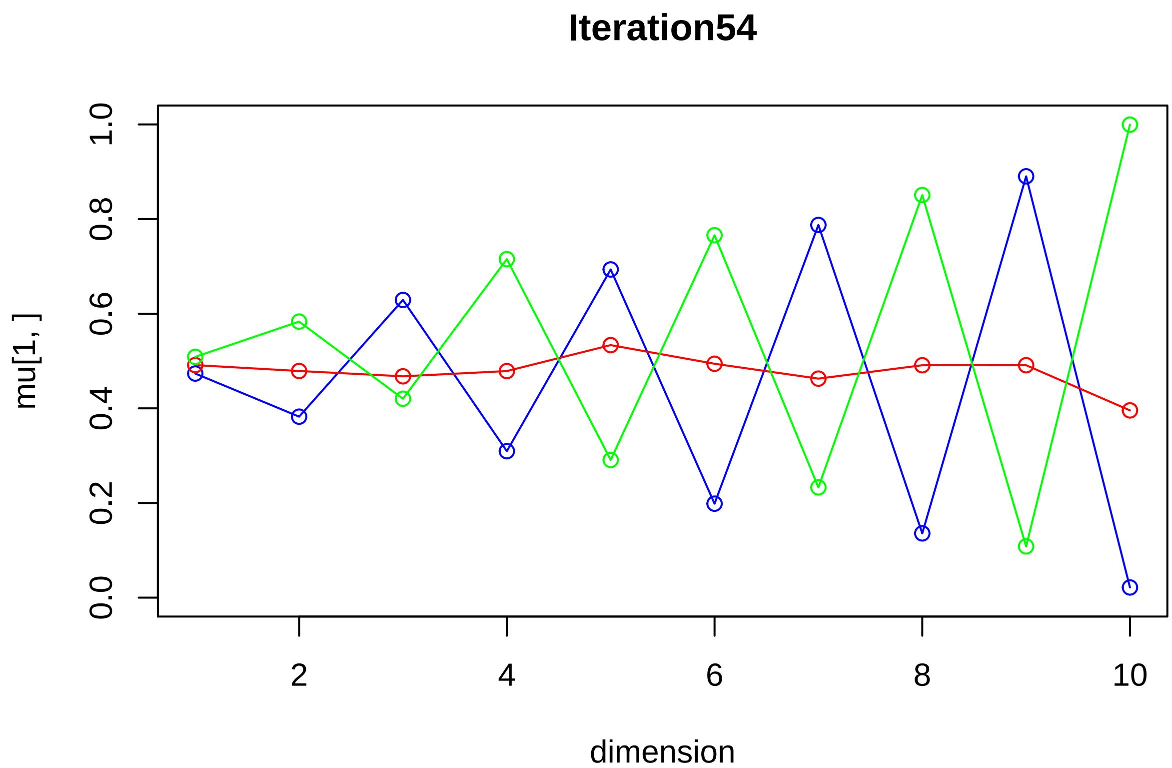

```
## iteration:  51 log likelihood:  -6744.627
```



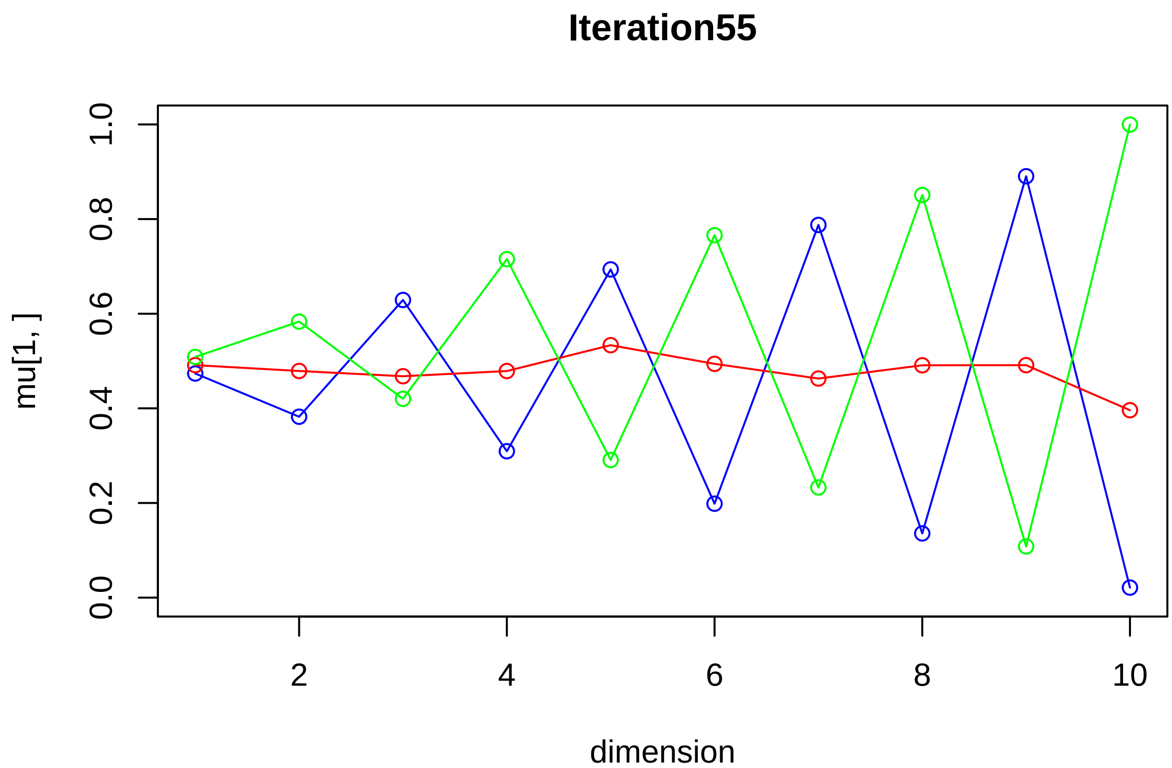

```
## iteration:  52 log likelihood:  -6744.483
```



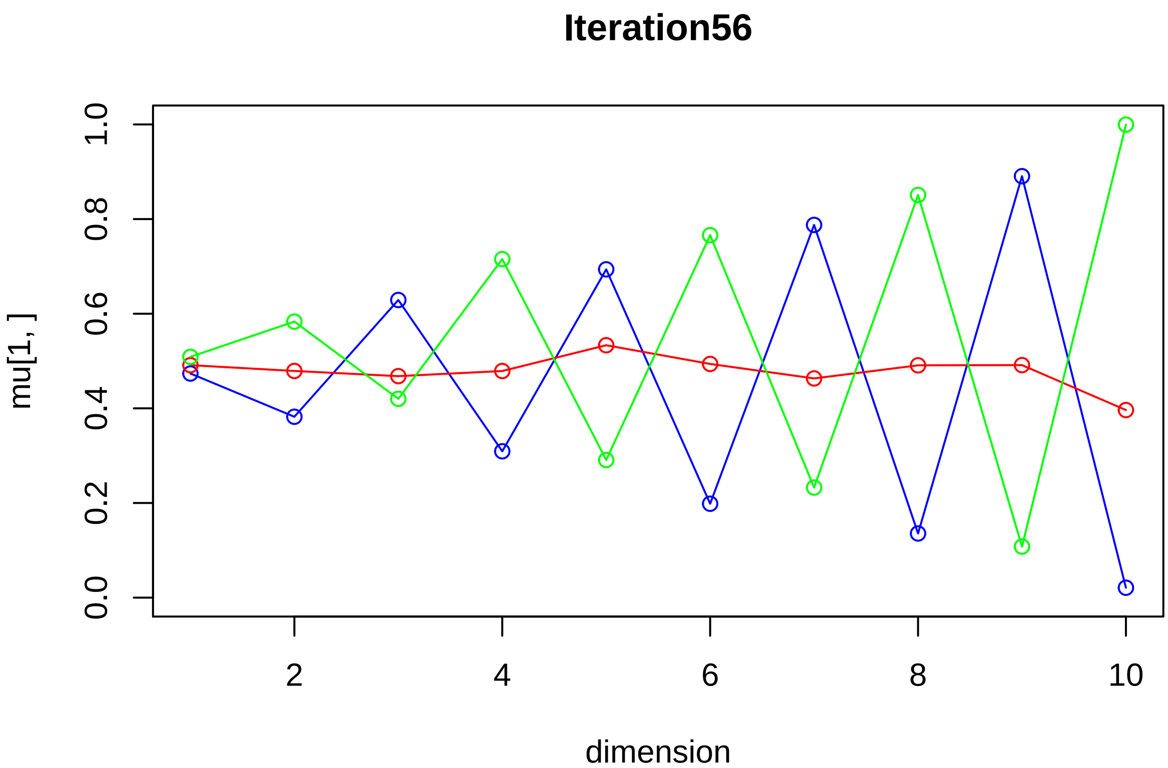

```
## iteration:  53 log likelihood:  -6744.344
```



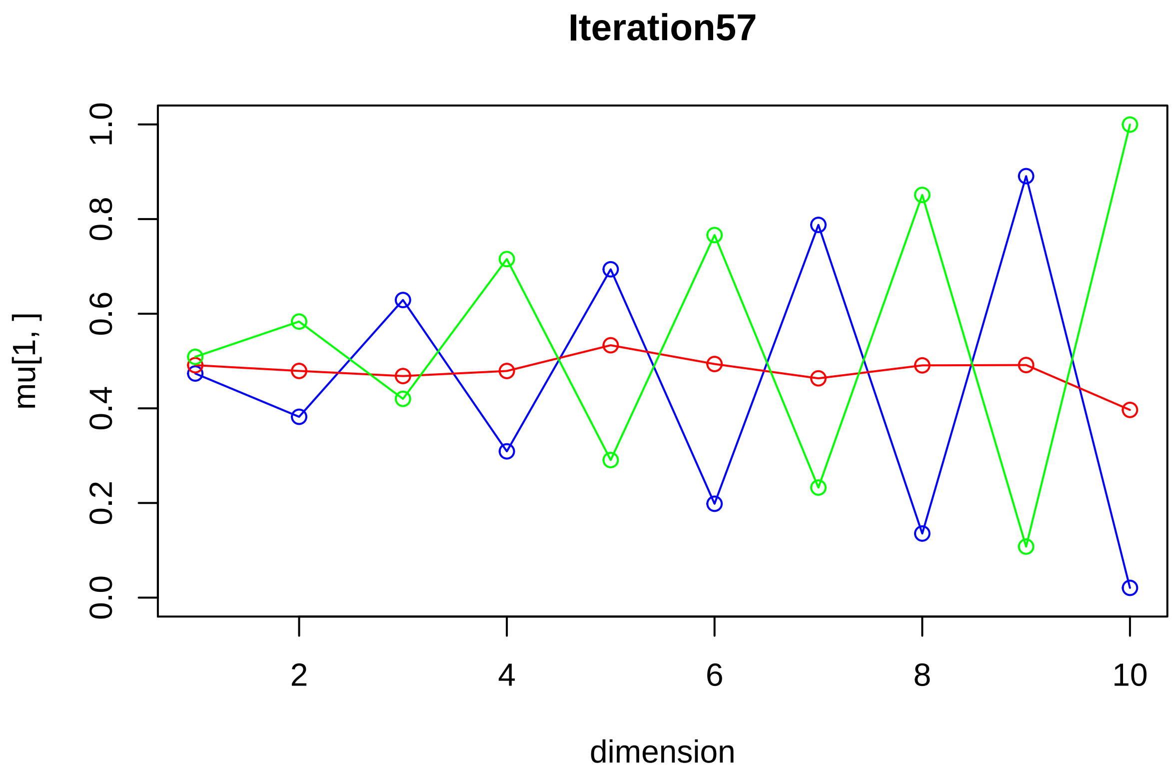

```
## iteration:  54 log likelihood:  -6744.212
```



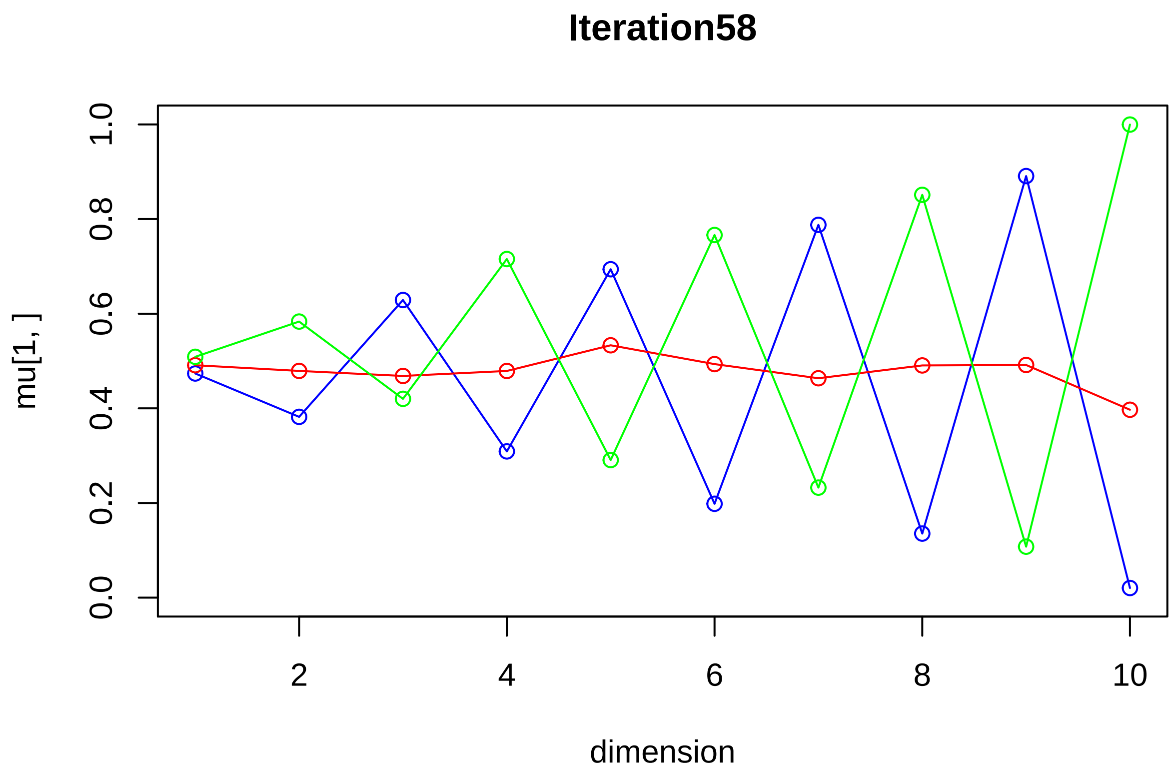

```
## iteration:  55 log likelihood:  -6744.086
```



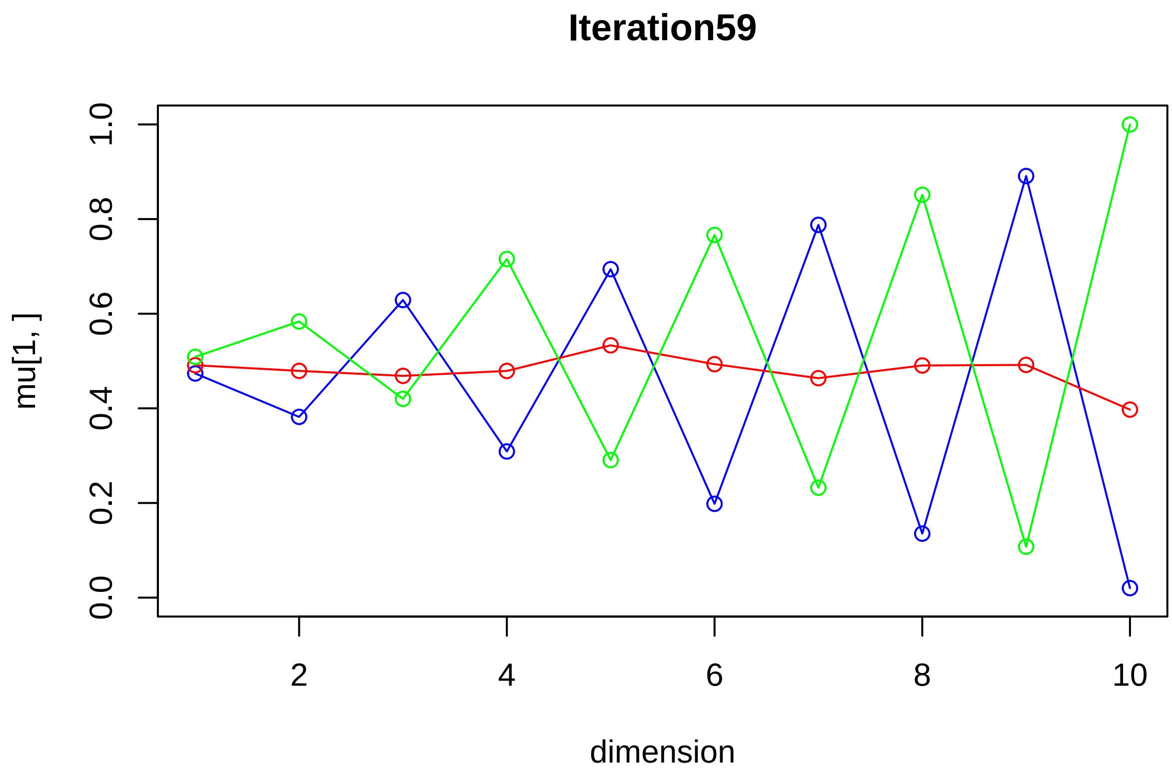

```
## iteration:  56 log likelihood:  -6743.964
```



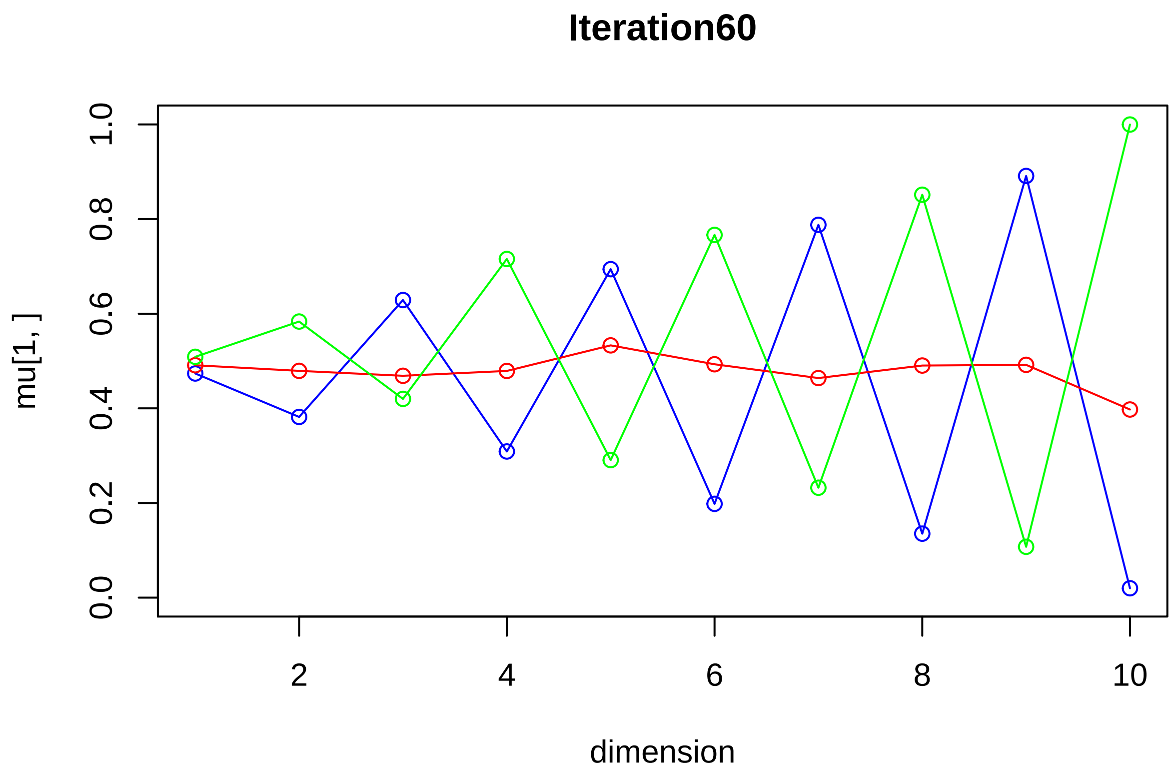

```
## iteration:  57 log likelihood:  -6743.848
```



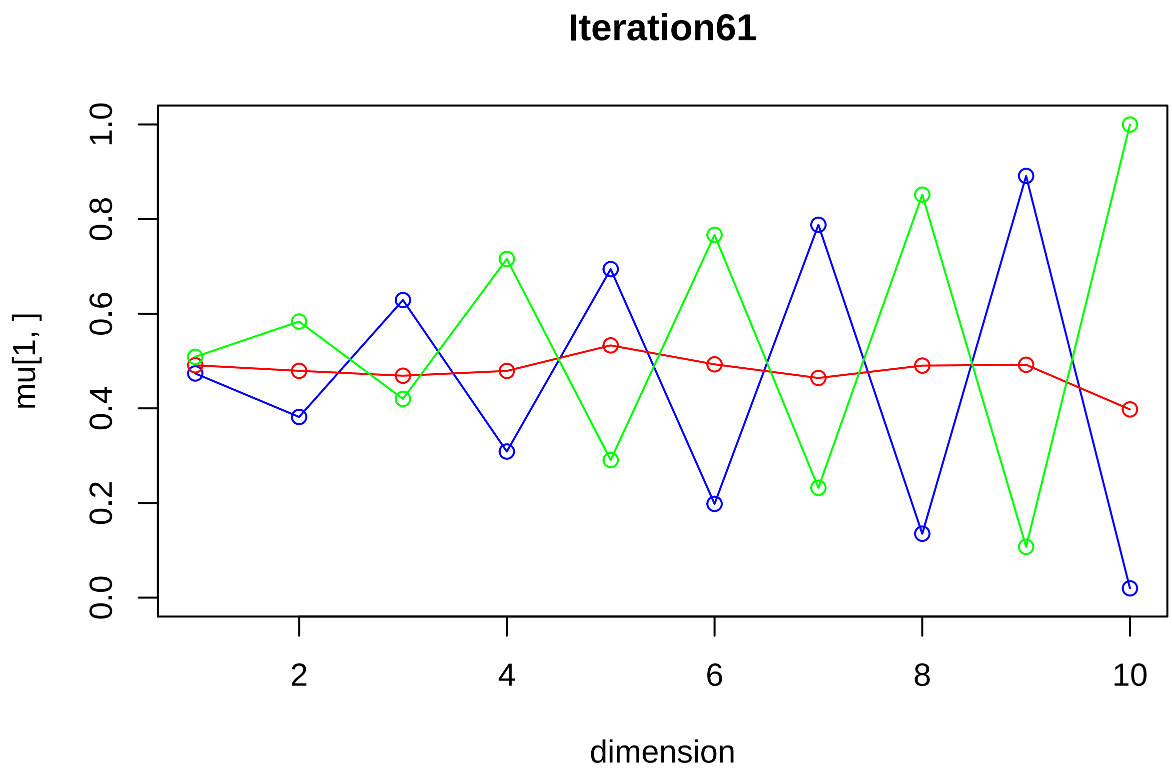

```
## iteration:  58 log likelihood:  -6743.736
```



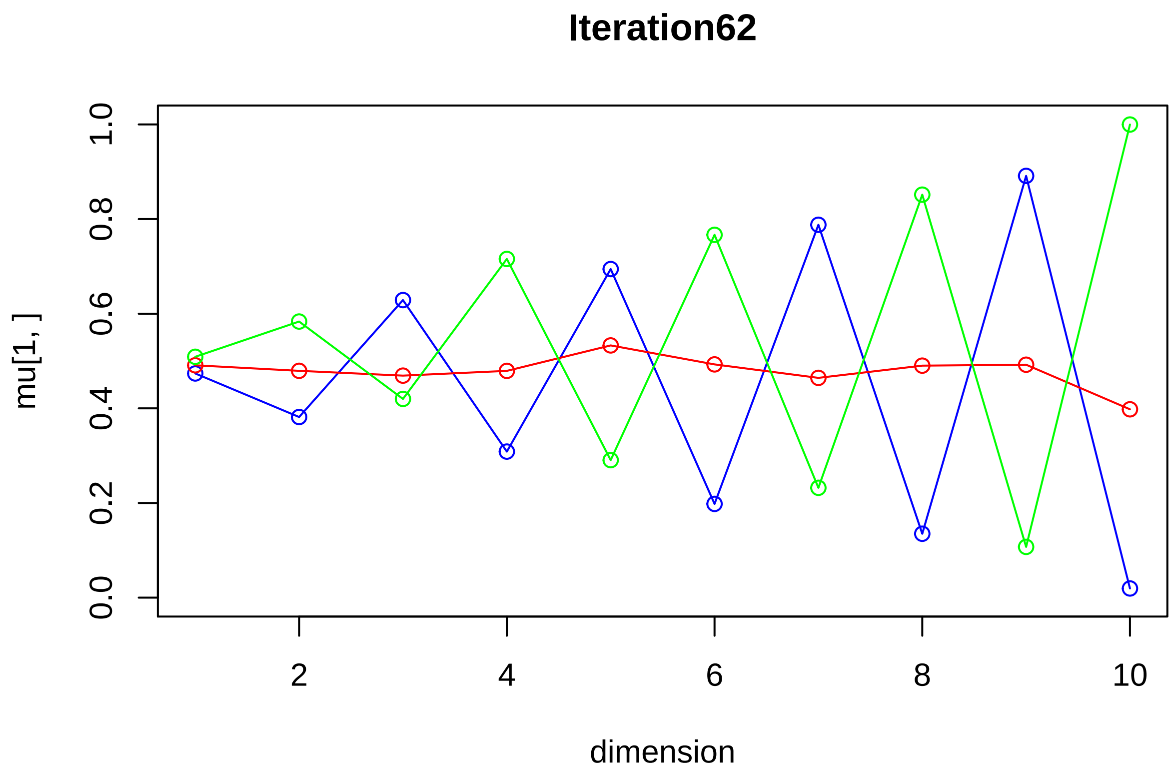

```
## iteration:  59 log likelihood:  -6743.628
```



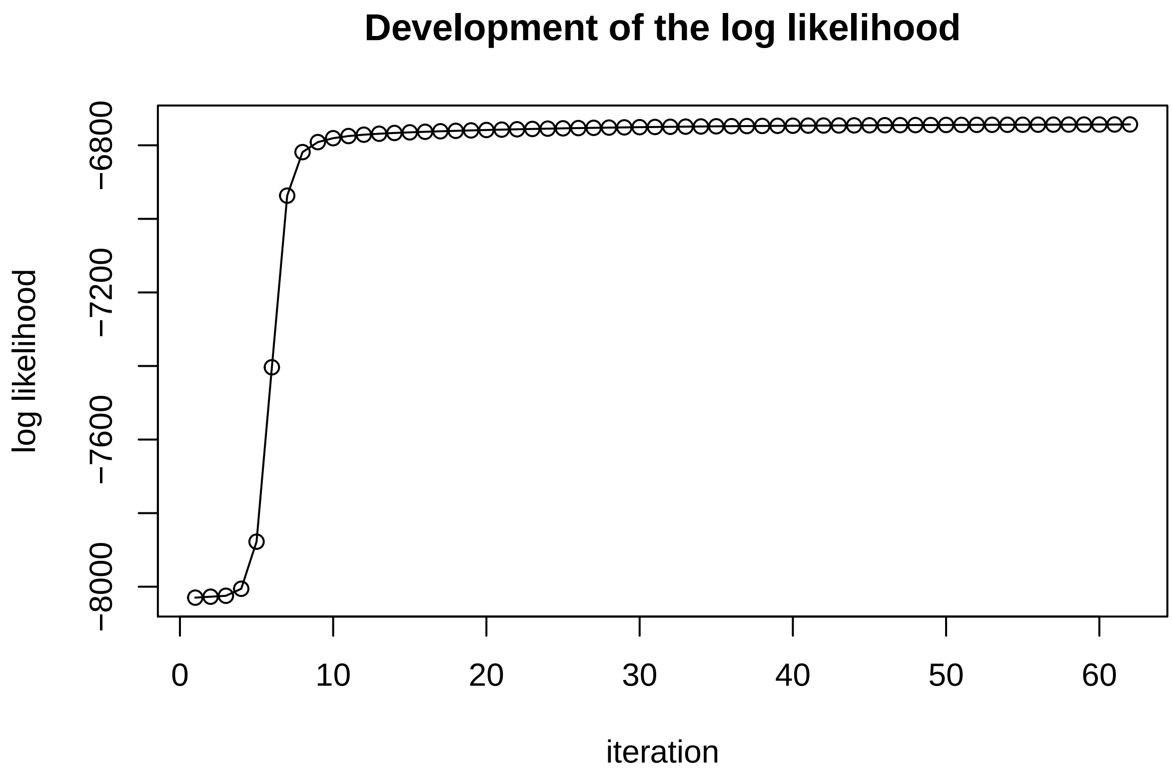

```
## iteration:  60 log likelihood:  -6743.524
```




```
## iteration:  61 log likelihood:  -6743.423
```



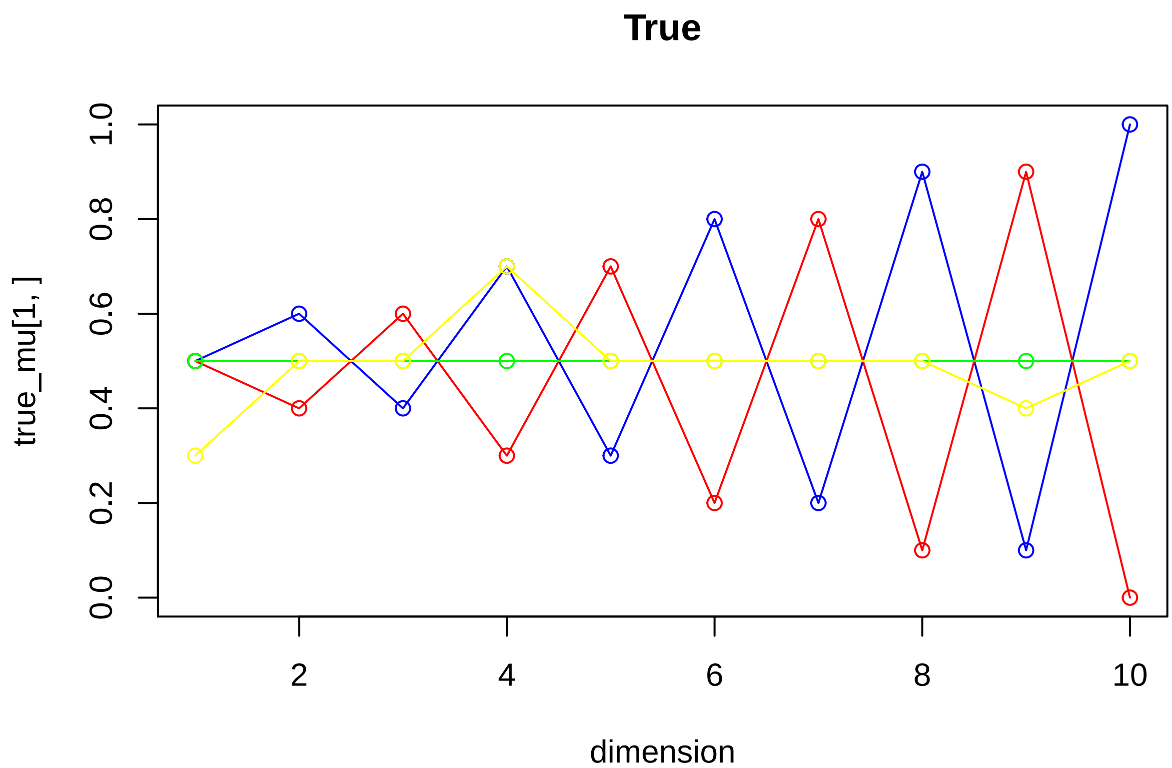

```
## iteration:  62 log likelihood:  -6743.326
```

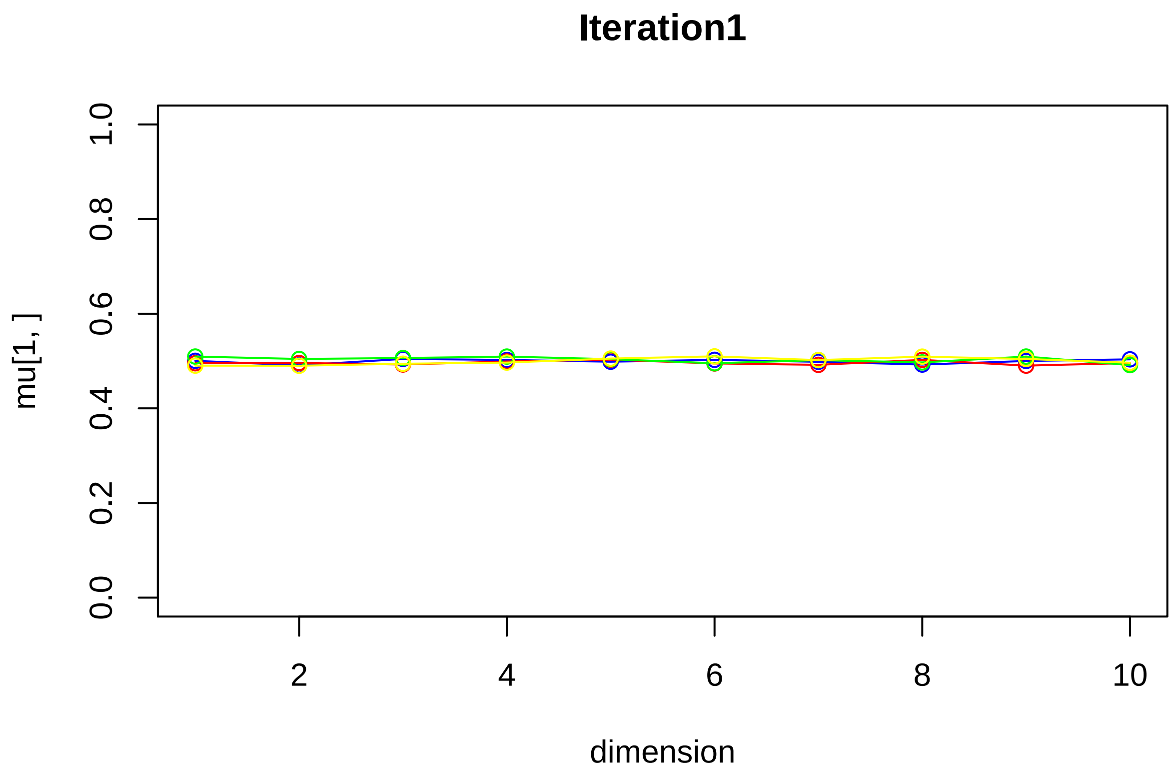



```
## $pi
## [1] 0.3259592 0.3044579 0.3695828
##
## $mu
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4737193 0.3817120 0.6288021 0.3086143 0.6943731 0.1980896 0.7879447
## [2,] 0.4909874 0.4793213 0.4691560 0.4791793 0.5329895 0.4928830 0.4643990
## [3,] 0.5089571 0.5834802 0.4199272 0.7157107 0.2905703 0.7667258 0.2320784
##           [,8]      [,9]      [,10]
## [1,] 0.1349651 0.8912534 0.01937869
## [2,] 0.4902682 0.4922194 0.39798407
## [3,] 0.8516111 0.1072226 0.99981353
##
## $logLikelihoodDevelopment
## NULL
```

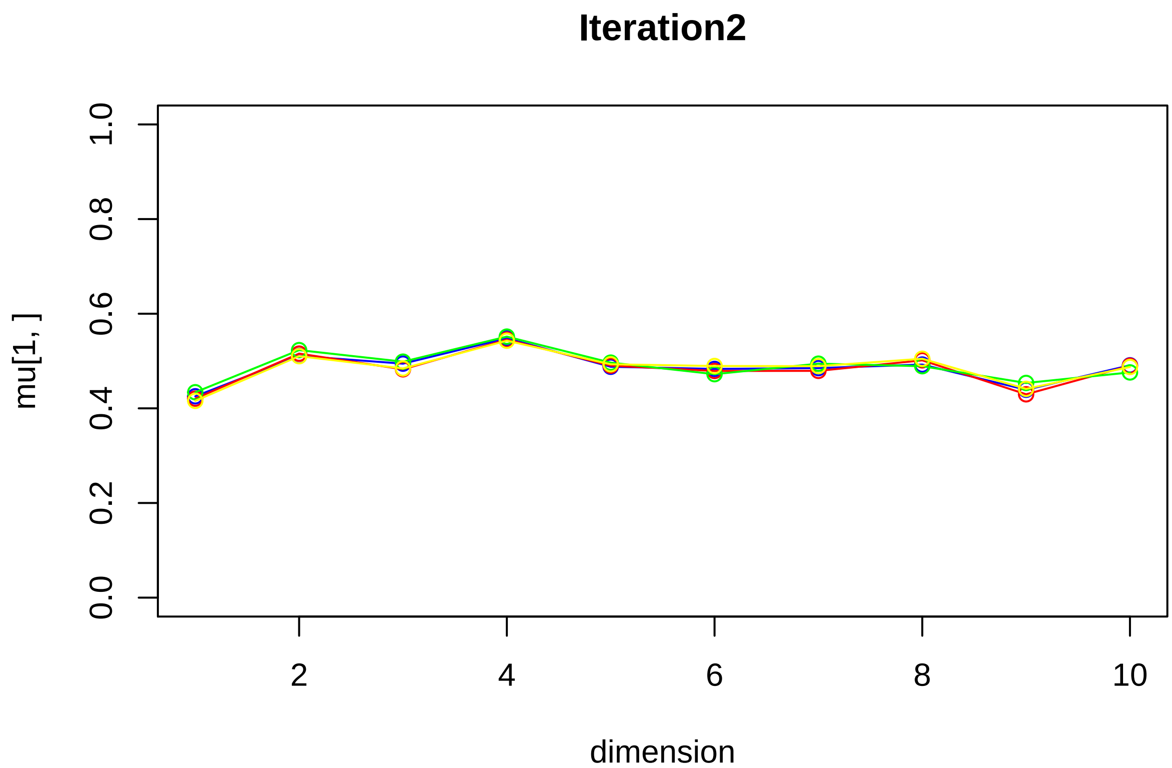
4. K=4

```
em_loop(4)
```

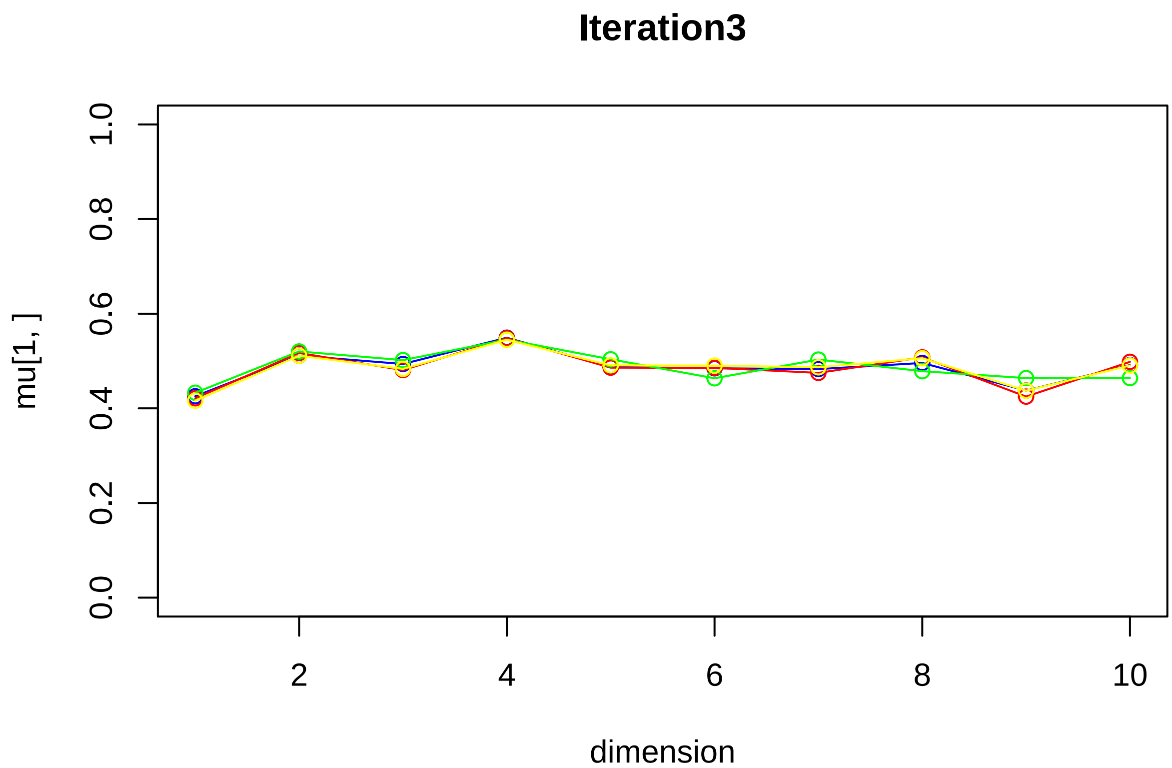

True
true_mu[1,]
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
dimension



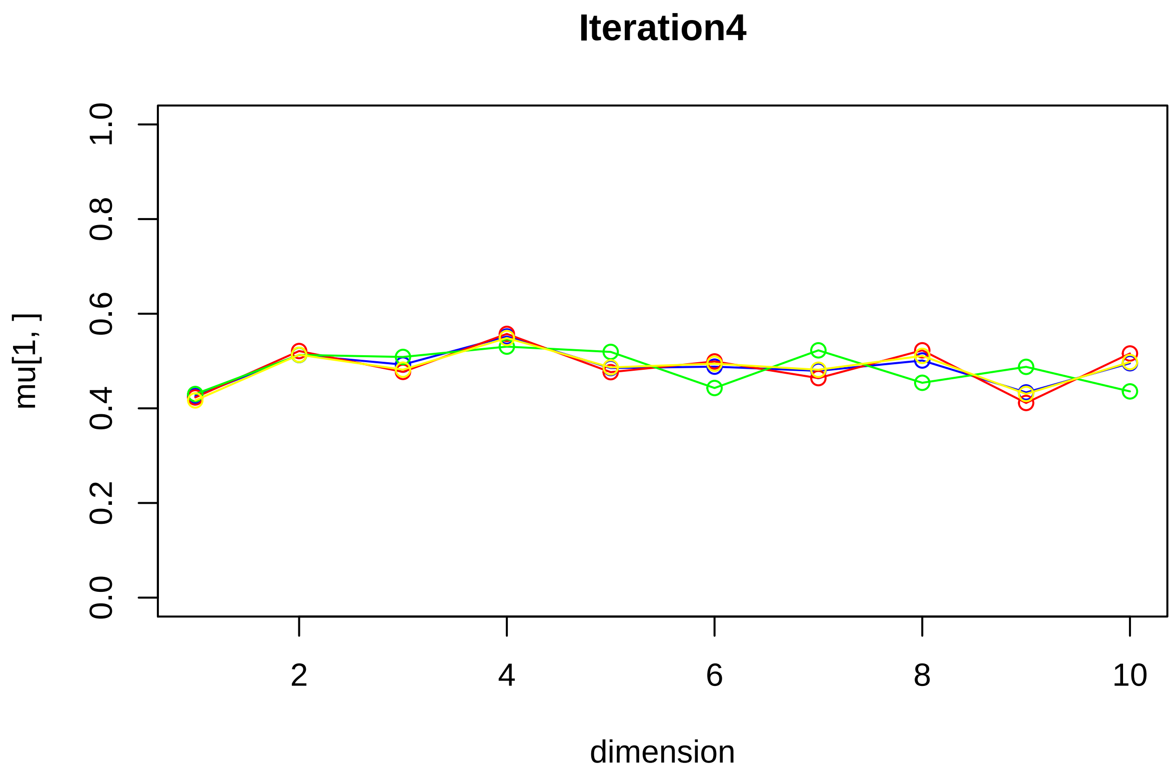

```
## iteration:  1 log likelihood:  -8316.904
```



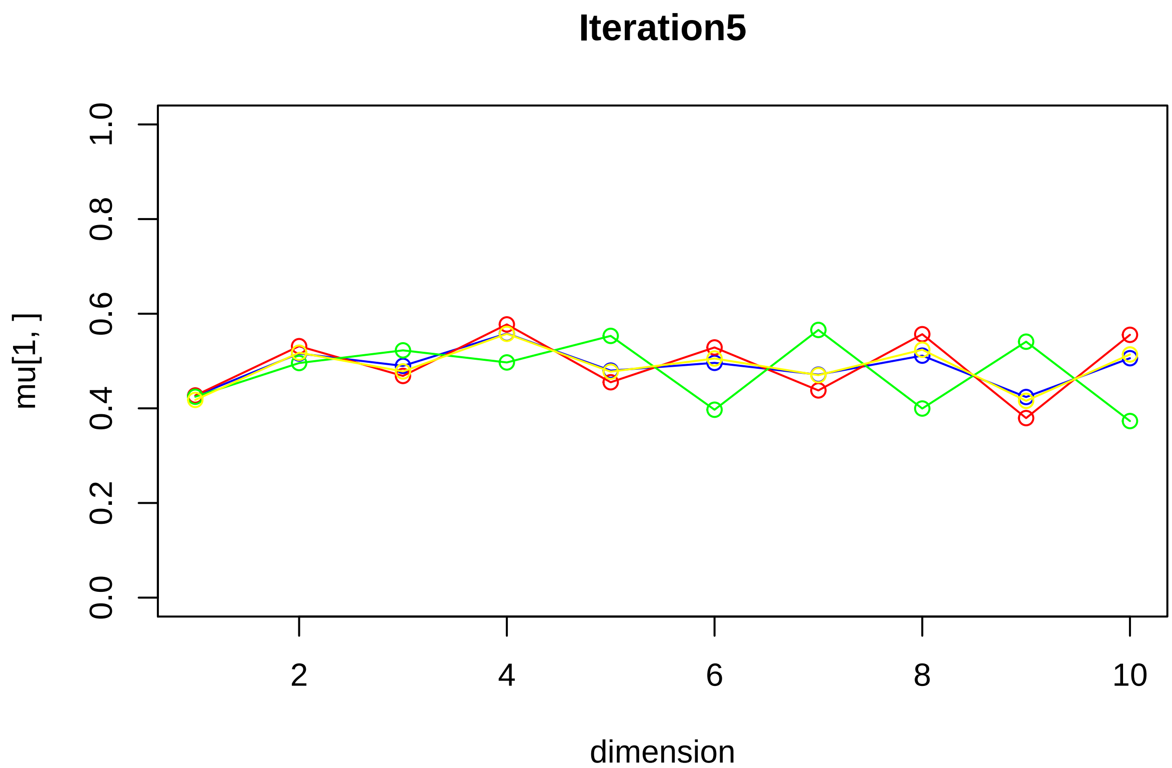

```
## iteration:  2 log likelihood:  -8291.114
```



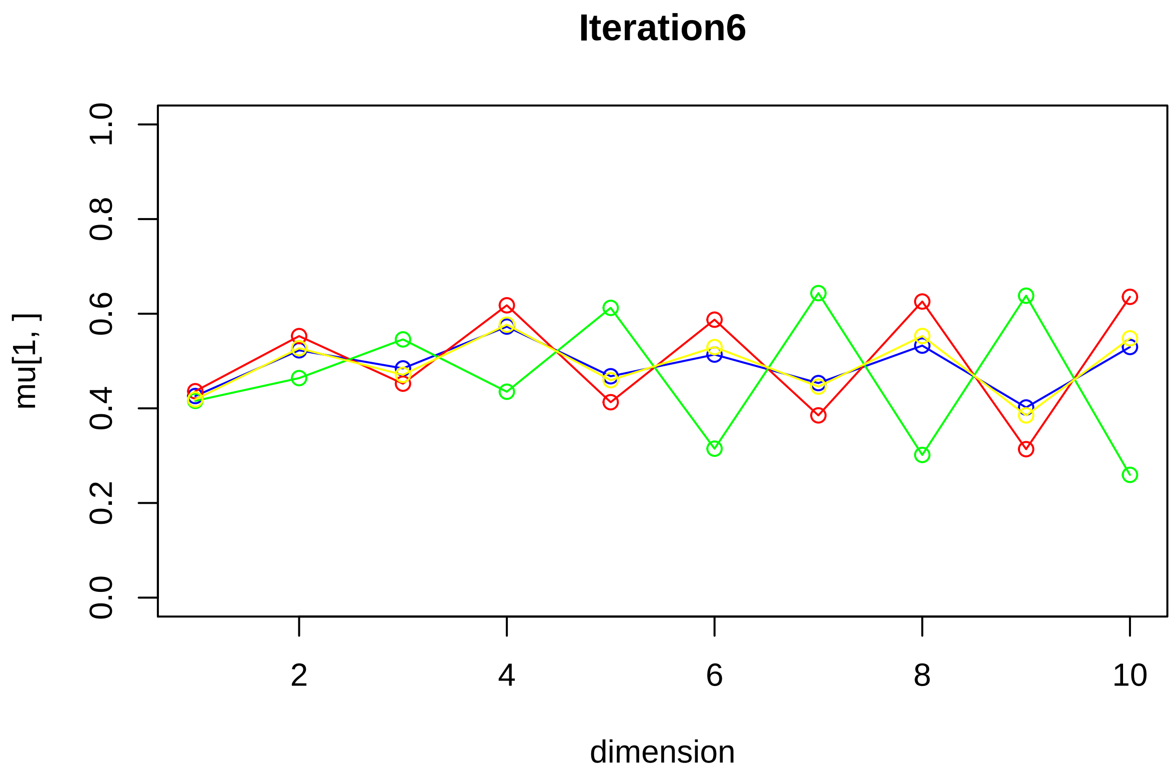

```
## iteration:  3 log likelihood:  -8286.966
```



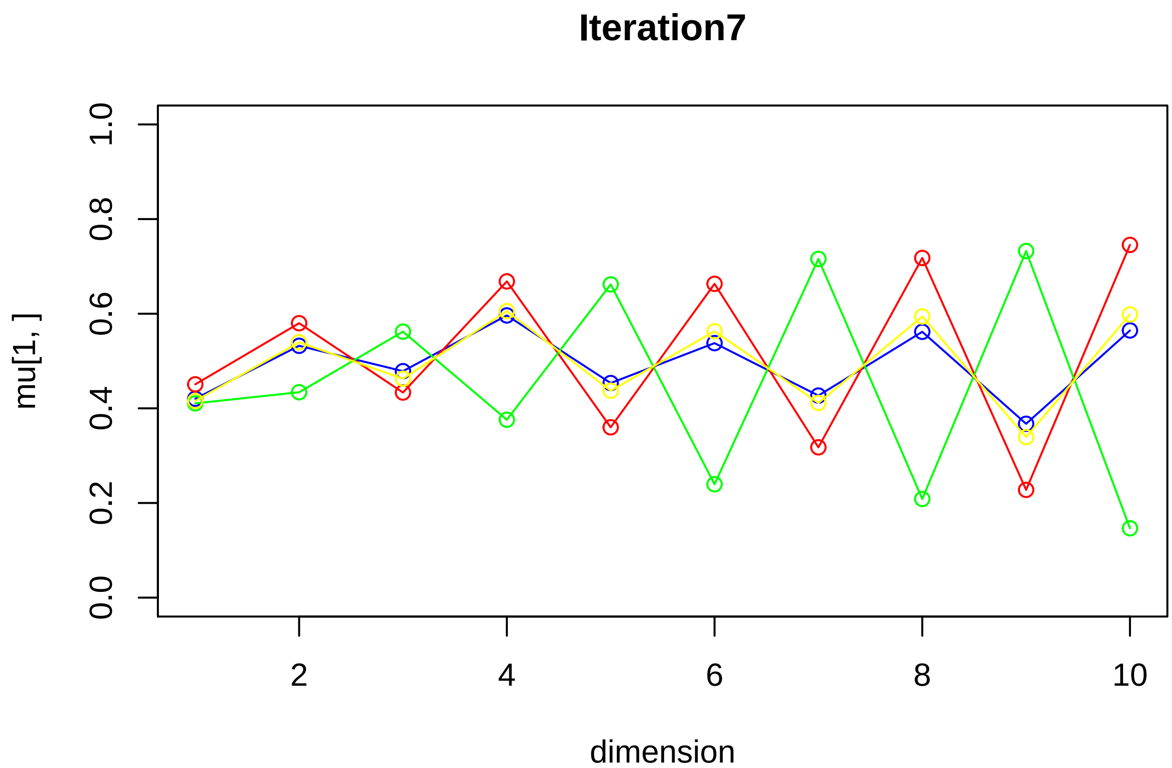

```
## iteration:  4 log likelihood:  -8264.806
```



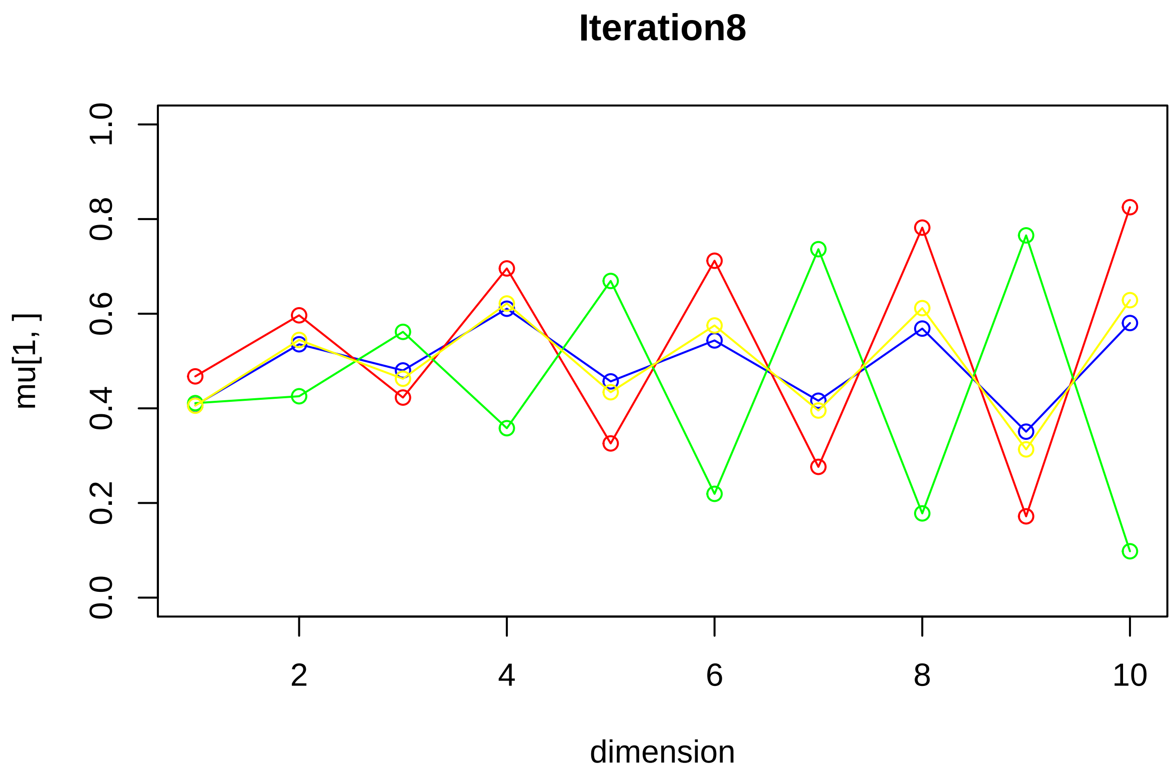

```
## iteration:  5 log likelihood:  -8161.19
```



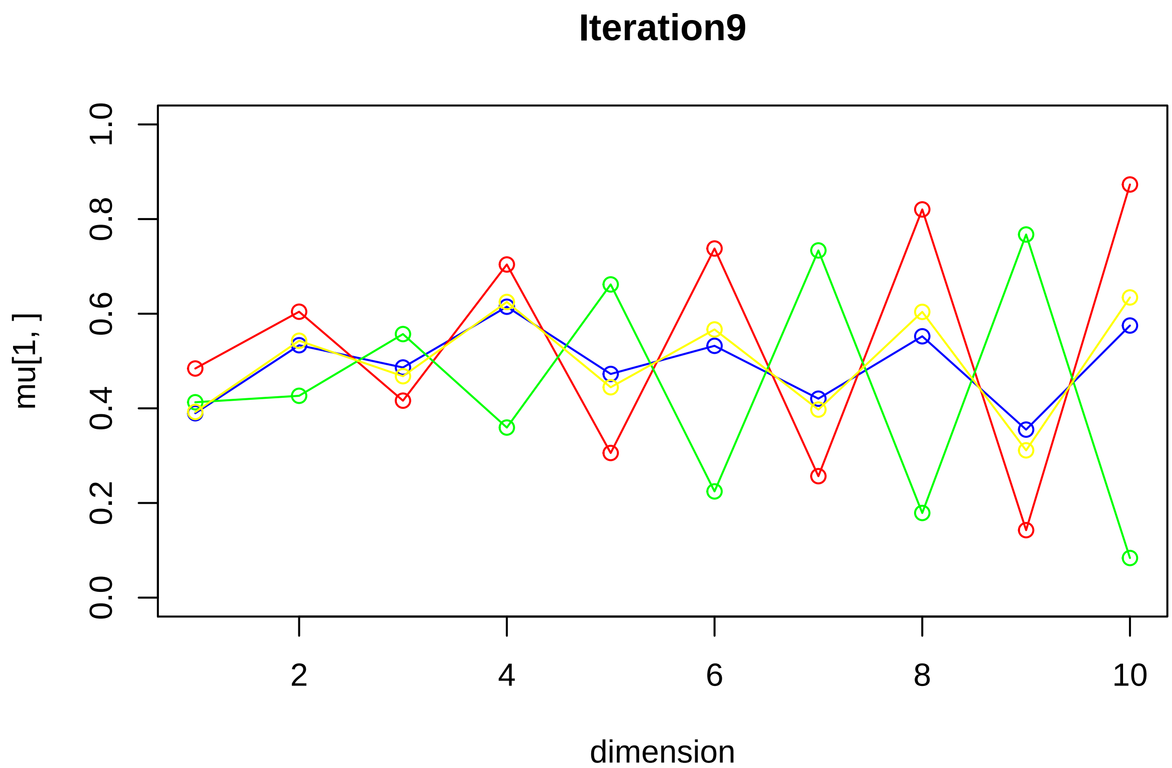

```
## iteration:  6 log likelihood:  -7868.89
```



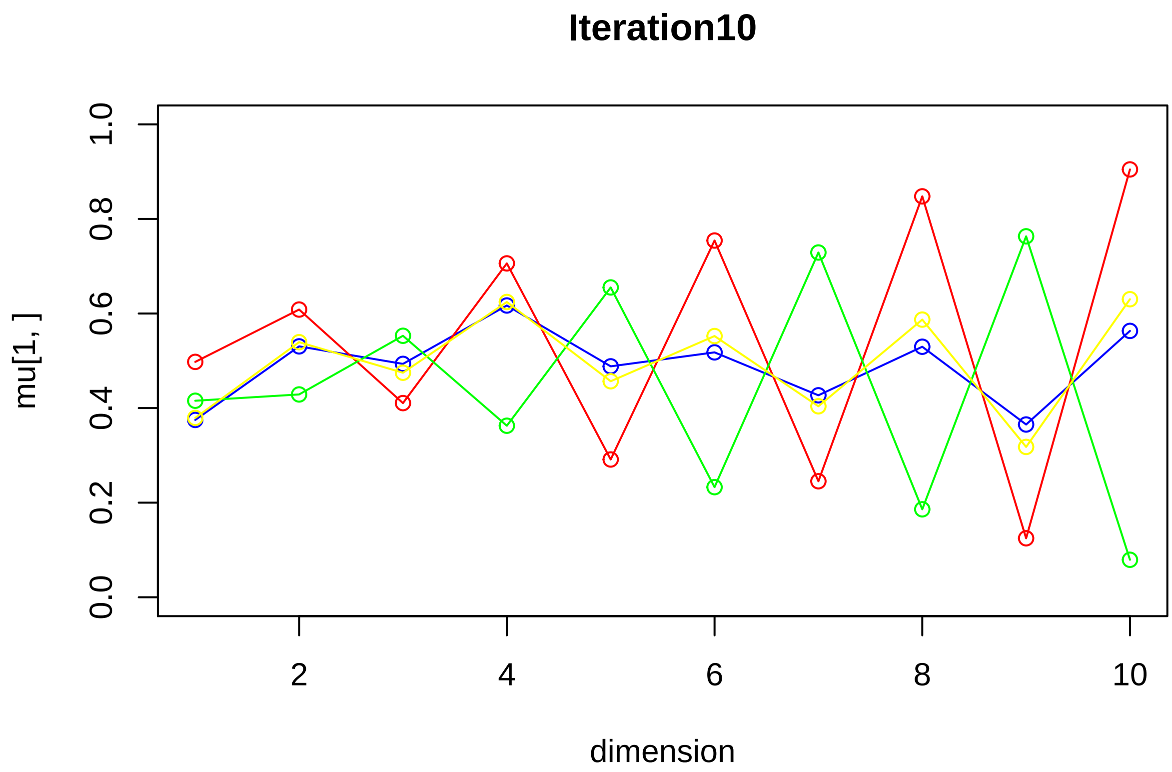

```
## iteration:  7 log likelihood:  -7570.873
```



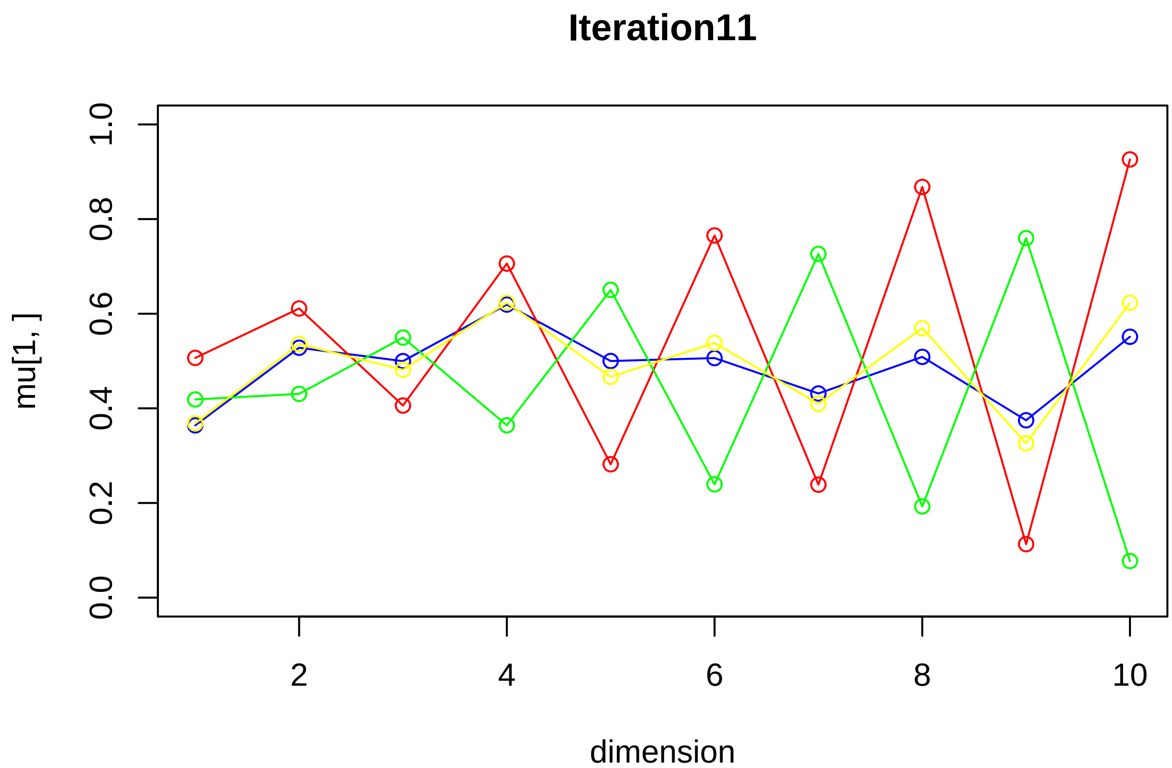

```
## iteration:  8 log likelihood:  -7445.719
```



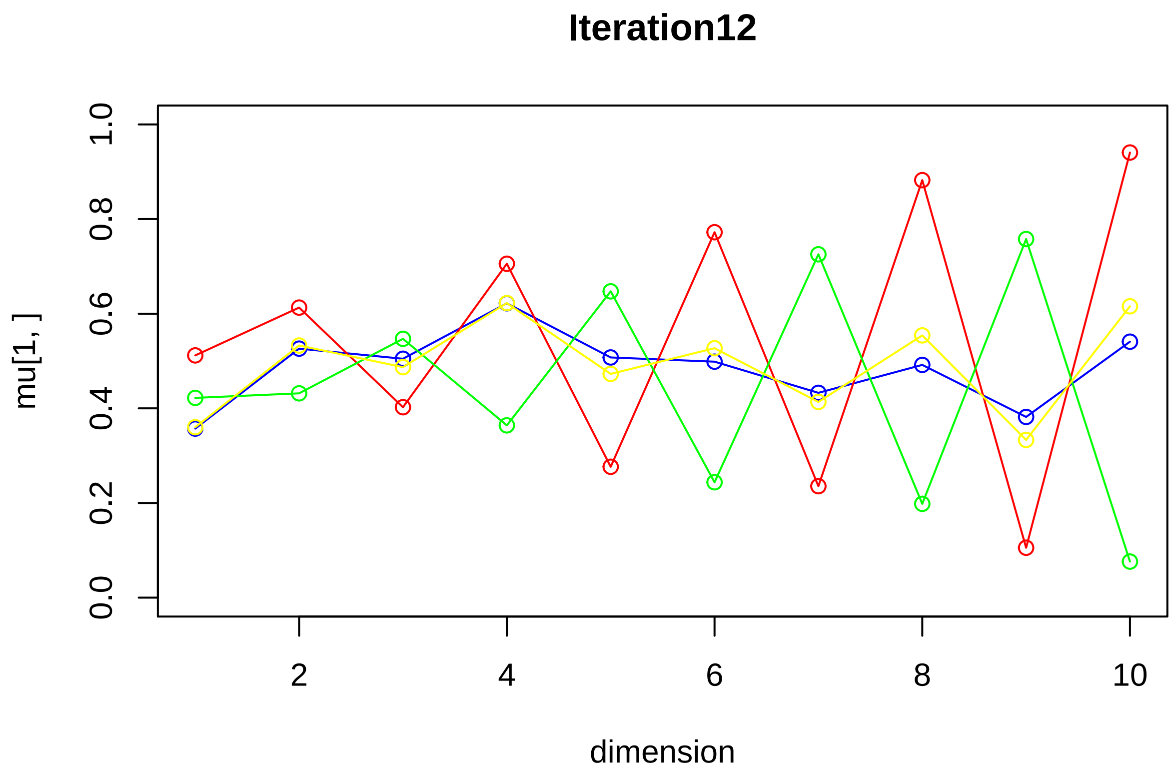

```
## iteration:  9 log likelihood:  -7389.741
```



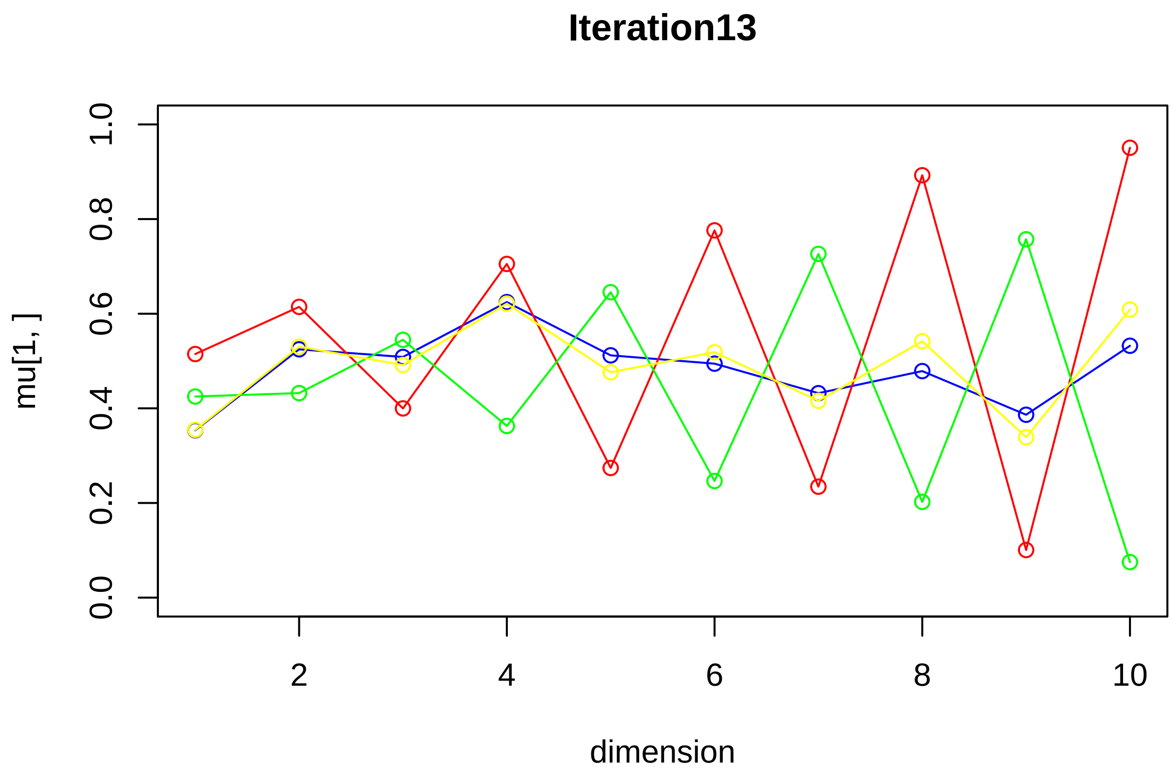

```
## iteration:  10 log likelihood:  -7356.803
```



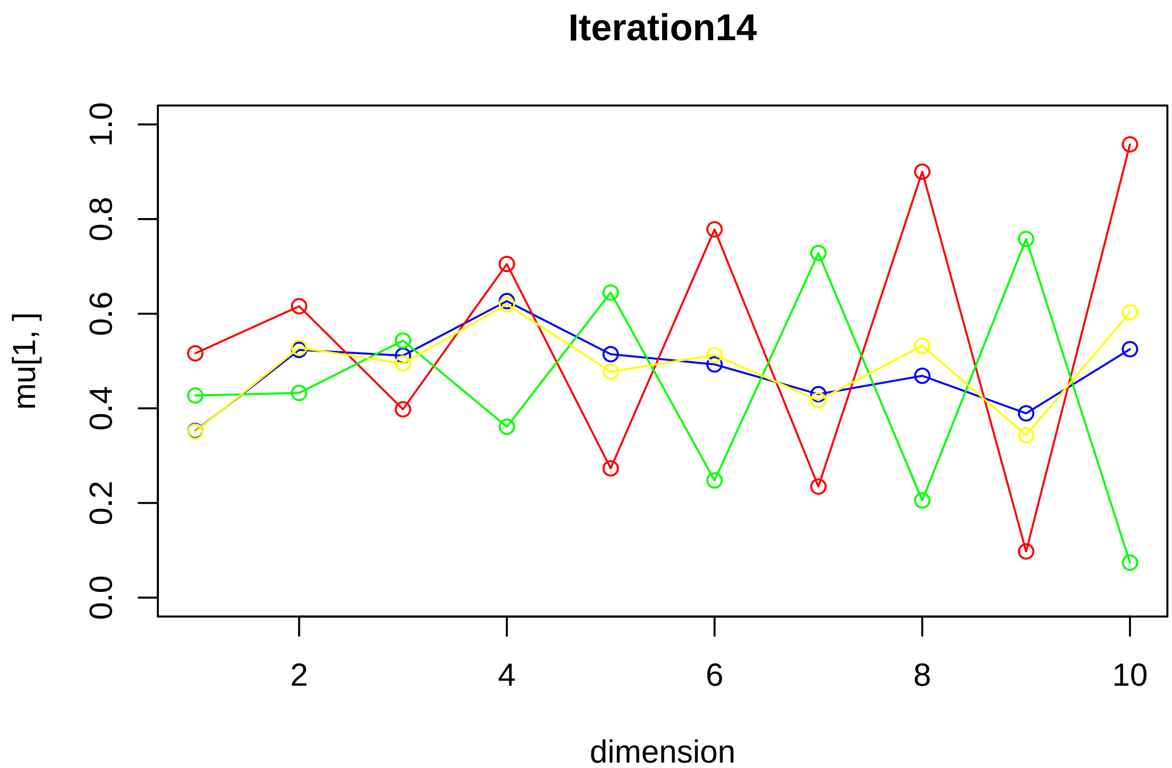

```
## iteration:  11 log likelihood:  -7337.208
```



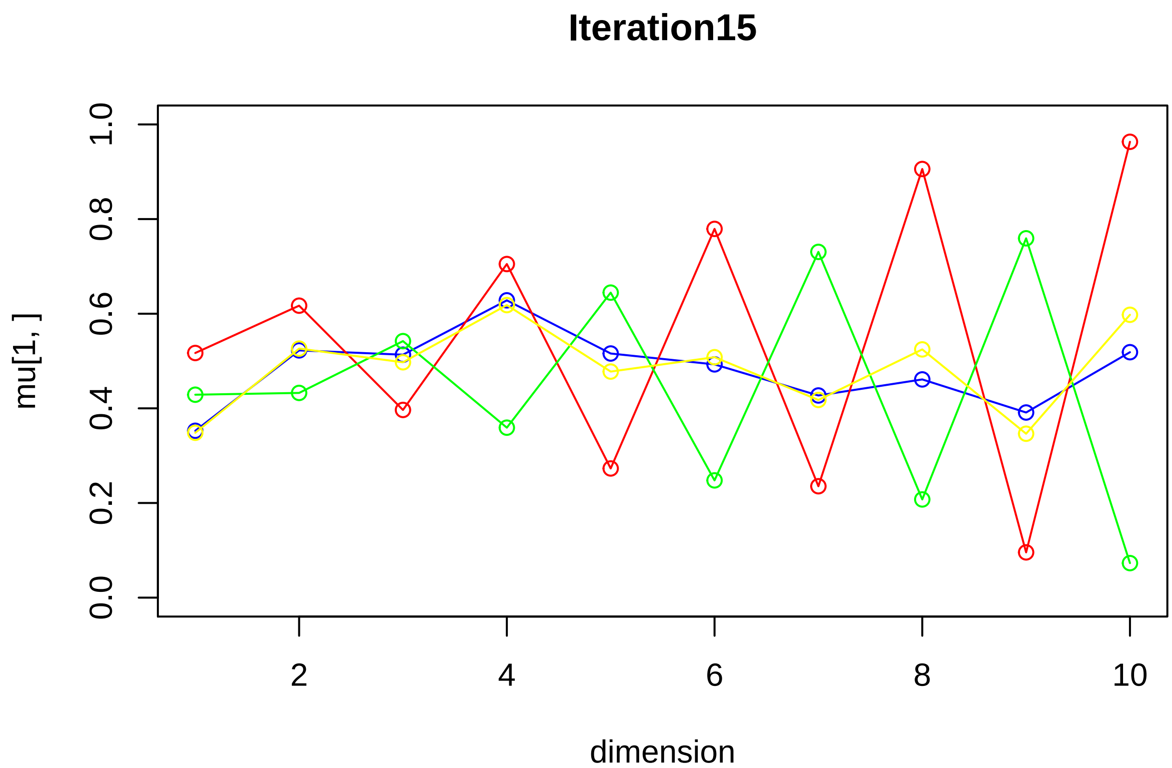

```
## iteration:  12 log likelihood:  -7326.118
```



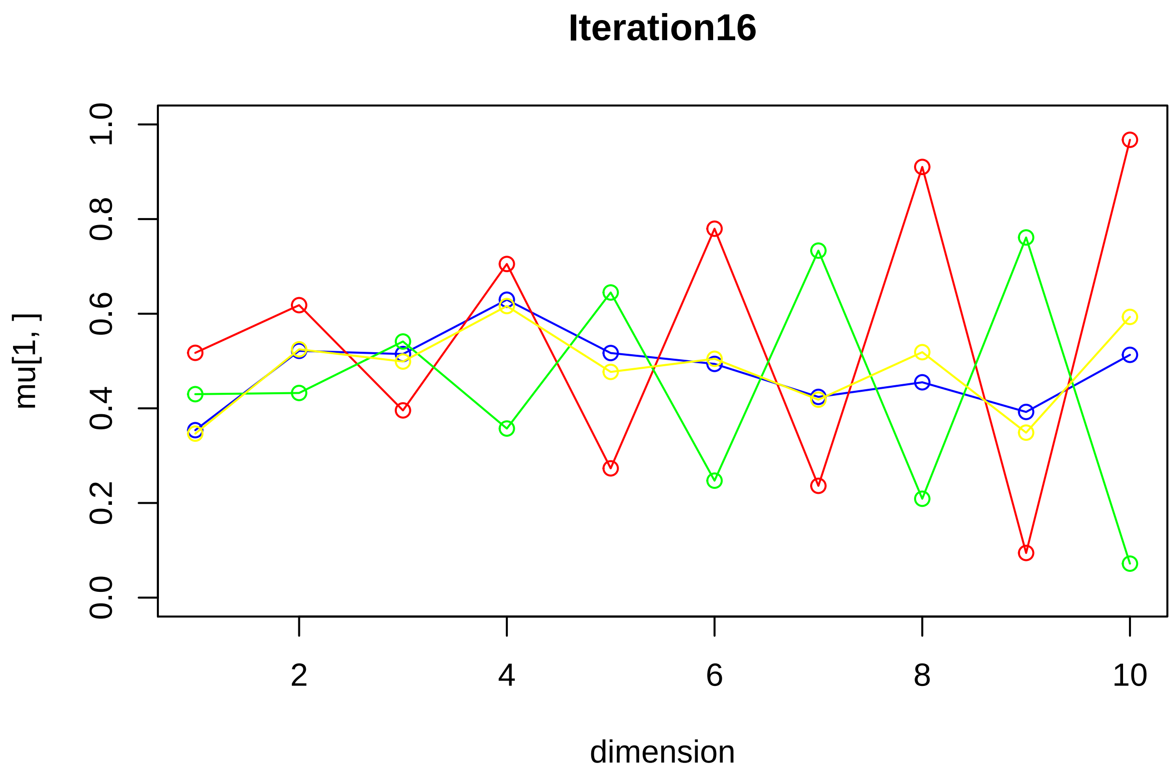

```
## iteration:  13 log likelihood:  -7319.998
```



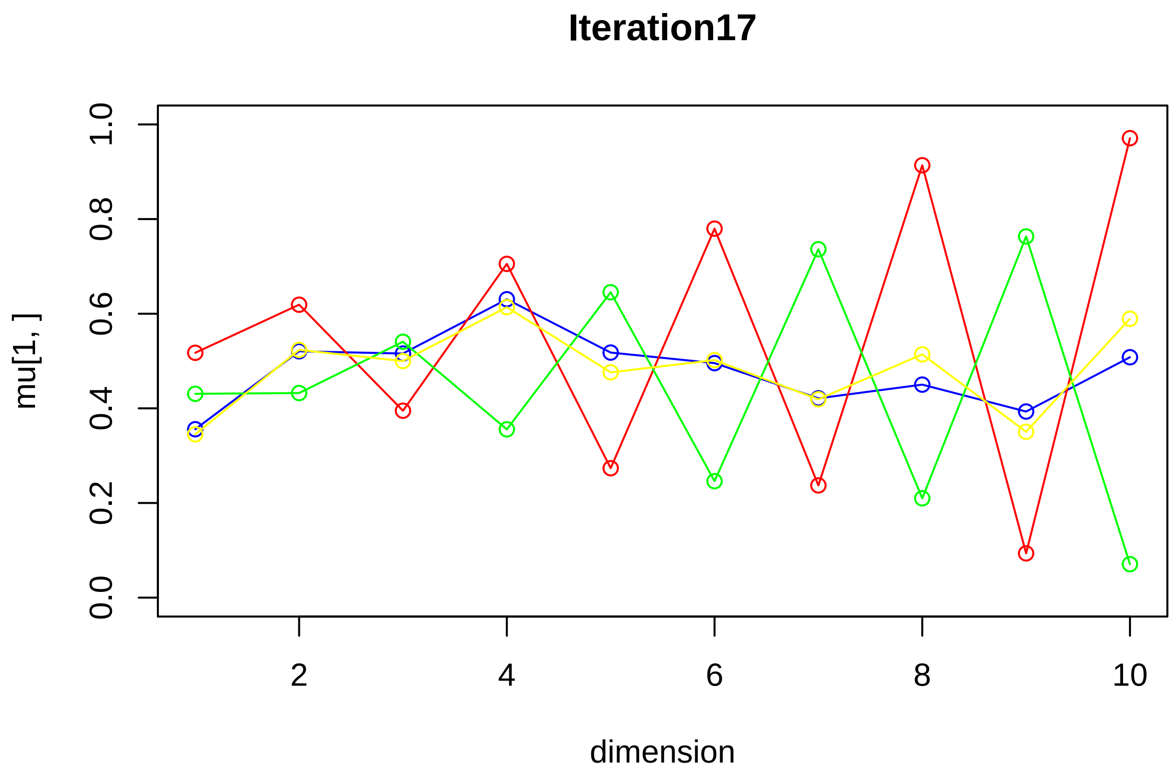

```
## iteration:  14 log likelihood:  -7316.6
```



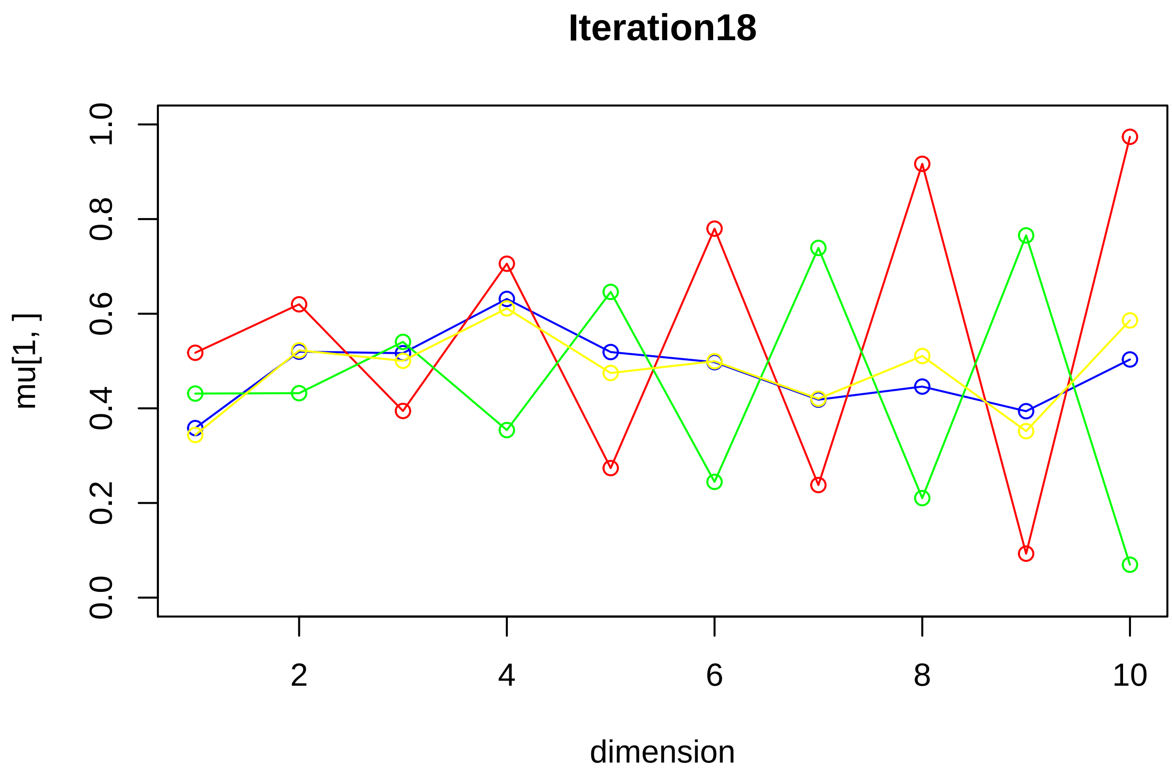

```
## iteration:  15 log likelihood:  -7314.666
```



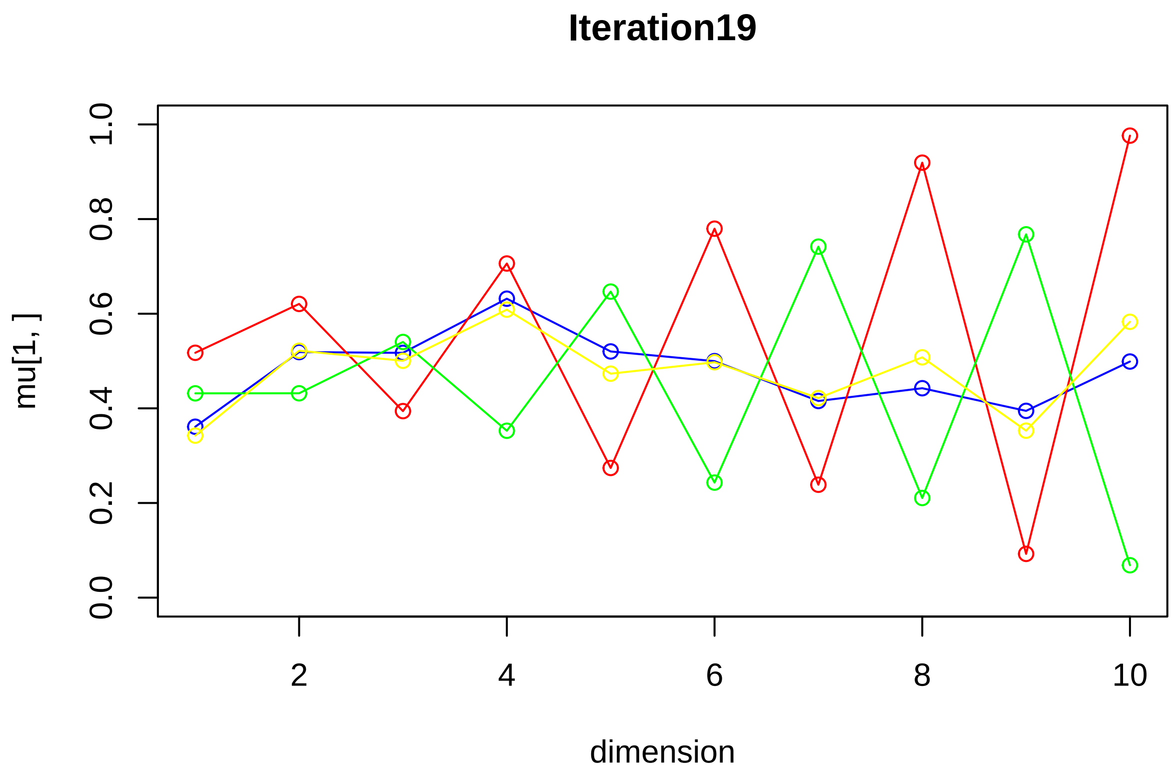

```
## iteration:  16 log likelihood:  -7313.528
```



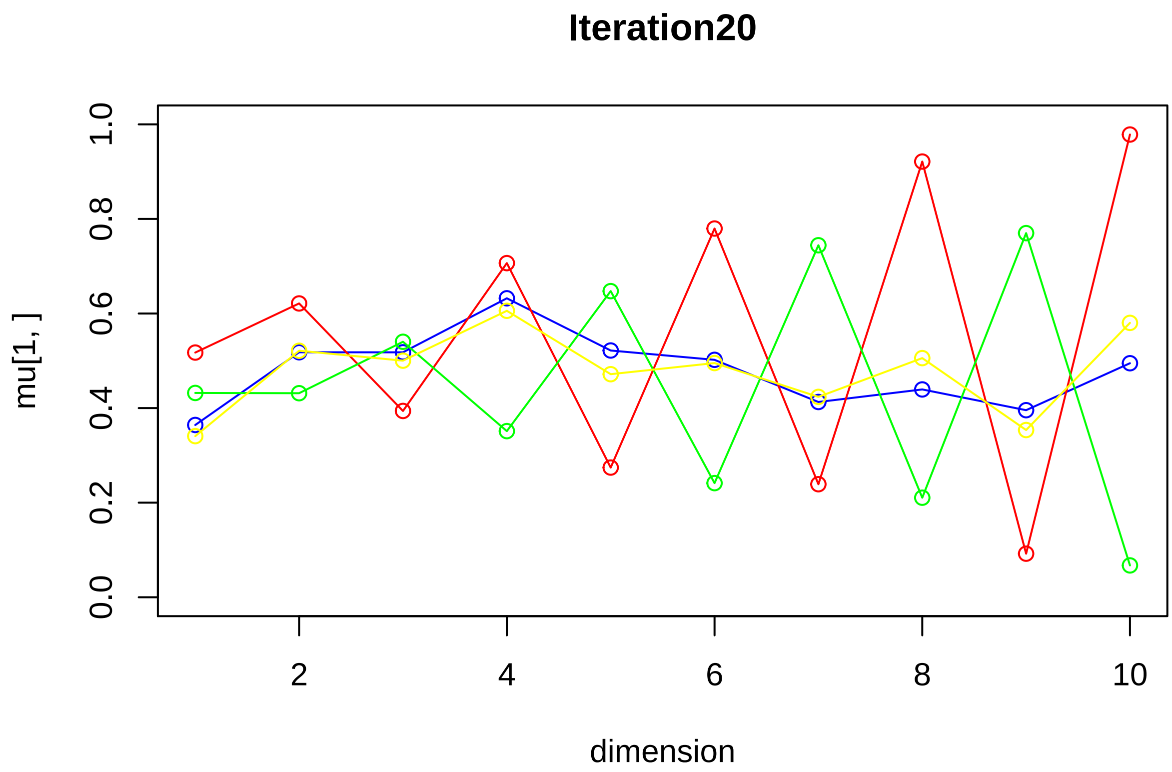

```
## iteration:  17 log likelihood:  -7312.829
```



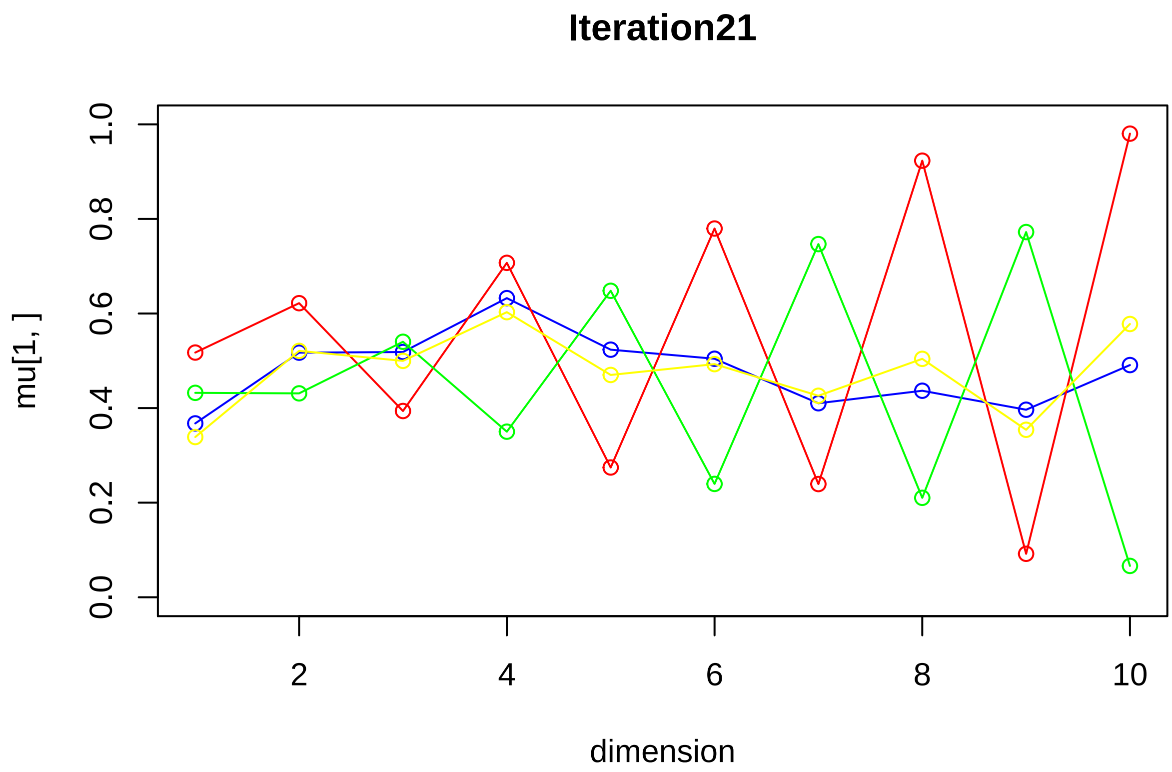

```
## iteration:  18 log likelihood:  -7312.367
```



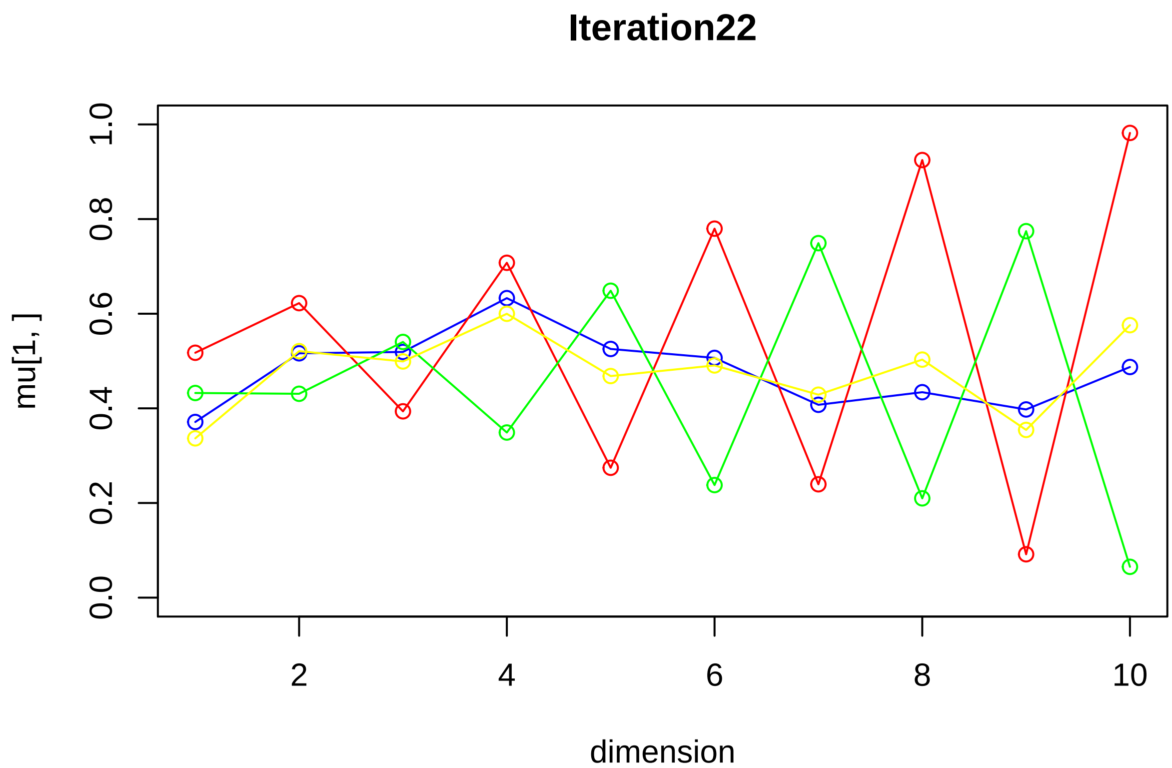

```
## iteration:  19 log likelihood:  -7312.024
```



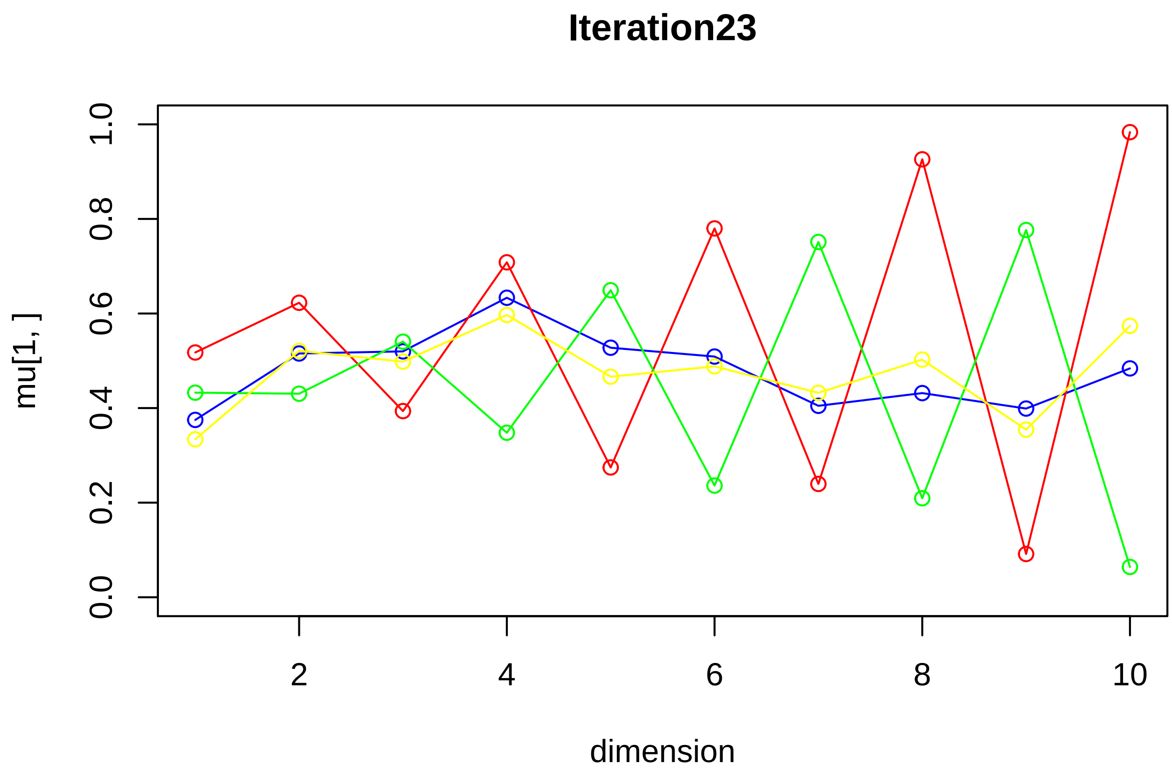

```
## iteration:  20 log likelihood:  -7311.723
```



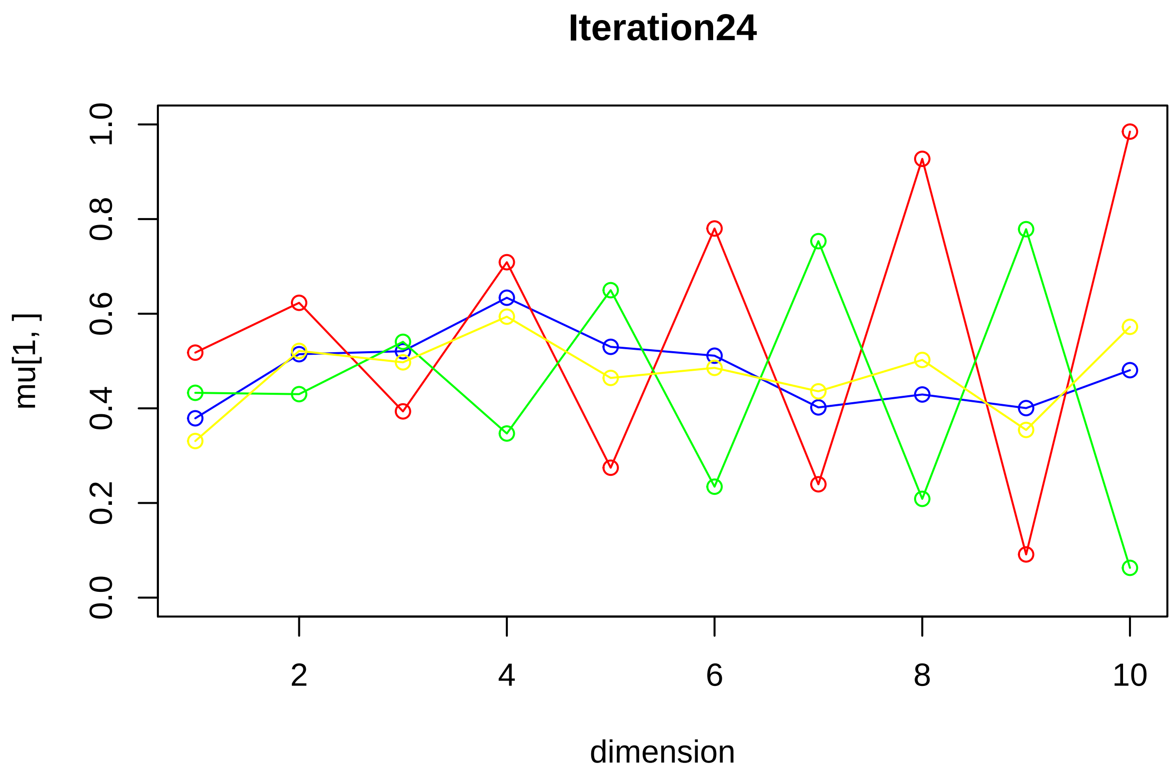

```
## iteration:  21 log likelihood:  -7311.407
```



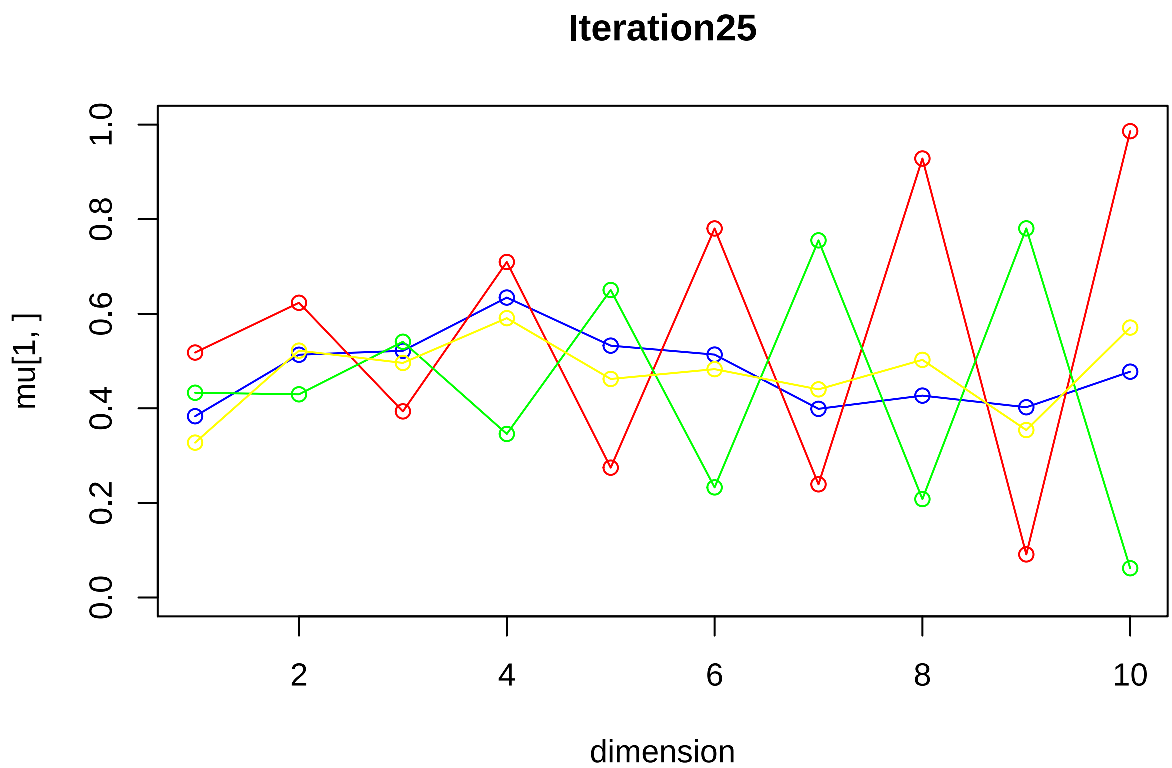

```
## iteration:  22 log likelihood:  -7311.036
```



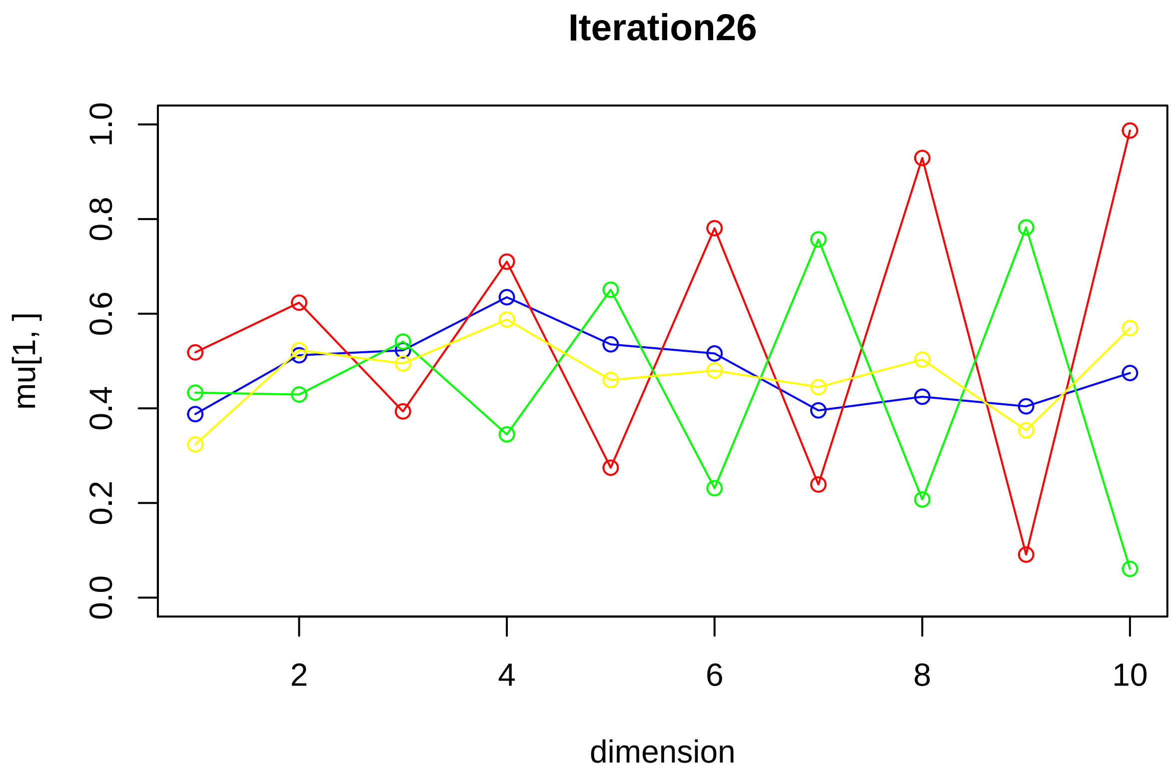

```
## iteration:  23 log likelihood:  -7310.574
```



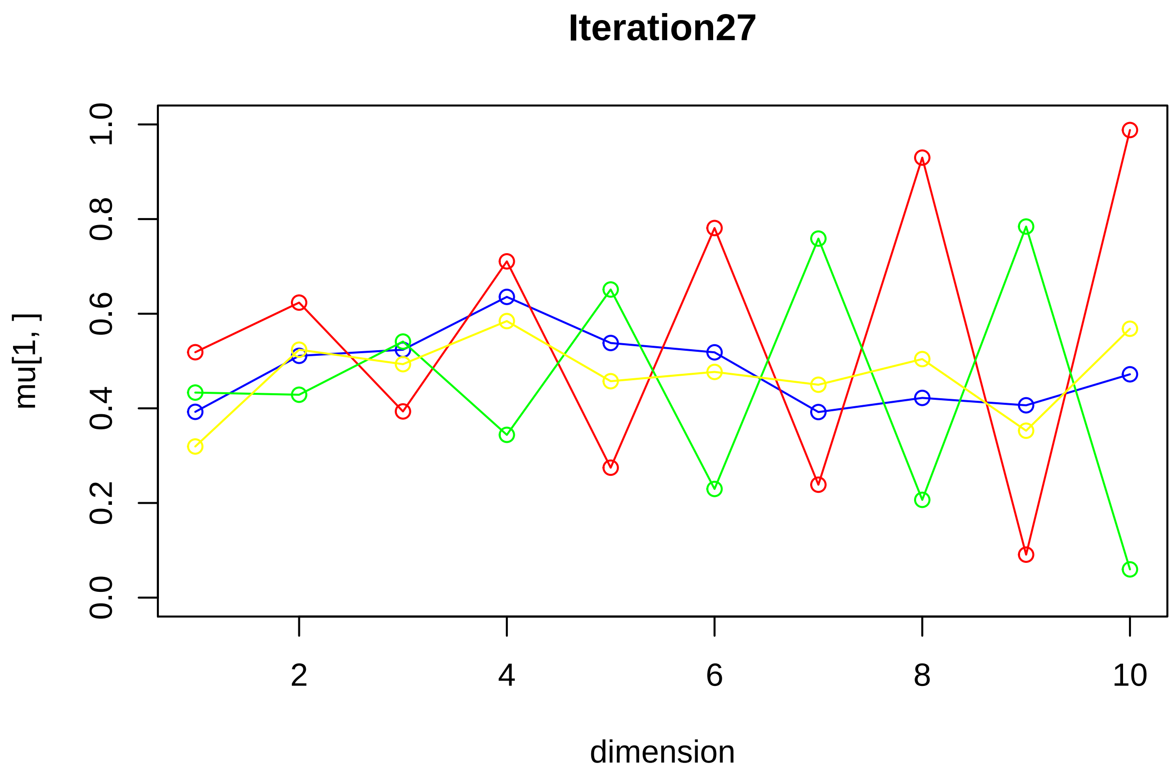

```
## iteration:  24 log likelihood:  -7309.988
```



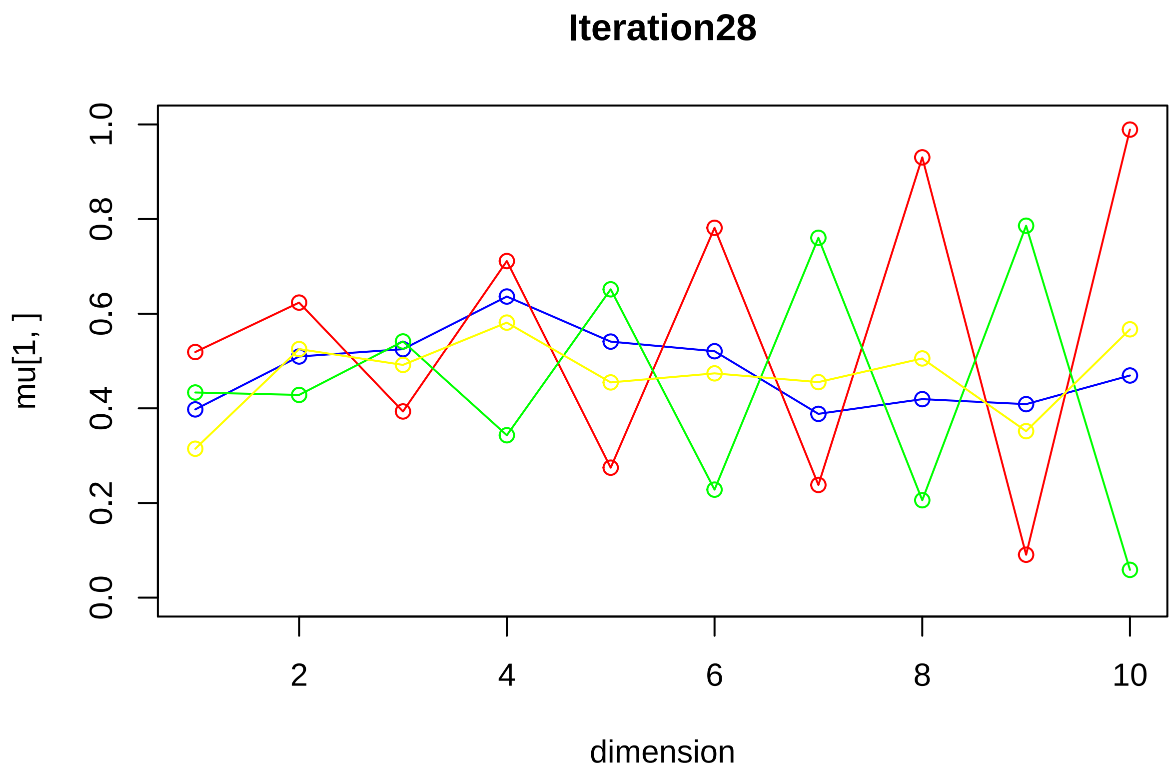

```
## iteration:  25 log likelihood:  -7309.248
```



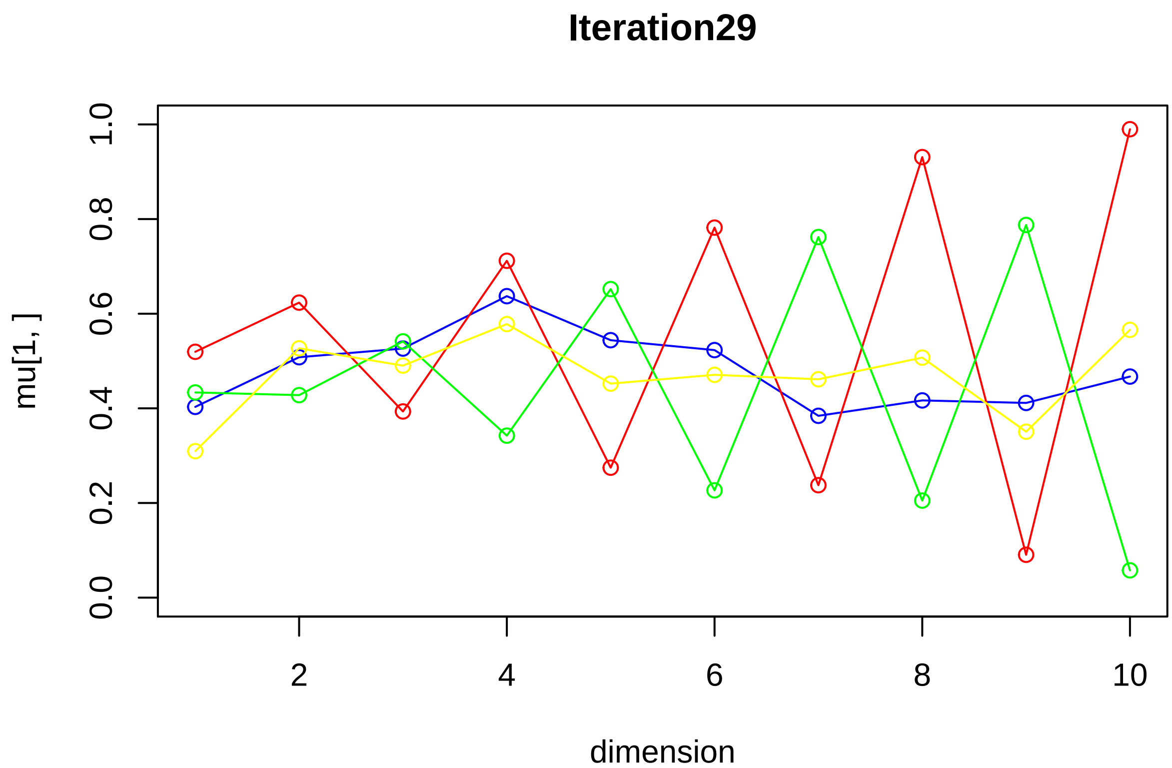

```
## iteration:  26 log likelihood:  -7308.322
```



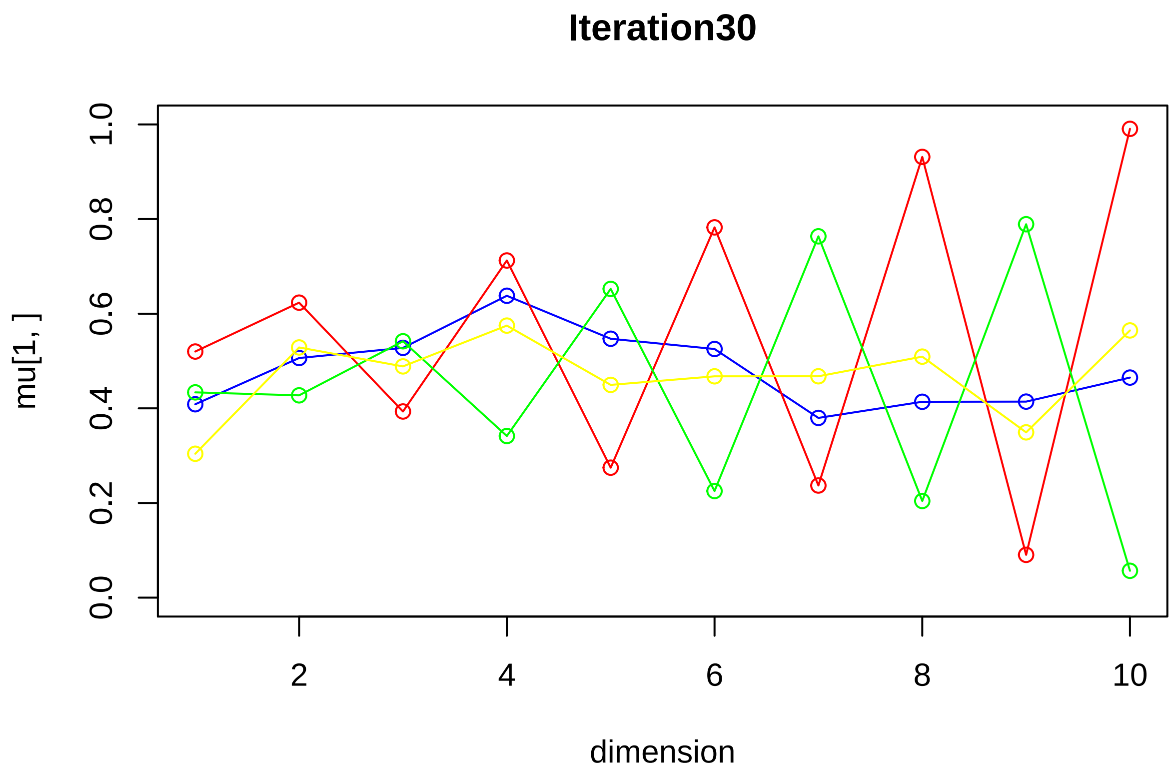

```
## iteration:  27 log likelihood:  -7307.185
```



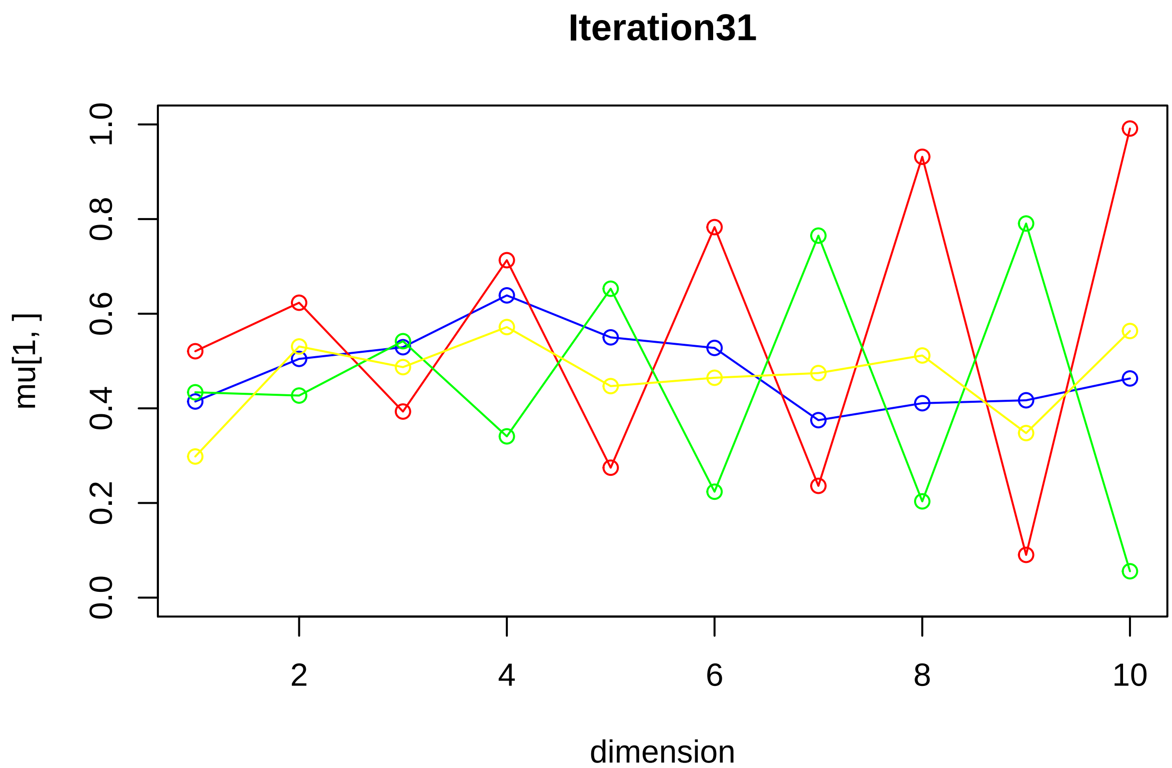

```
## iteration:  28 log likelihood:  -7305.809
```



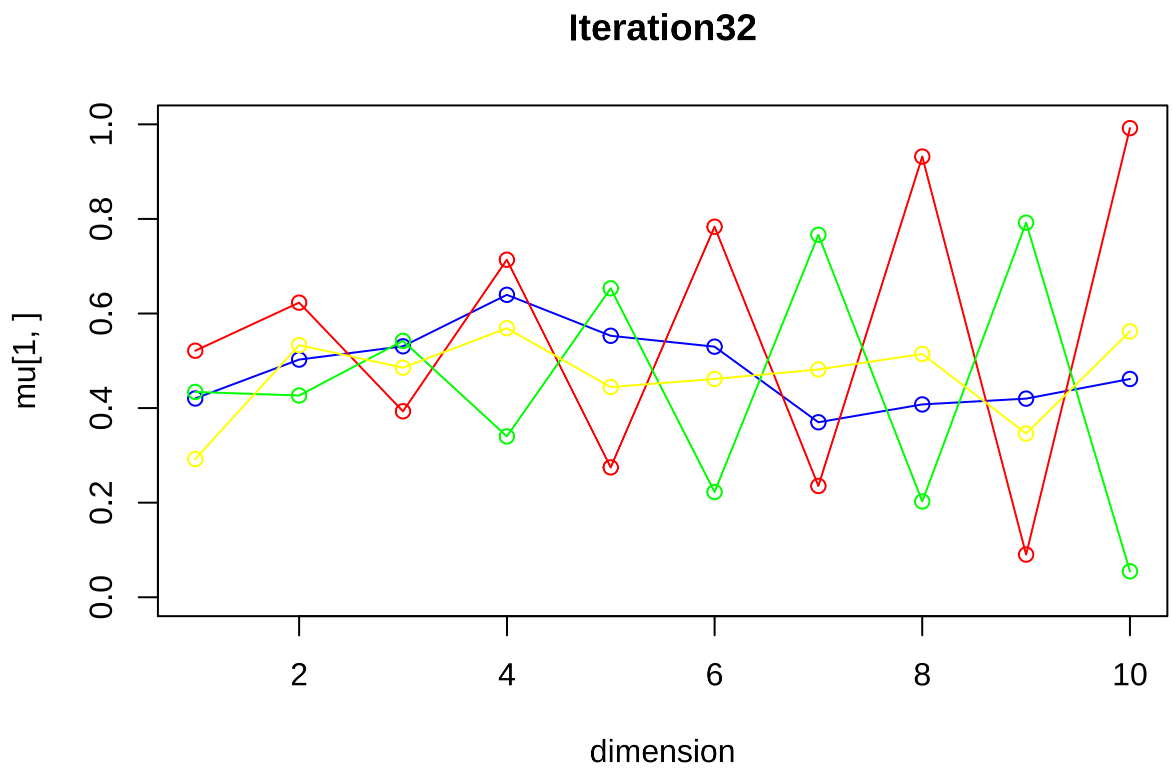

```
## iteration:  29 log likelihood:  -7304.176
```



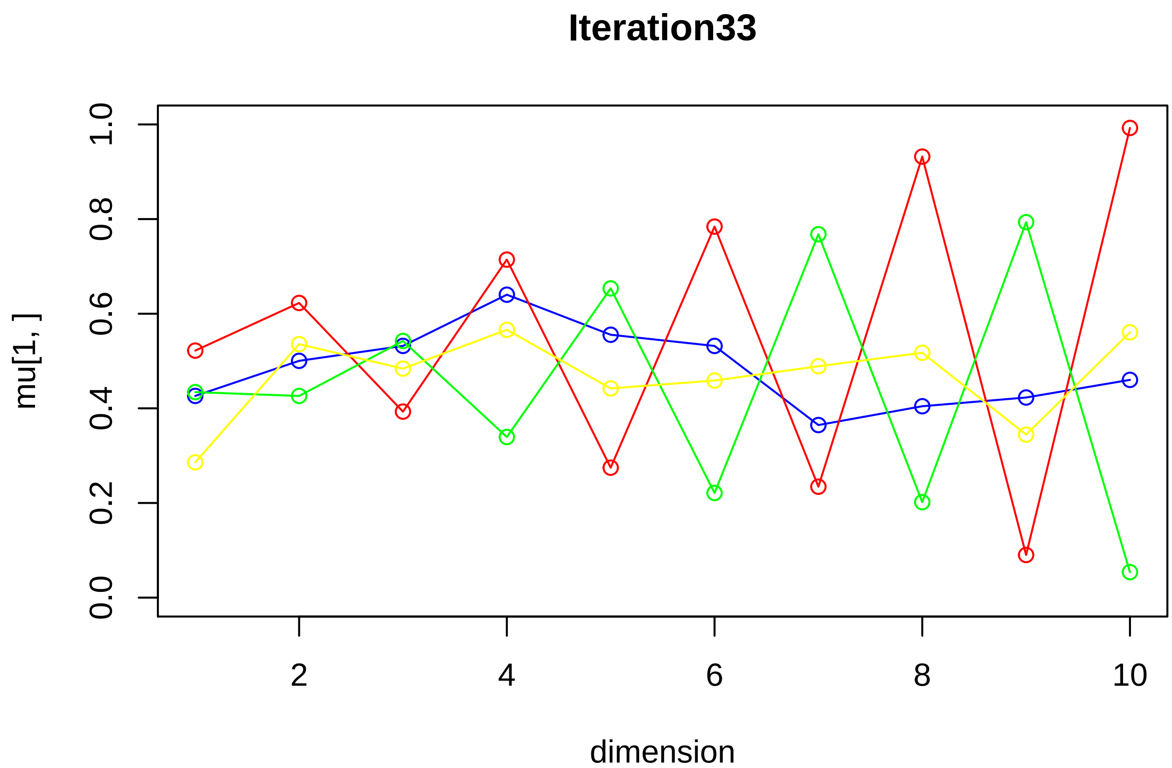

```
## iteration:  30 log likelihood:  -7302.273
```



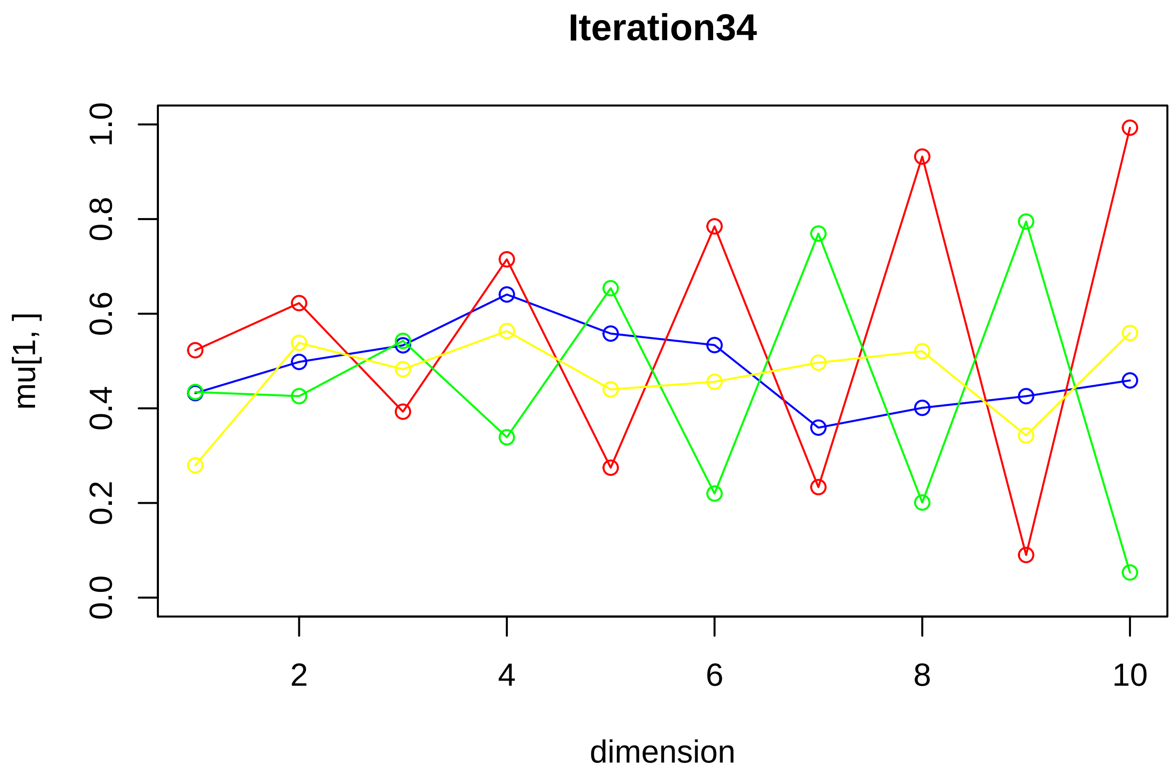

```
## iteration:  31 log likelihood:  -7300.1
```



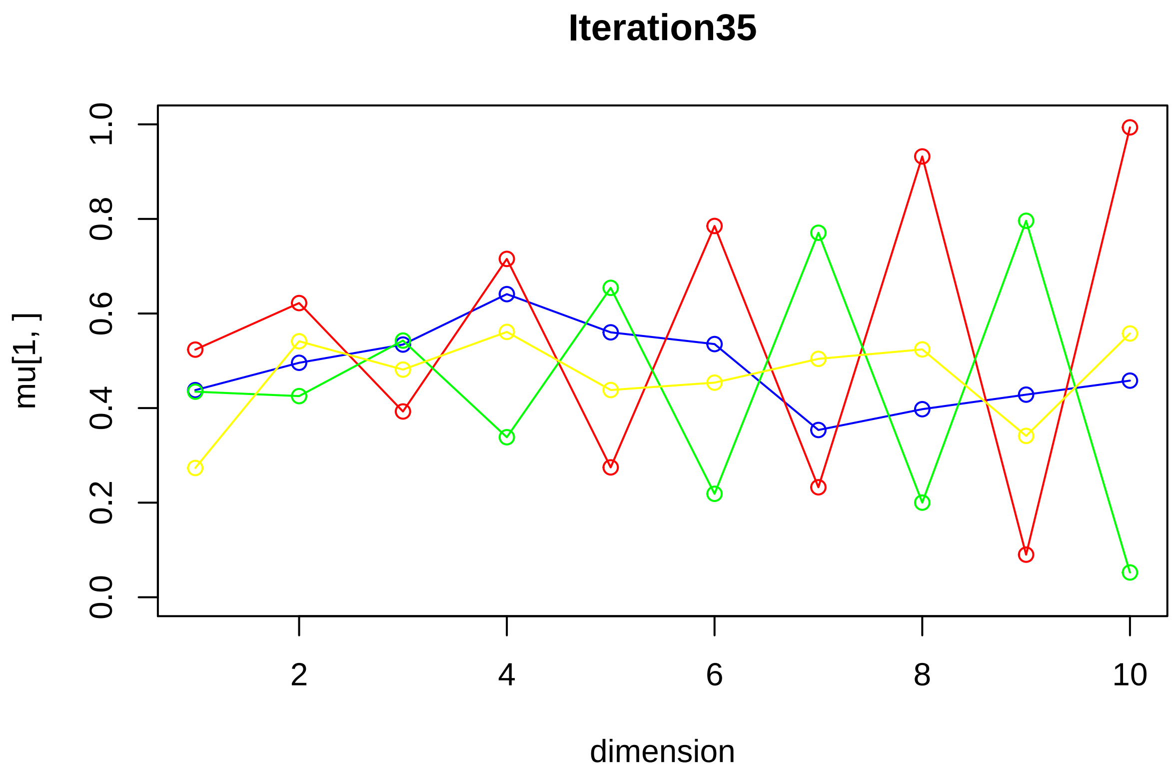

```
## iteration:  32 log likelihood:  -7297.671
```



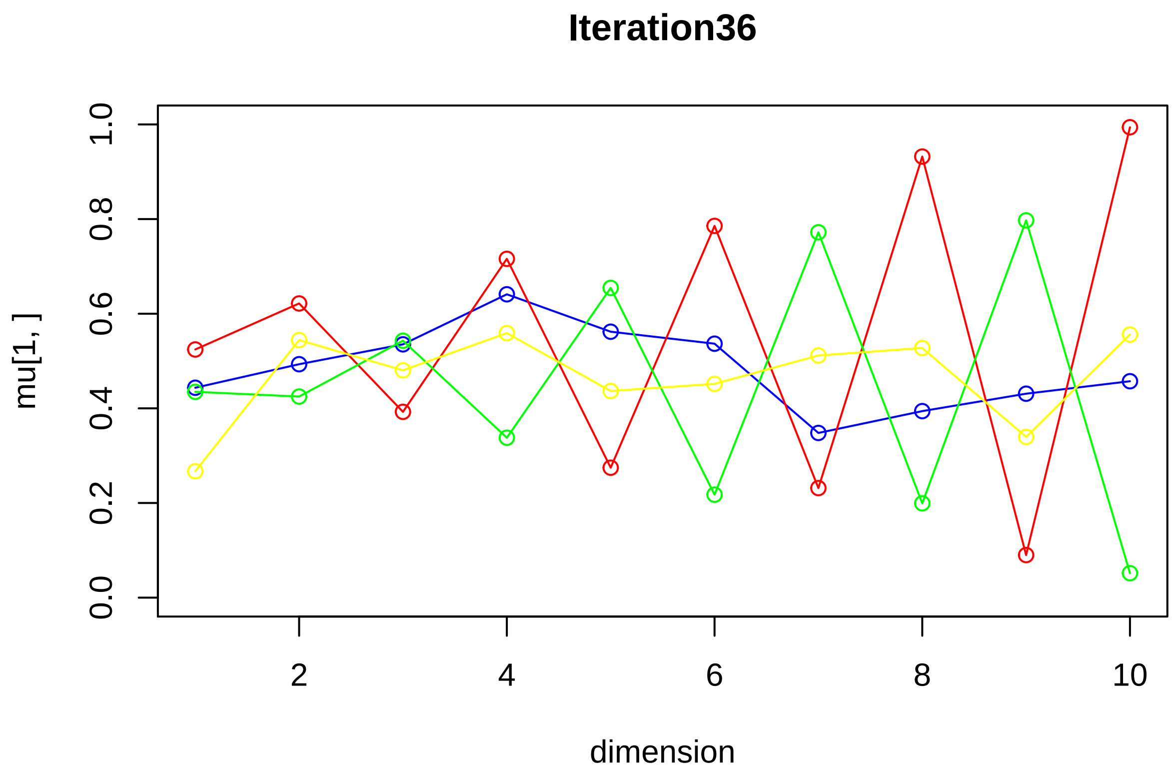

```
## iteration:  33 log likelihood:  -7295.014
```



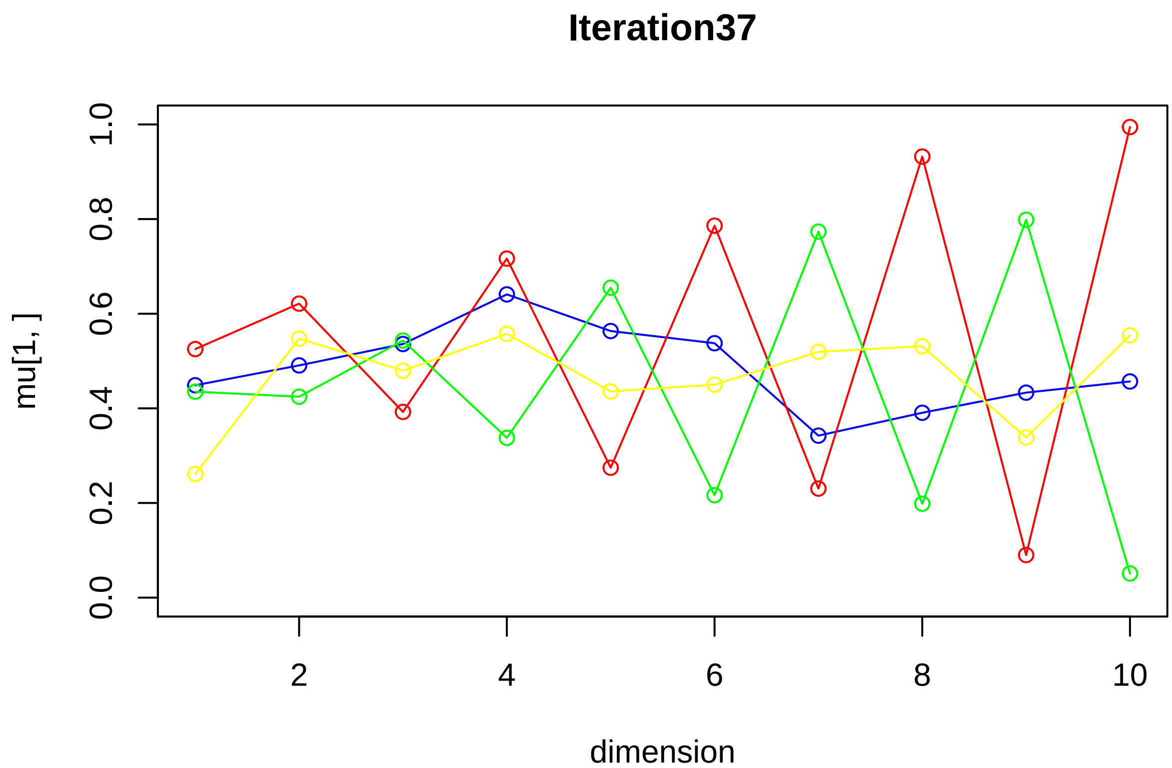

```
## iteration:  34 log likelihood:  -7292.171
```



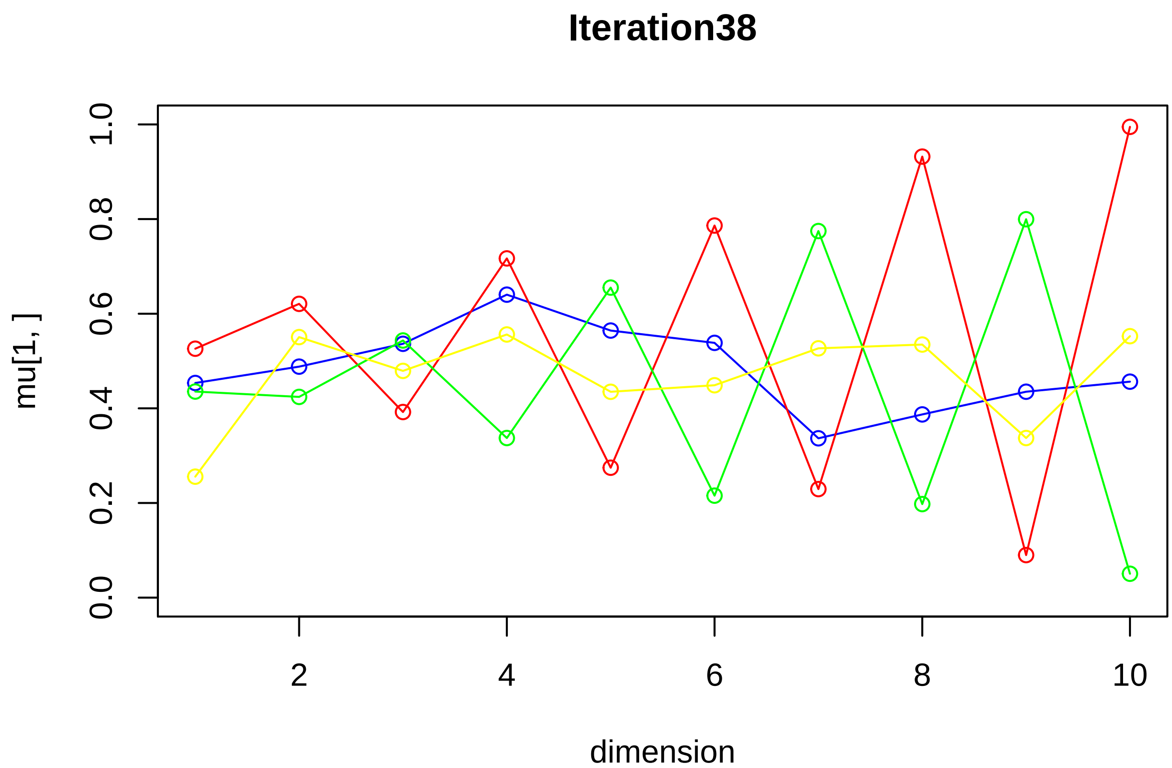

```
## iteration:  35 log likelihood:  -7289.196
```



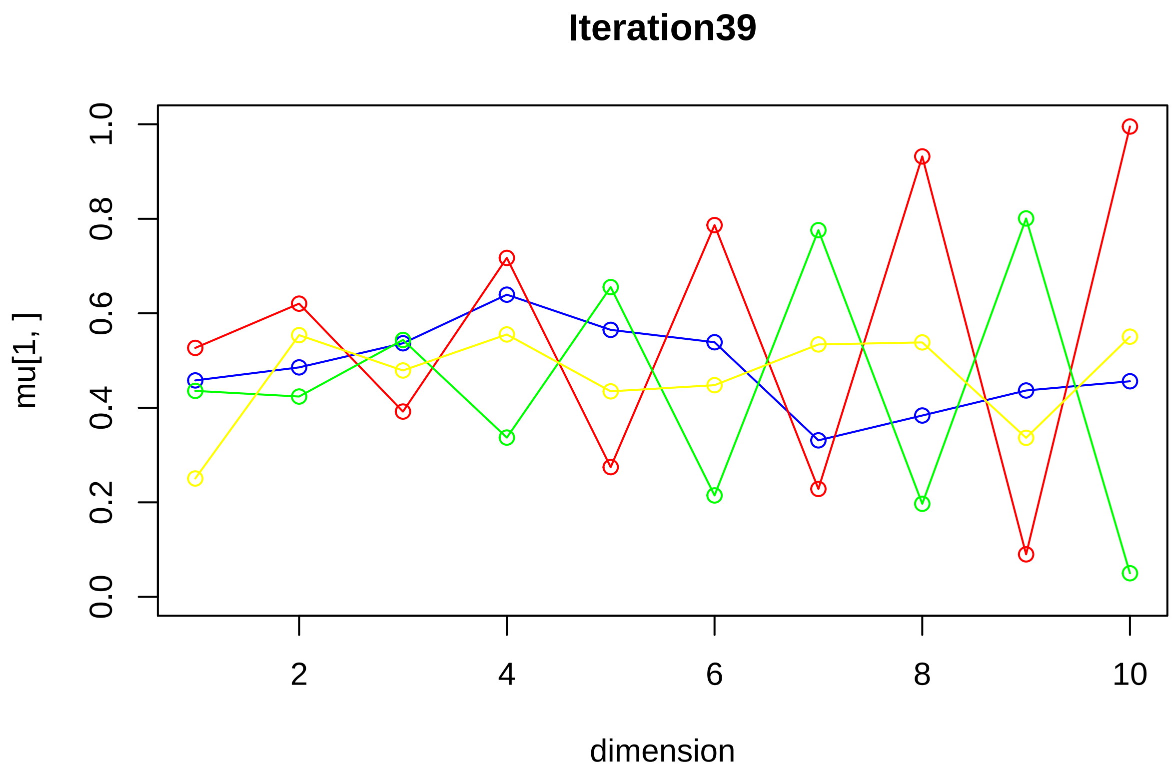

```
## iteration:  36 log likelihood:  -7286.15
```



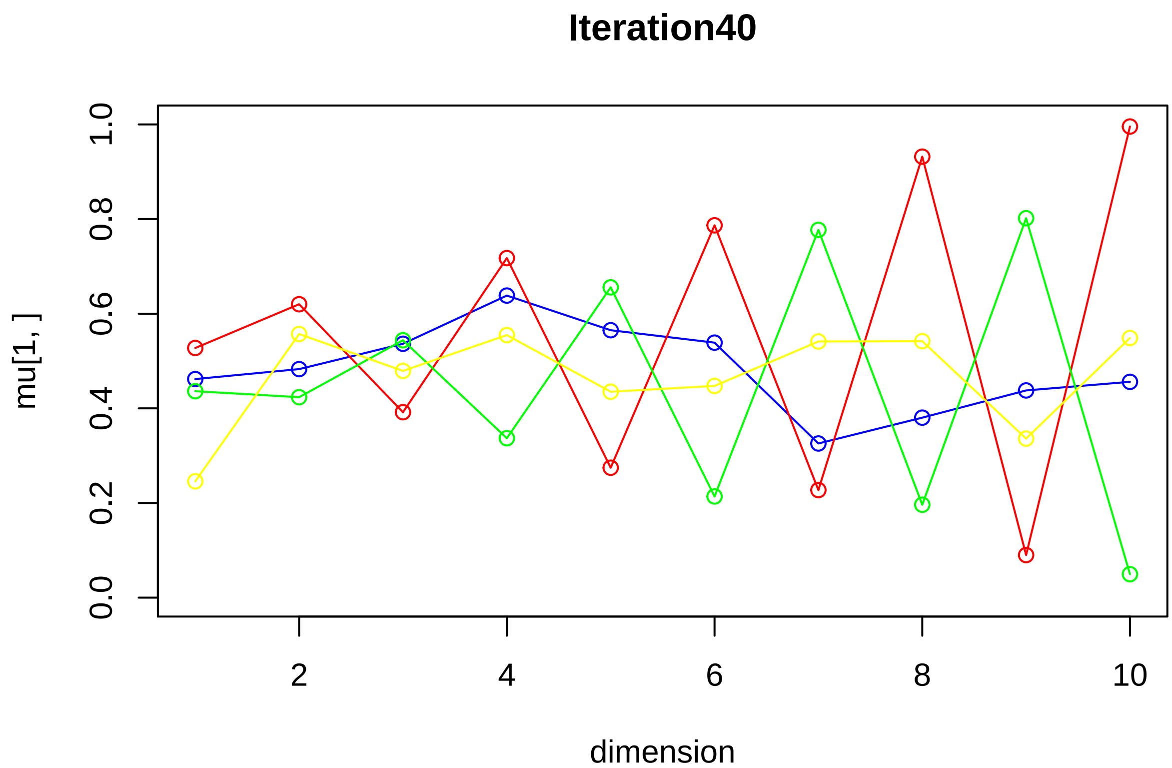

```
## iteration:  37 log likelihood:  -7283.093
```



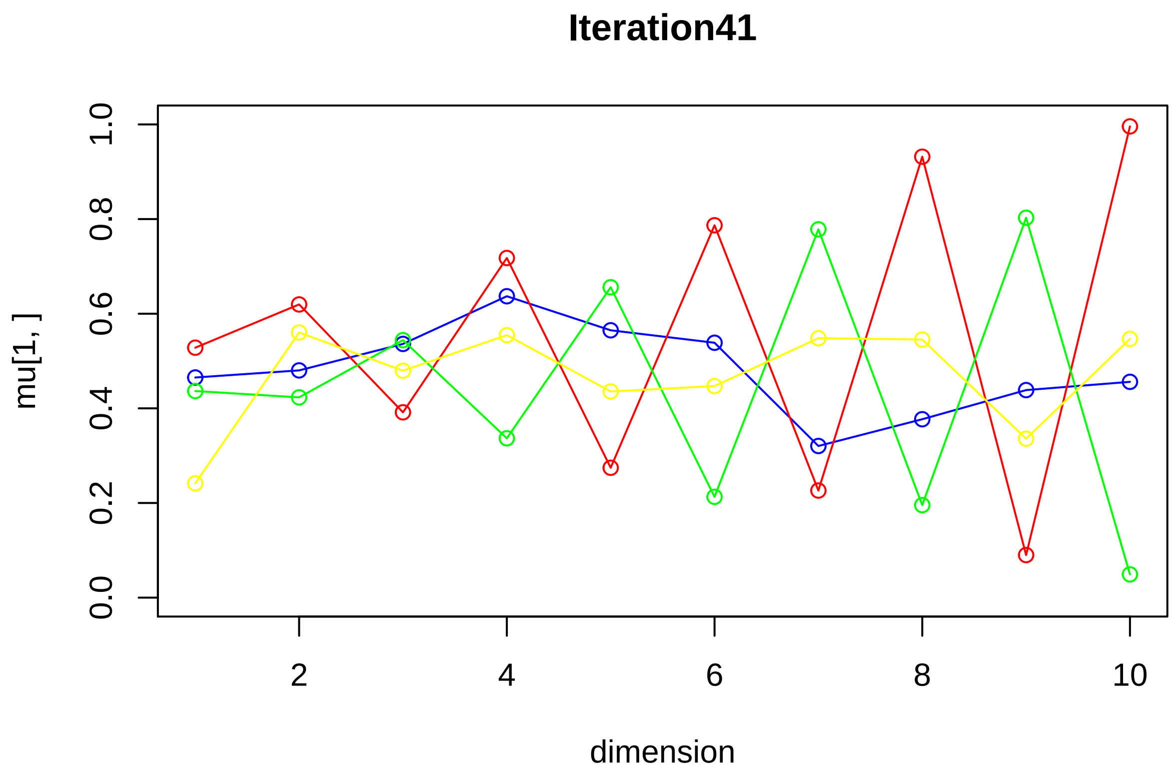

```
## iteration:  38 log likelihood:  -7280.079
```



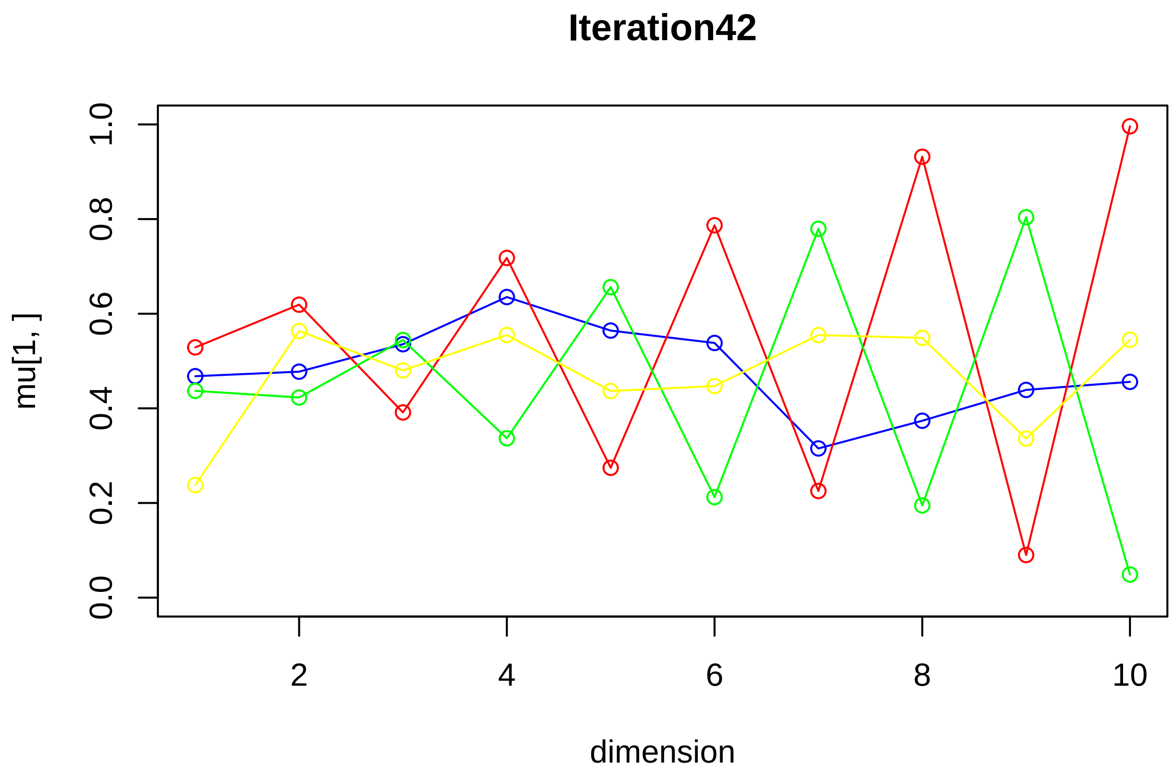

```
## iteration:  39 log likelihood:  -7277.151
```



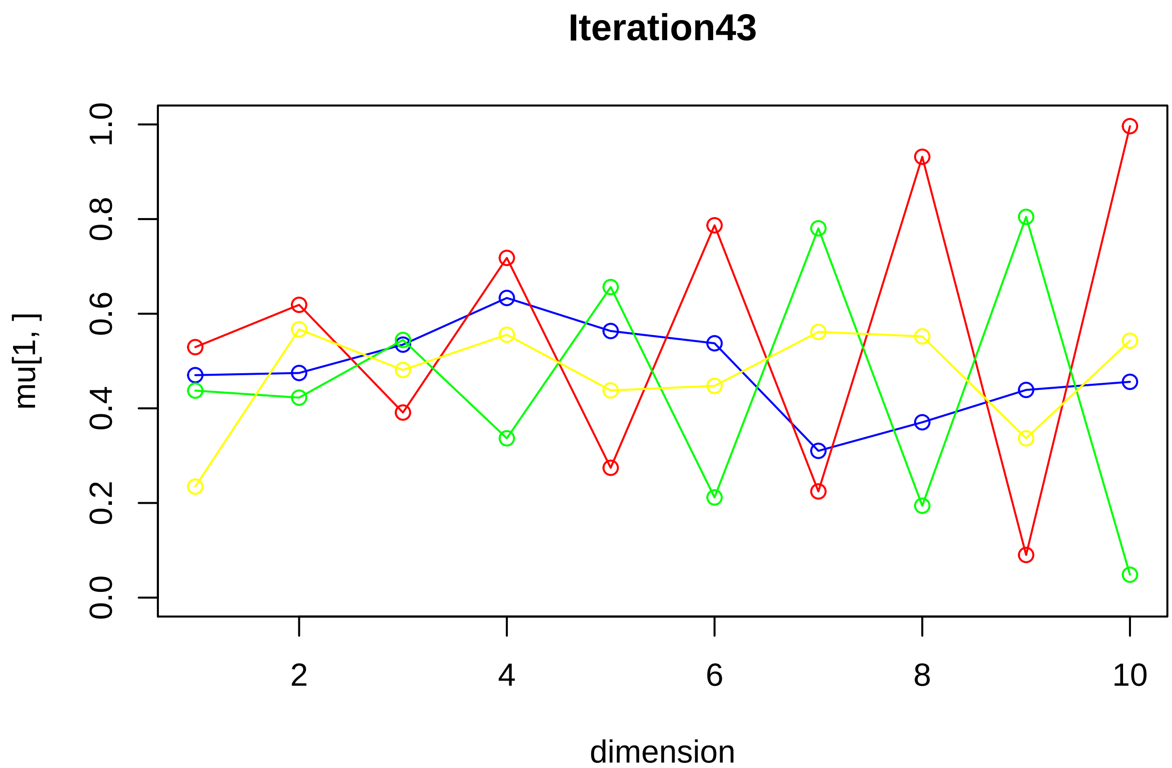

```
## iteration:  40 log likelihood:  -7274.34
```



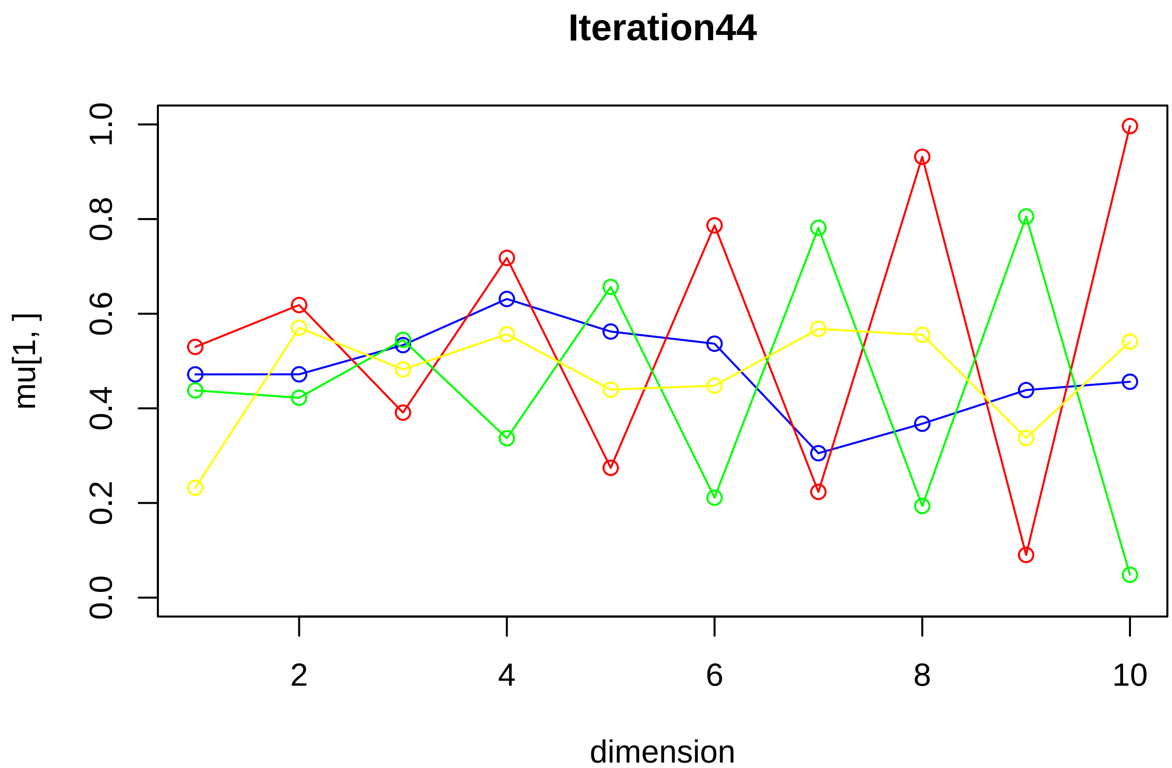

```
## iteration:  41 log likelihood:  -7271.66
```



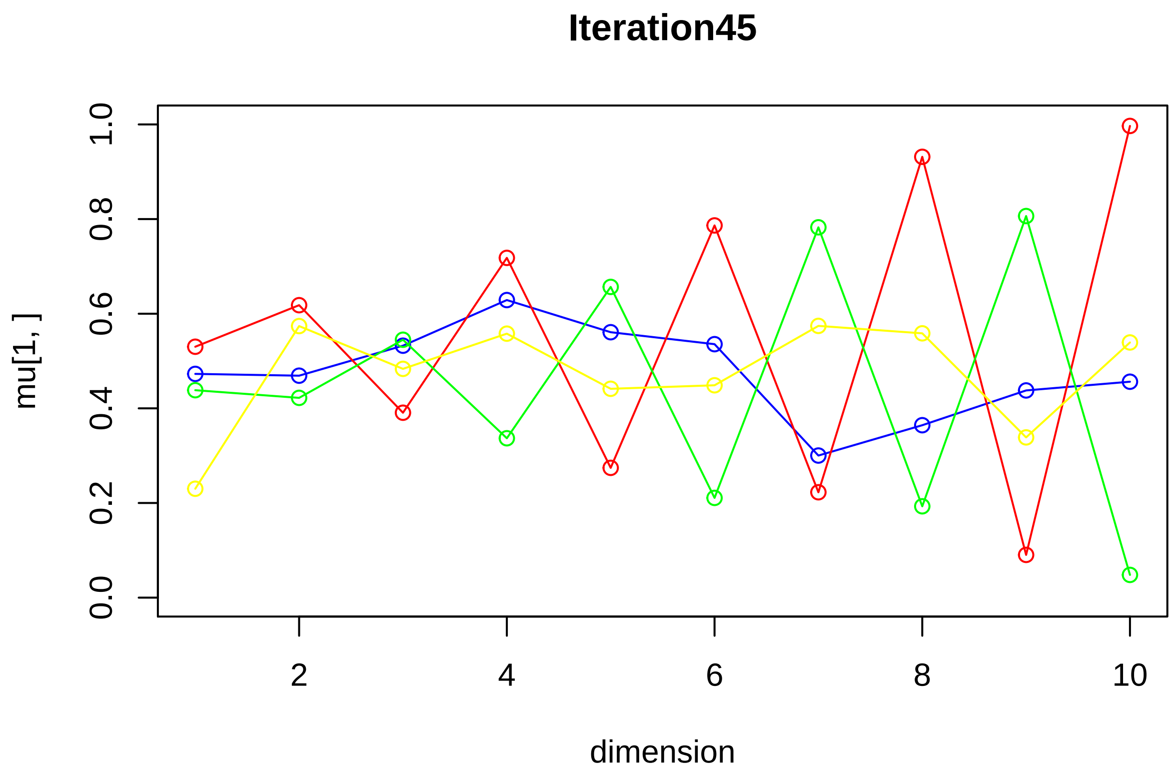

```
## iteration:  42 log likelihood:  -7269.116
```



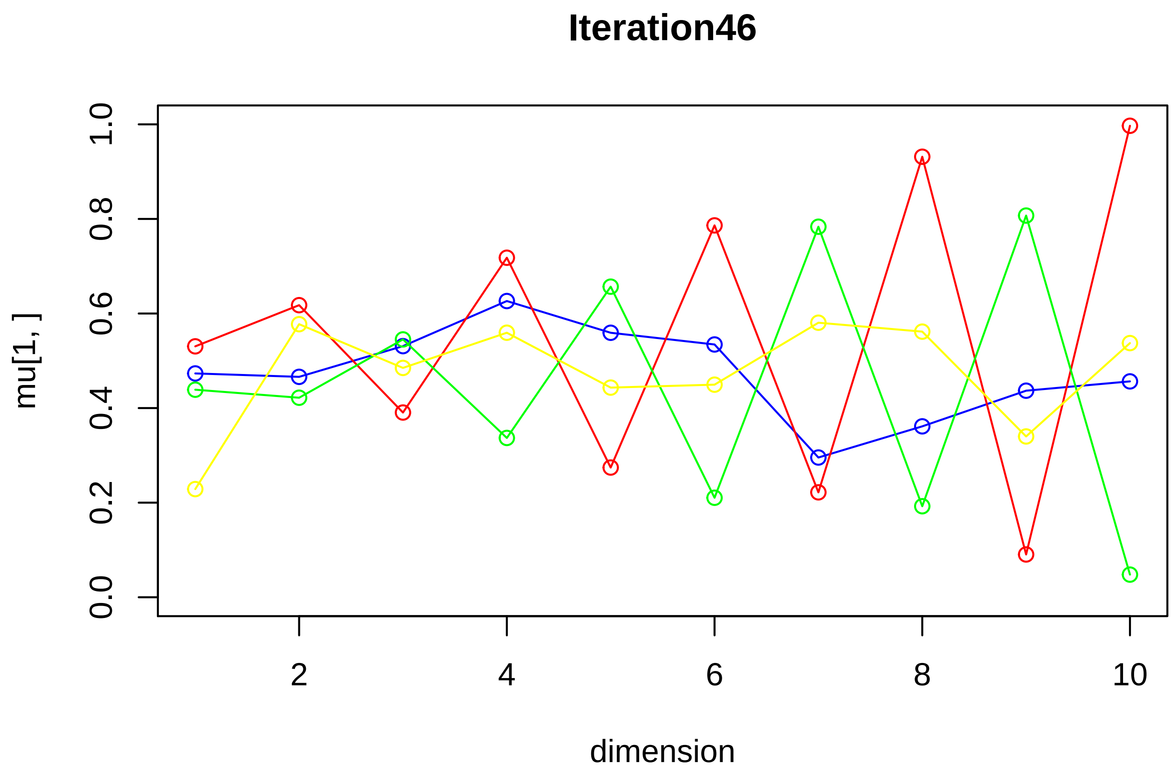

```
## iteration:  43 log likelihood:  -7266.7
```



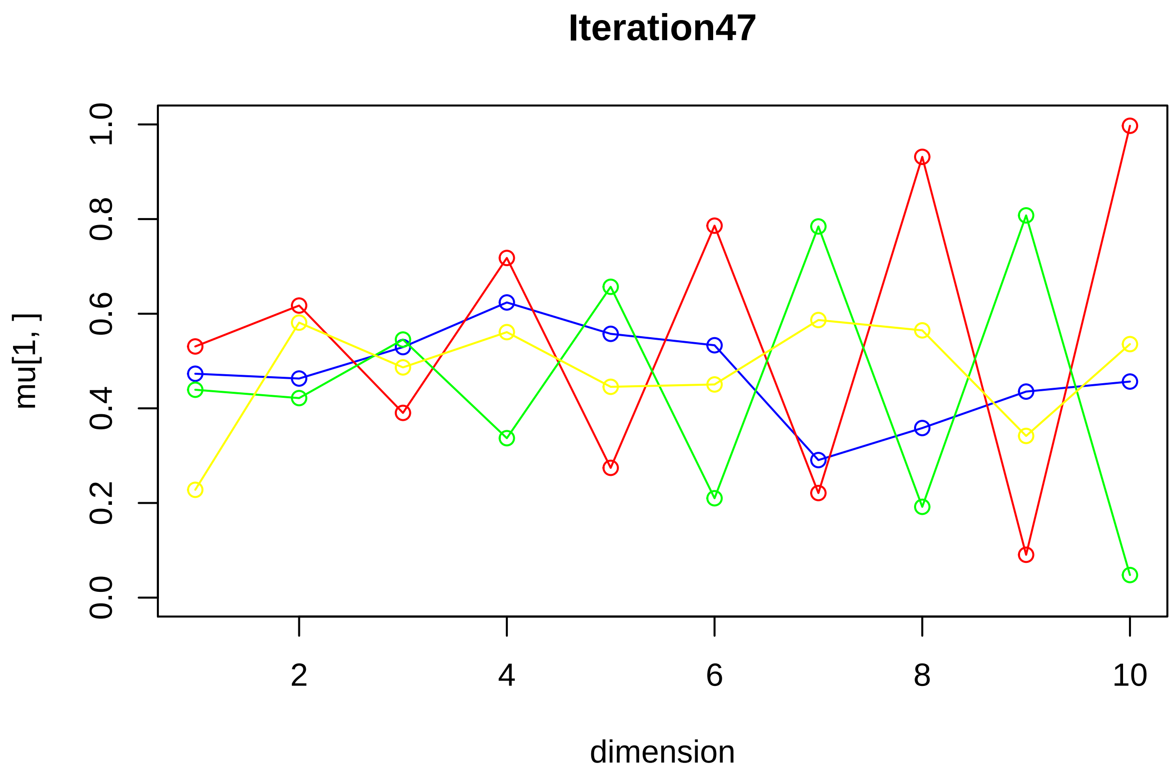

```
## iteration:  44 log likelihood:  -7264.398
```



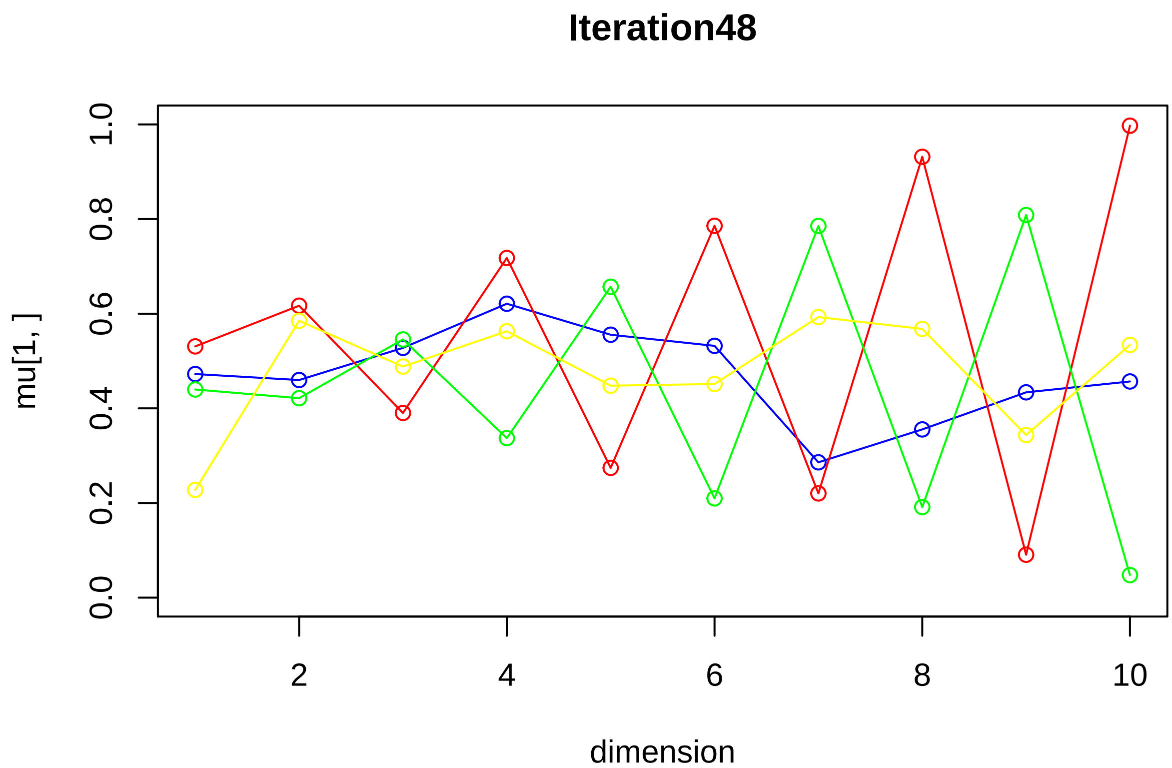

```
## iteration:  45 log likelihood:  -7262.189
```



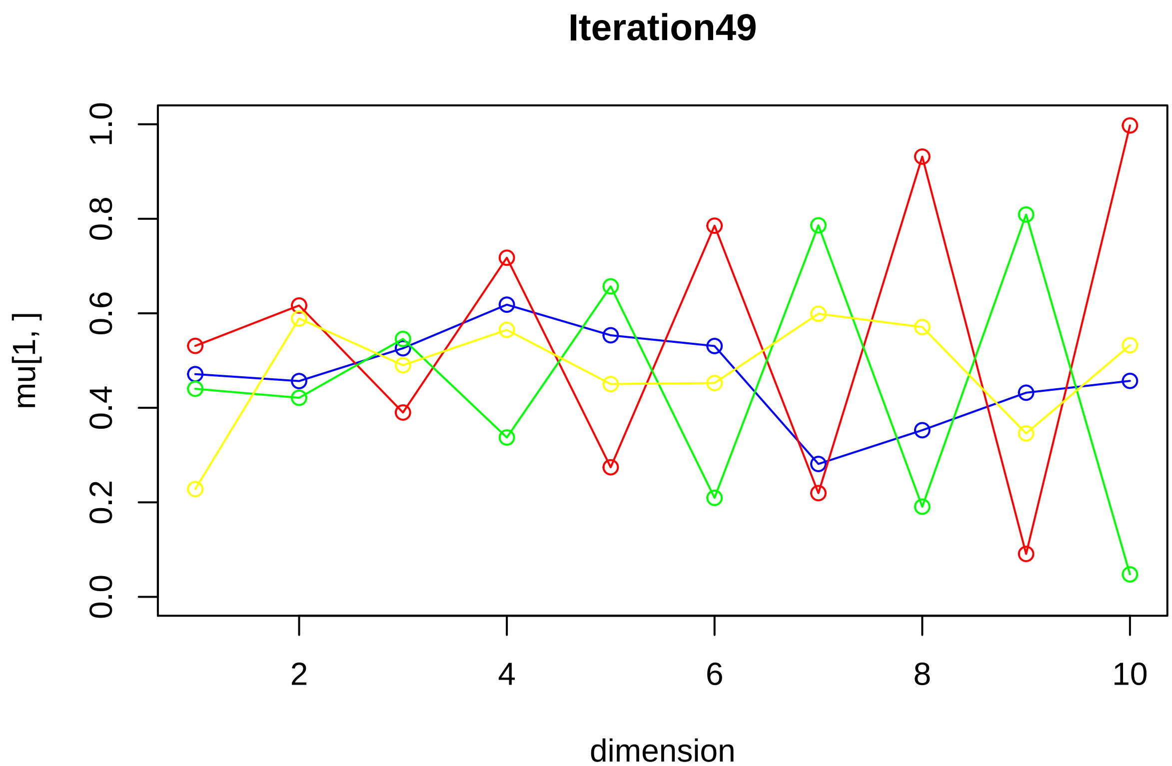

```
## iteration:  46 log likelihood:  -7260.051
```



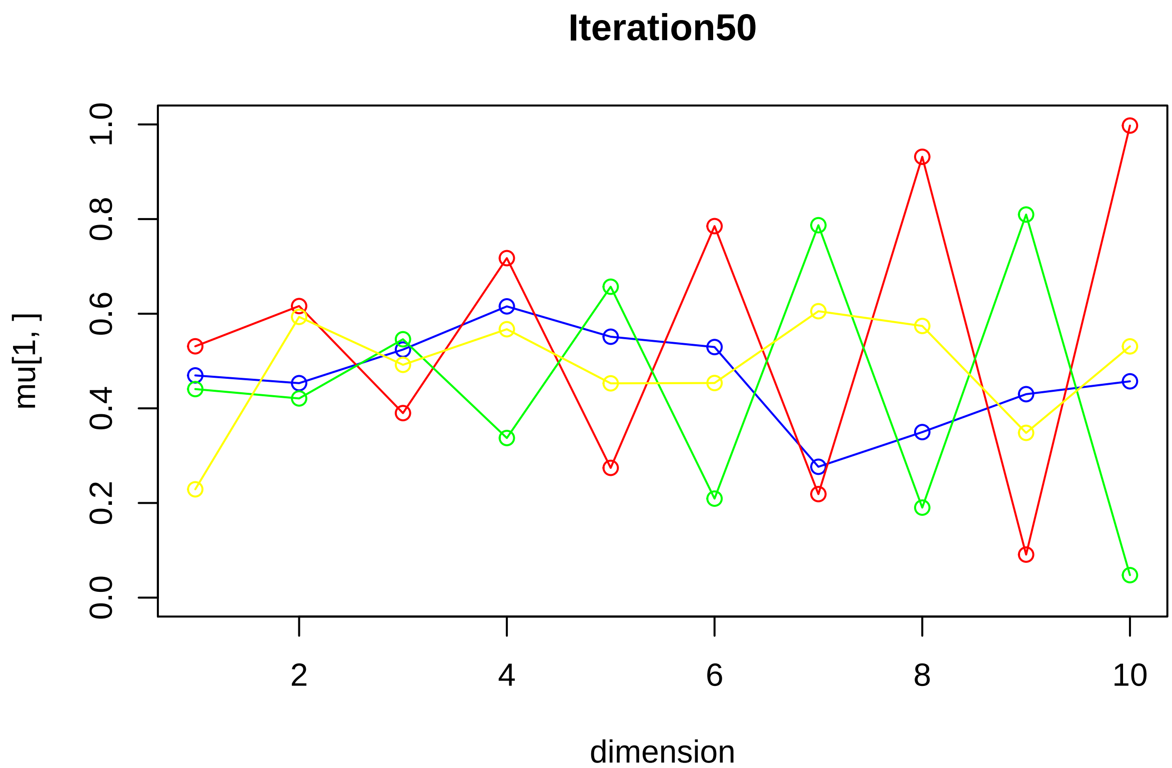

```
## iteration:  47 log likelihood:  -7257.96
```



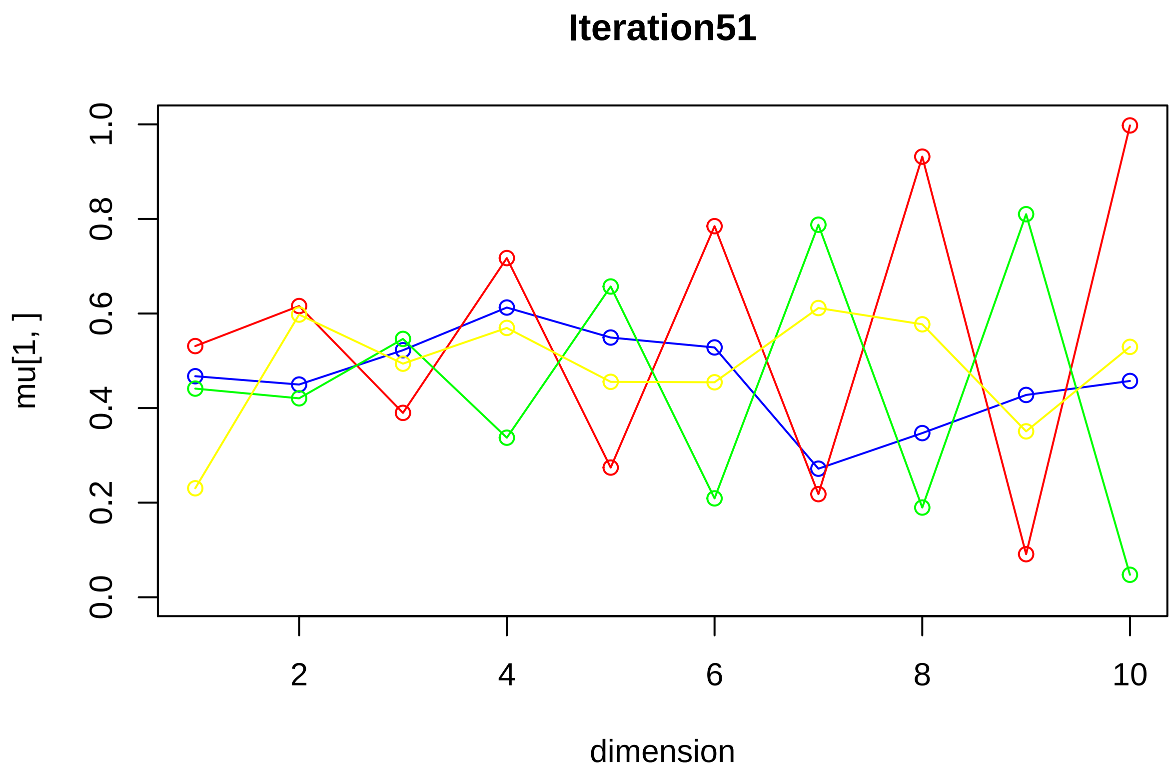

```
## iteration:  48 log likelihood:  -7255.892
```



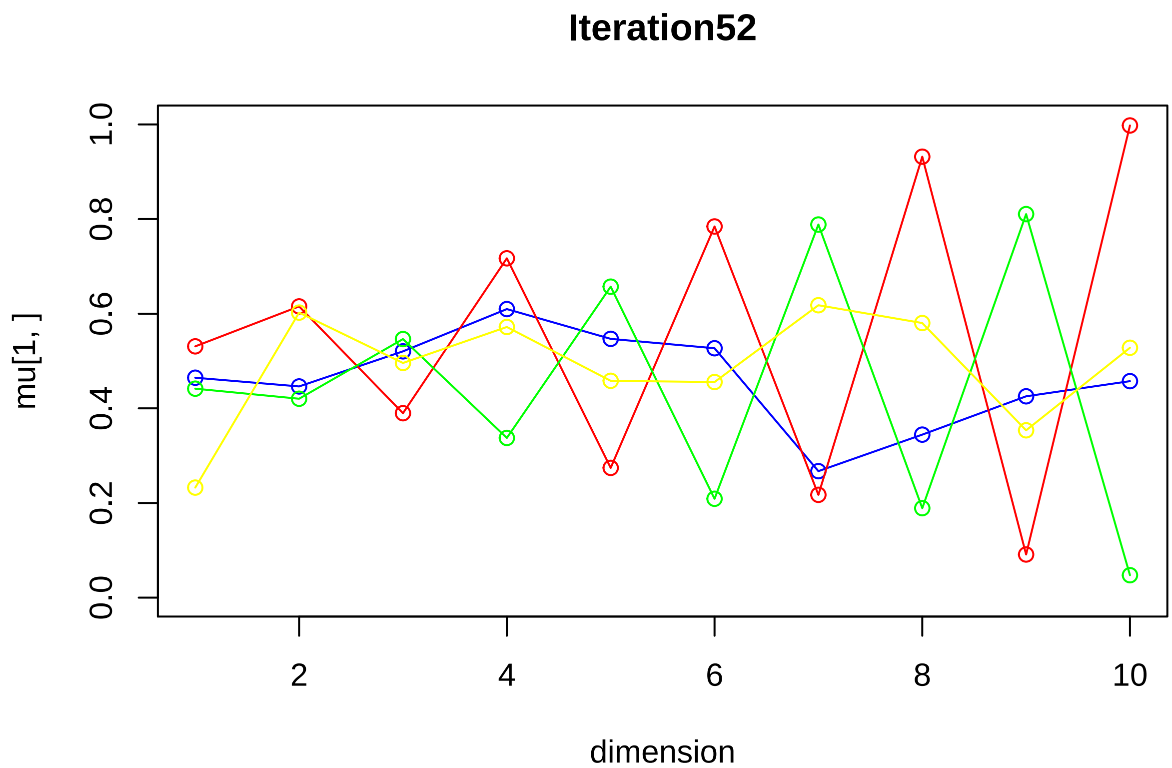

```
## iteration:  49 log likelihood:  -7253.824
```



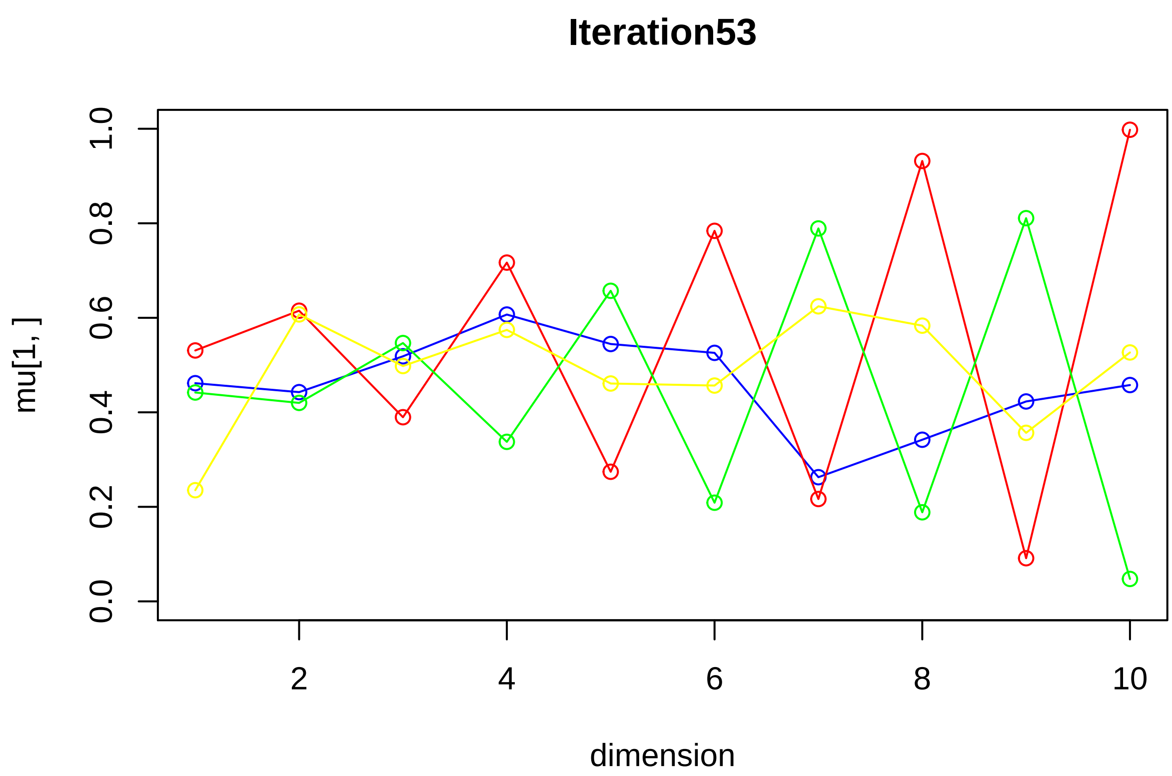

```
## iteration:  50 log likelihood:  -7251.733
```



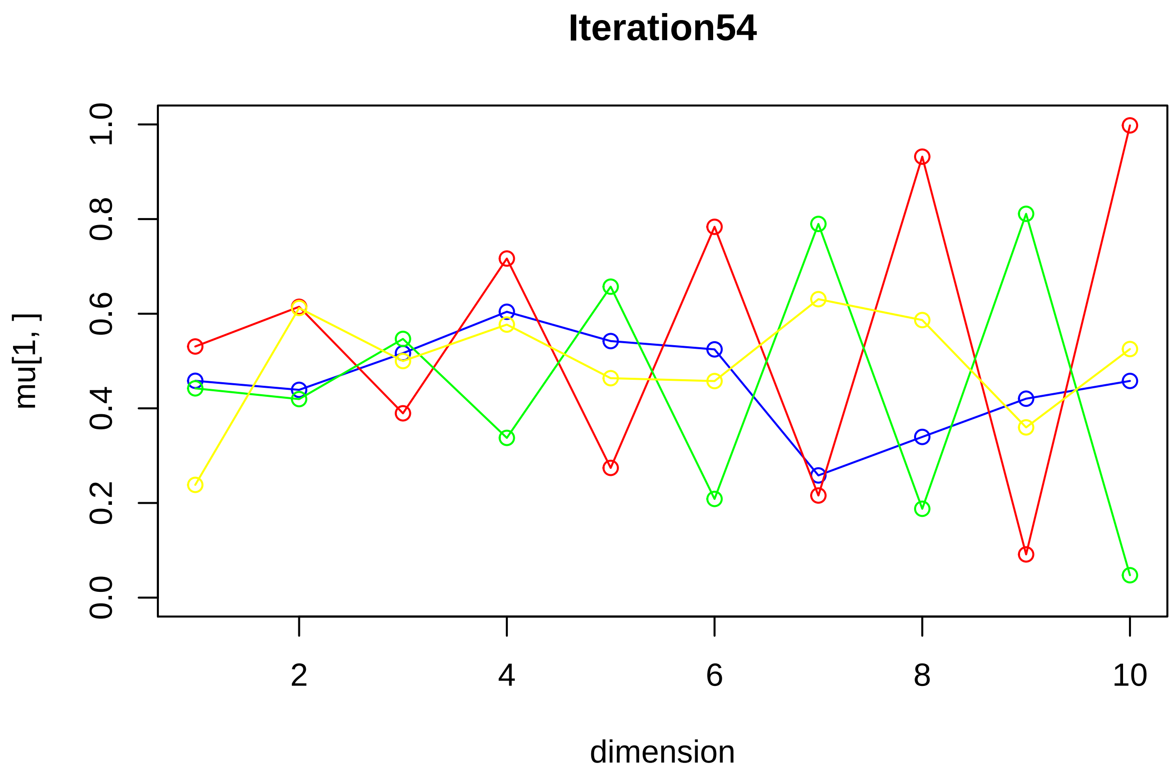

```
## iteration:  51 log likelihood:  -7249.603
```



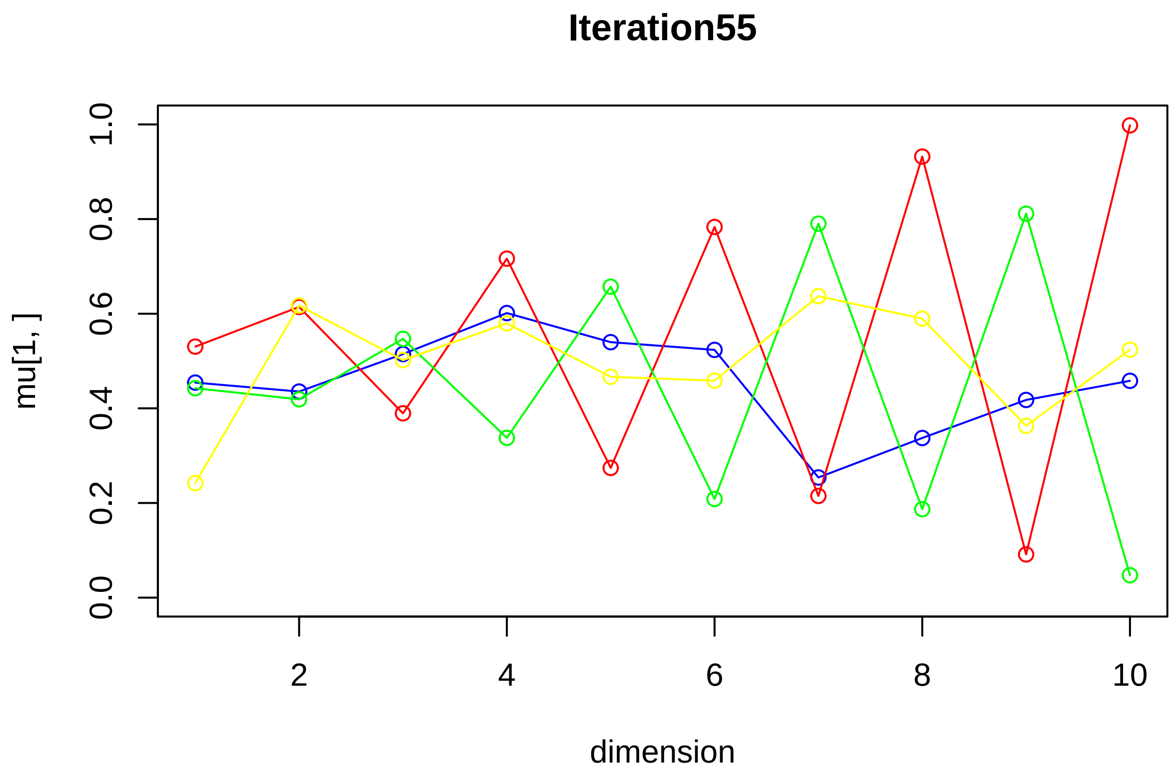

```
## iteration:  52 log likelihood:  -7247.419
```



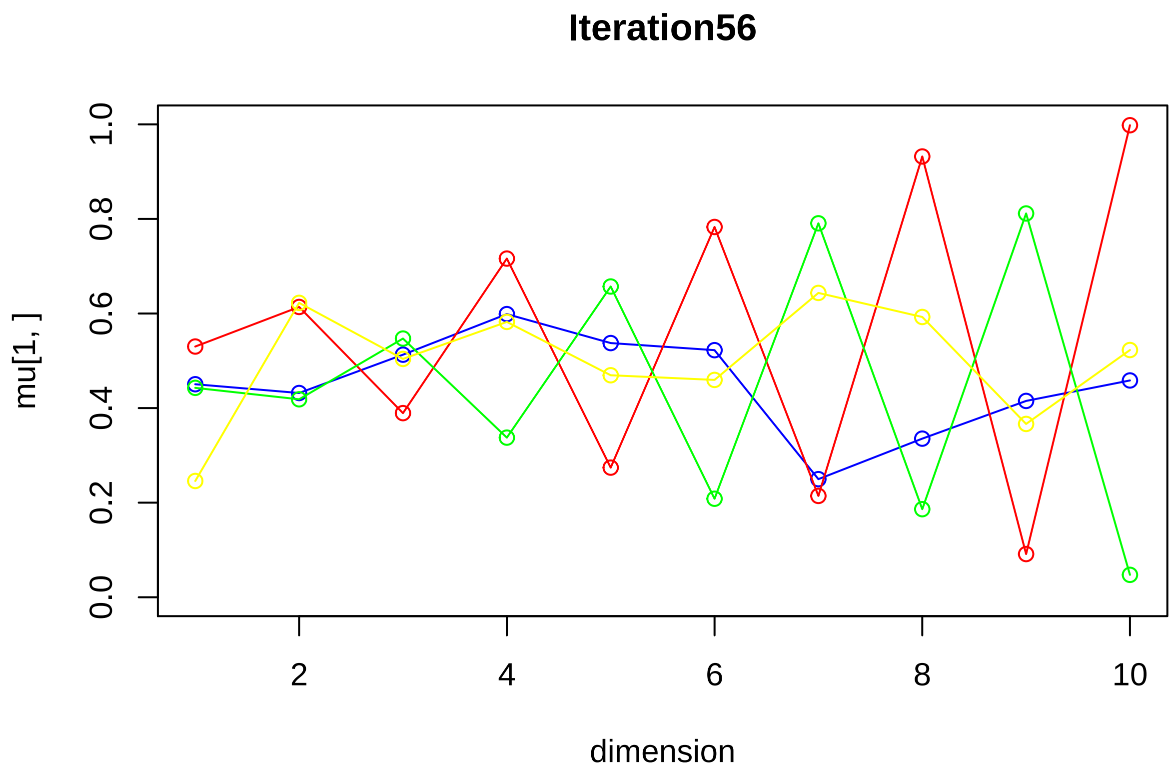

```
## iteration:  53 log likelihood:  -7245.17
```



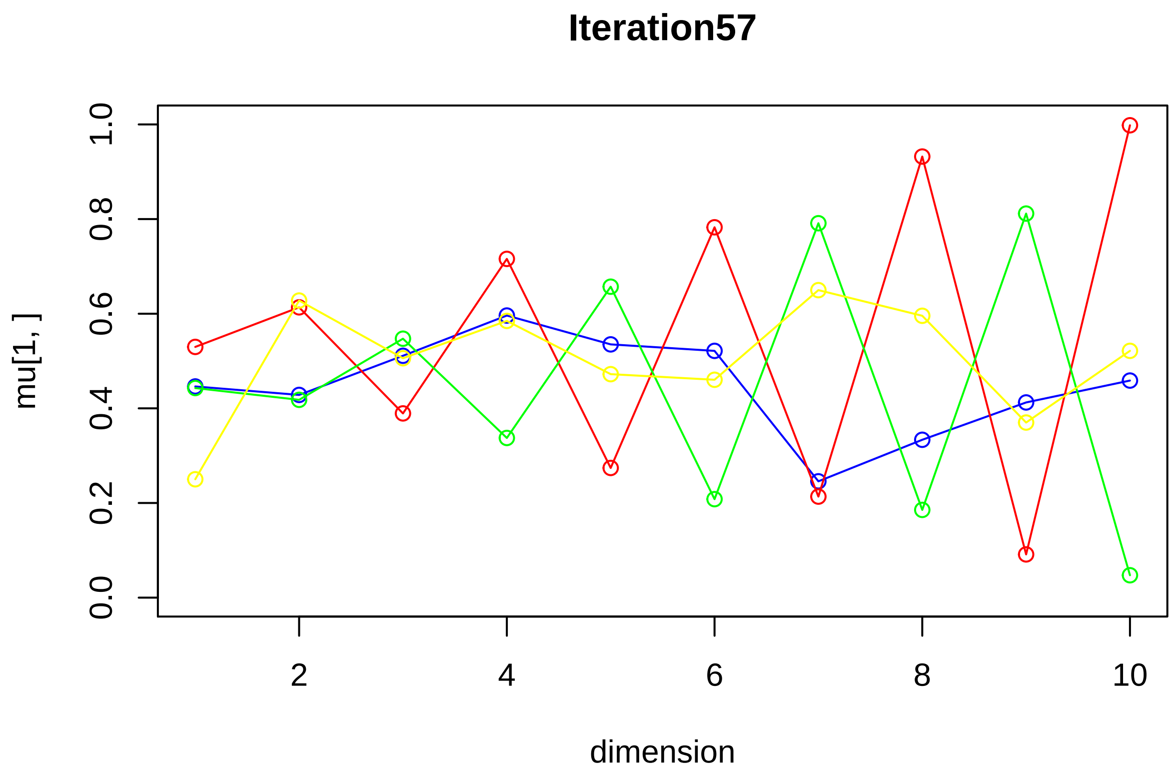

```
## iteration:  54 log likelihood:  -7242.853
```



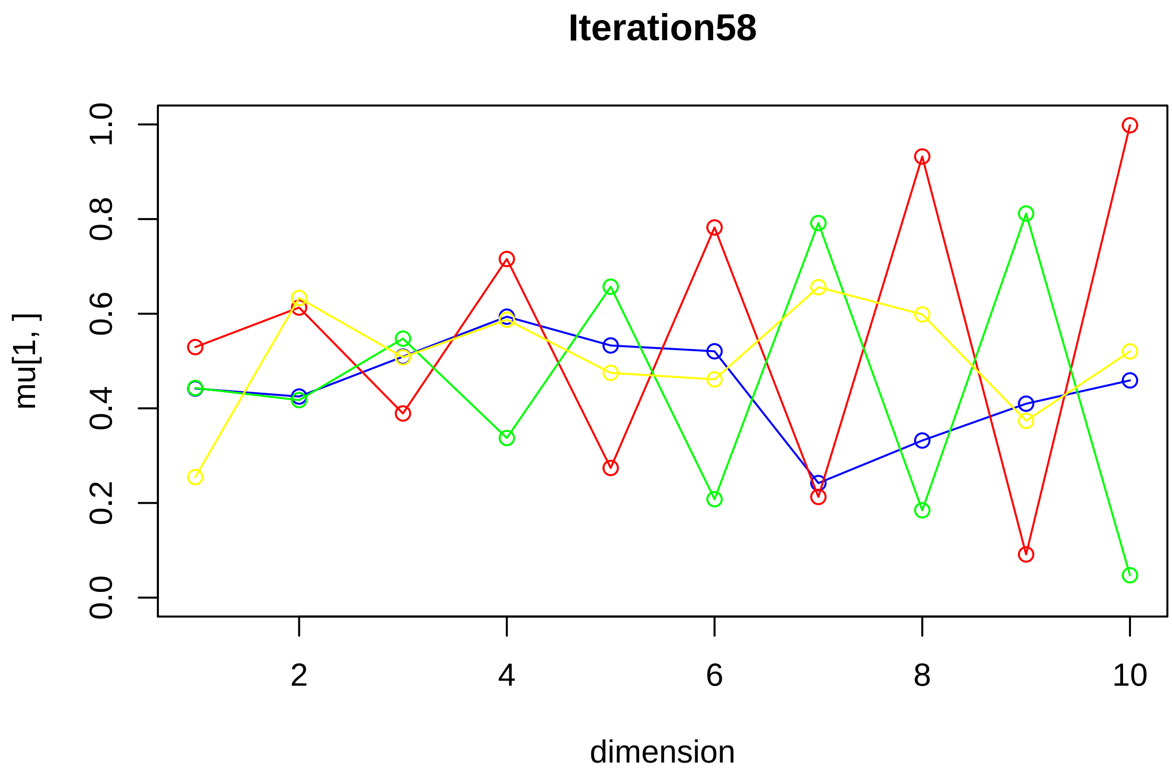

```
## iteration:  55 log likelihood:  -7240.472
```



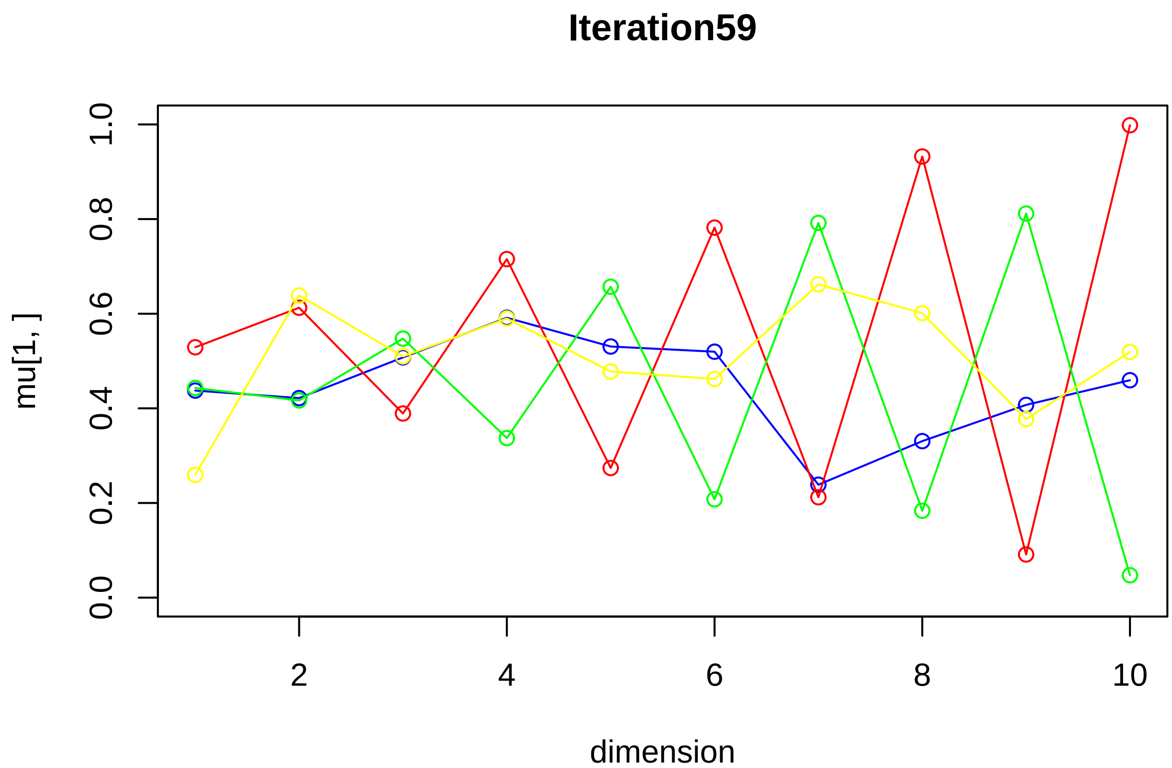

```
## iteration:  56 log likelihood:  -7238.038
```



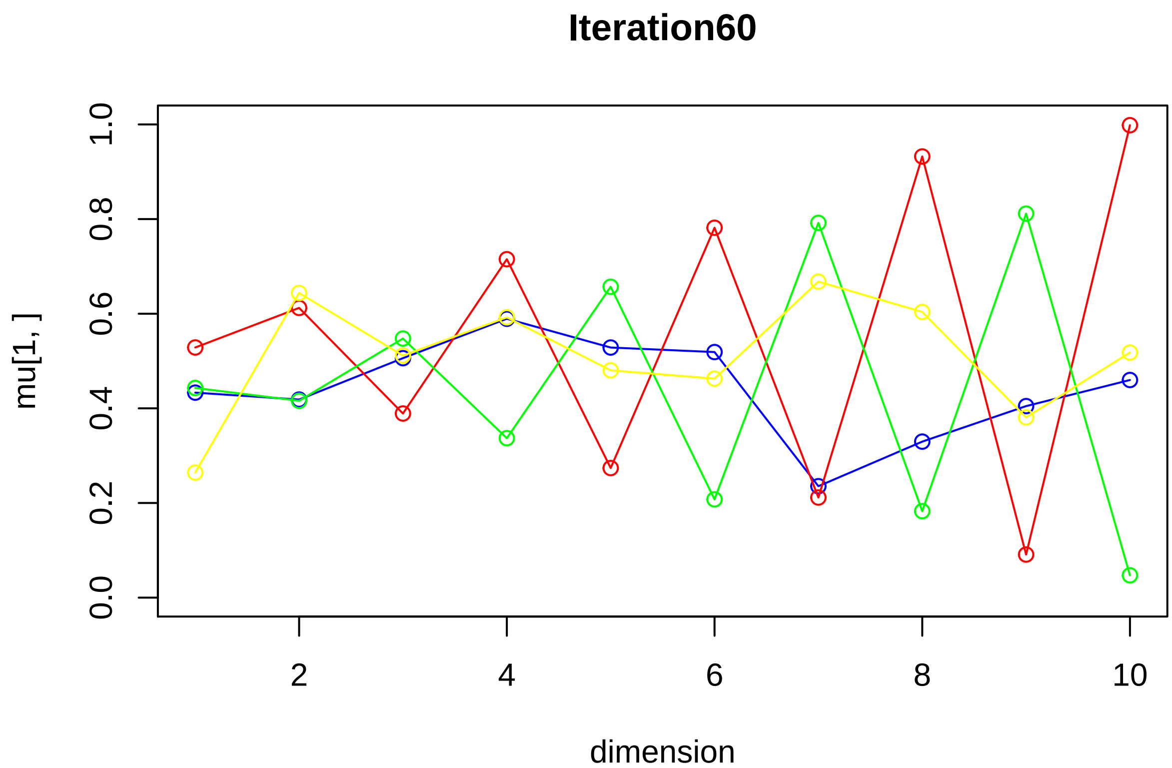

```
## iteration:  57 log likelihood:  -7235.571
```



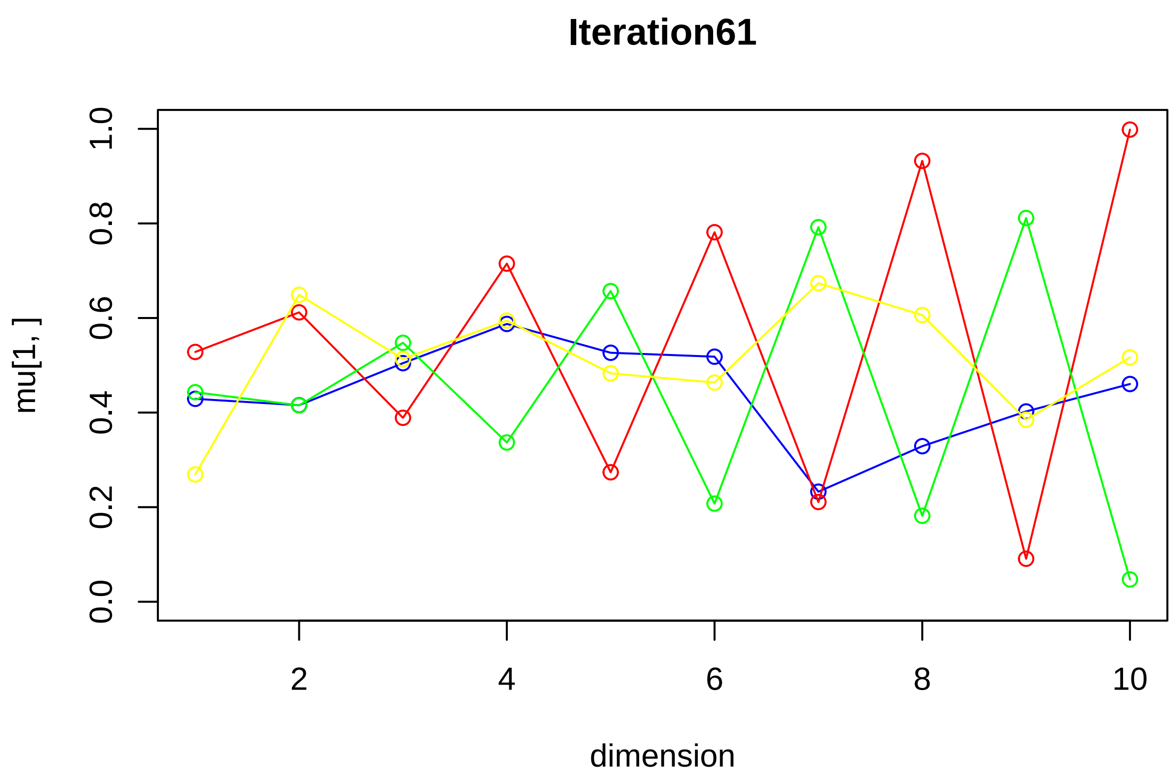

```
## iteration:  58 log likelihood:  -7233.095
```



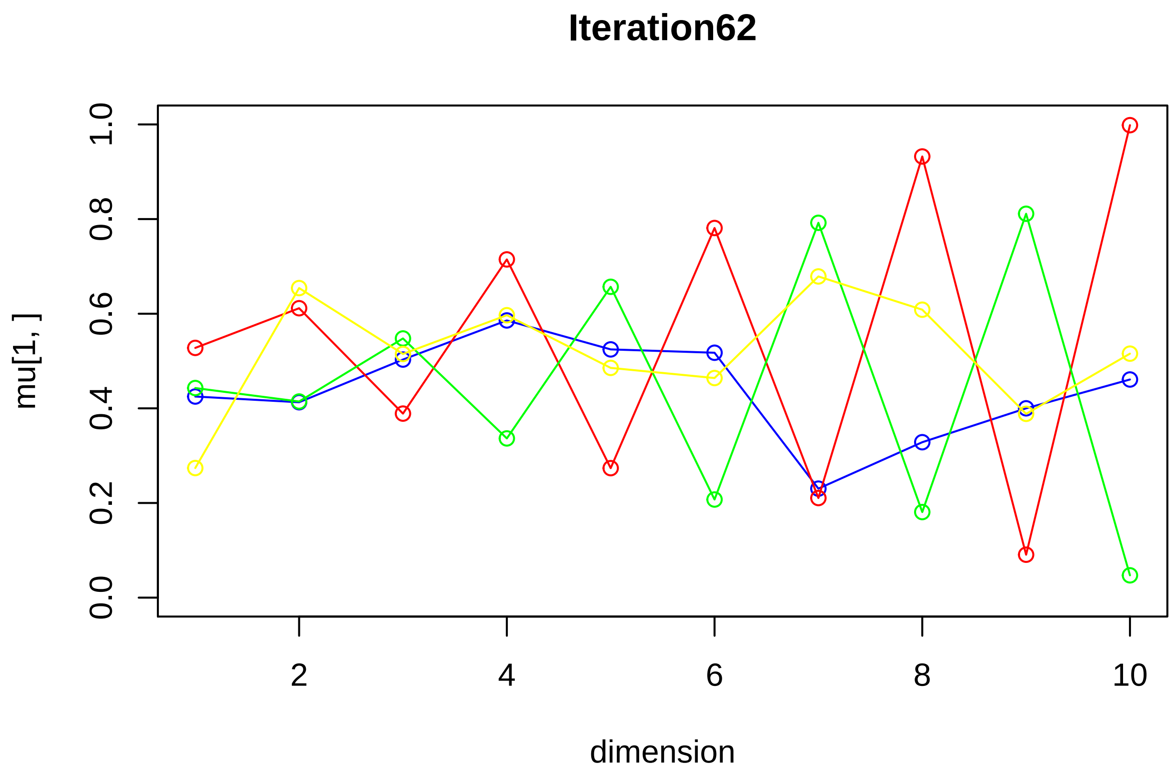

```
## iteration:  59 log likelihood:  -7230.64
```



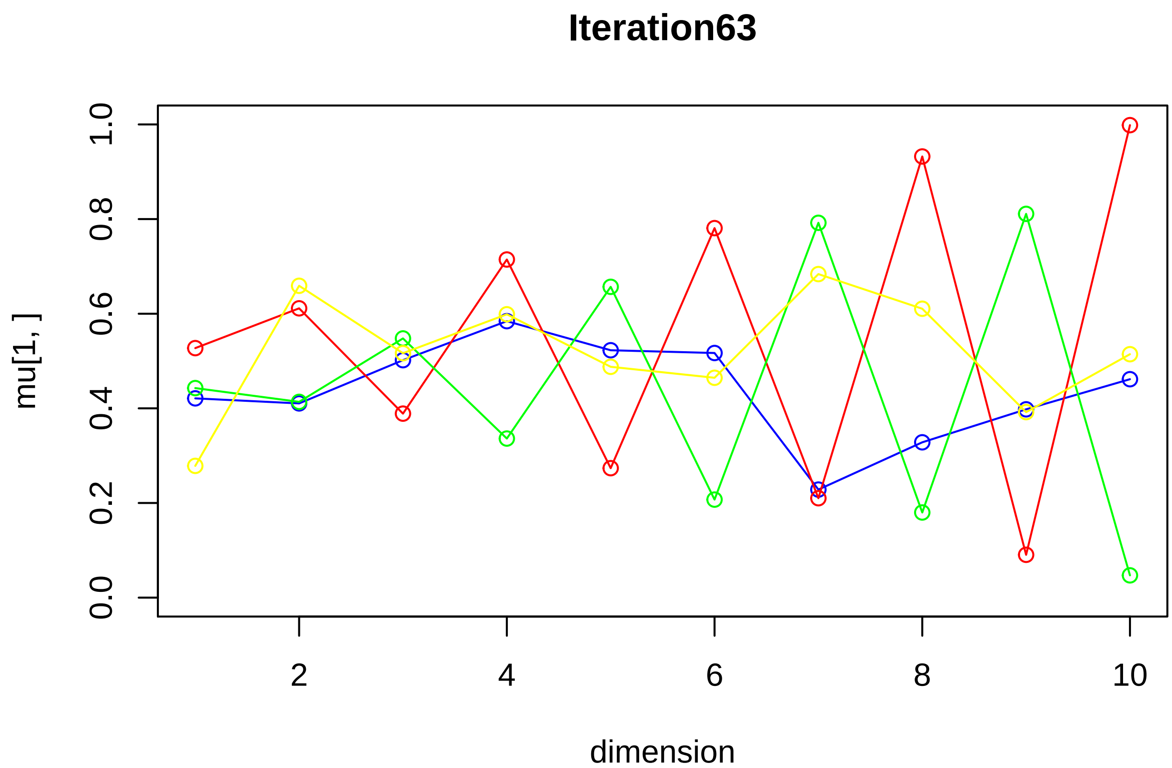

```
## iteration:  60 log likelihood:  -7228.239
```



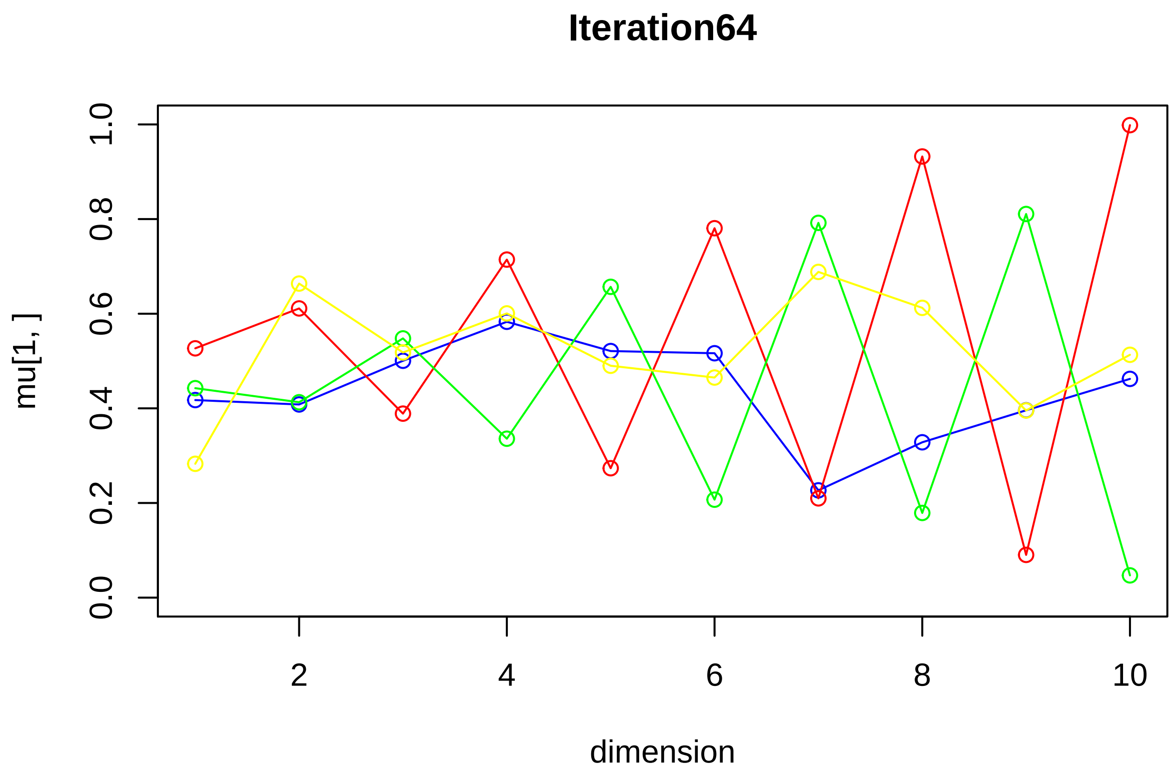

```
## iteration:  61 log likelihood:  -7225.925
```



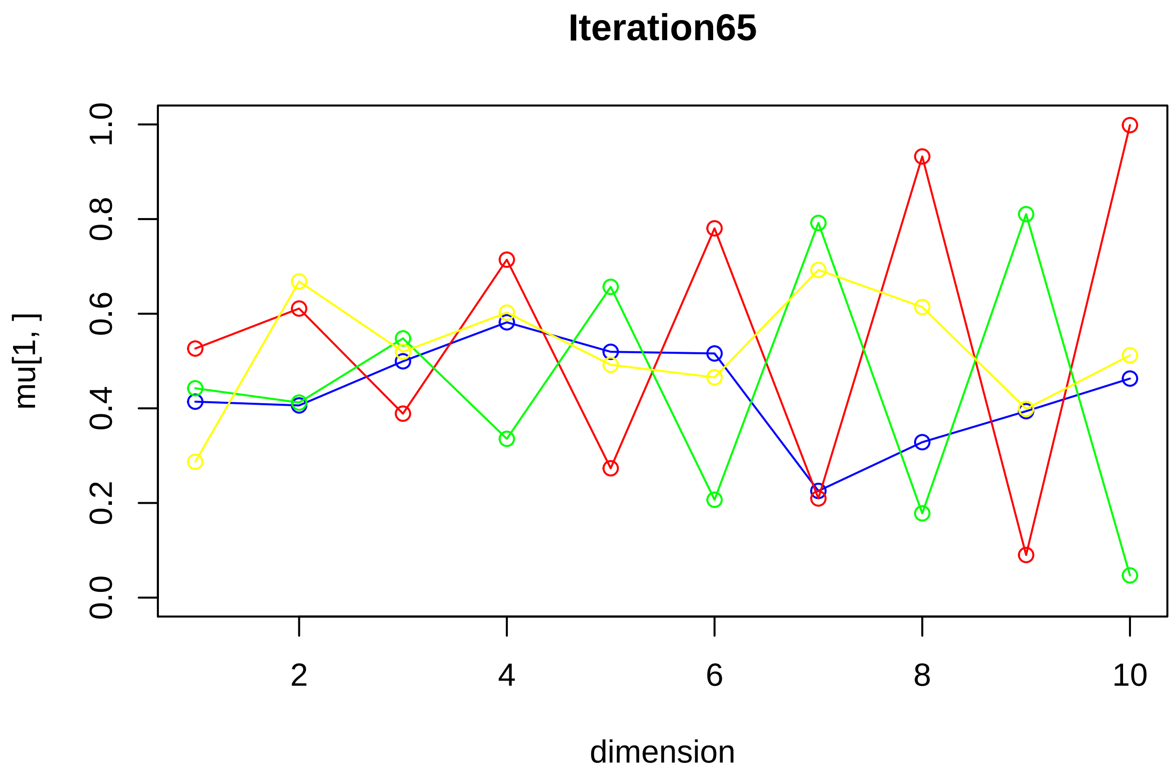

```
## iteration:  62 log likelihood:  -7223.725
```



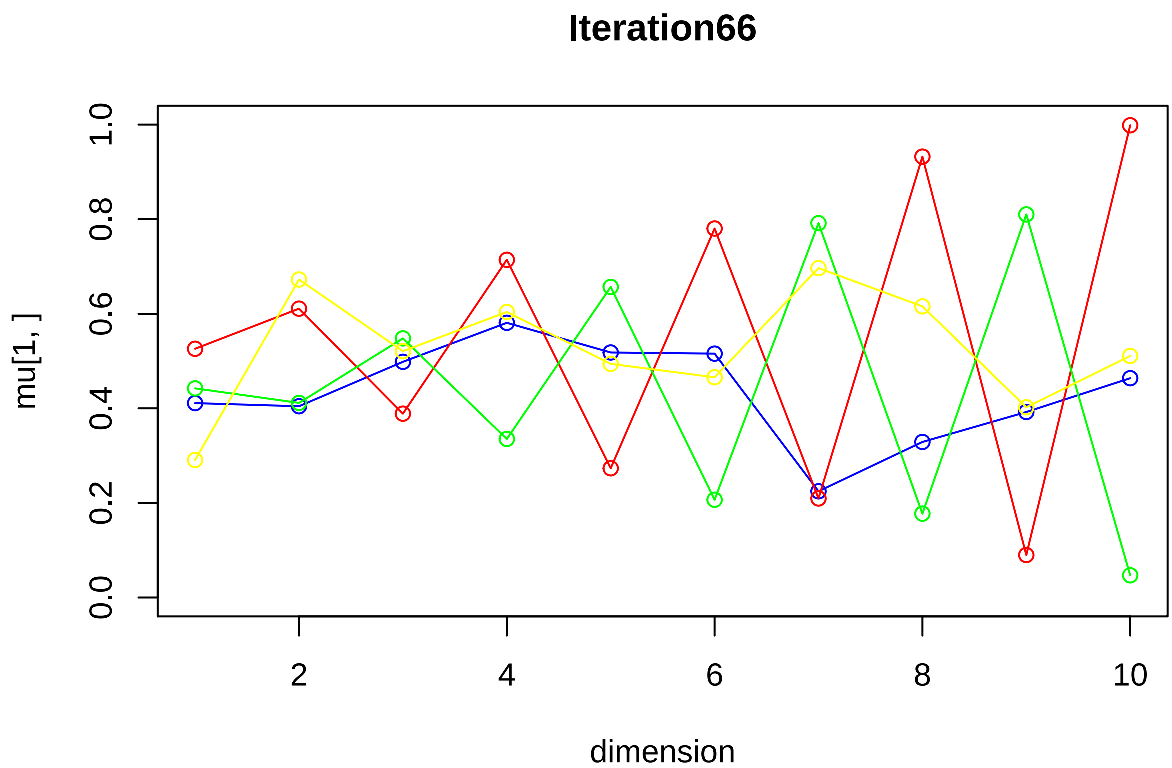

```
## iteration:  63 log likelihood:  -7221.663
```



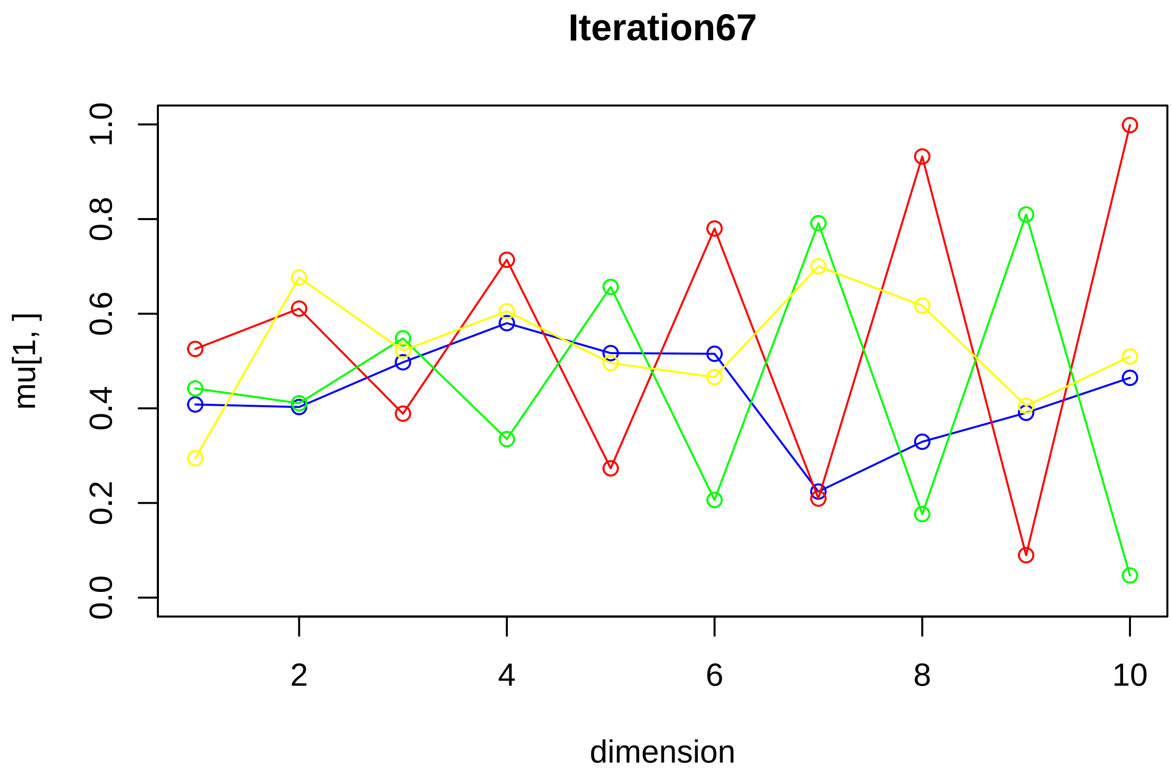

```
## iteration:  64 log likelihood:  -7219.755
```



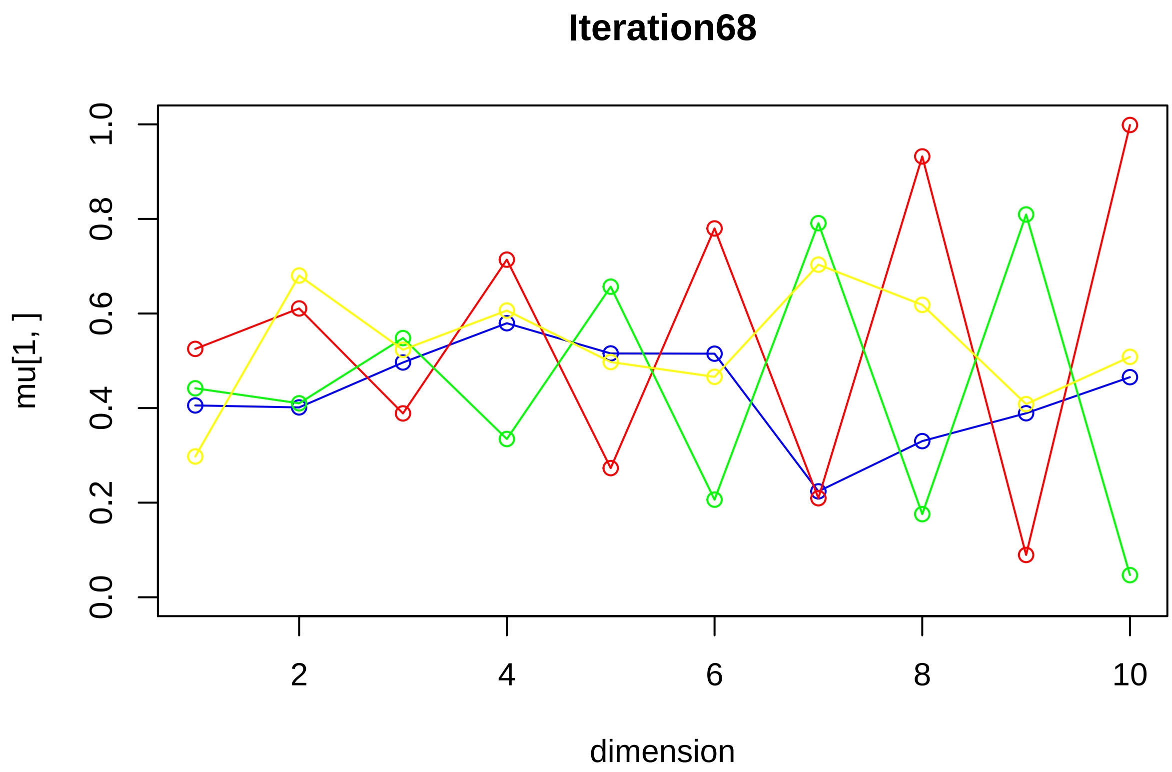

```
## iteration:  65 log likelihood:  -7218.01
```



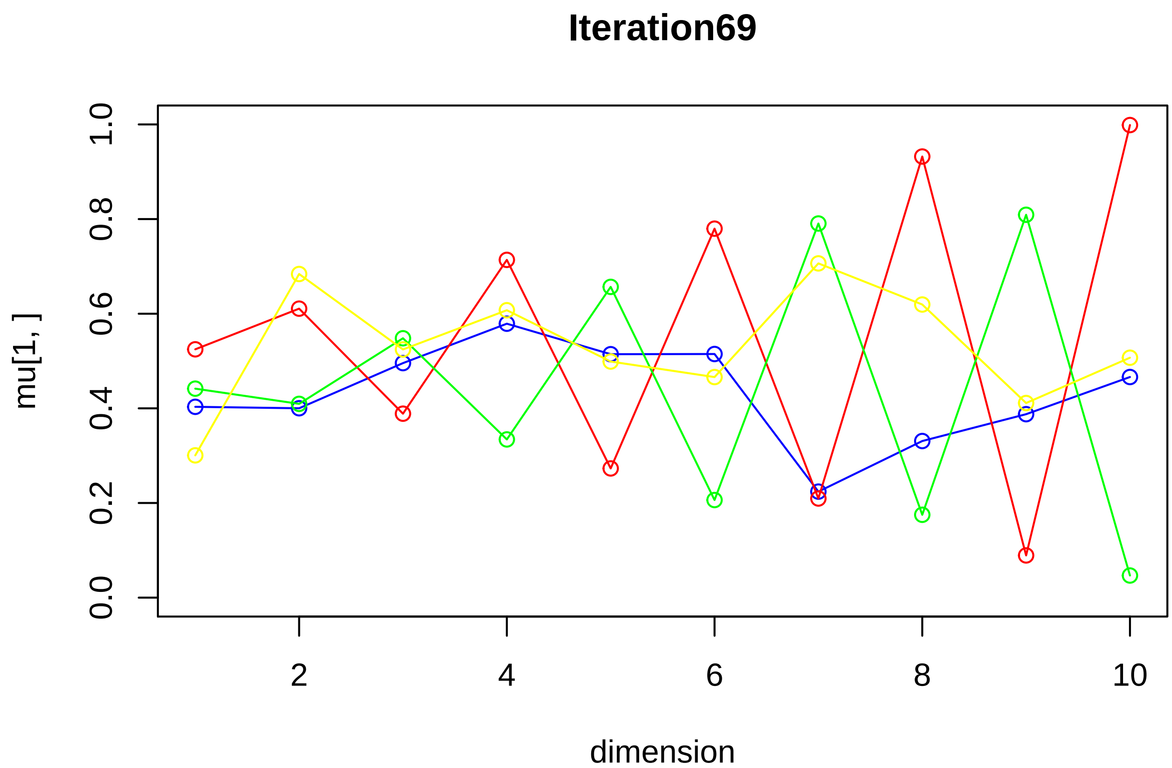

```
## iteration:  66 log likelihood:  -7216.431
```



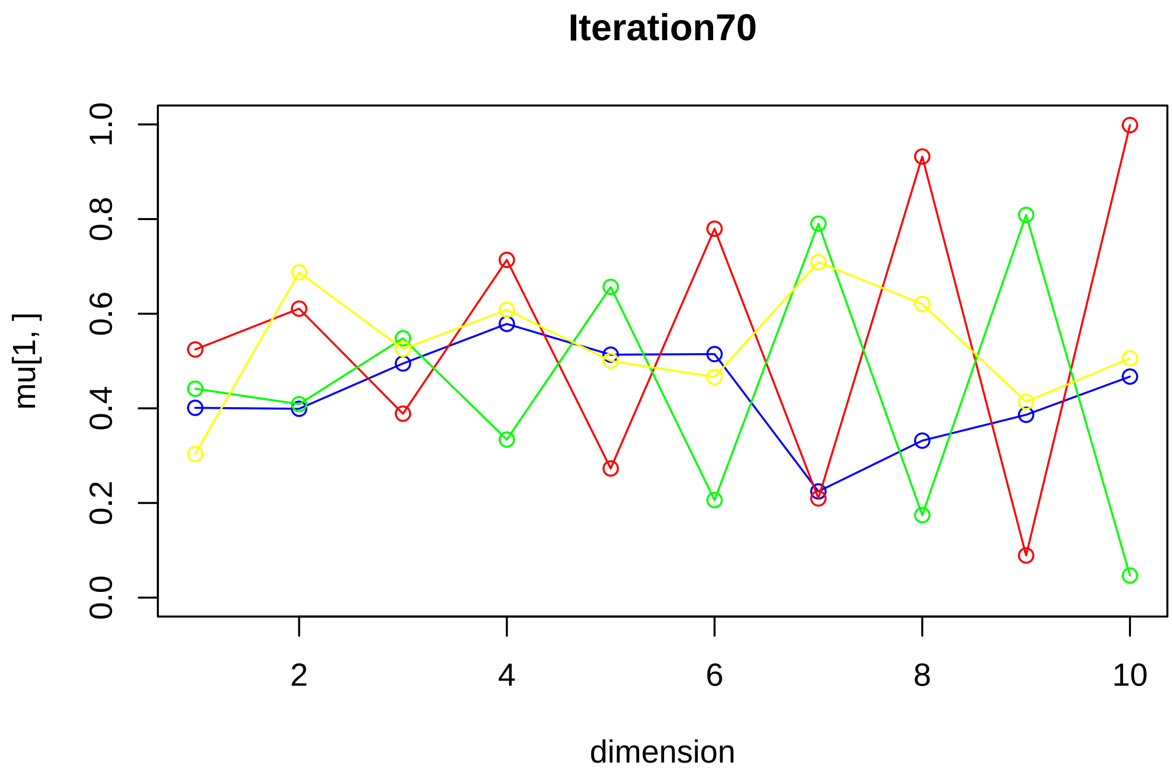

```
## iteration:  67 log likelihood:  -7215.013
```



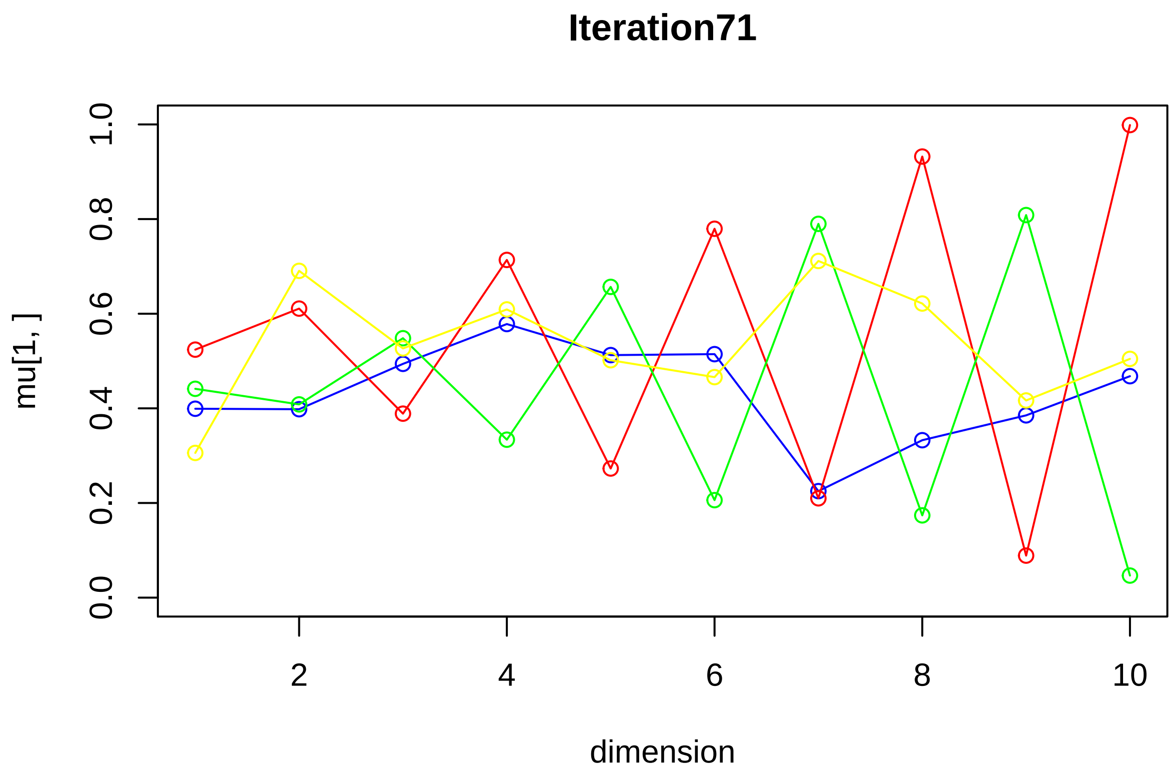

```
## iteration:  68 log likelihood:  -7213.748
```



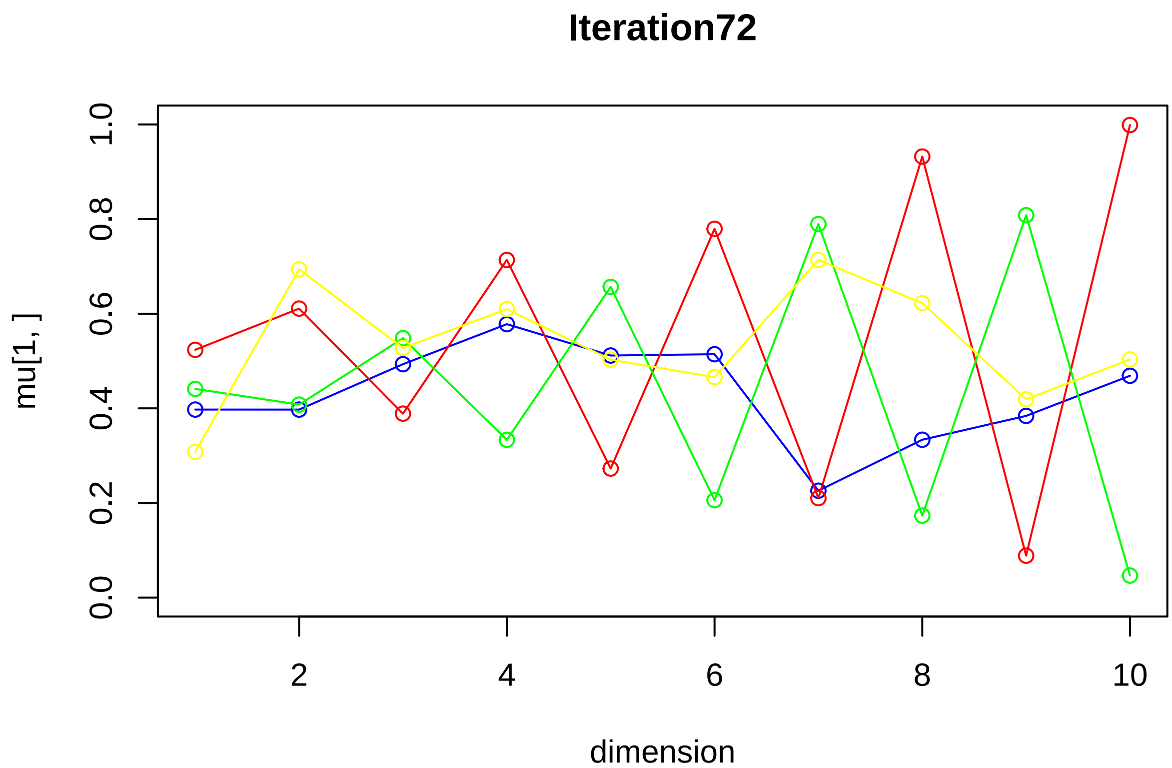

```
## iteration:  69 log likelihood:  -7212.621
```



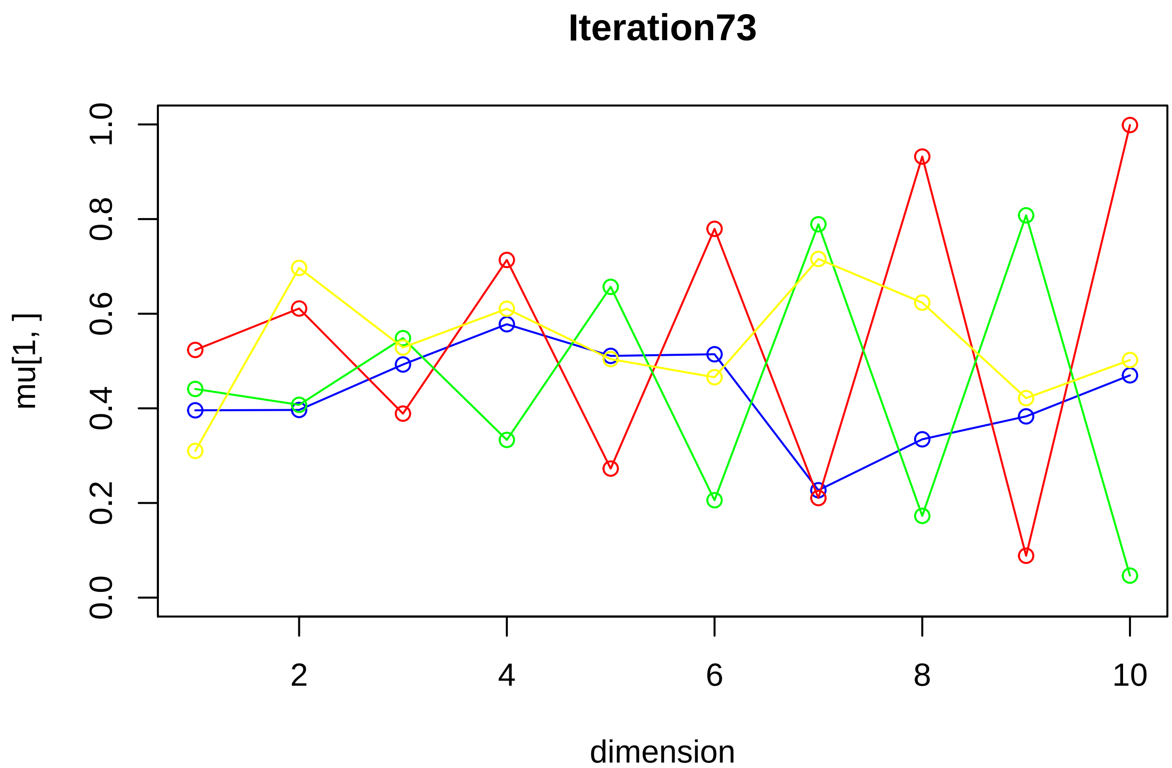

```
## iteration:  70 log likelihood:  -7211.62
```



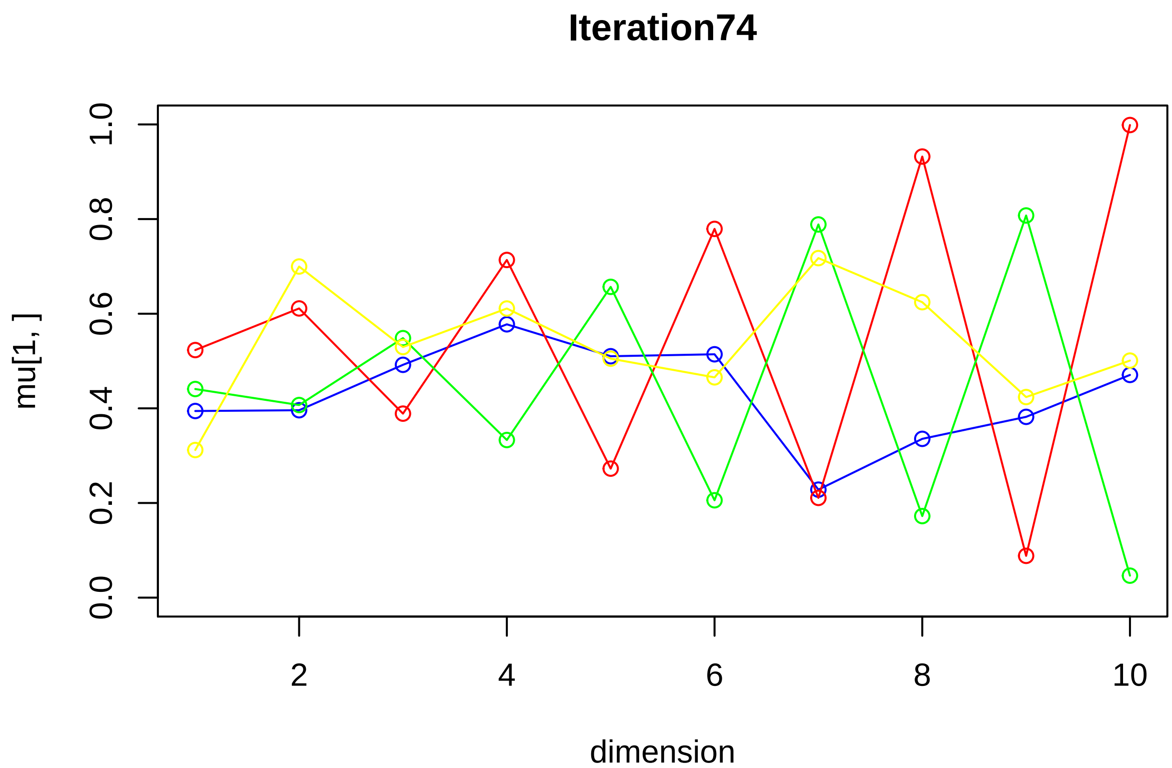

```
## iteration:  71 log likelihood:  -7210.727
```



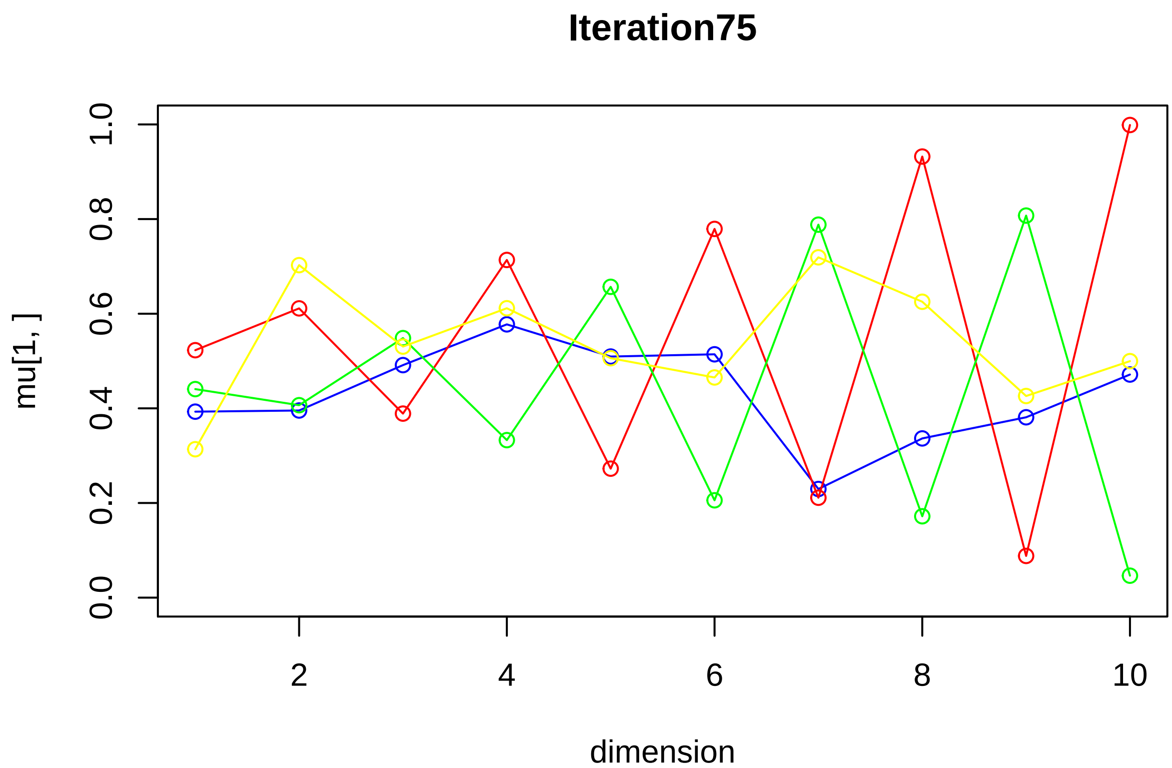

```
## iteration:  72 log likelihood:  -7209.929
```



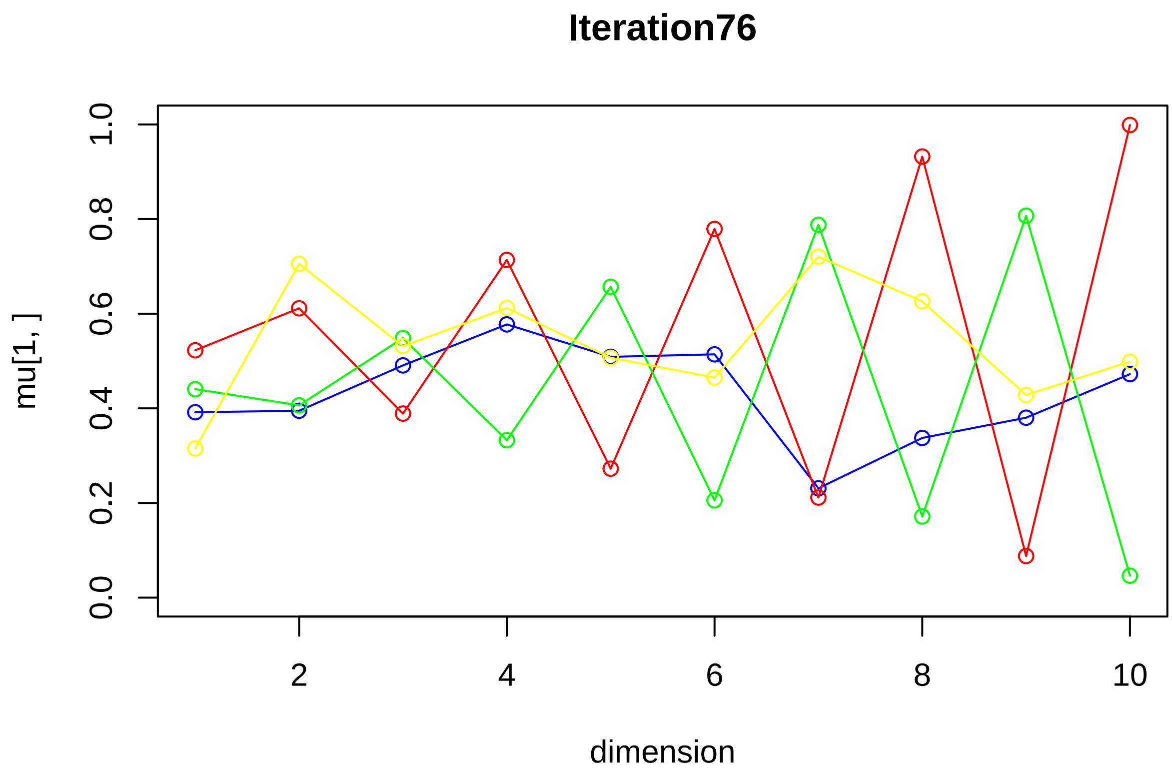

```
## iteration:  73 log likelihood:  -7209.208
```



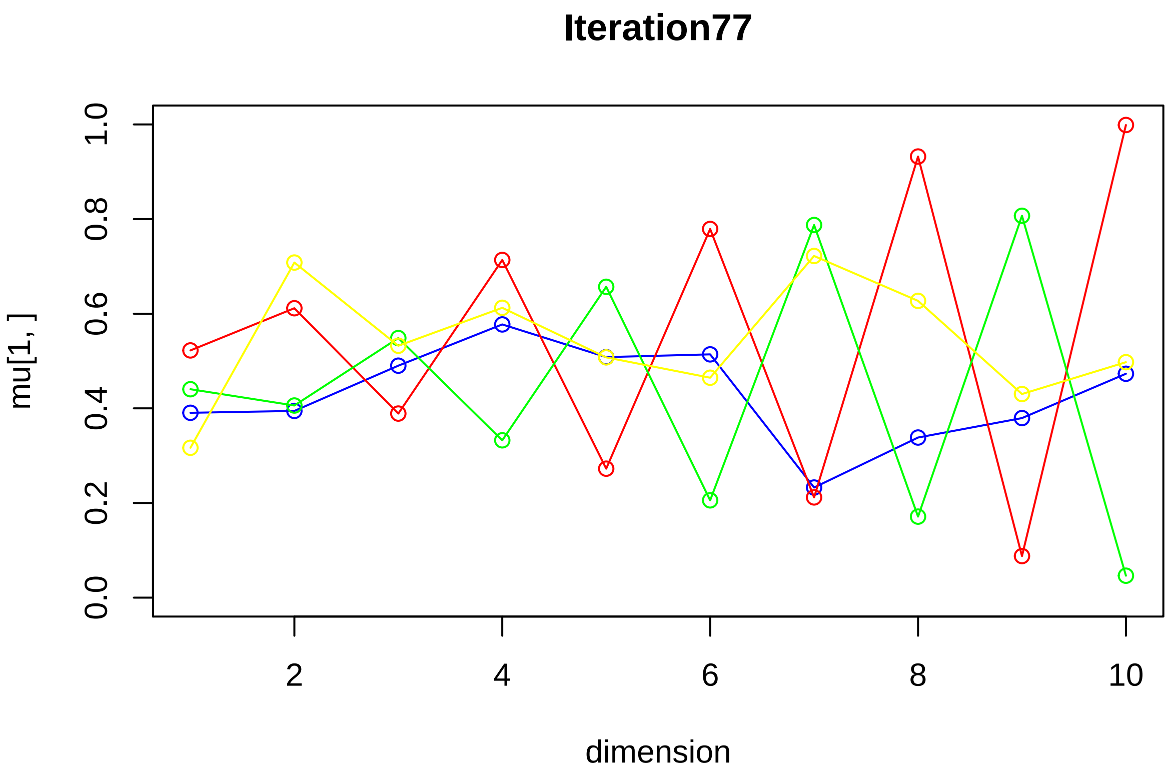

```
## iteration:  74 log likelihood:  -7208.552
```



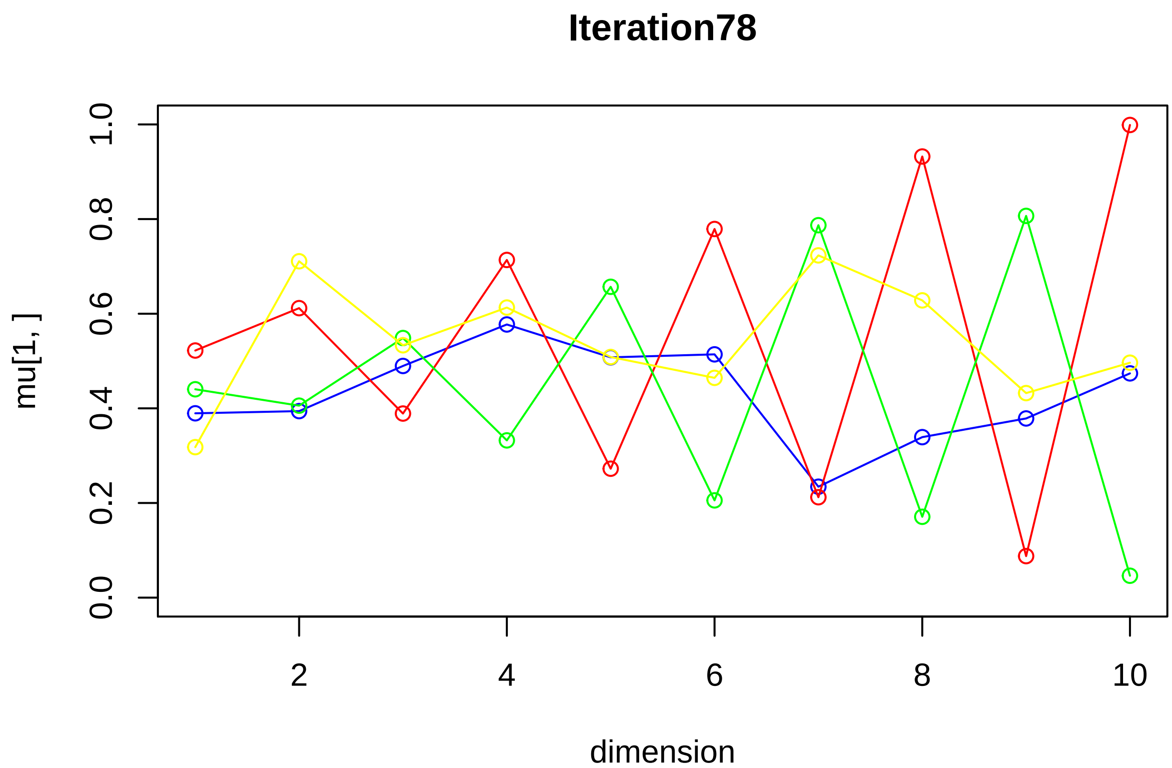

```
## iteration:  75 log likelihood:  -7207.946
```



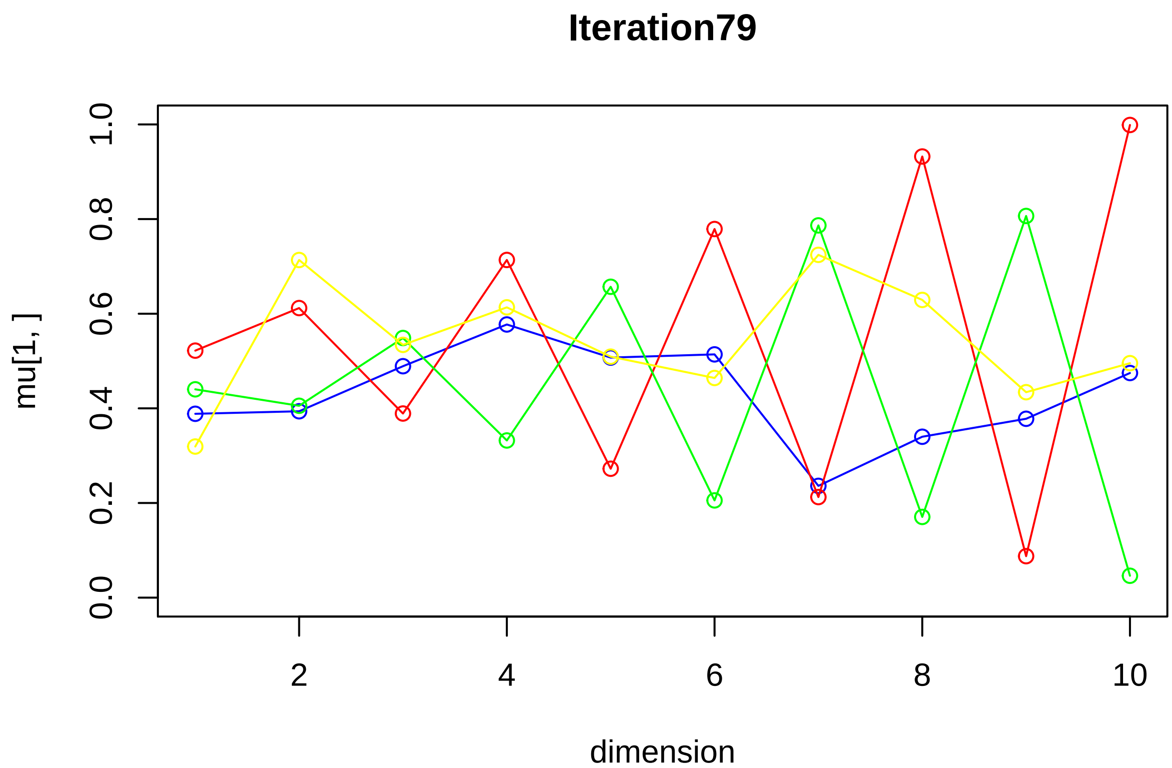

```
## iteration:  76 log likelihood:  -7207.38
```



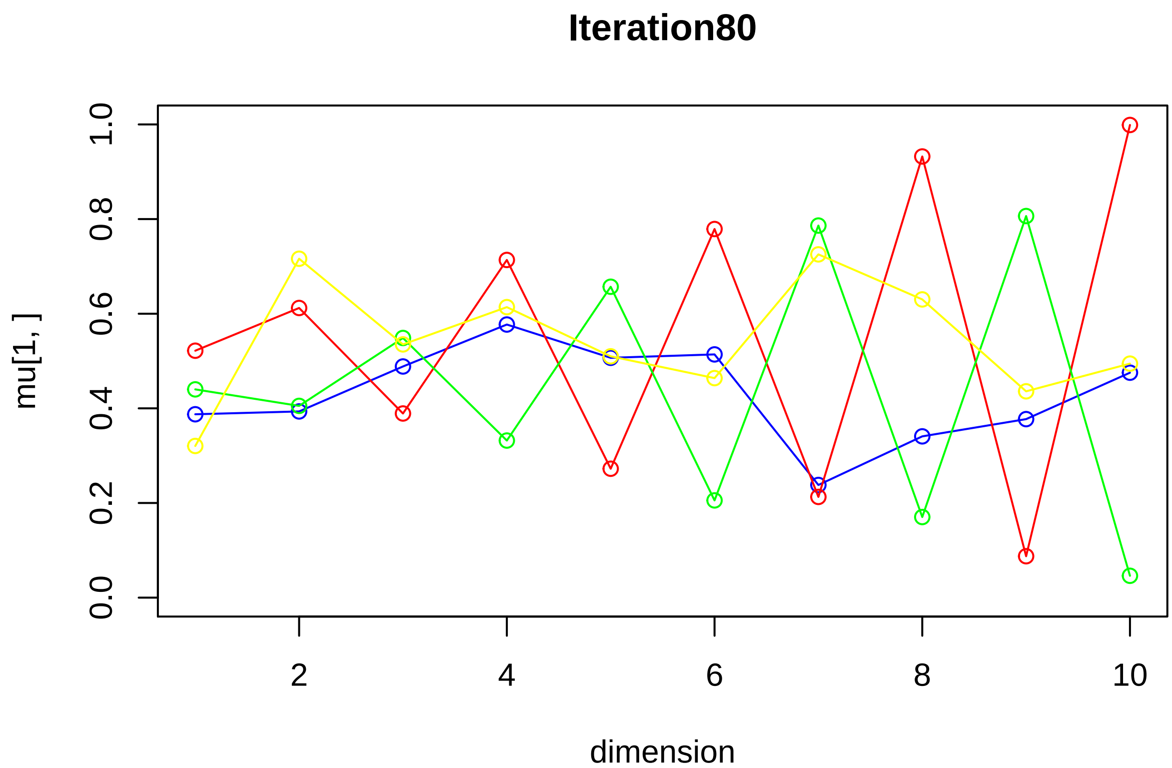

```
## iteration:  77 log likelihood:  -7206.844
```



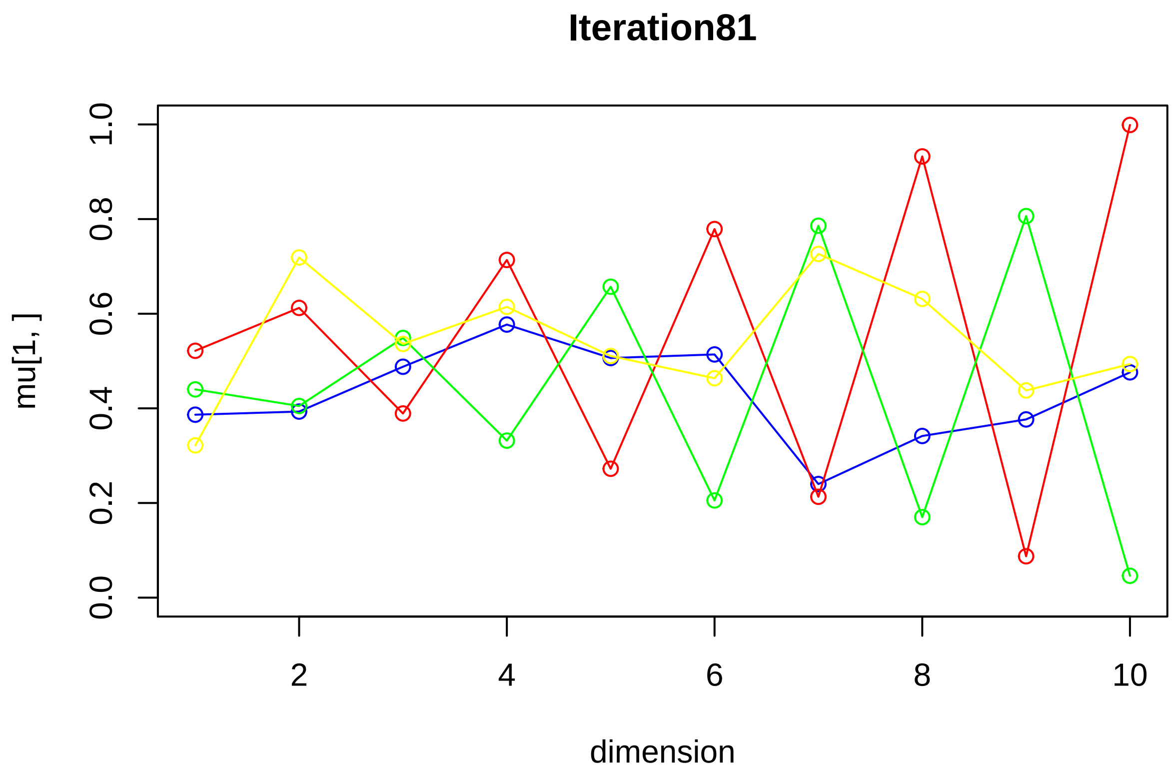

```
## iteration:  78 log likelihood:  -7206.327
```



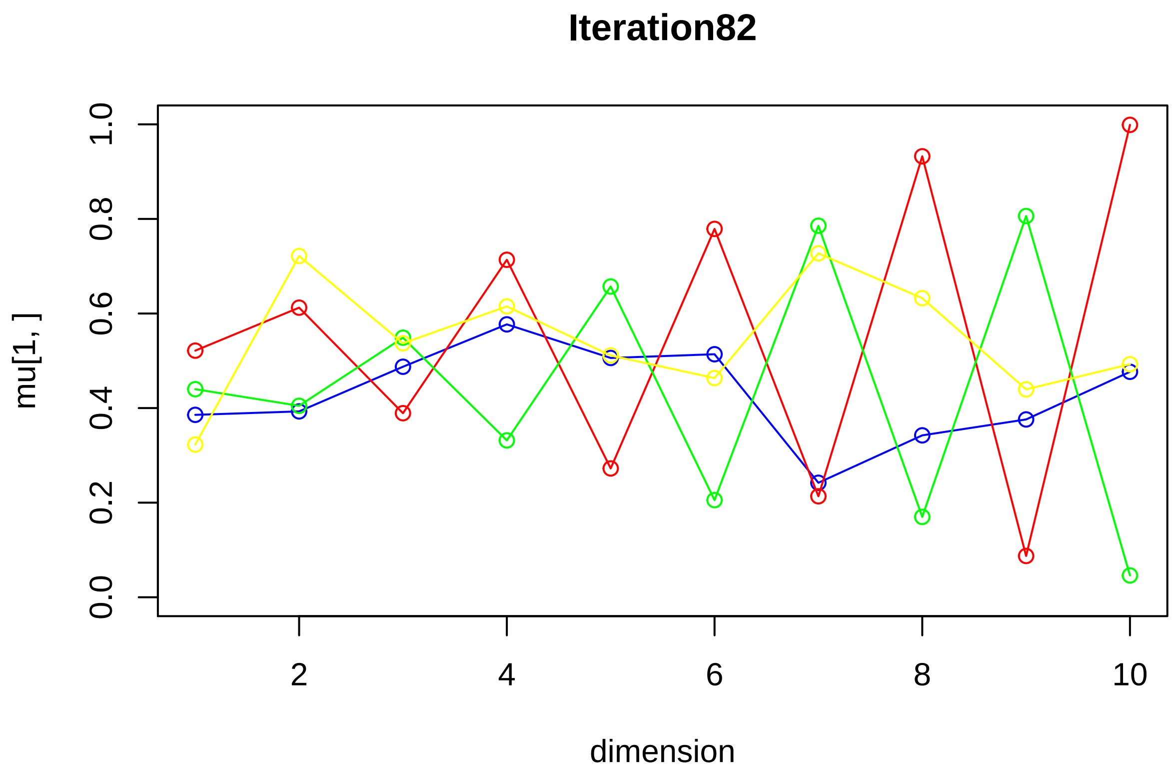

```
## iteration:  79 log likelihood:  -7205.824
```



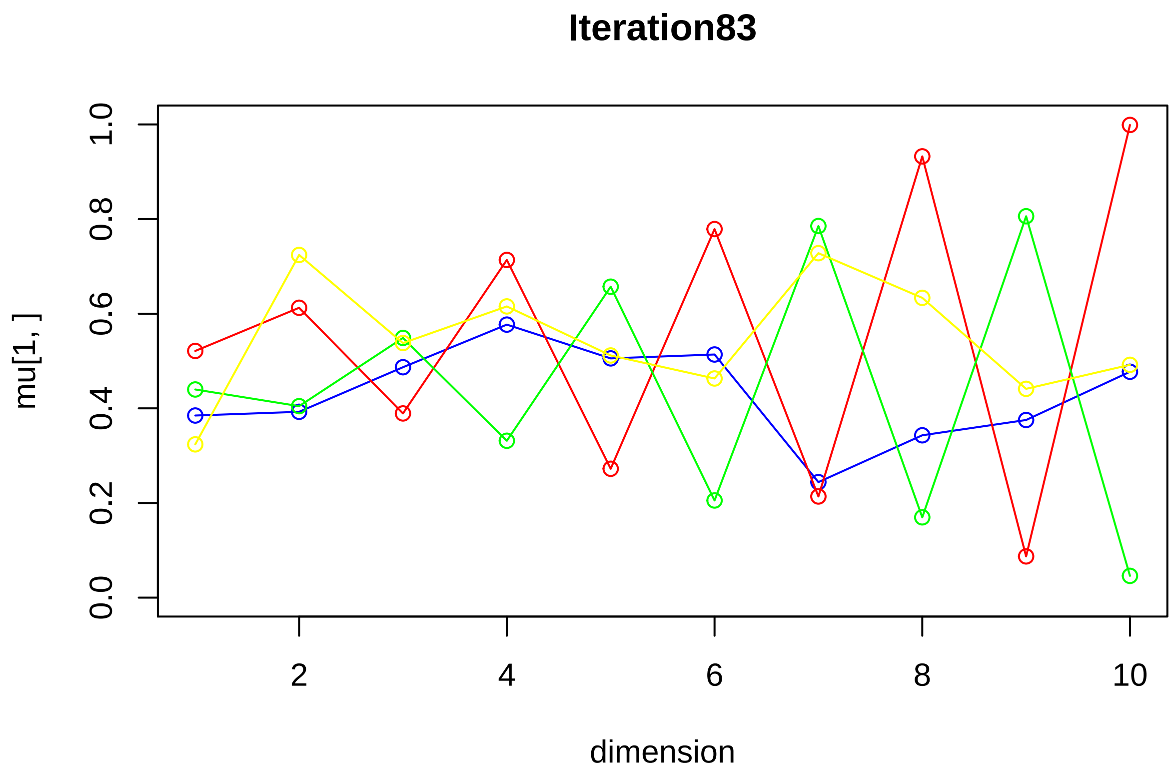

```
## iteration:  80 log likelihood:  -7205.326
```



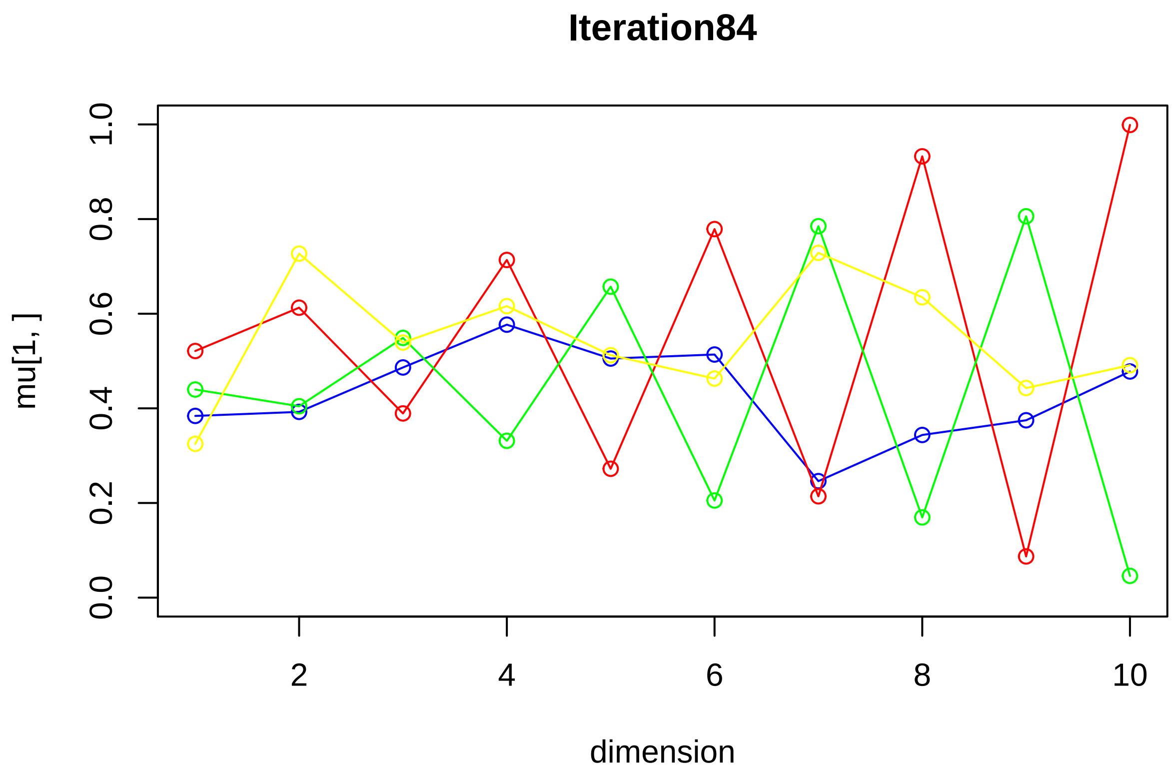

```
## iteration:  81 log likelihood:  -7204.829
```



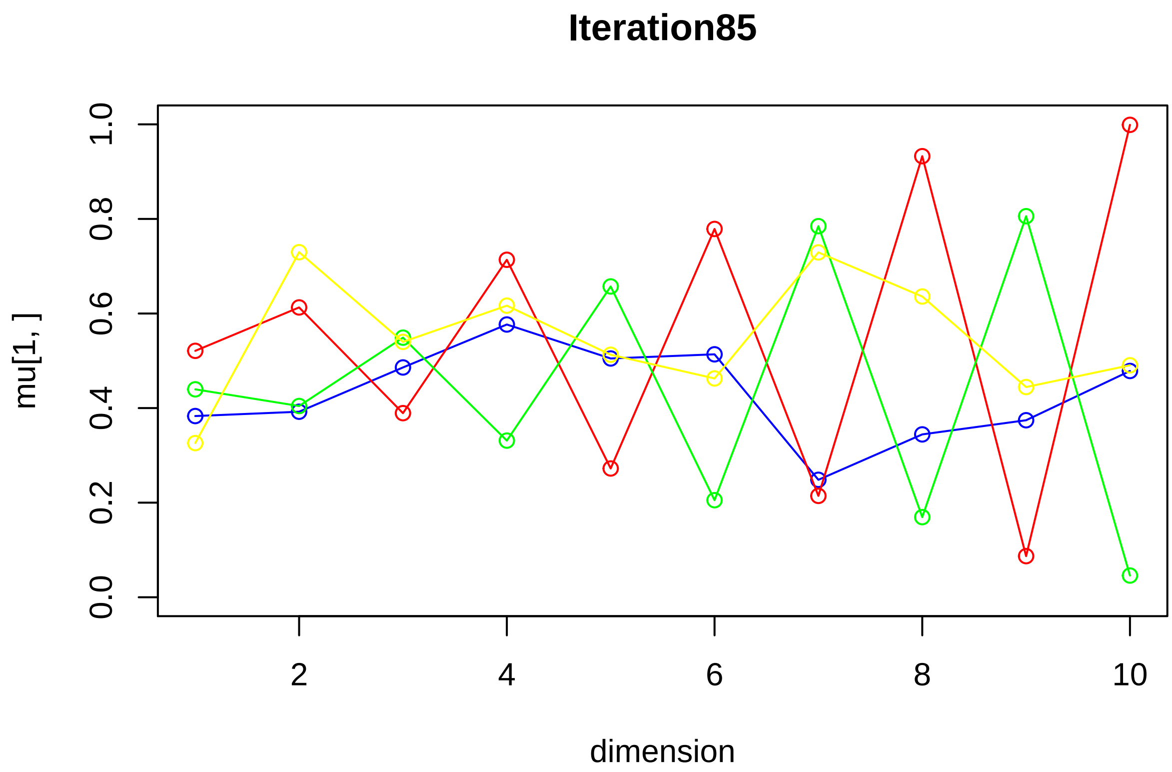

```
## iteration:  82 log likelihood:  -7204.327
```



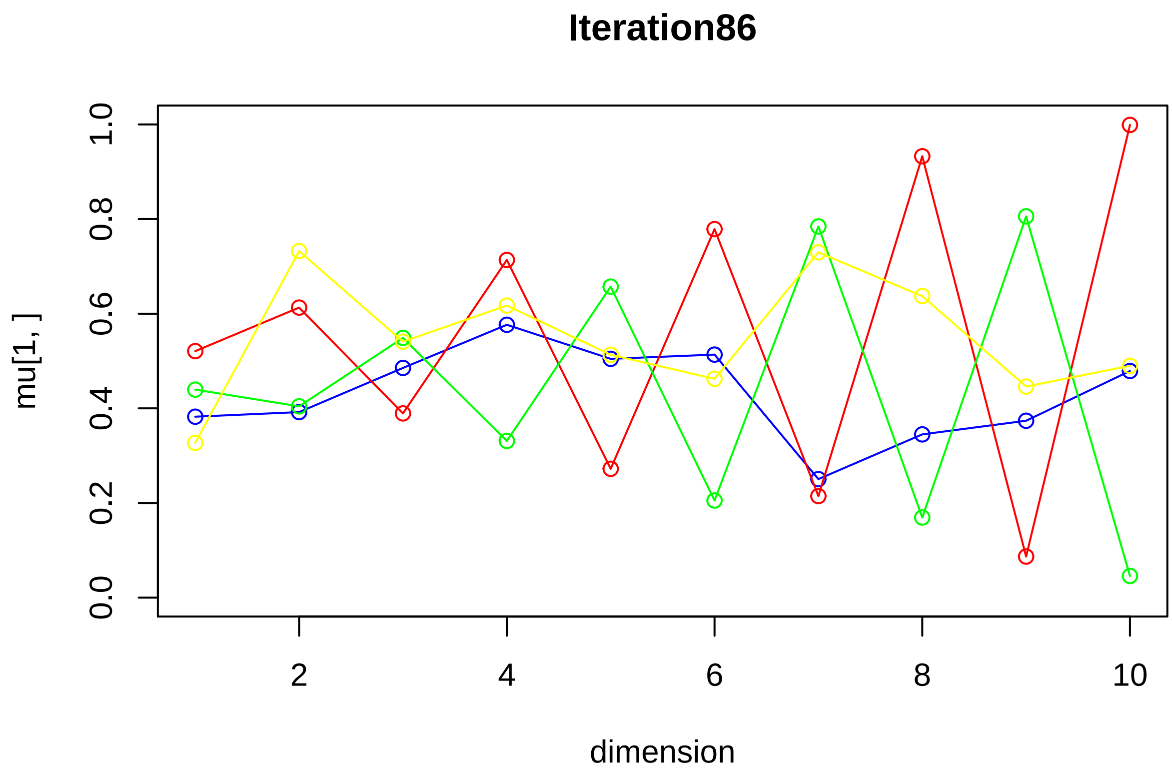

```
## iteration:  83 log likelihood:  -7203.816
```



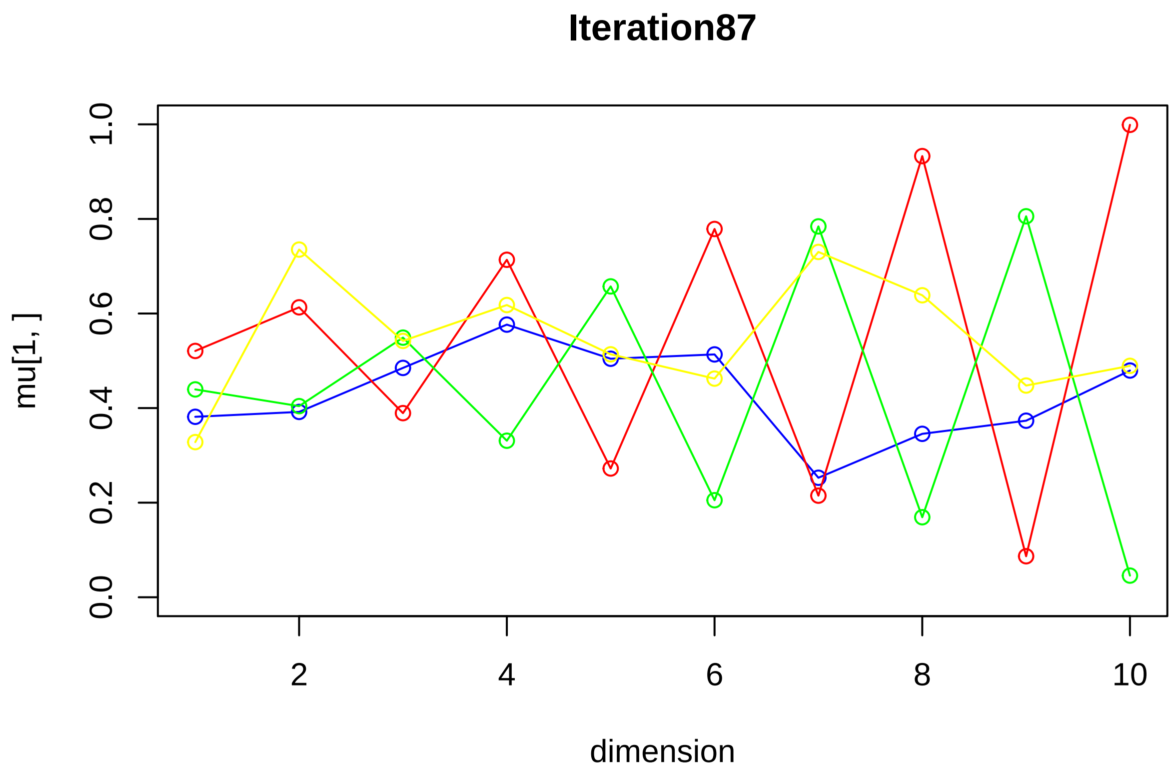

```
## iteration:  84 log likelihood:  -7203.294
```



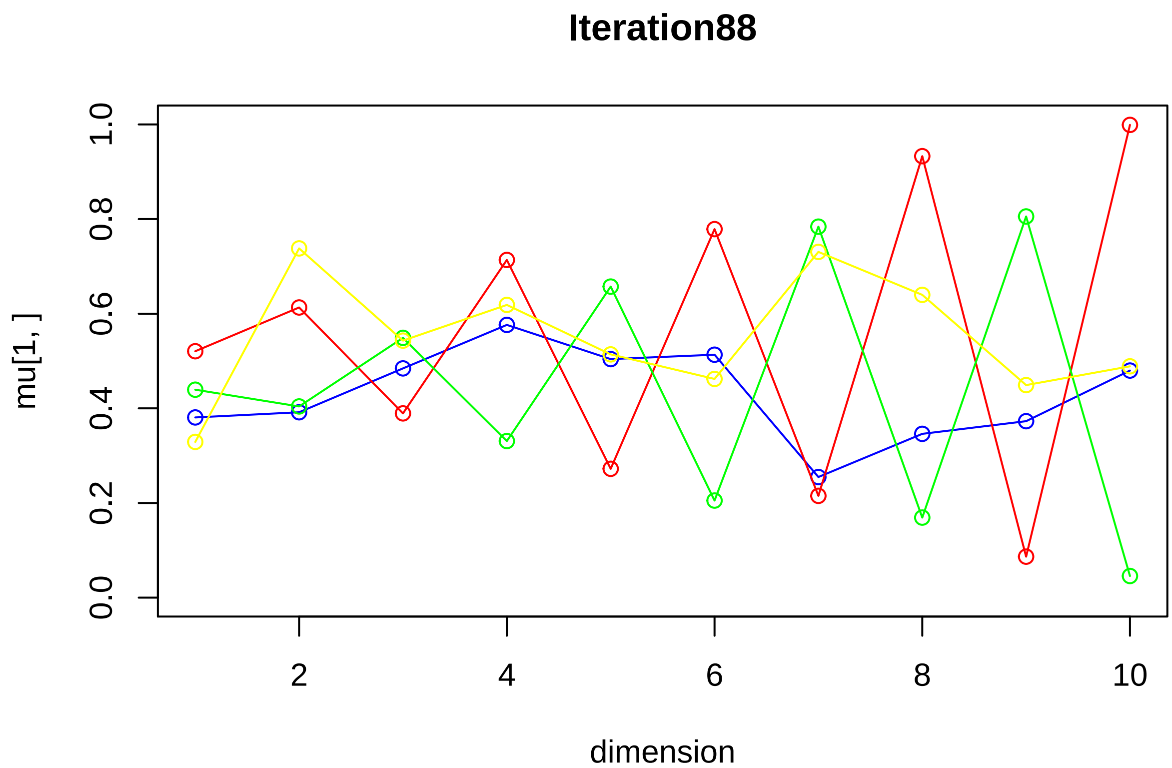

```
## iteration:  85 log likelihood:  -7202.756
```



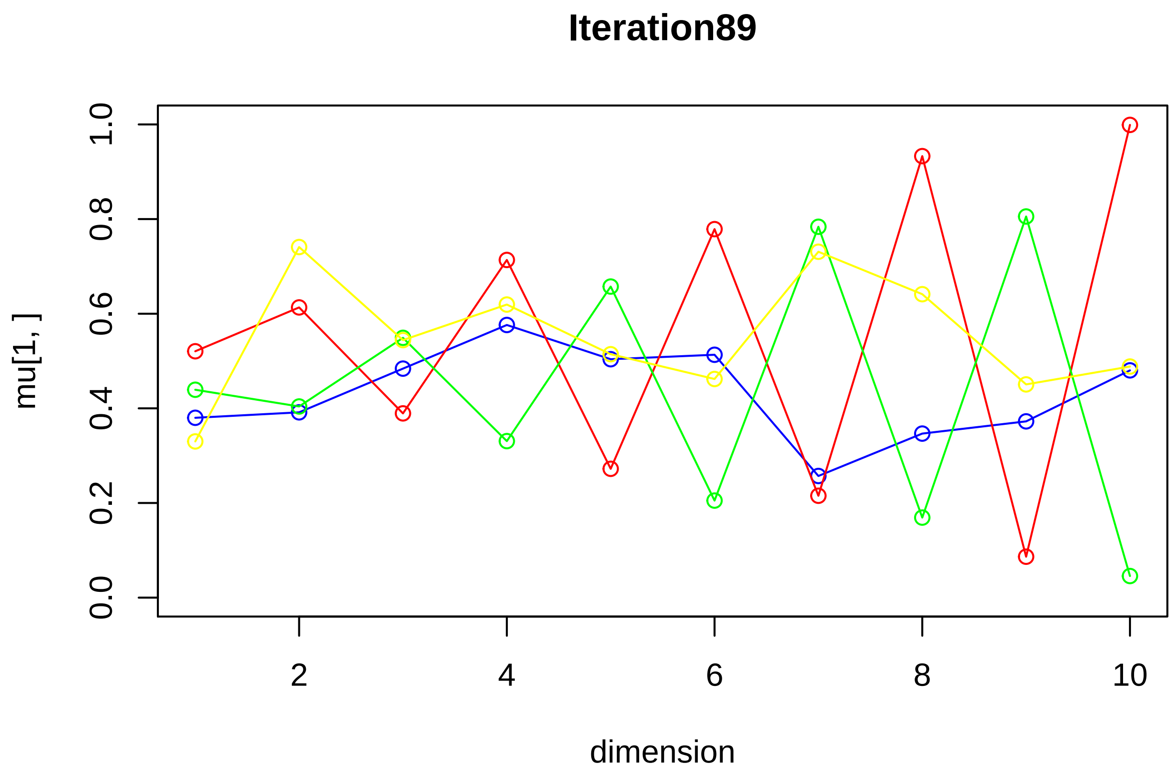

```
## iteration:  86 log likelihood:  -7202.201
```



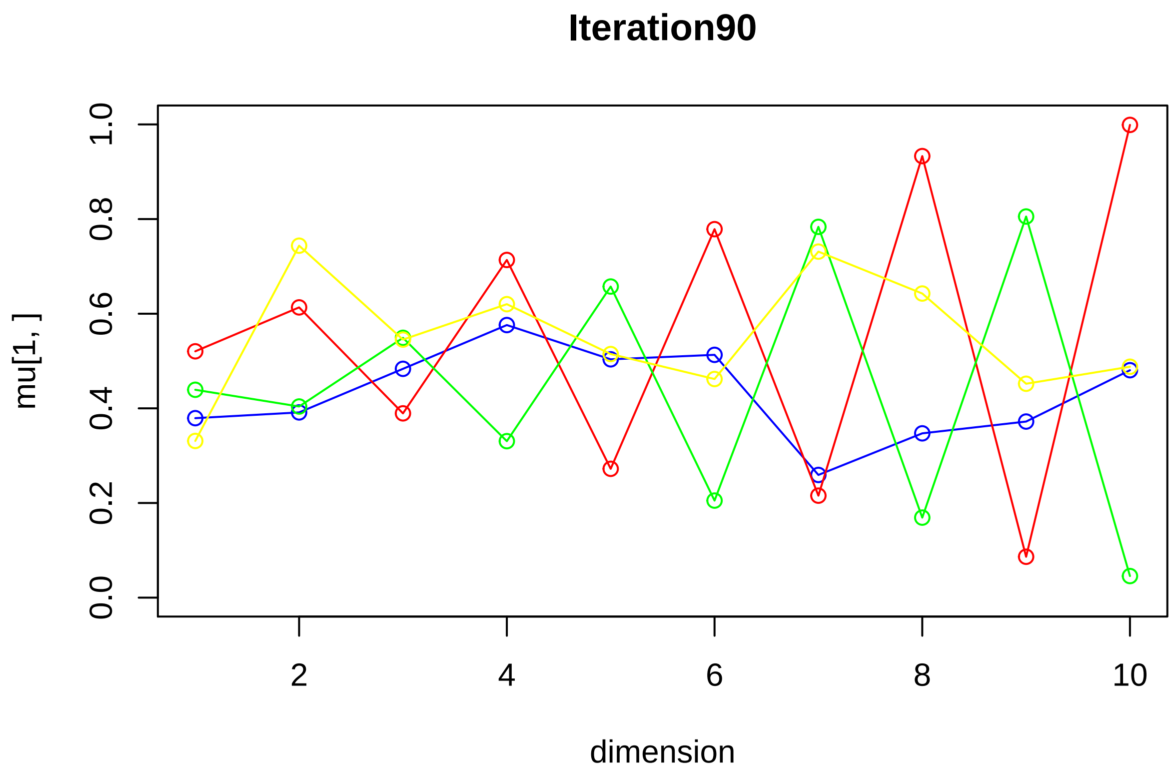

```
## iteration:  87 log likelihood:  -7201.627
```



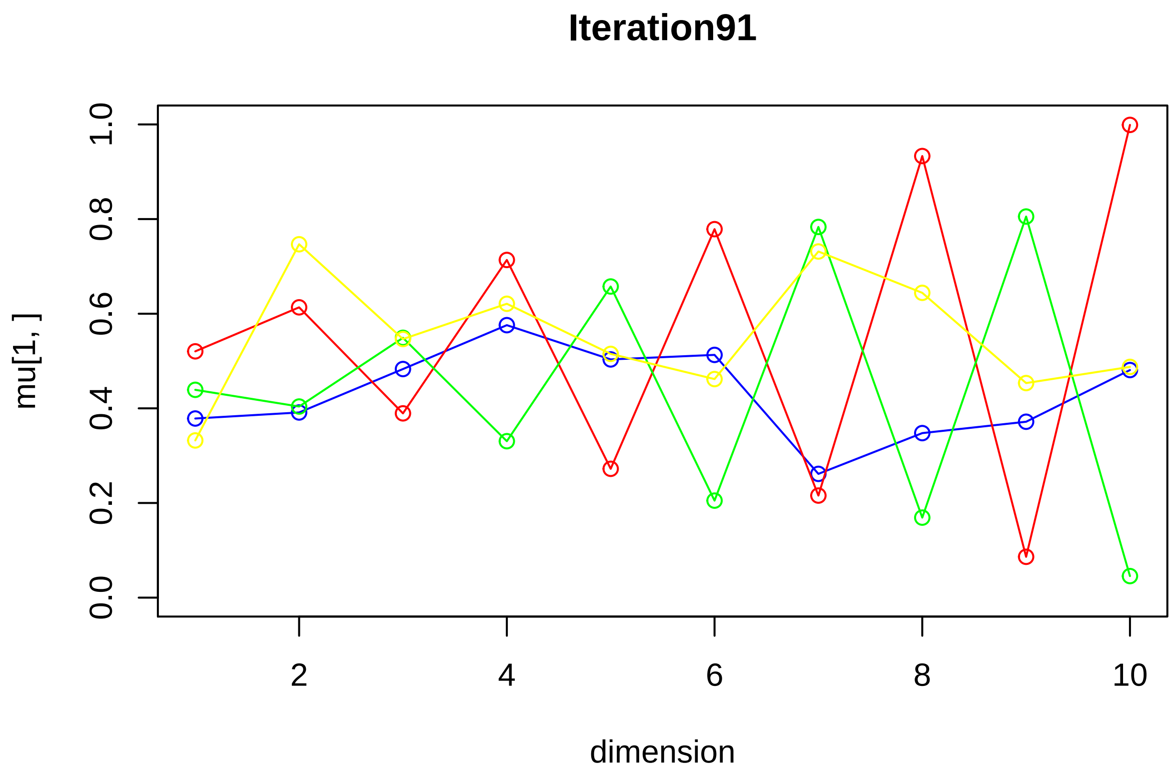

```
## iteration:  88 log likelihood:  -7201.032
```



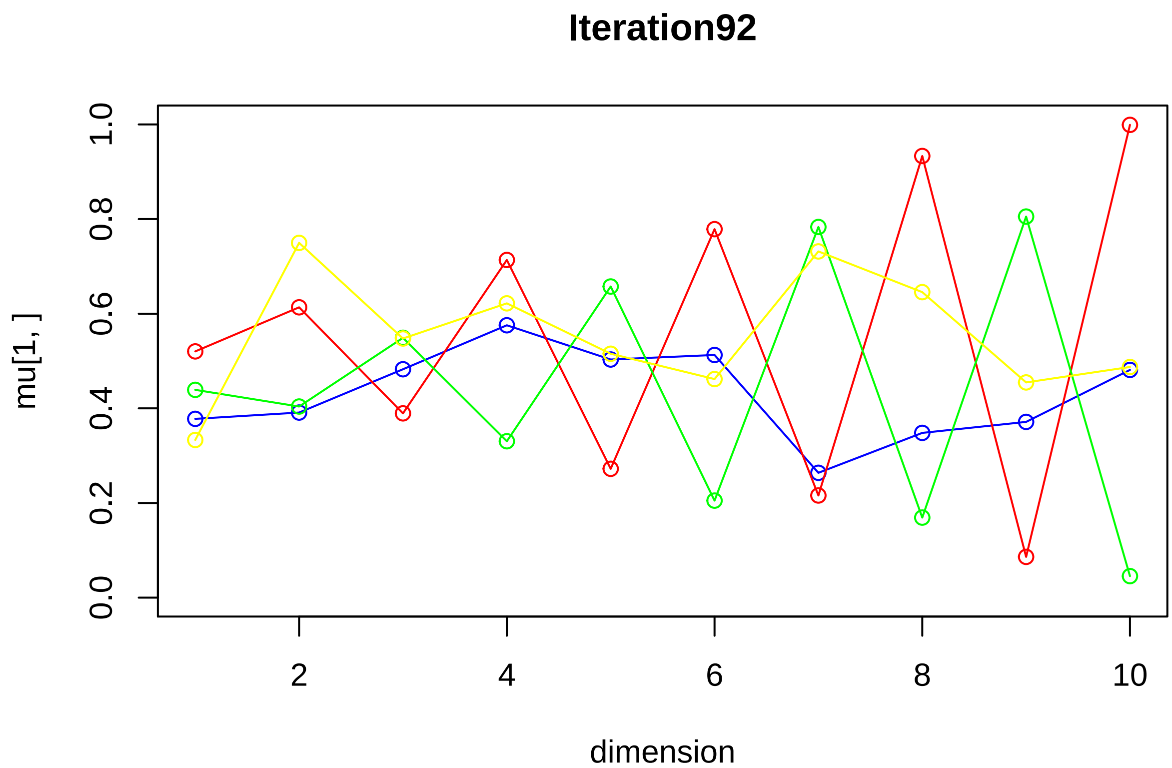

```
## iteration:  89 log likelihood:  -7200.414
```



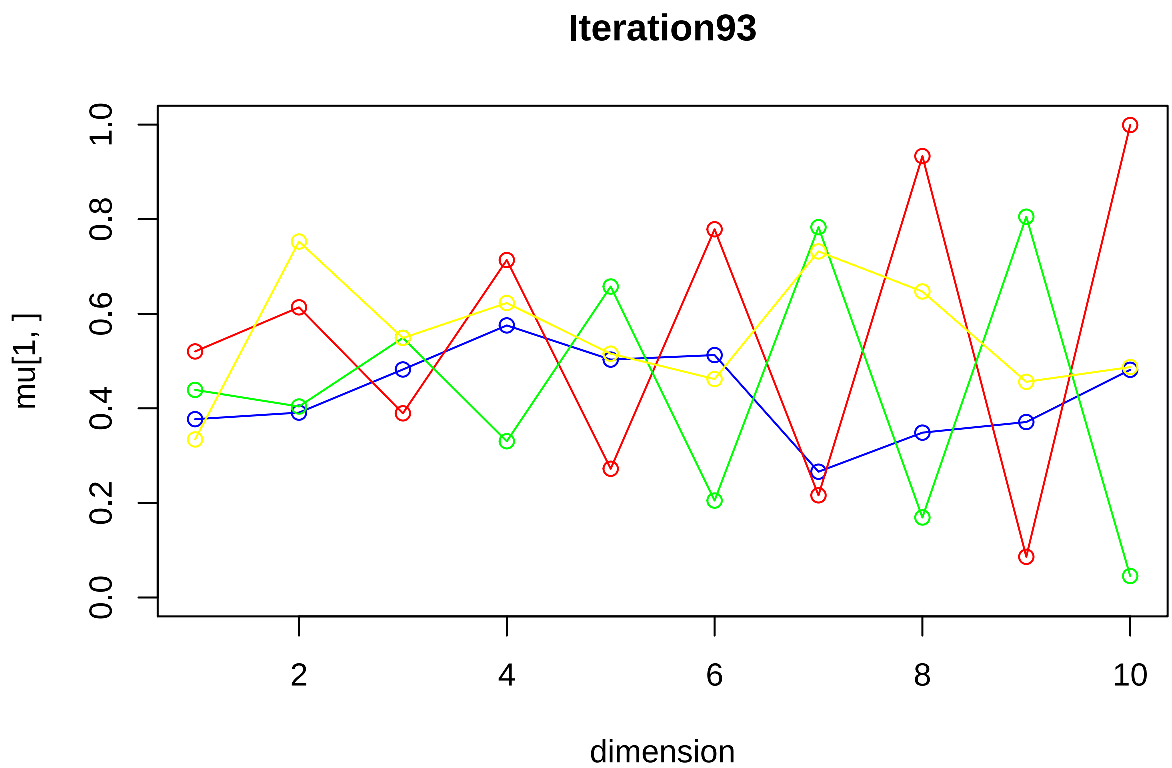

```
## iteration:  90 log likelihood:  -7199.773
```



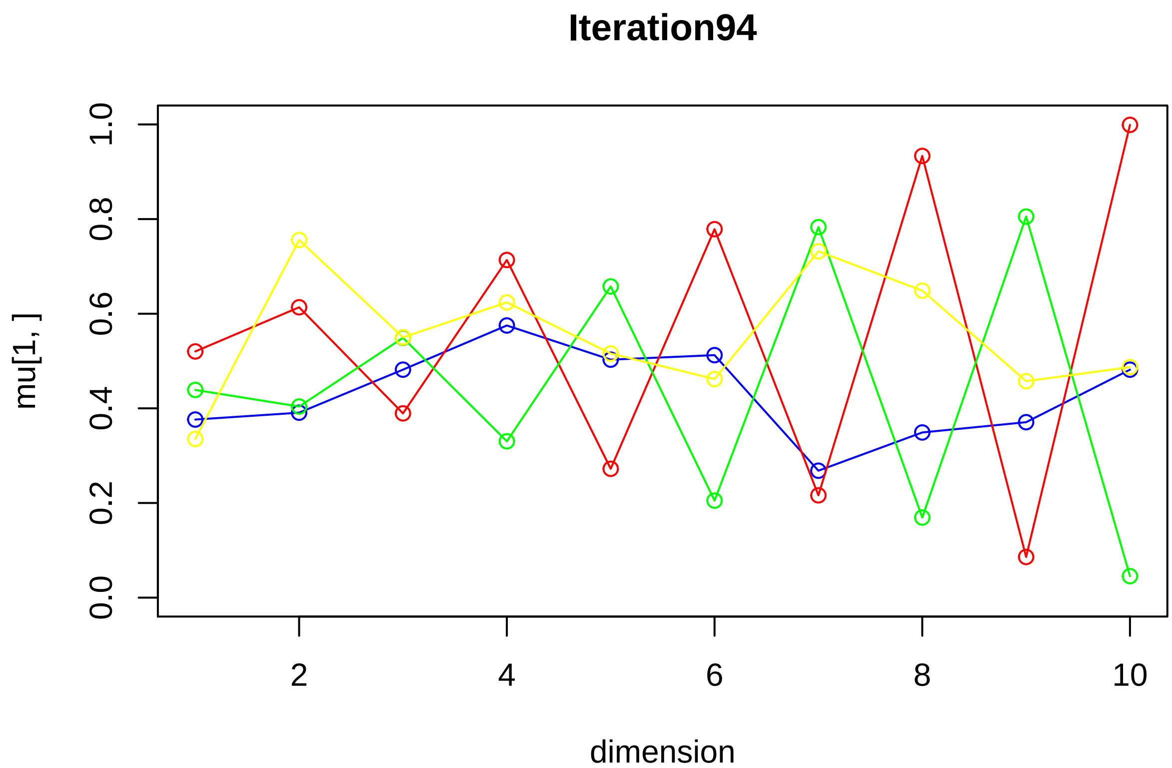

```
## iteration:  91 log likelihood:  -7199.107
```



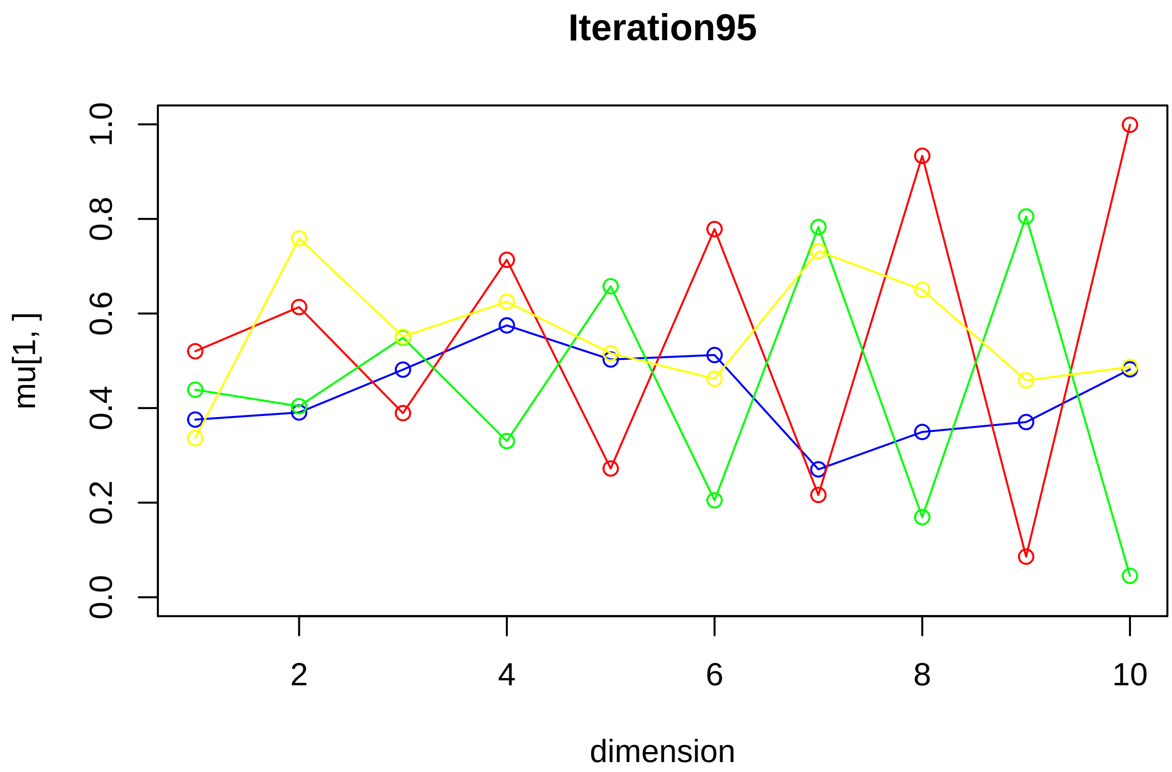

```
## iteration:  92 log likelihood:  -7198.416
```



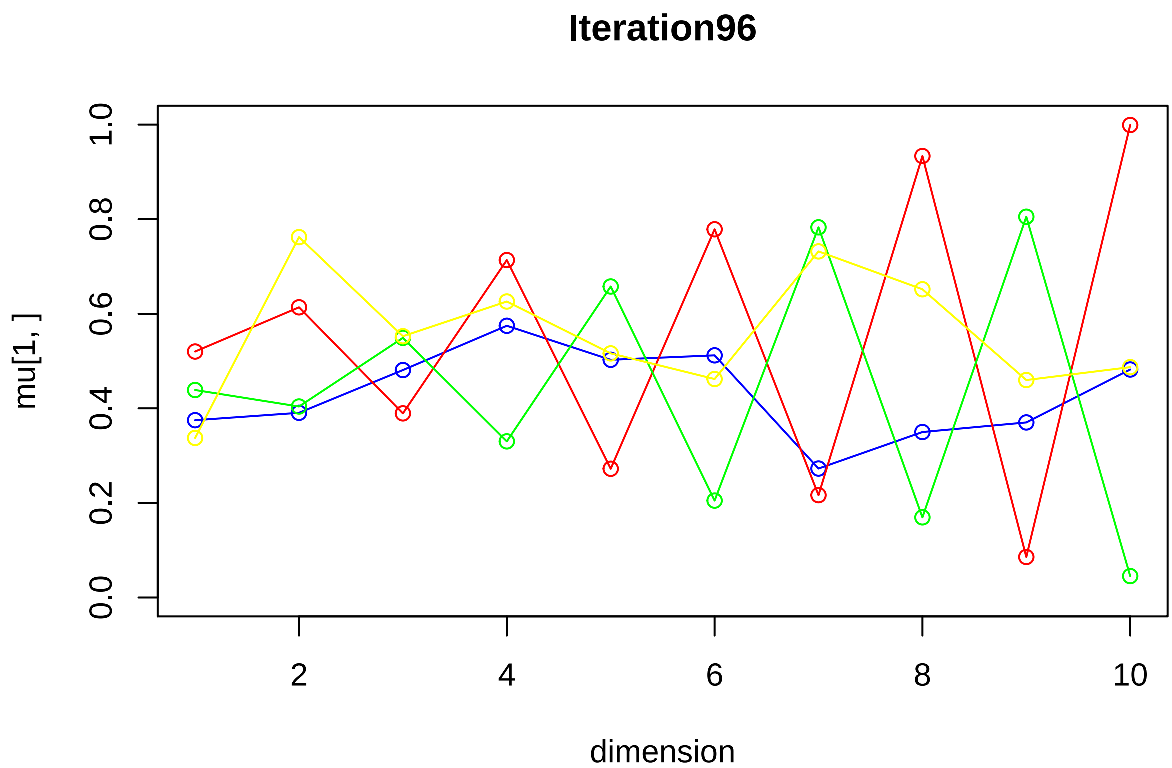

```
## iteration:  93 log likelihood:  -7197.7
```



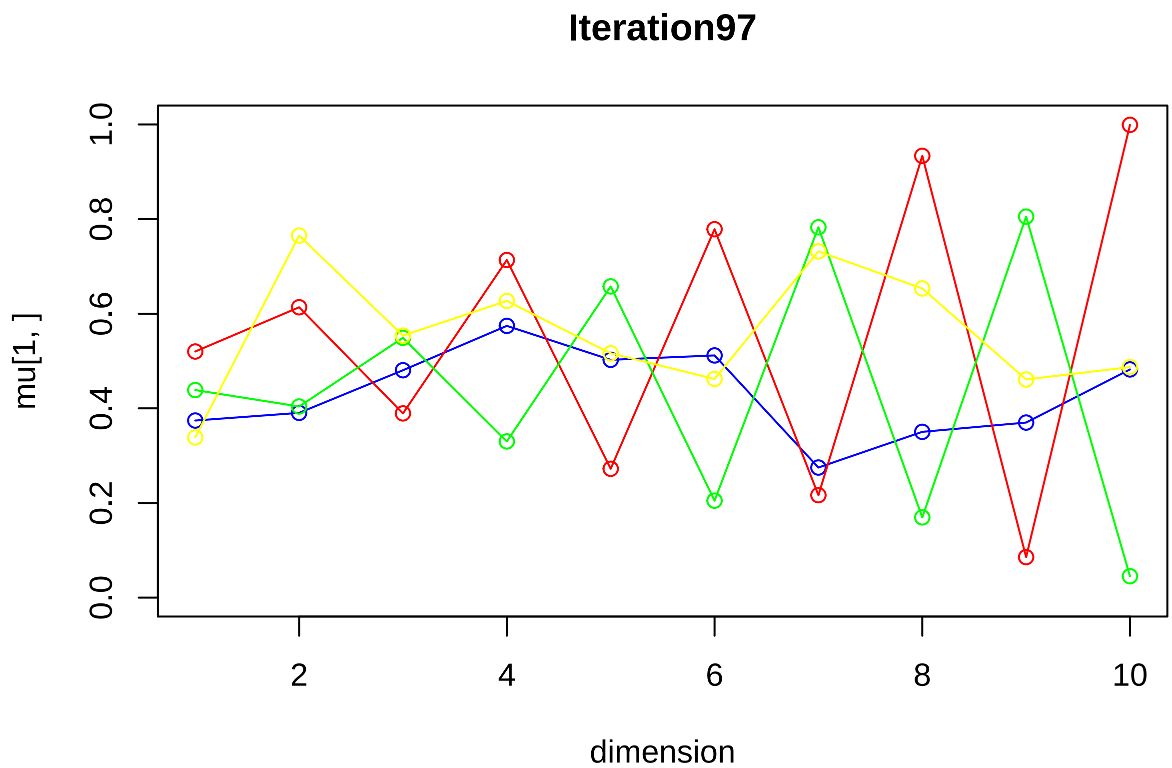

```
## iteration:  94 log likelihood:  -7196.957
```



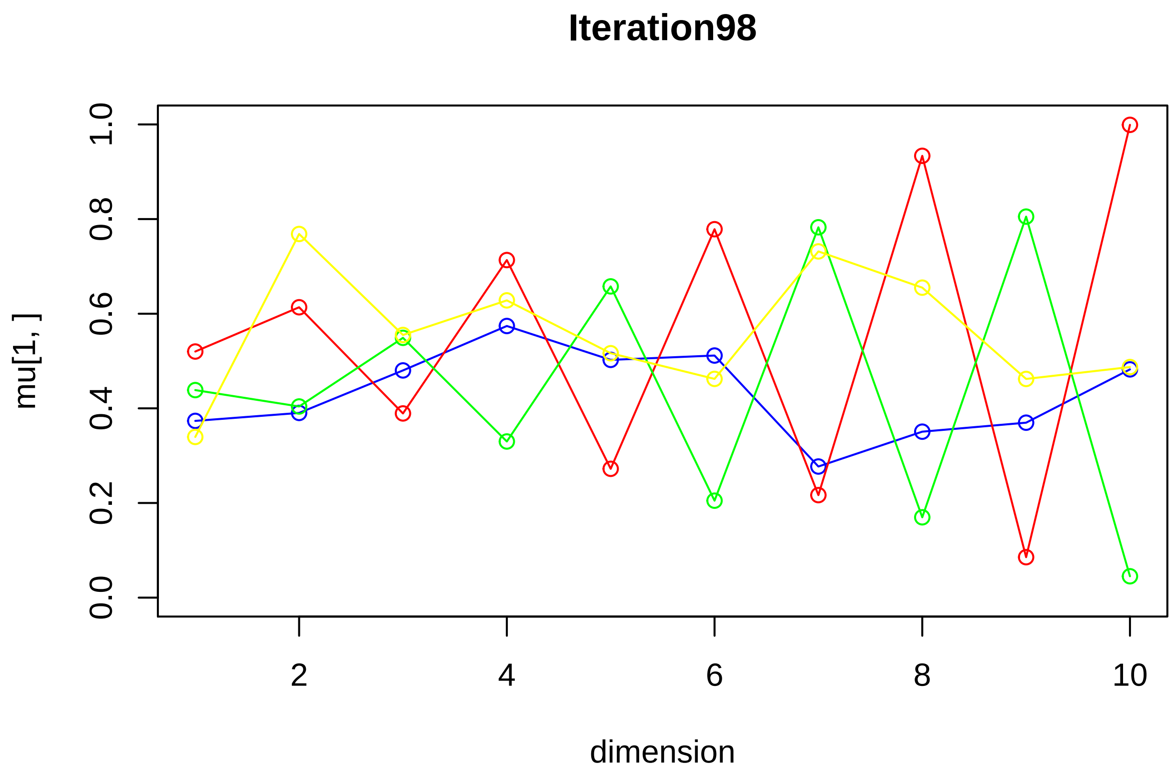

```
## iteration:  95 log likelihood:  -7196.188
```



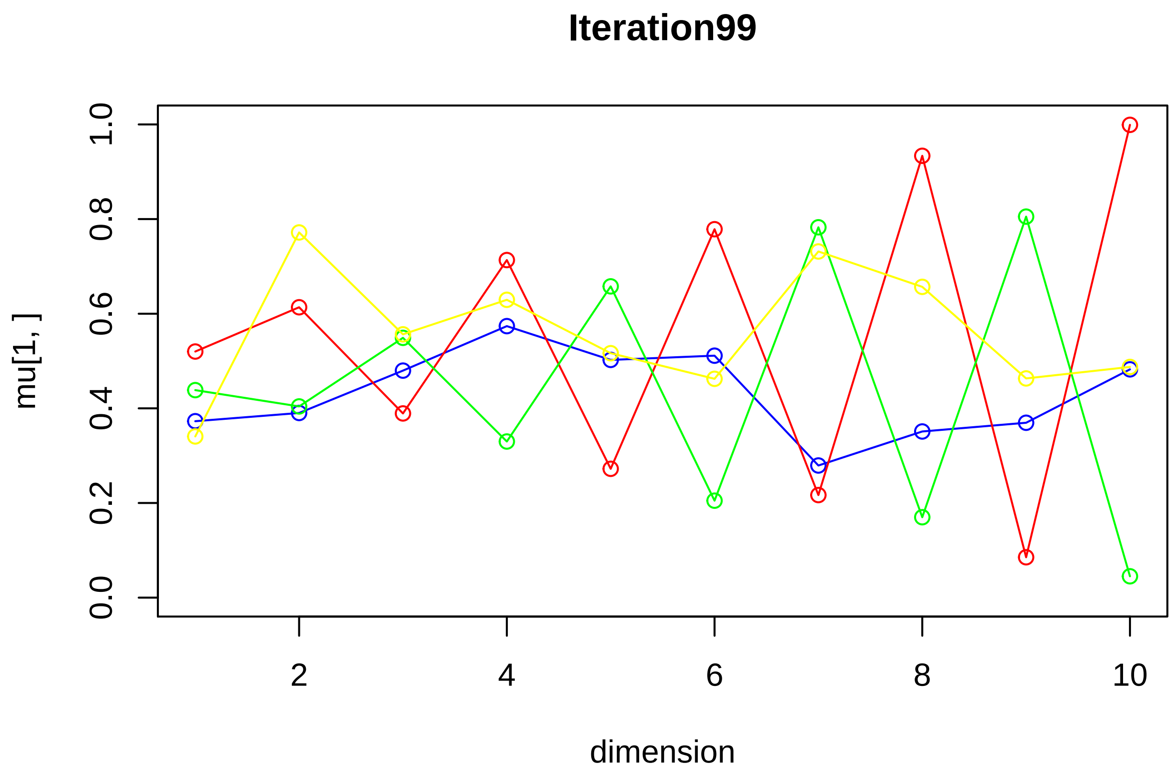

```
## iteration:  96 log likelihood:  -7195.392
```



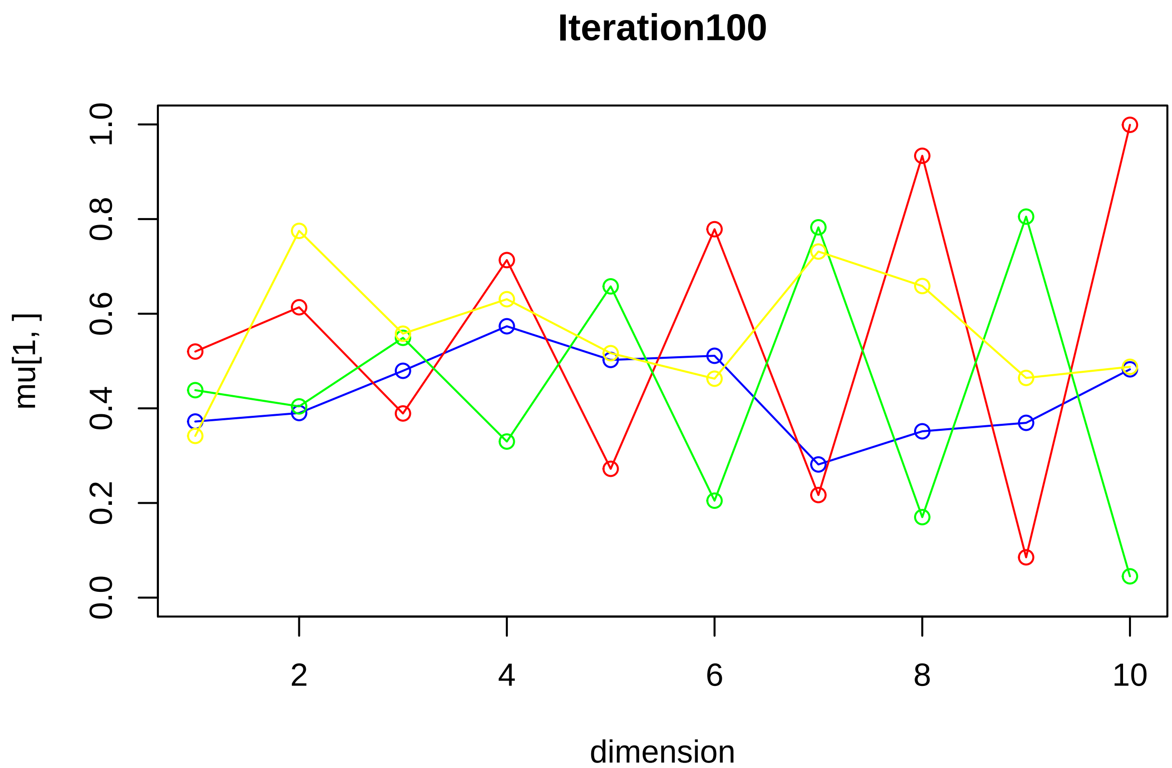

```
## iteration:  97 log likelihood:  -7194.57
```



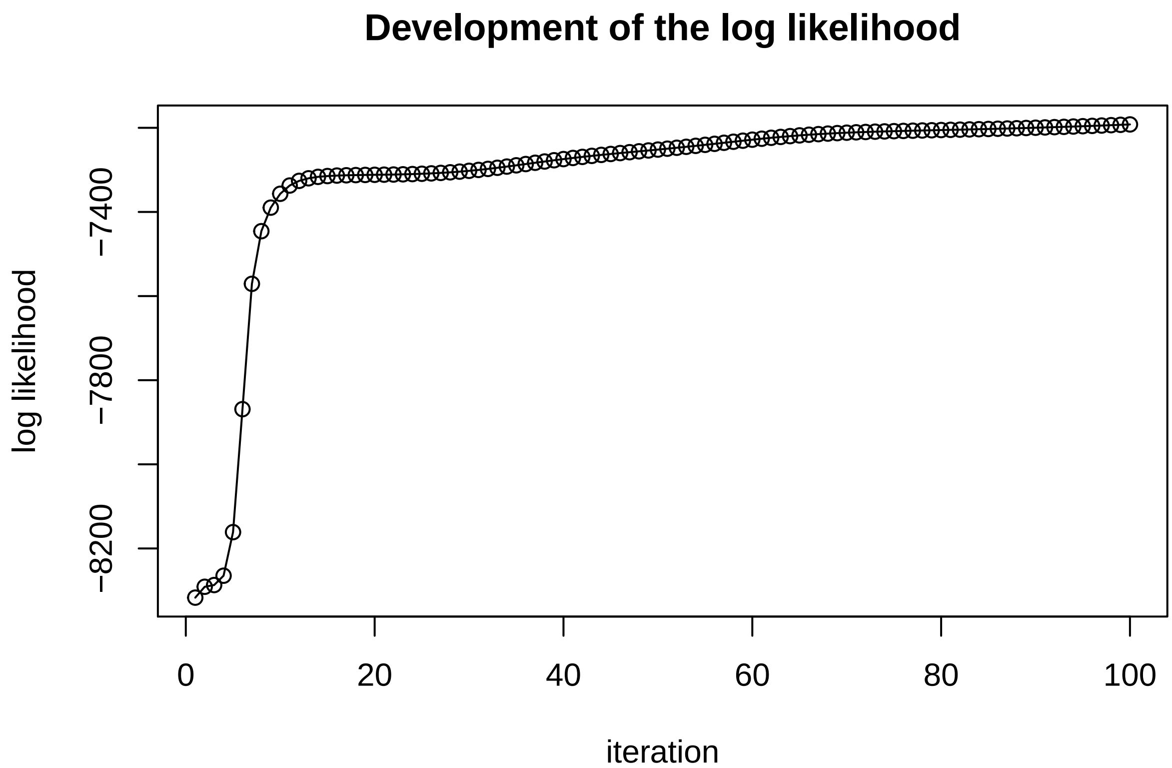

```
## iteration:  98 log likelihood:  -7193.722
```




```
## iteration:  99 log likelihood:  -7192.847
```




```
## iteration:  100 log likelihood:  -7191.946
```




```
## $pi
## [1] 0.2880470 0.2533761 0.2933710 0.1652060
##
## $mu
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.3714855 0.3899958 0.4790260 0.5731886 0.5022651 0.5108478 0.2835691
## [2,] 0.5199997 0.6135841 0.3891214 0.7132736 0.2722448 0.7785461 0.2168891
## [3,] 0.4383456 0.4042497 0.5489526 0.3298363 0.6578057 0.2049012 0.7825505
## [4,] 0.3428531 0.7784238 0.5591637 0.6319621 0.5167044 0.4629058 0.7311279
##           [,8]       [,9]      [,10]
## [1,] 0.3519184 0.36924863 0.48252239
## [2,] 0.9337959 0.08504806 0.99916297
## [3,] 0.1703330 0.80517853 0.04500171
## [4,] 0.6601375 0.46532151 0.48814639
##
## $logLikelihoodDevelopment
## NULL
```

Function for EM Algorithm

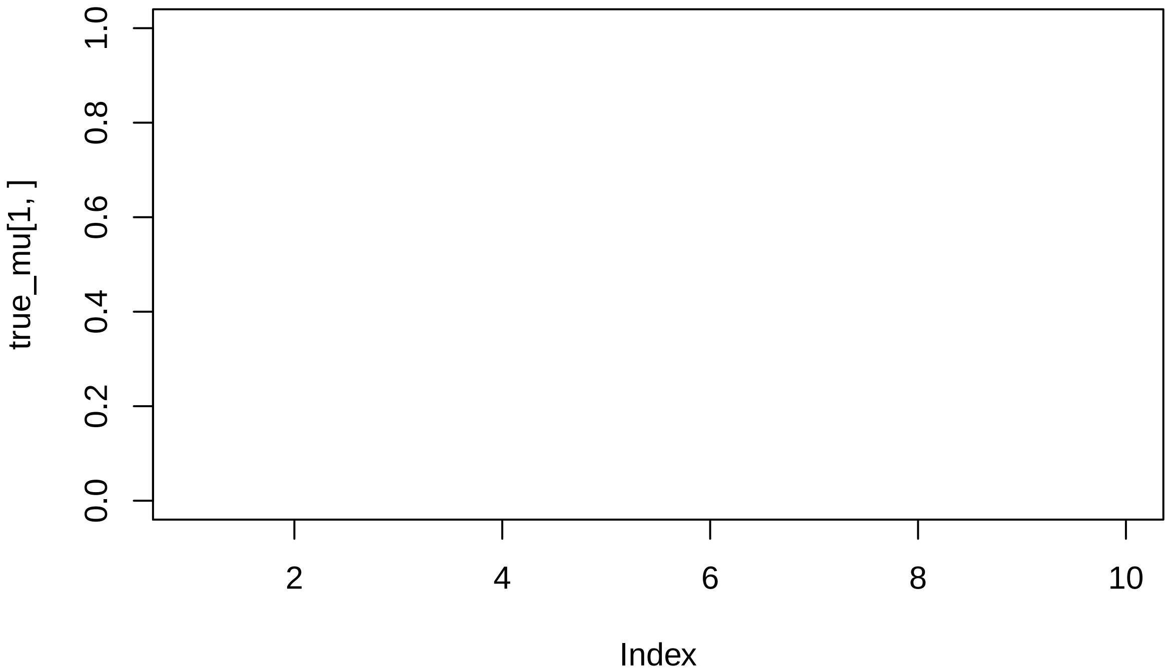
```
myem <- function(K){
  set.seed(1234567890)

max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
```

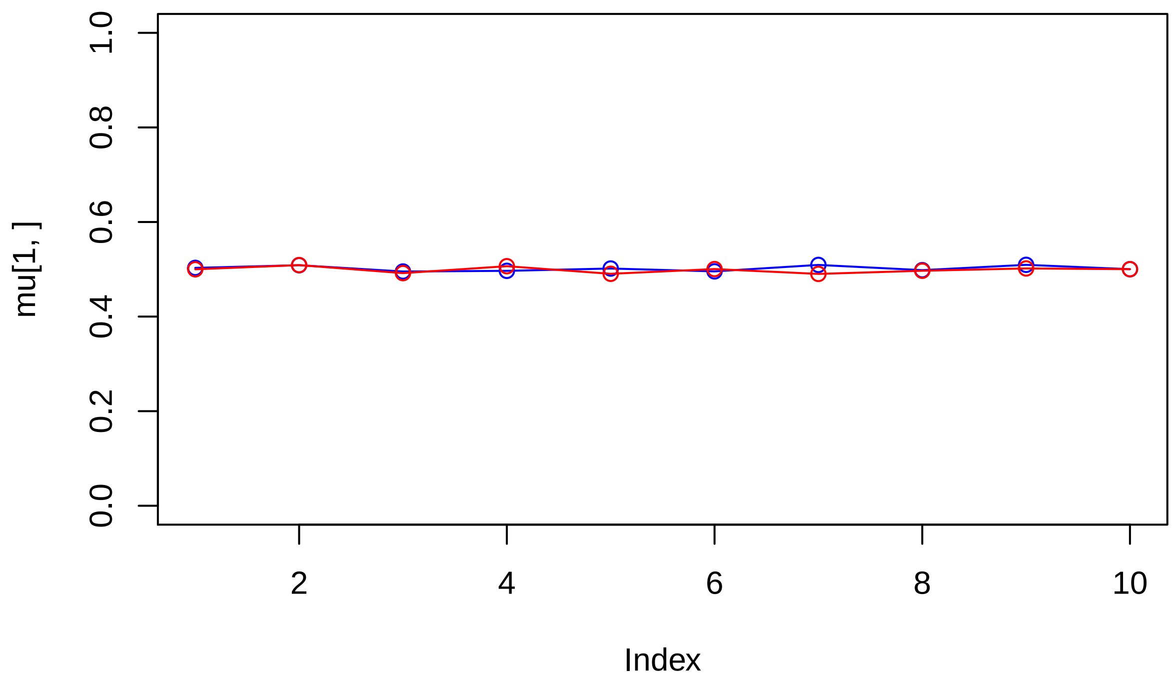
```r
N=1000 # number of training points
D=10 # number of dimensions
x <- matrix(nrow=N, ncol=D) # training data
true_pi <- vector(length = K) # true mixing coefficients
true_mu <- matrix(nrow=K, ncol=D) # true conditional distributions
true_pi=c(rep(1/3, K))

if(K == 2){
  plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
  points(true_mu[2,], type="o", col="red")

  true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
  true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
}else if(K == 3){
    plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
    points(true_mu[2,], type="o", col="red")
    points(true_mu[3,], type="o", col="green")

  true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
  true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
  true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
}else {
    plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
    points(true_mu[2,], type="o", col="red")
    points(true_mu[3,], type="o", col="green")
    points(true_mu[4,], type="o", col="yellow")

    true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
    true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
    true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
    true_mu[4,] = c(0.3,0.5,0.5,0.7,0.5,0.5,0.5,0.5,0.4,0.5)}

# Producing the training data
for(n in 1:N) {
k <- sample(1:K,1,prob=true_pi)
for(d in 1:D) {
x[n,d] <- rbinom(1,1,true_mu[k,d])
}
}

z <- matrix(nrow=N, ncol=K) # fractional component assignments
pi <- vector(length = K) # mixing coefficients
mu <- matrix(nrow=K, ncol=D) # conditional distributions
llik <- vector(length = max_it) # log likelihood of the EM iterations
# Random initialization of the paramters
pi <- runif(K,0.49,0.51)
pi <- pi / sum(pi)


for(k in 1:K) {
mu[k,] <- runif(D,0.49,0.51)
}
```

```r
for(it in 1:max_it) {

if(K == 2){
  plot(mu[1,], type="o", col="blue", ylim=c(0,1))
  points(mu[2,], type="o", col="red")
}else if(K == 3){
  plot(mu[1,], type="o", col="blue", ylim=c(0,1))
  points(mu[2,], type="o", col="red")
  points(mu[3,], type="o", col="green")
}else{
    plot(mu[1,], type="o", col="blue", ylim=c(0,1))
    points(mu[2,], type="o", col="red")
    points(mu[3,], type="o", col="green")
    points(mu[4,], type="o", col="yellow")}


Sys.sleep(0.5)
# E-step: Computation of the fractional component assignments

for(k in 1:K)
prod <- exp(x %*% log(t(mu))) * exp((1-x) %*% t(1-mu))

num = matrix(rep(pi,N), ncol = K, byrow = TRUE) * prod
dem = rowSums(num)
poster = num/dem

#Log likelihood computation.
llik[it] = sum(log(dem))
# Your code here
cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
flush.console()
# Stop if the lok likelihood has not changed significantly
if( it != 1){
if(abs(llik[it] - llik[it-1]) < min_change){break}
}
#M-step: ML parameter estimation from the data and fractional component assignments
# Your code here
num_pi = colSums(poster)
pi = num_pi/N
mu = (t(poster) %*% x)/num_pi
}

#Printing pi, mu and development of log likelihood at the end
return(list(
pi = pi,
mu = mu,
logLikelihoodDevelopment = plot(llik[1:it],
type = "o",
main = "Development of the log likelihood",
xlab = "iteration",
ylab = "log likelihood")
))
}
```

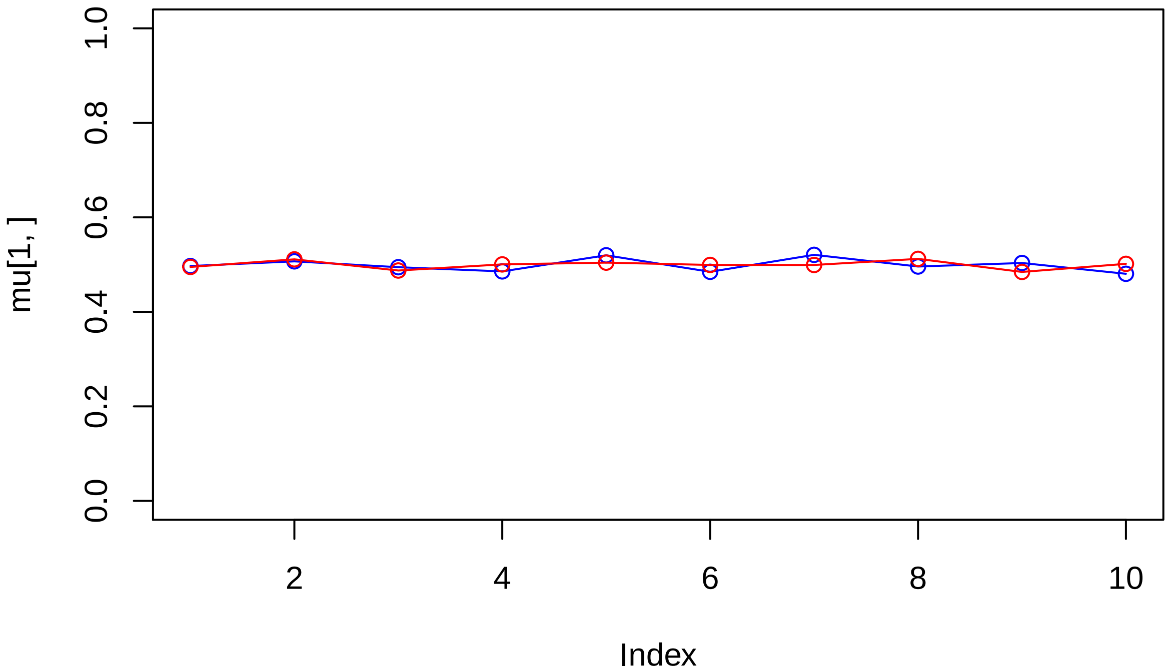
2. K = 2

```r
myem(K=2)
```

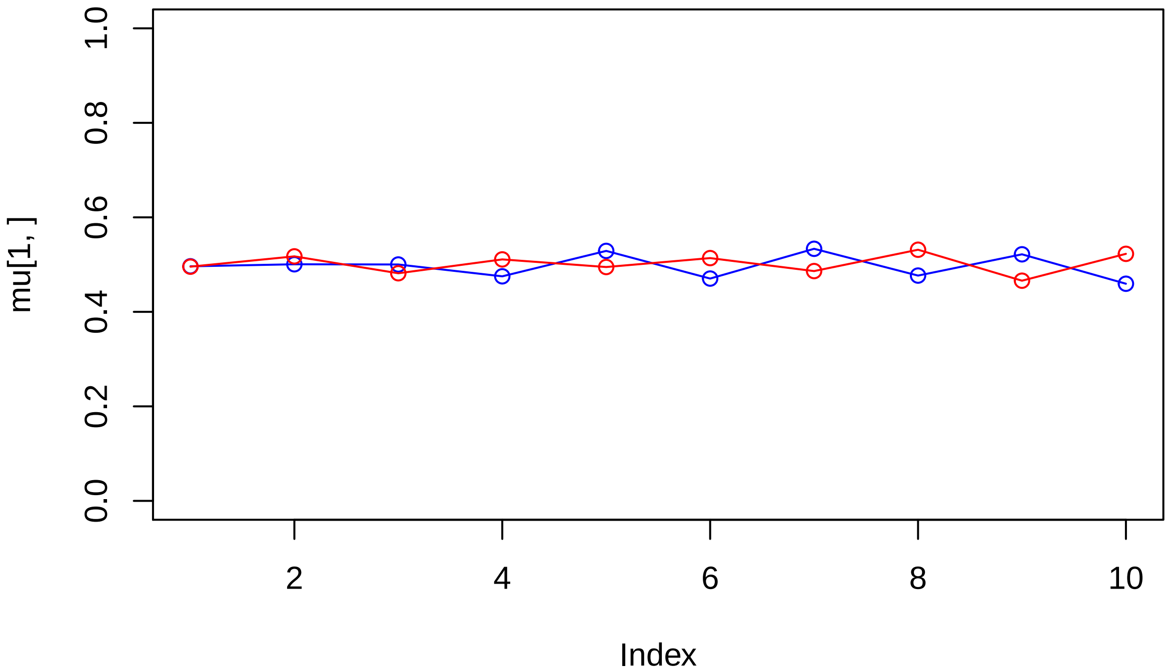




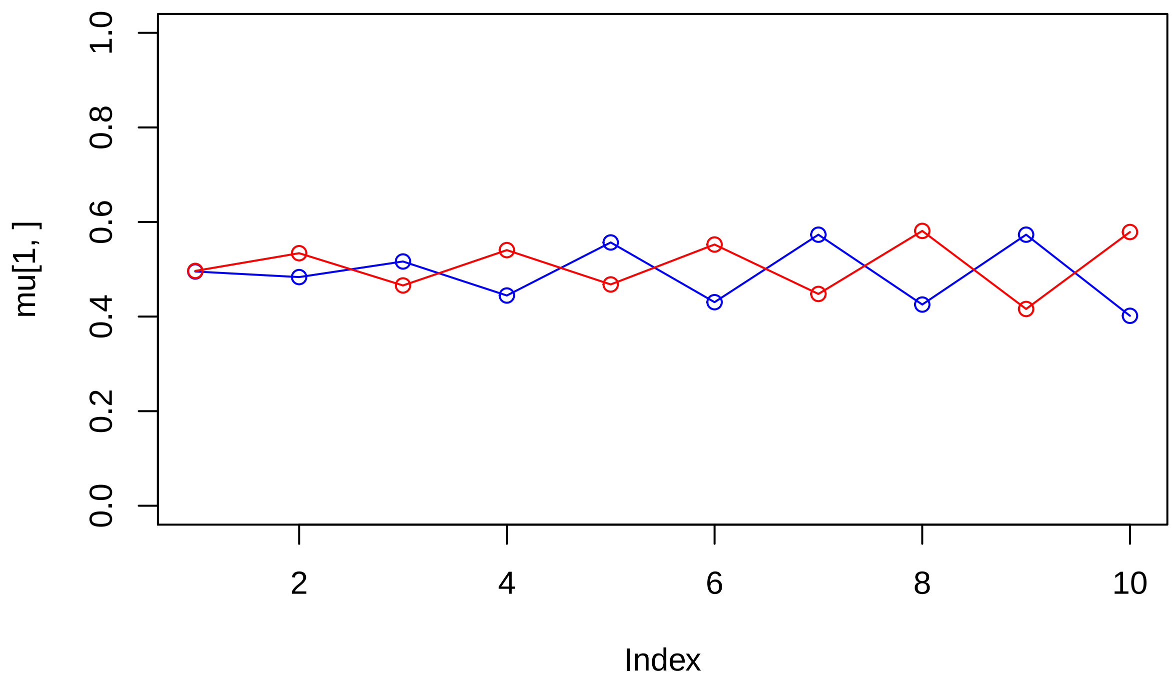

```
## iteration:  1 log likelihood:  -954.7133
```



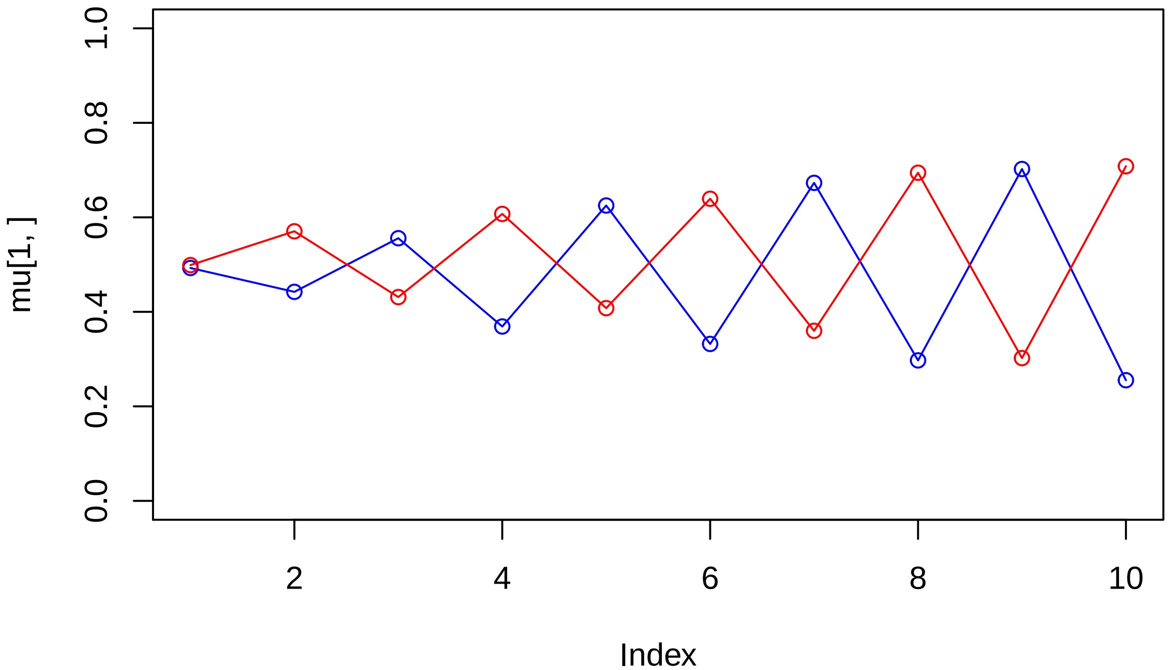

```
## iteration:  2 log likelihood:  -957.1002
```



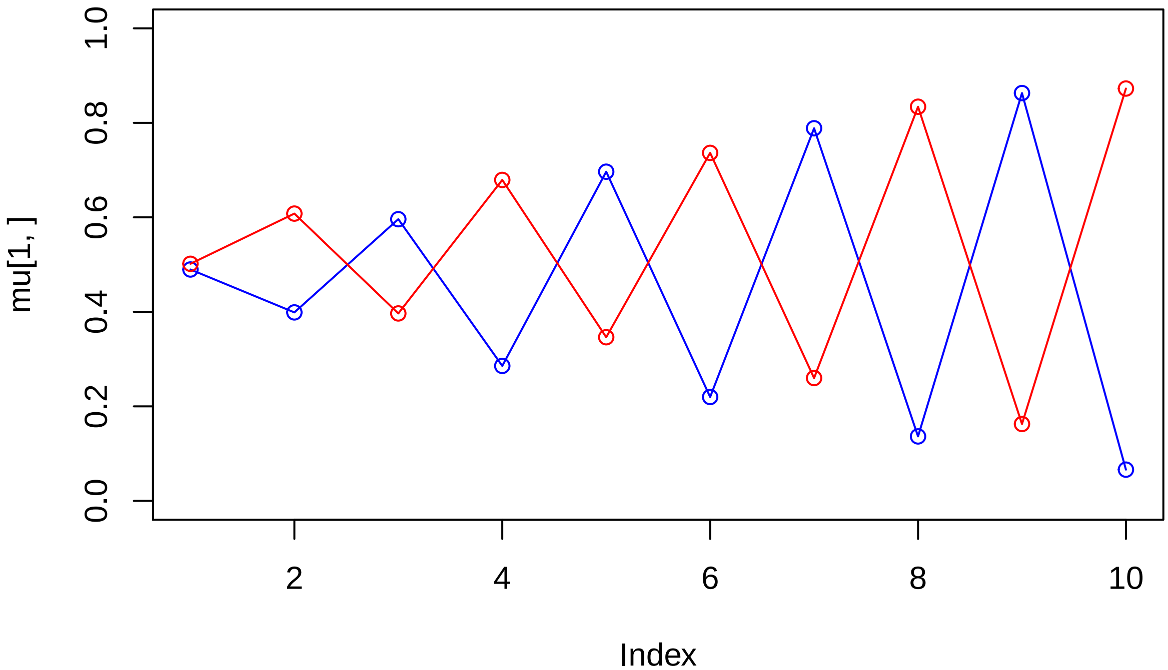

```
## iteration:  3 log likelihood:  -944.9229
```



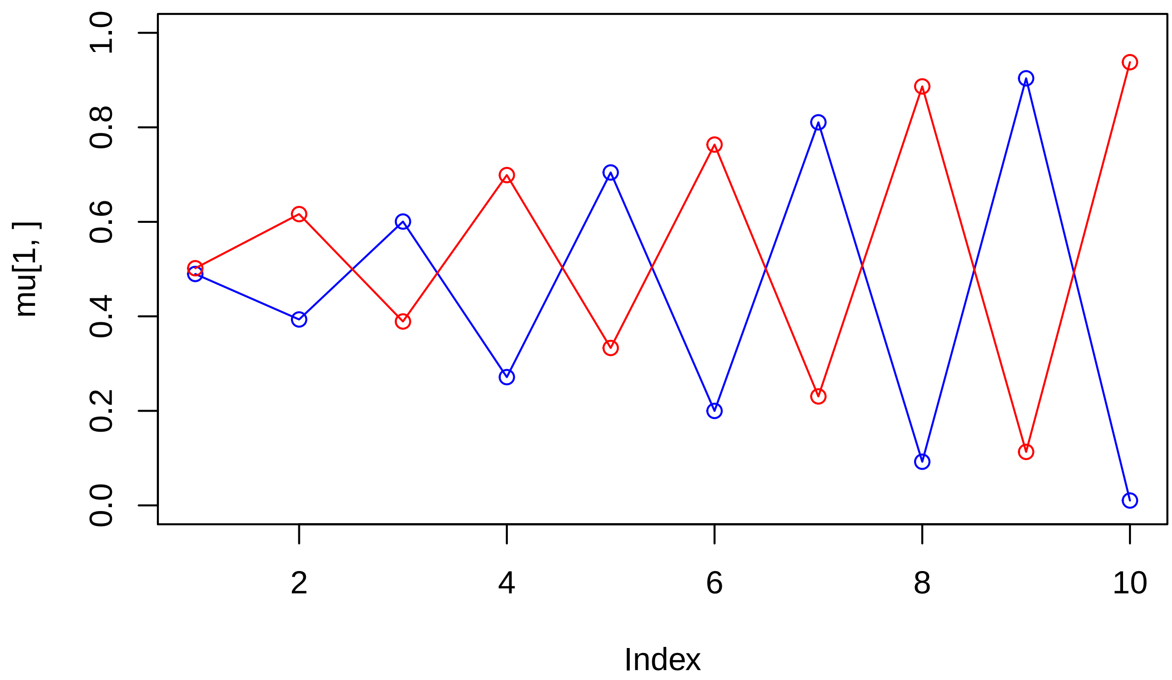

```
## iteration:  4 log likelihood:  -857.3443
```



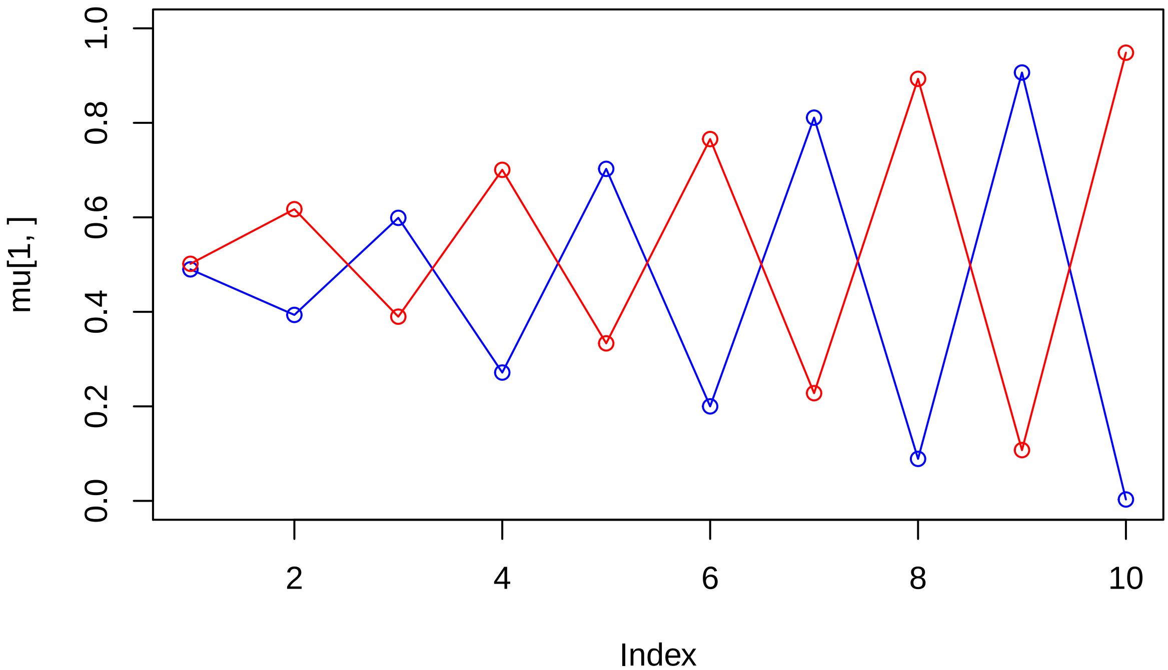

```
## iteration:  5 log likelihood:  -464.0063
```



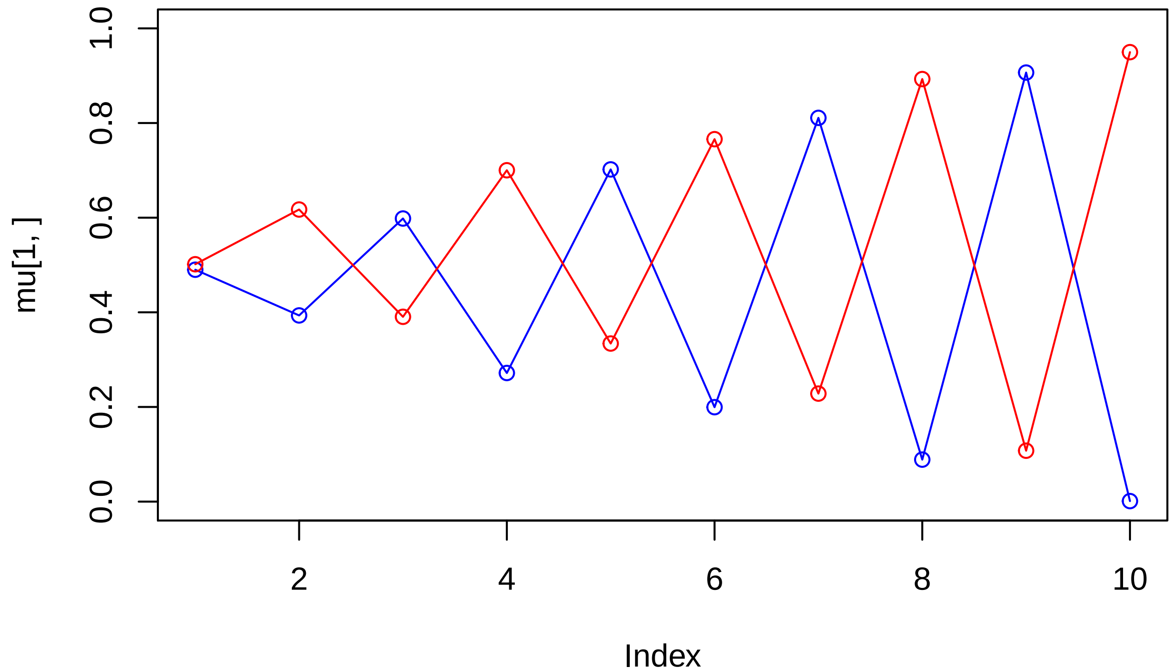

```
## iteration:  6 log likelihood:  50.2616
```



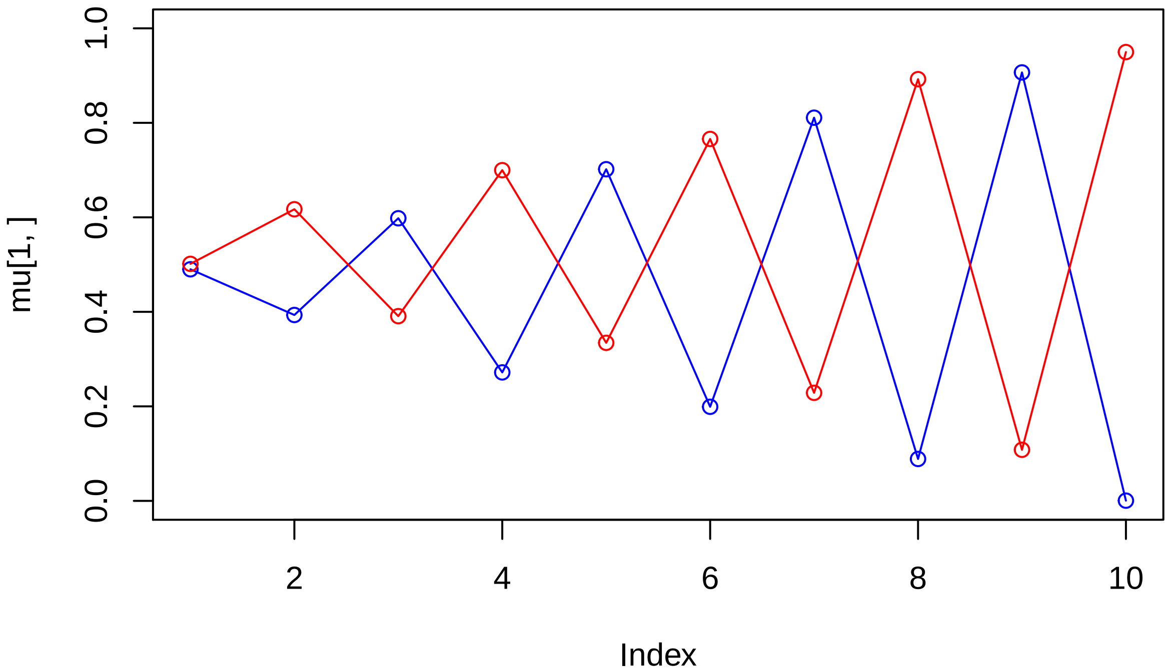

```
## iteration:  7 log likelihood:  177.0235
```



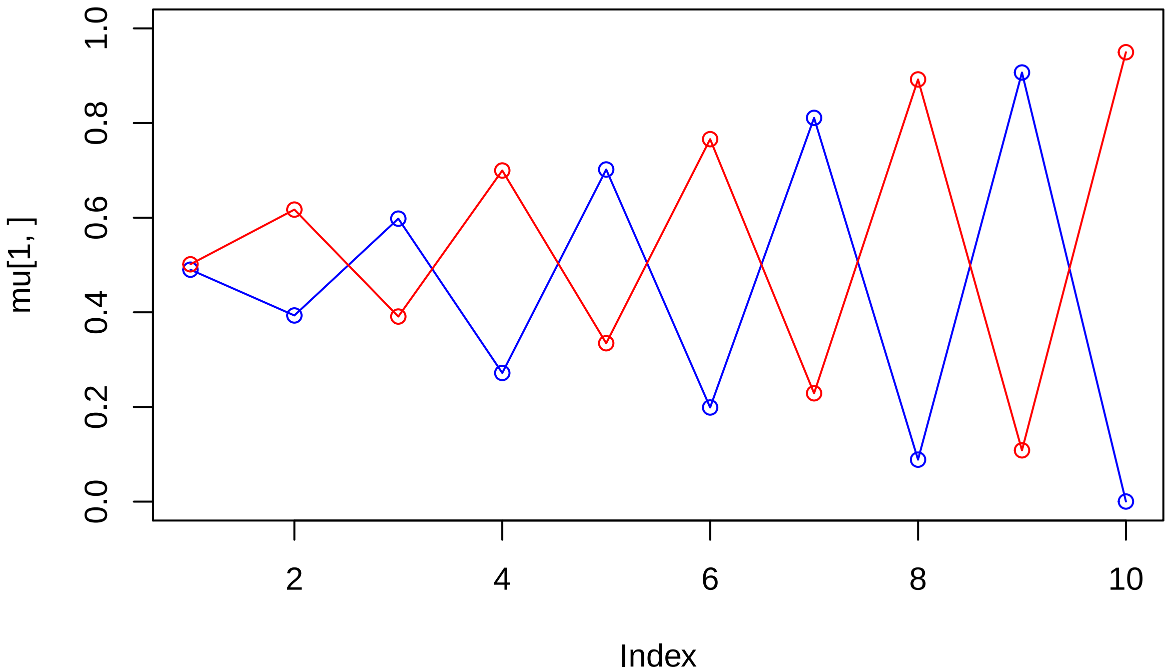

```
## iteration:  8 log likelihood:  189.1059
```



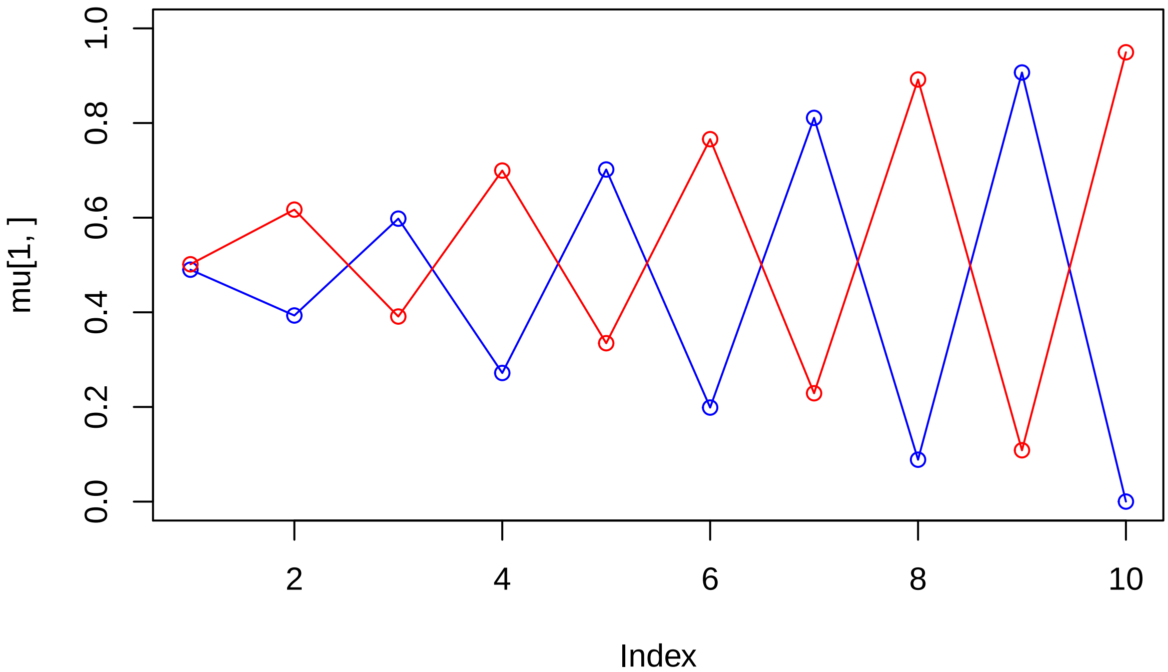

```
## iteration:  9 log likelihood:  190.0362
```



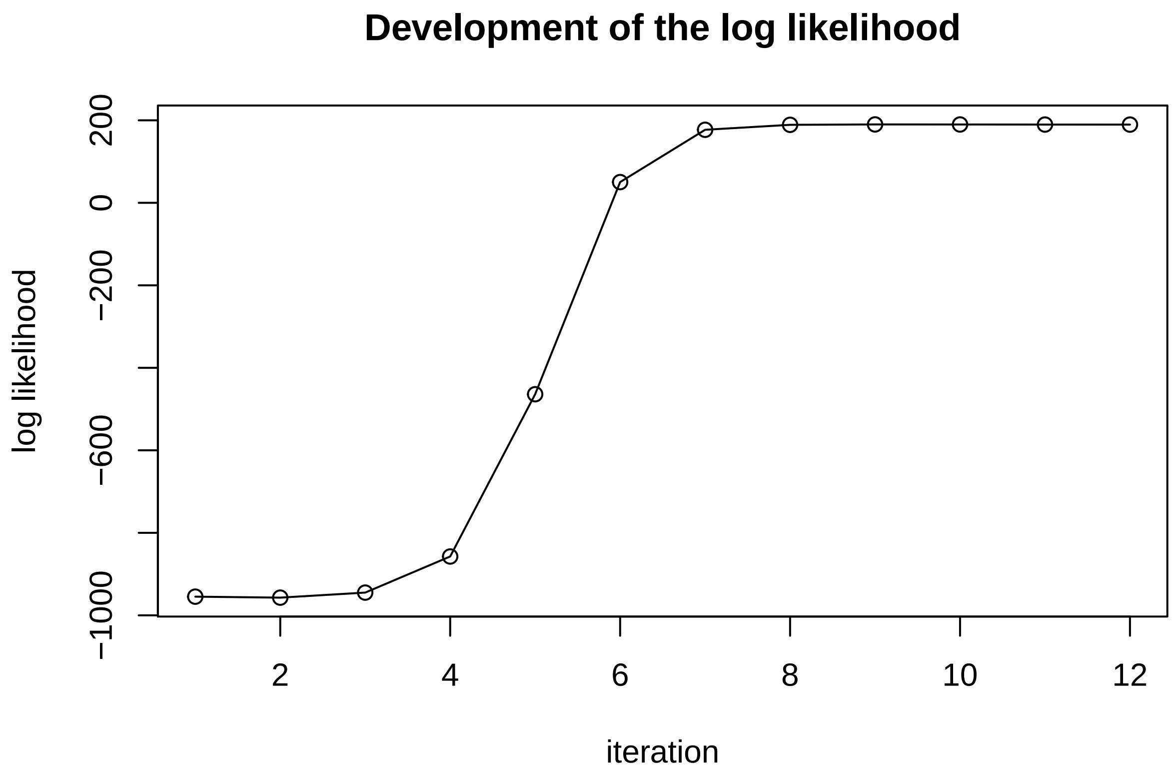

```
## iteration:  10 log likelihood:  189.9033
```




```
## iteration:  11 log likelihood:  189.7476
```



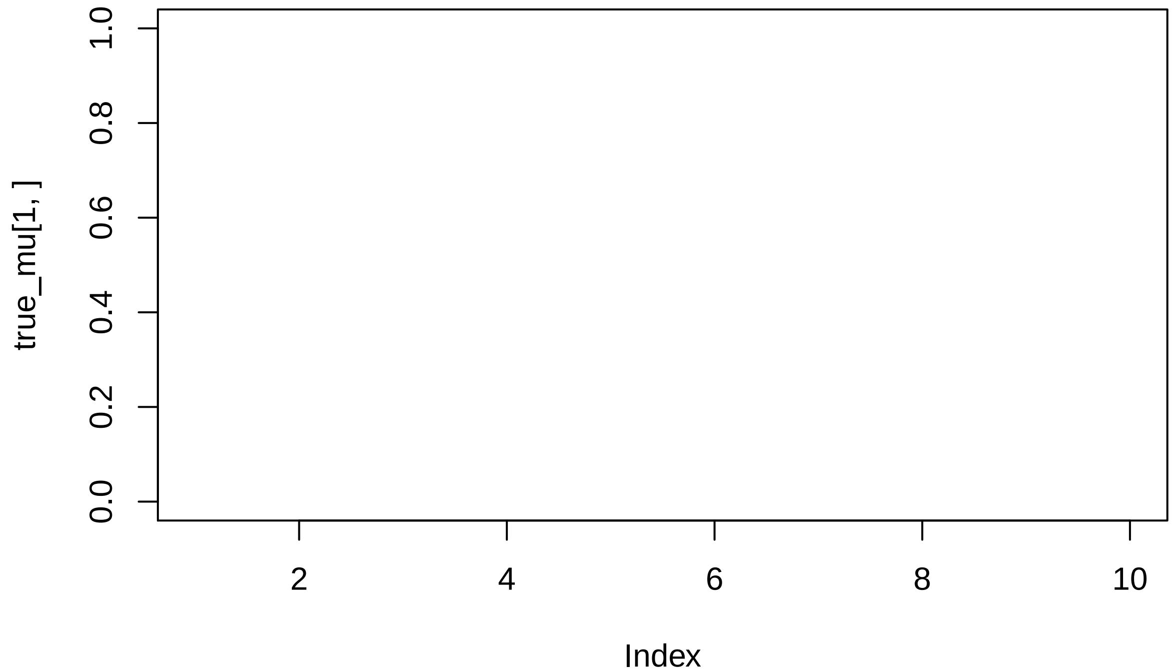

```
## iteration:  12 log likelihood:  189.6572
```

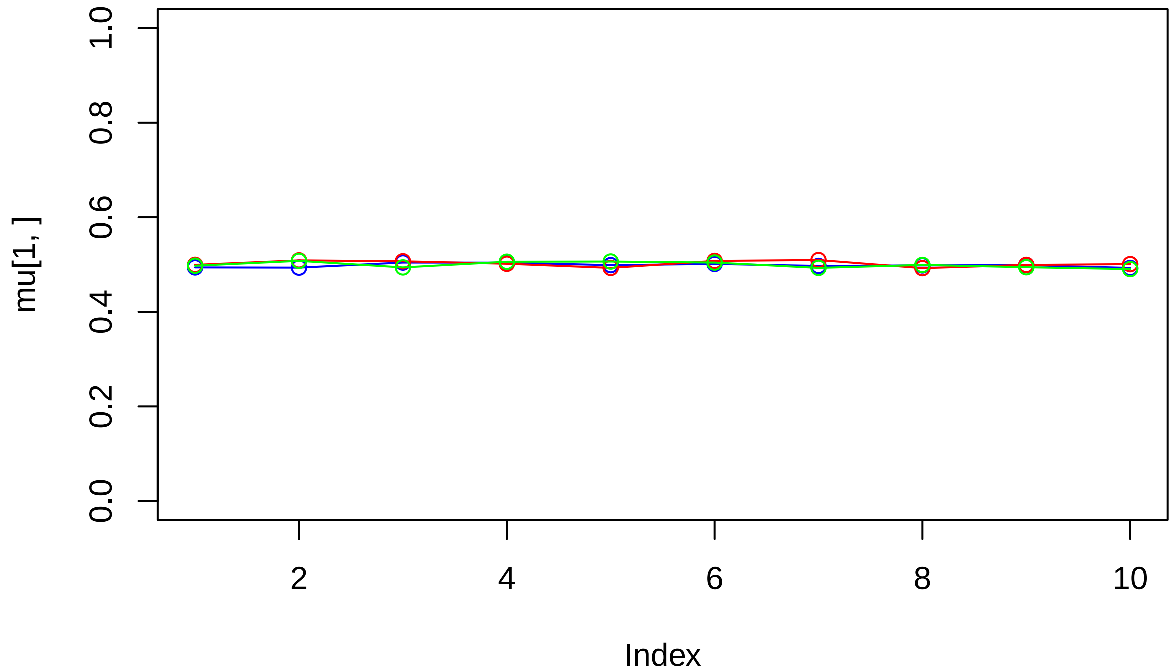



```
## $pi
## [1] 0.4829313 0.5170687
##
## $mu
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4900797 0.3933282 0.5979243 0.2718254 0.7017861 0.1987762 0.8108854
## [2,] 0.5015295 0.6170350 0.3911350 0.6995725 0.3347438 0.7658649 0.2289793
##            [,8]      [,9]         [,10]
## [1,] 0.08857256 0.9067548 0.00008952955
## [2,] 0.89200055 0.1084956 0.94950012035
##
## $logLikelihoodDevelopment
## NULL
```

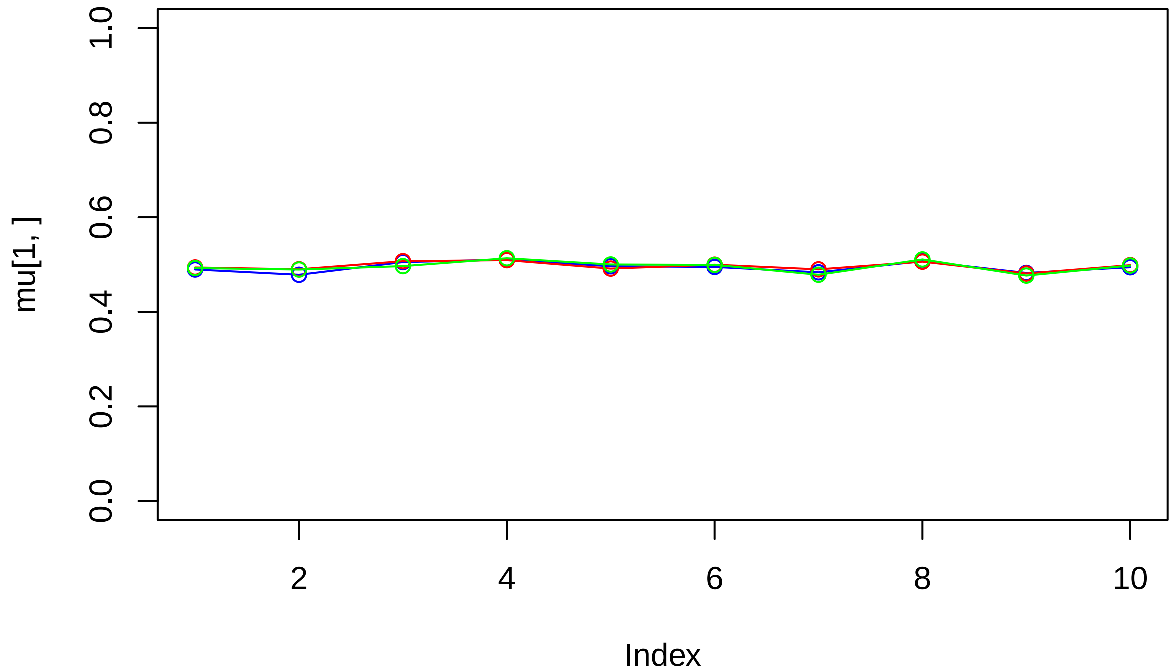
3. K = 3

```
myem(K=3)
```

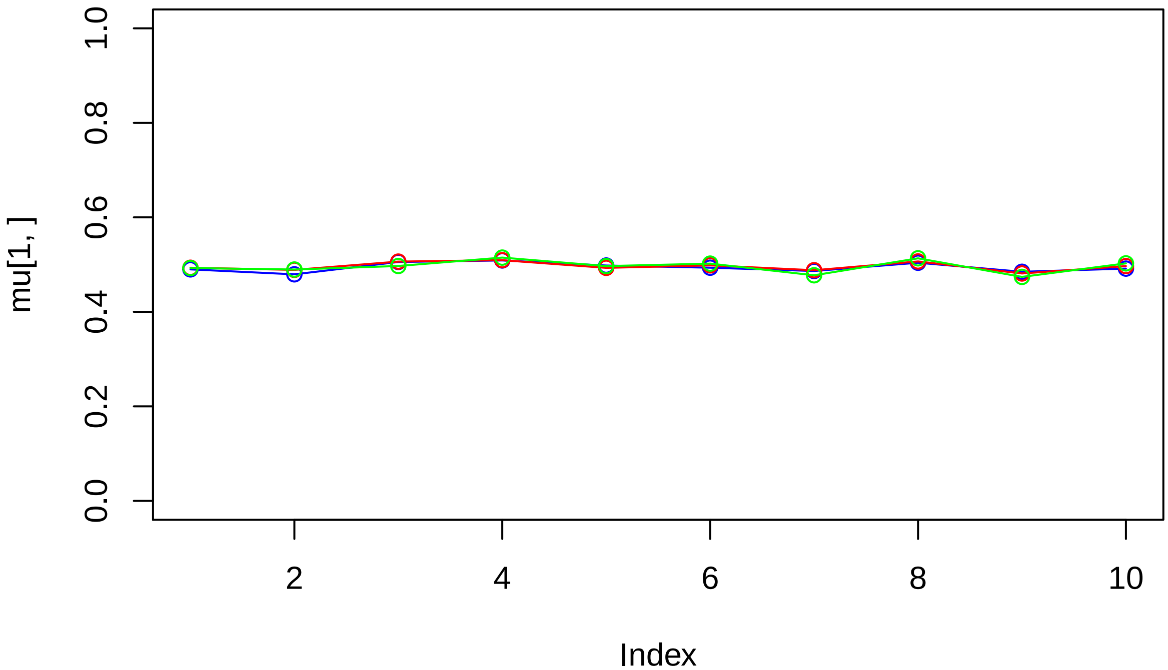

true_mu[1,]
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Index



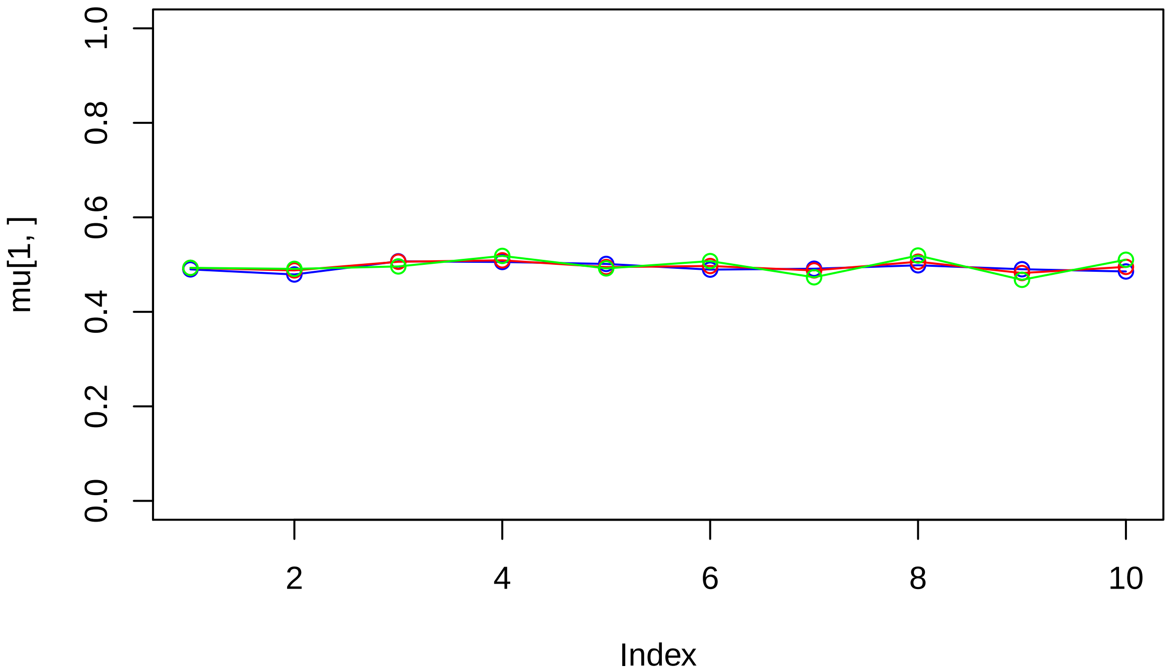

```
## iteration:  1 log likelihood:  -912.7567
```



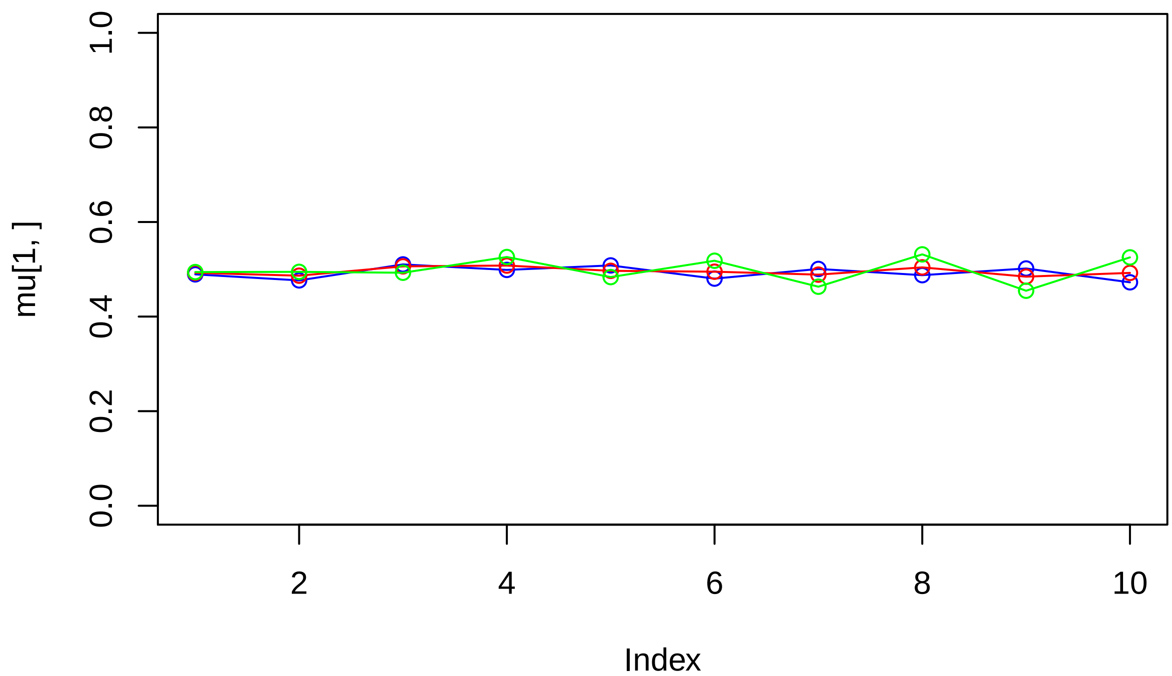

```
## iteration:  2 log likelihood:  -932.1921
```



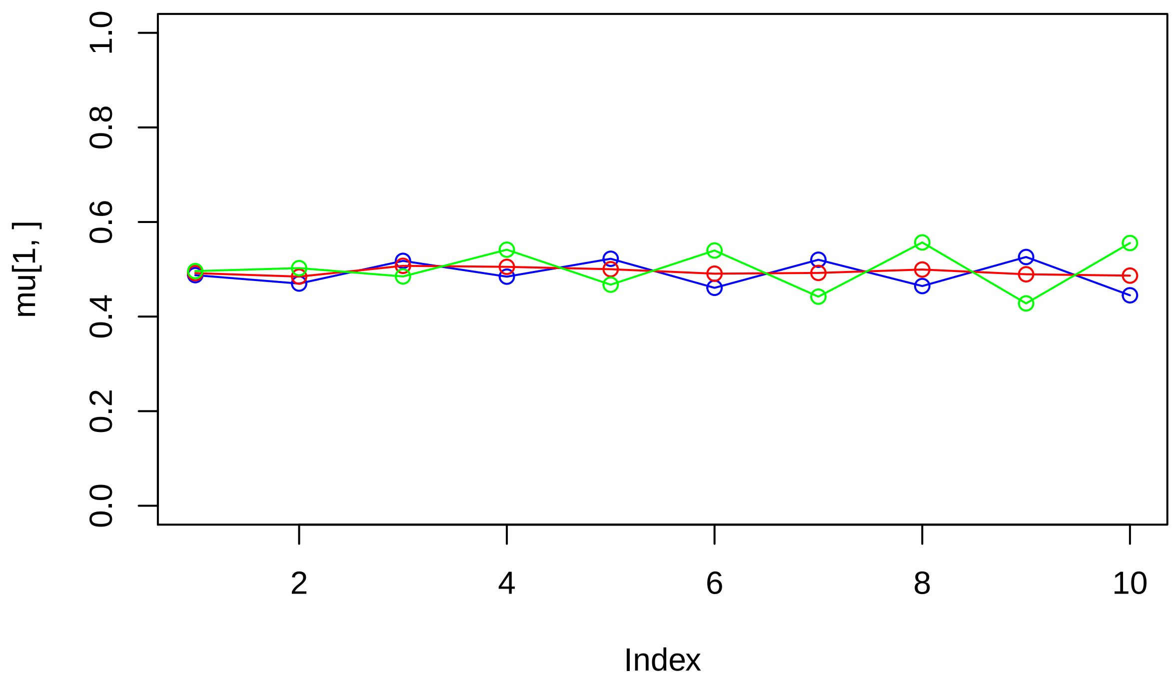

```
## iteration:  3 log likelihood:  -932.0234
```



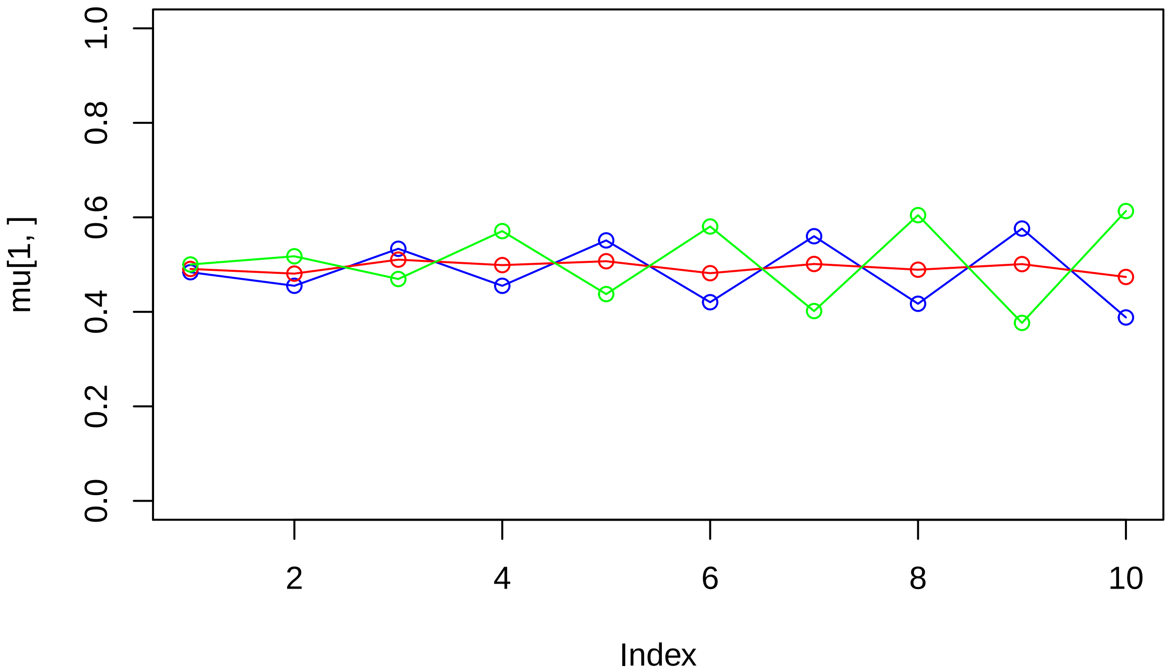

```
## iteration:  4 log likelihood:  -931.2587
```



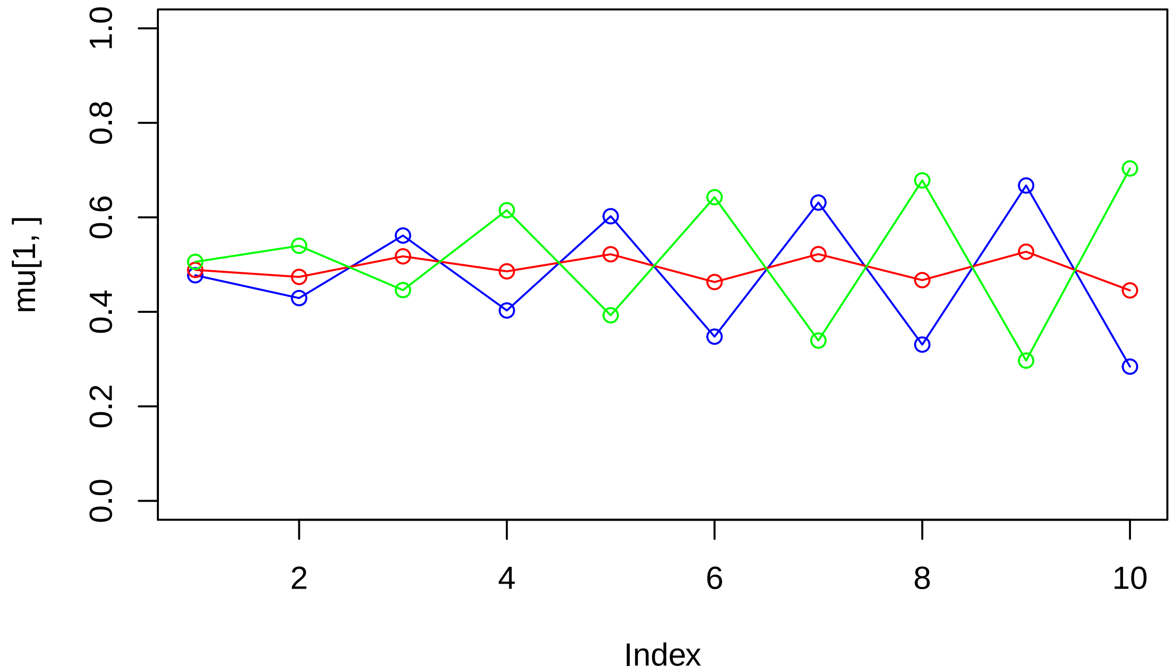

```
## iteration:  5 log likelihood:  -927.8881
```



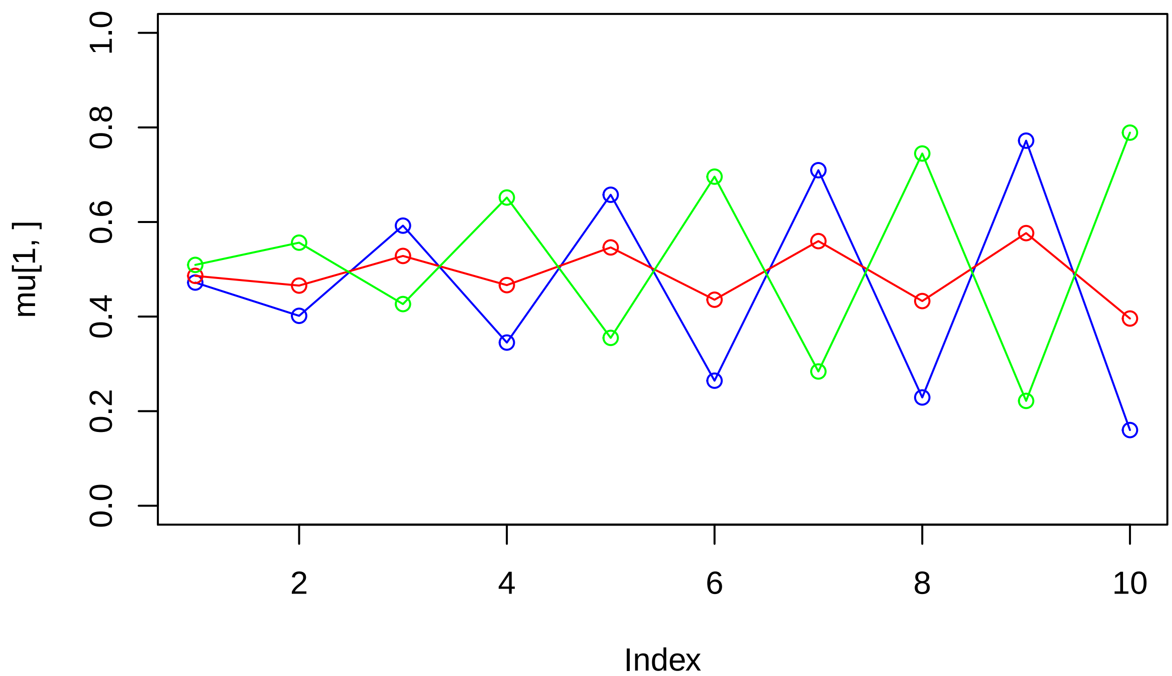

```
## iteration:  6 log likelihood:  -913.454
```



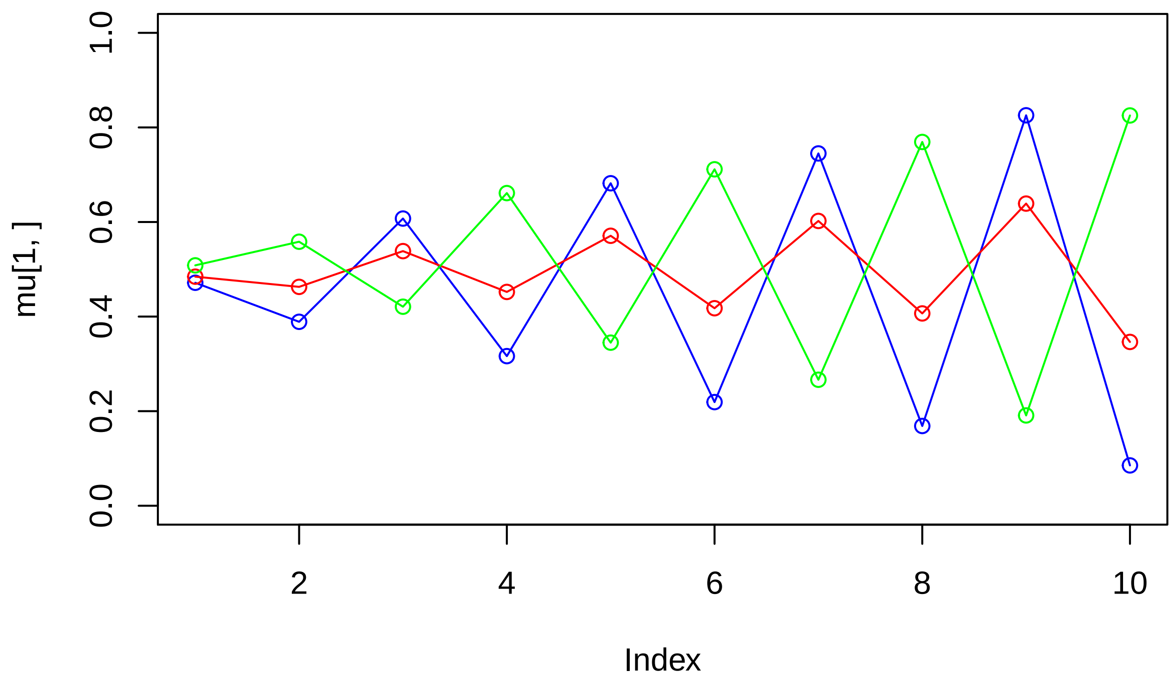

```
## iteration:  7 log likelihood:  -858.0583
```



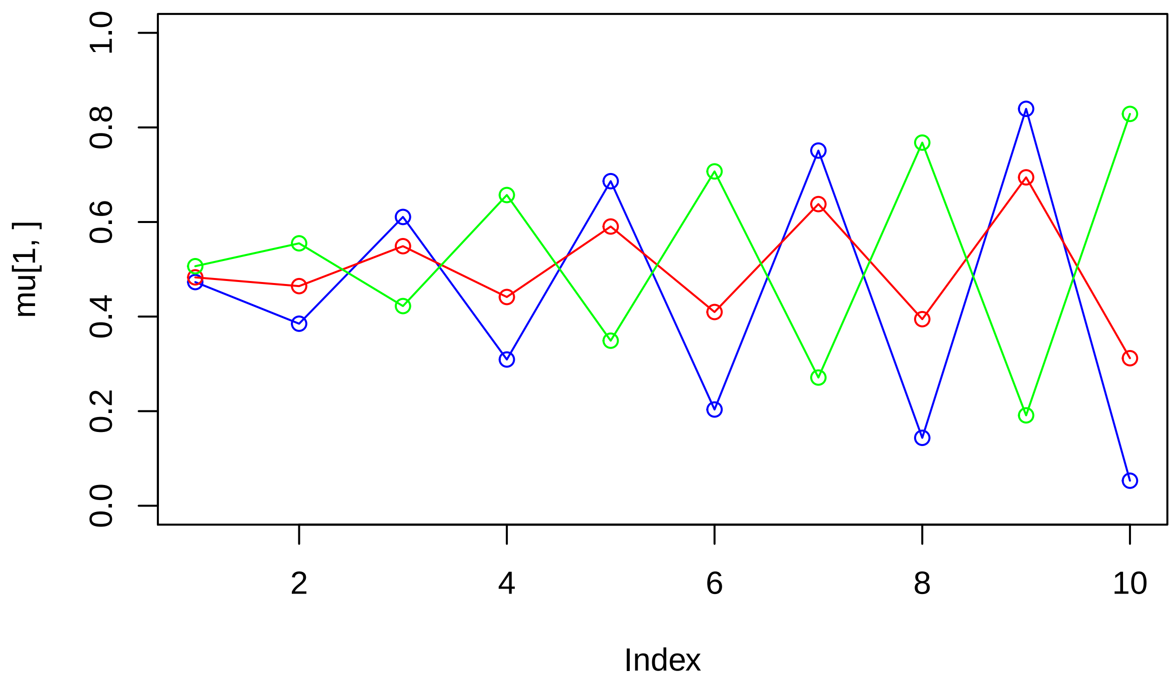

```
## iteration:  8 log likelihood:  -709.6665
```



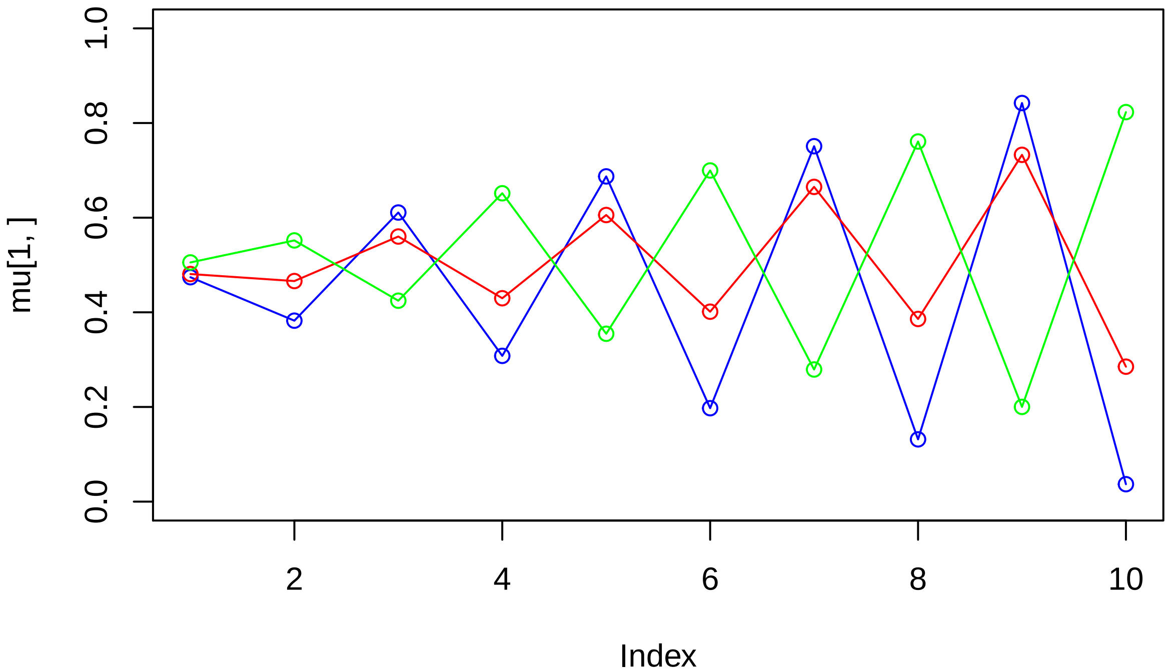

```
## iteration:  9 log likelihood:  -524.1097
```



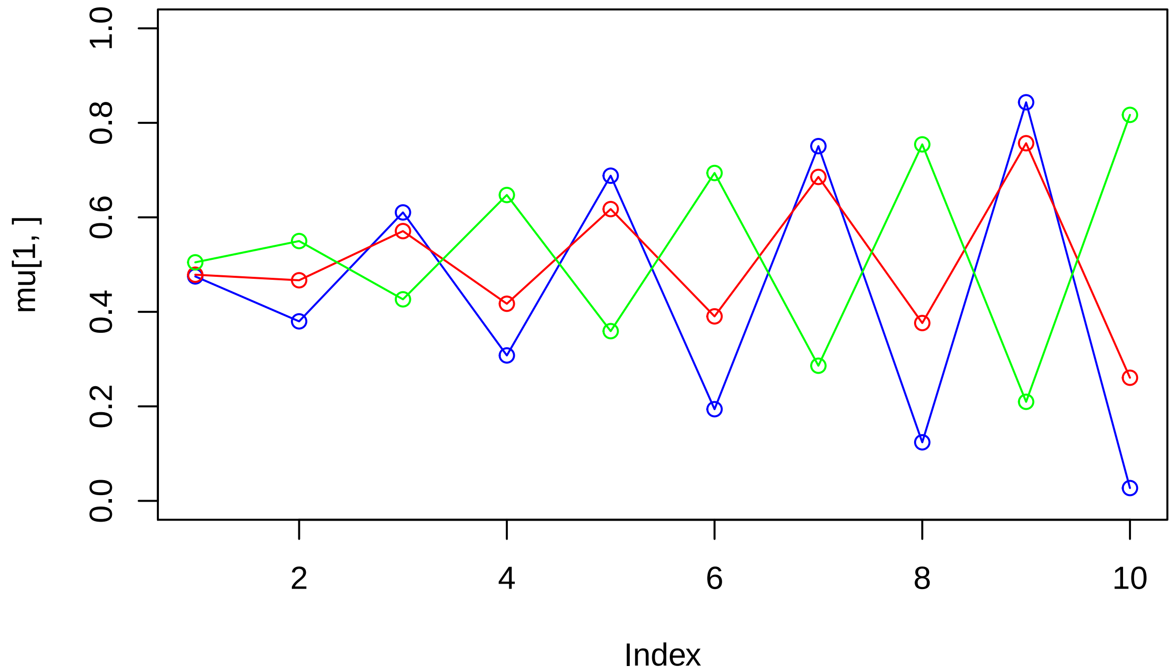

```
## iteration:  10 log likelihood:  -433.1614
```



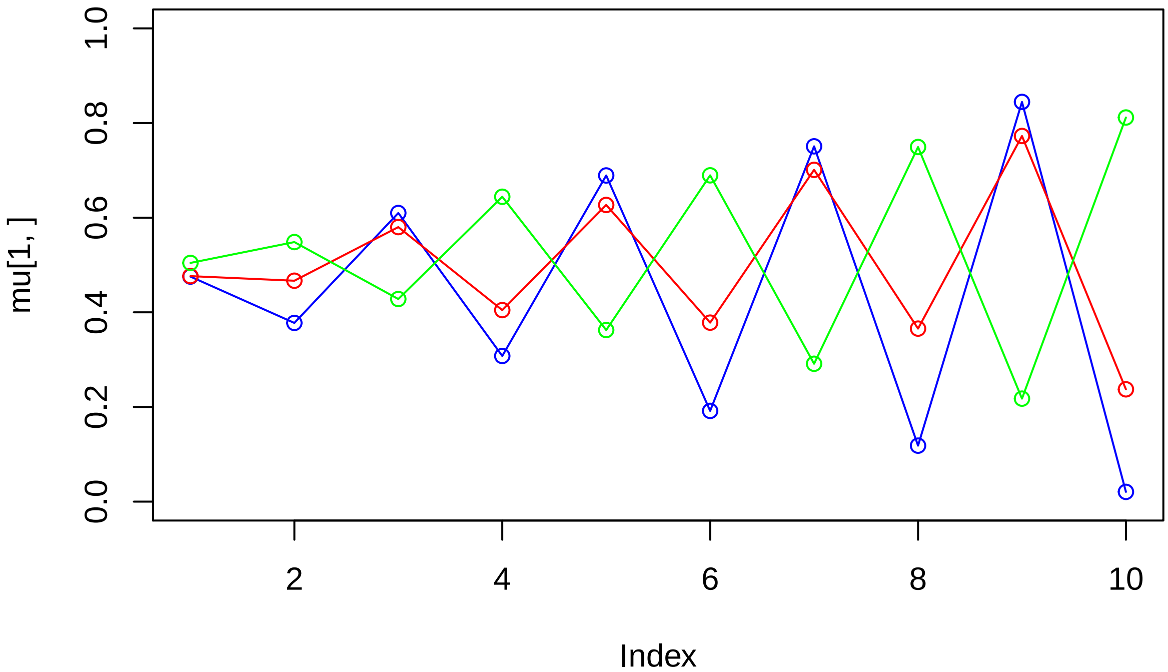

```
## iteration:  11 log likelihood:  -409.3331
```



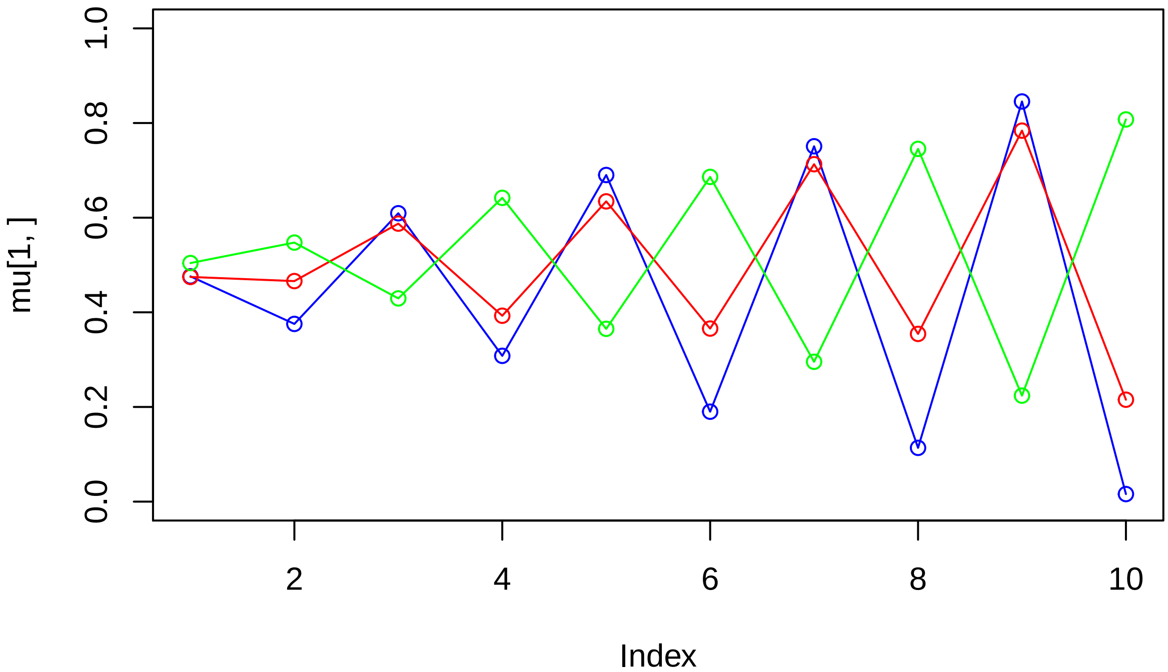

```
## iteration:  12 log likelihood:  -405.2132
```



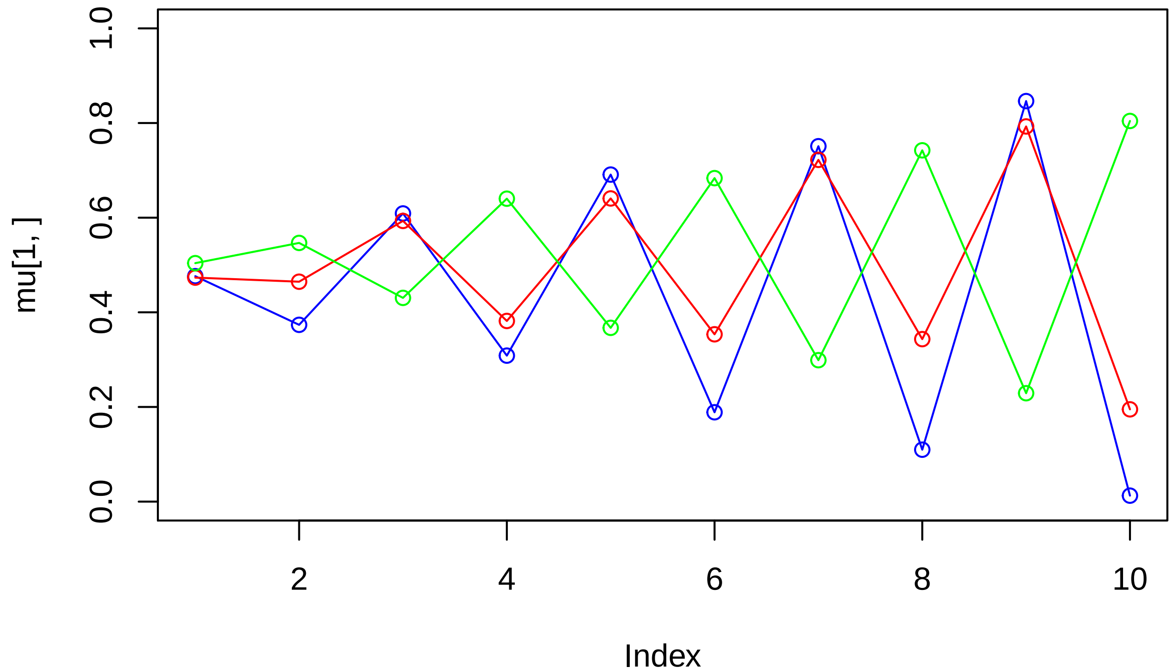

```
## iteration:  13 log likelihood:  -405.7233
```



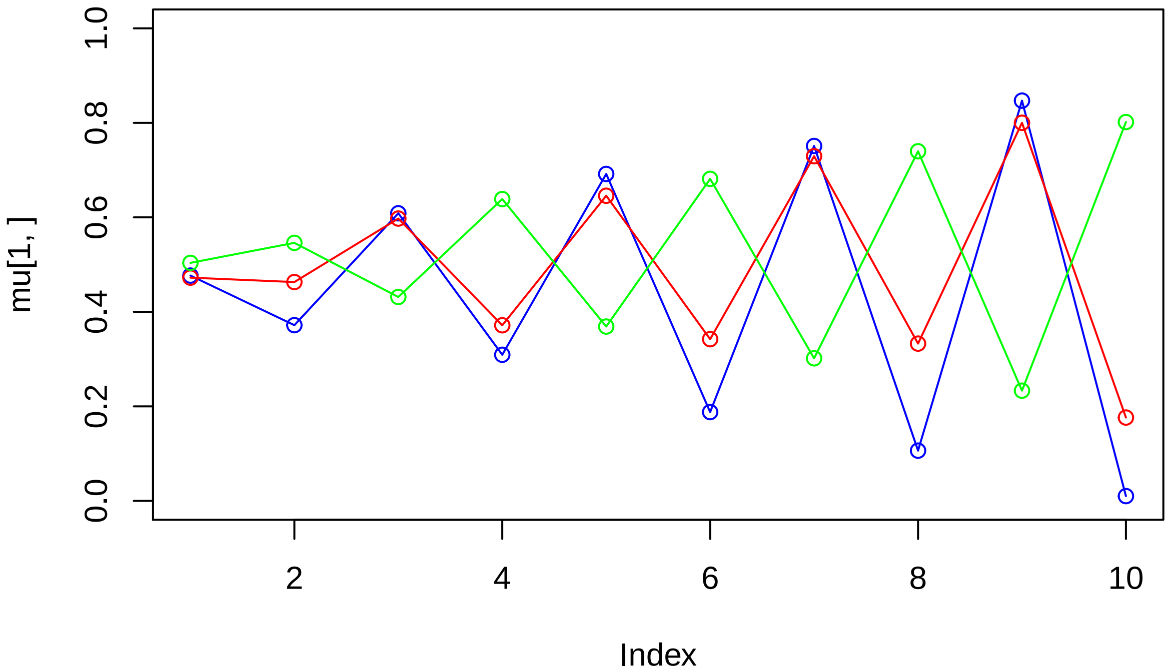

```
## iteration:  14 log likelihood:  -407.1621
```



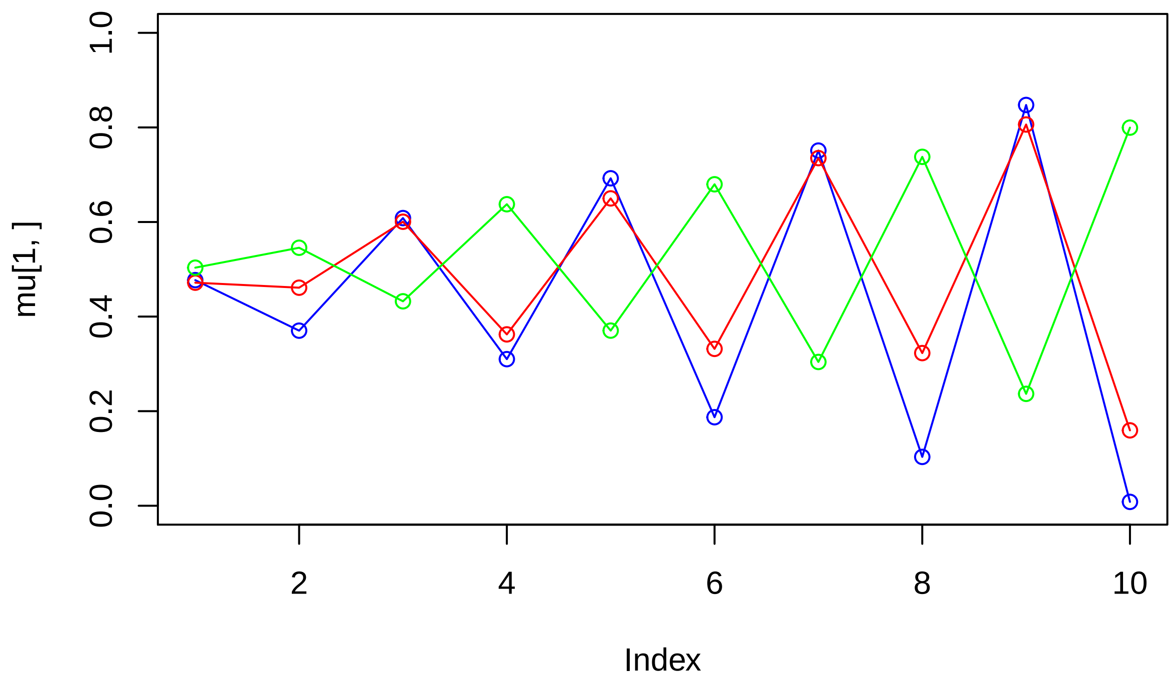

```
## iteration:  15 log likelihood:  -408.6475
```



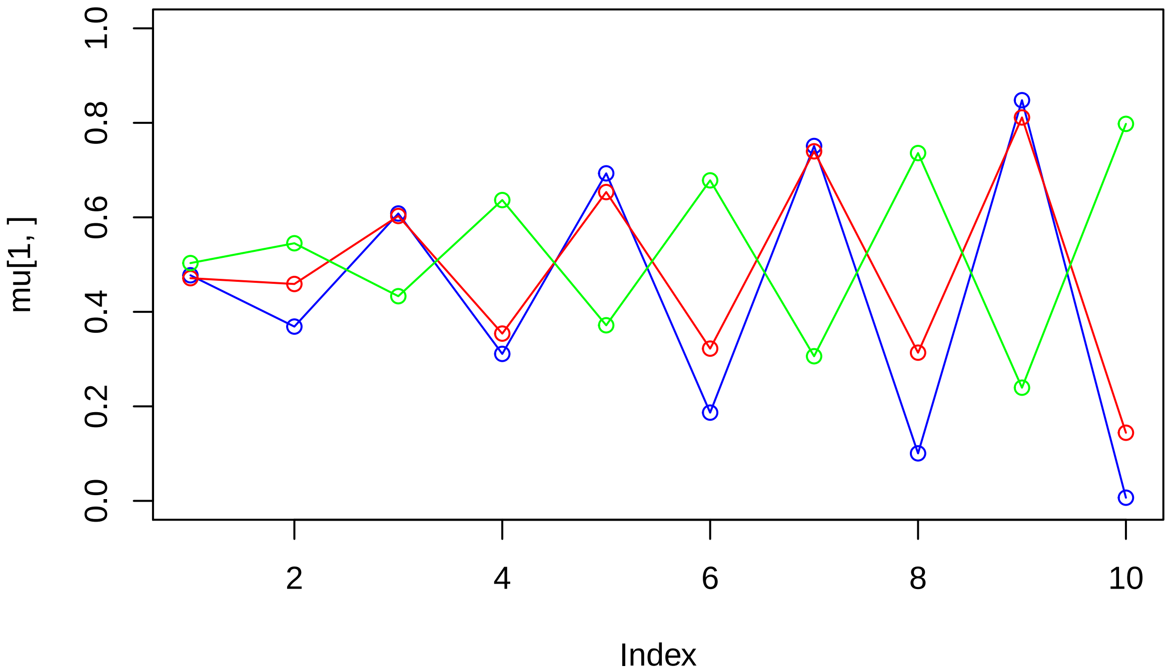

```
## iteration:  16 log likelihood:  -409.9879
```



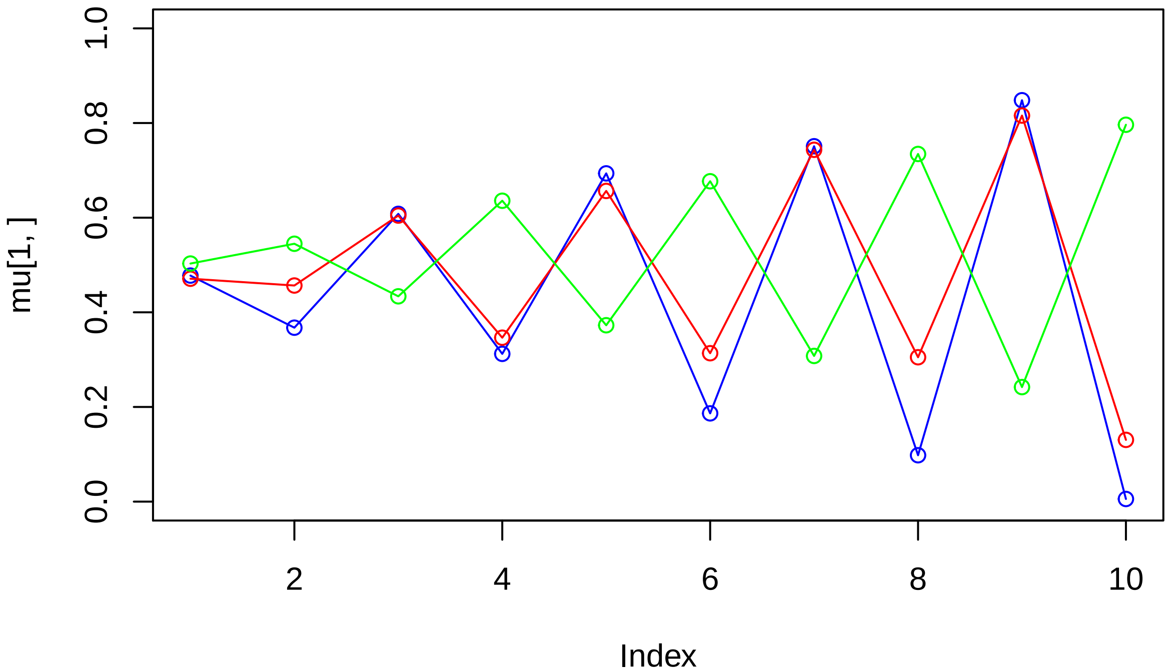

```
## iteration:  17 log likelihood:  -411.1645
```



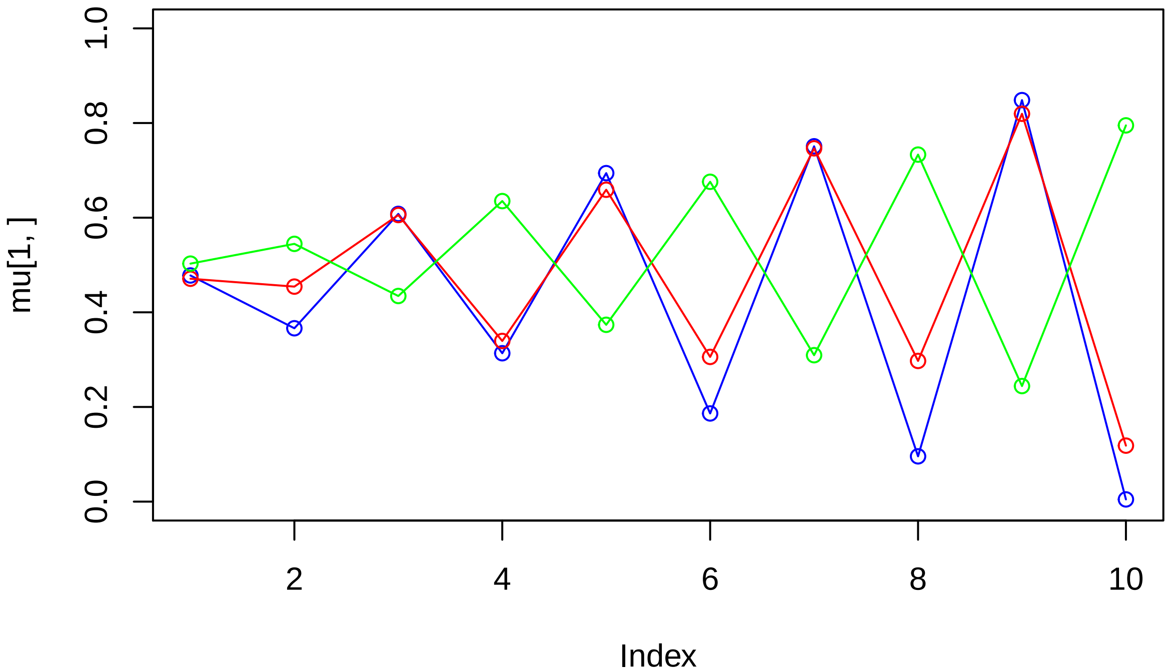

```
## iteration:  18 log likelihood:  -412.1979
```



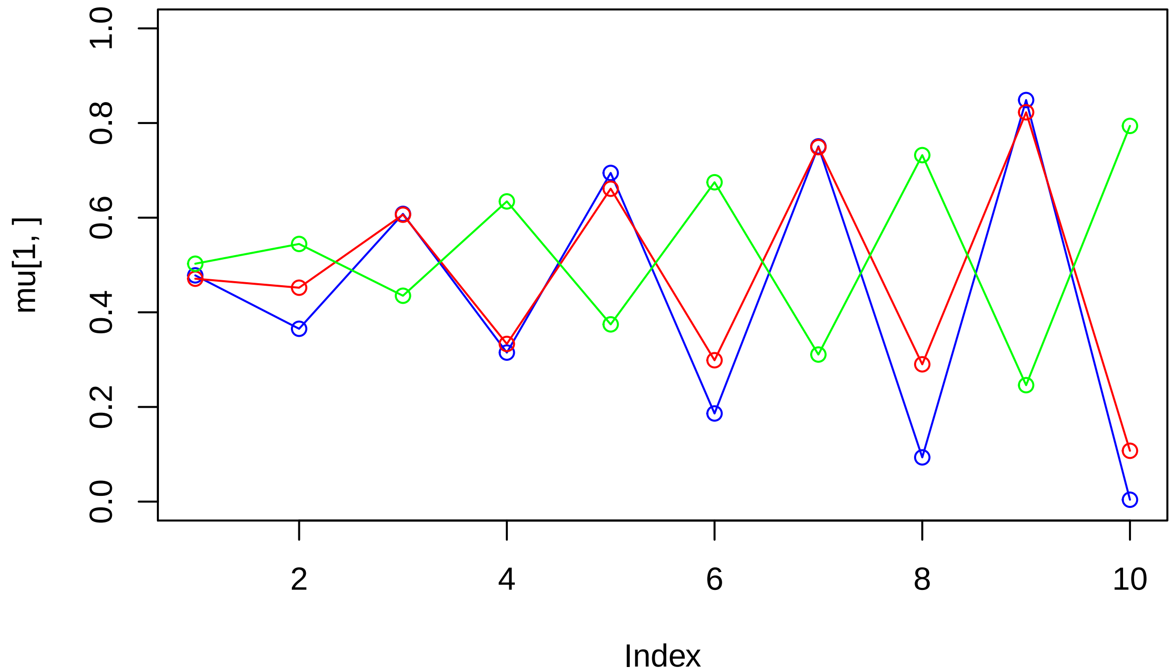

```
## iteration:  19 log likelihood:  -413.1139
```



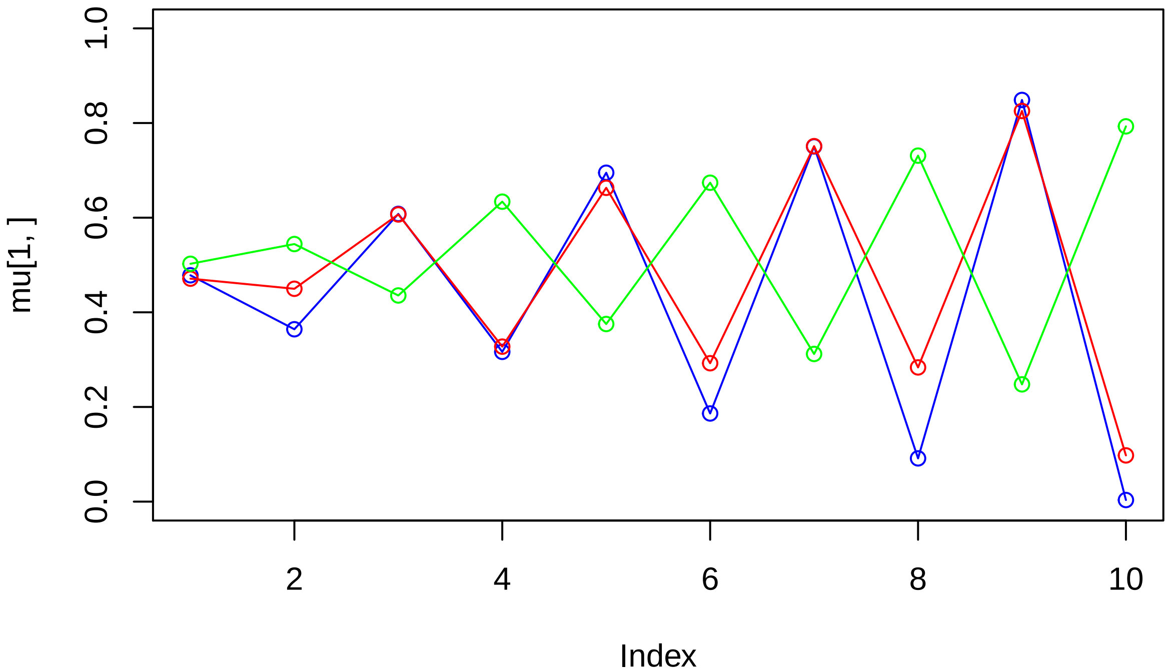

```
## iteration:  20 log likelihood:  -413.934
```



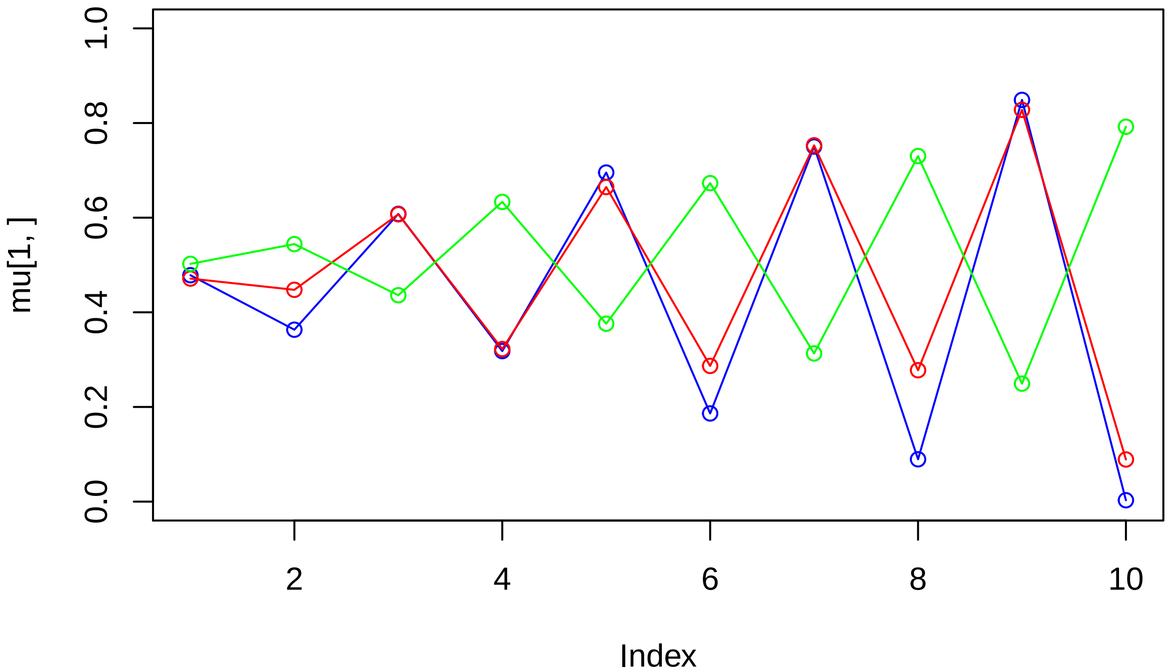

```
## iteration:  21 log likelihood:  -414.675
```



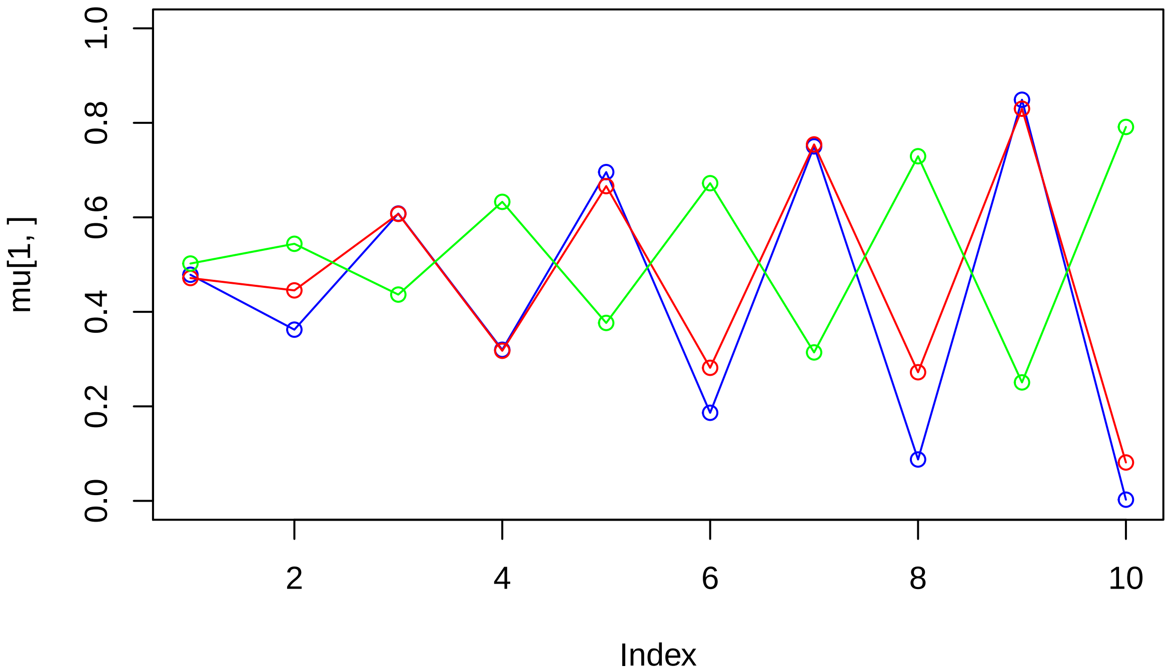

```
## iteration:  22 log likelihood:  -415.3492
```



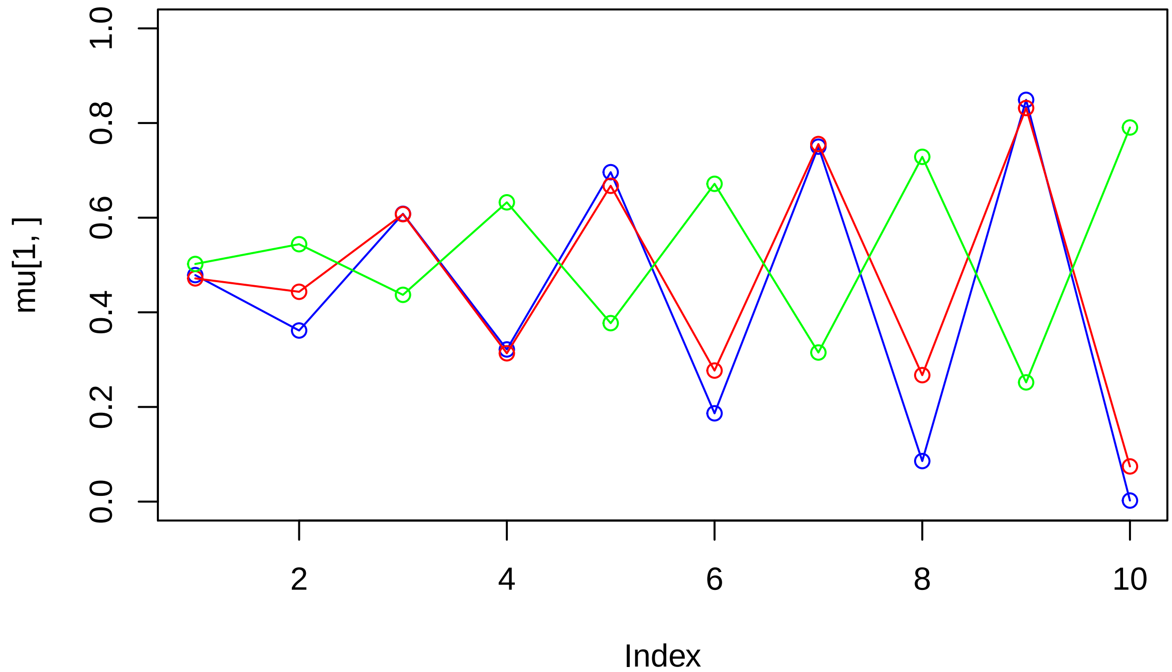

```
## iteration:  23 log likelihood:  -415.9659
```



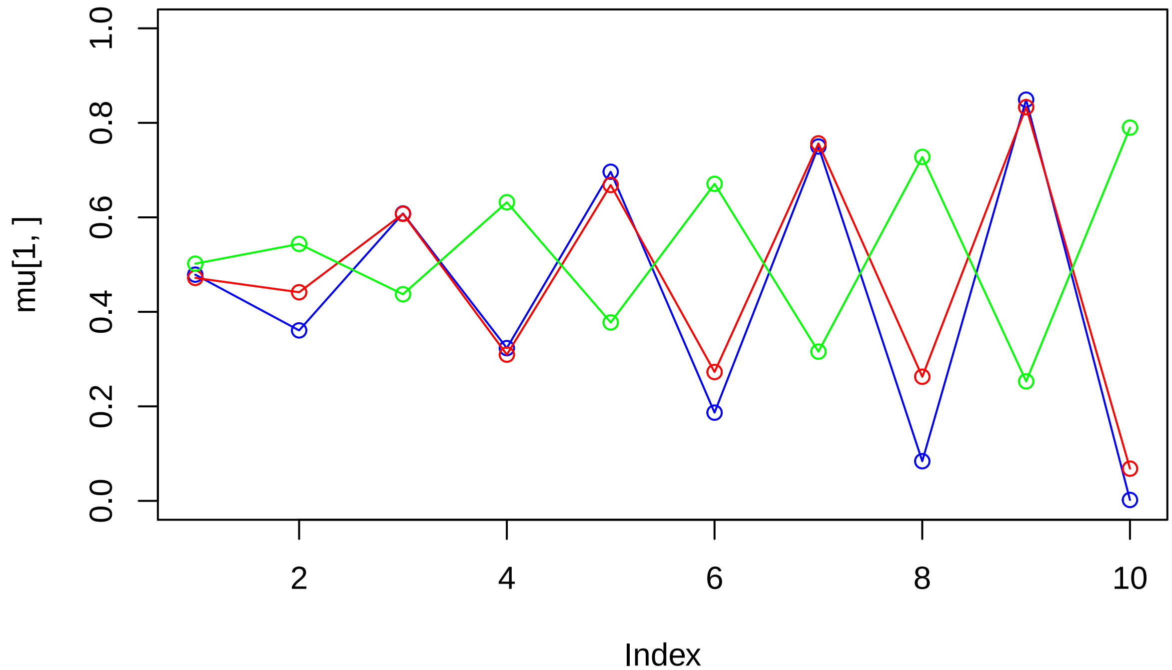

```
## iteration:  24 log likelihood:  -416.532
```



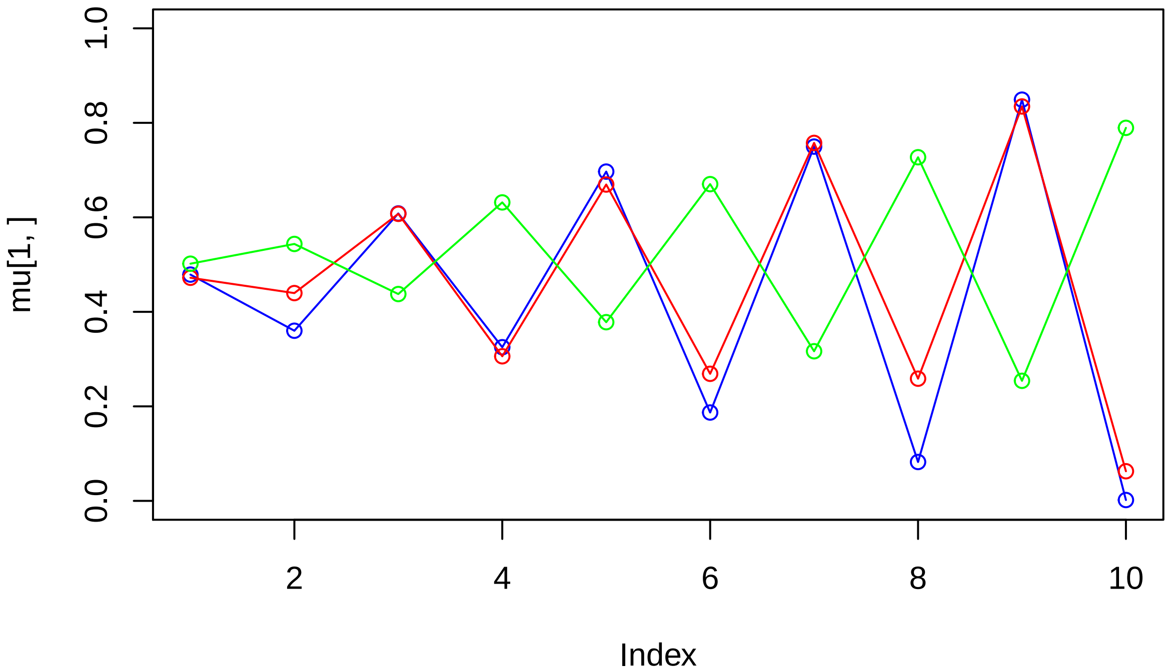

```
## iteration:  25 log likelihood:  -417.0528
```



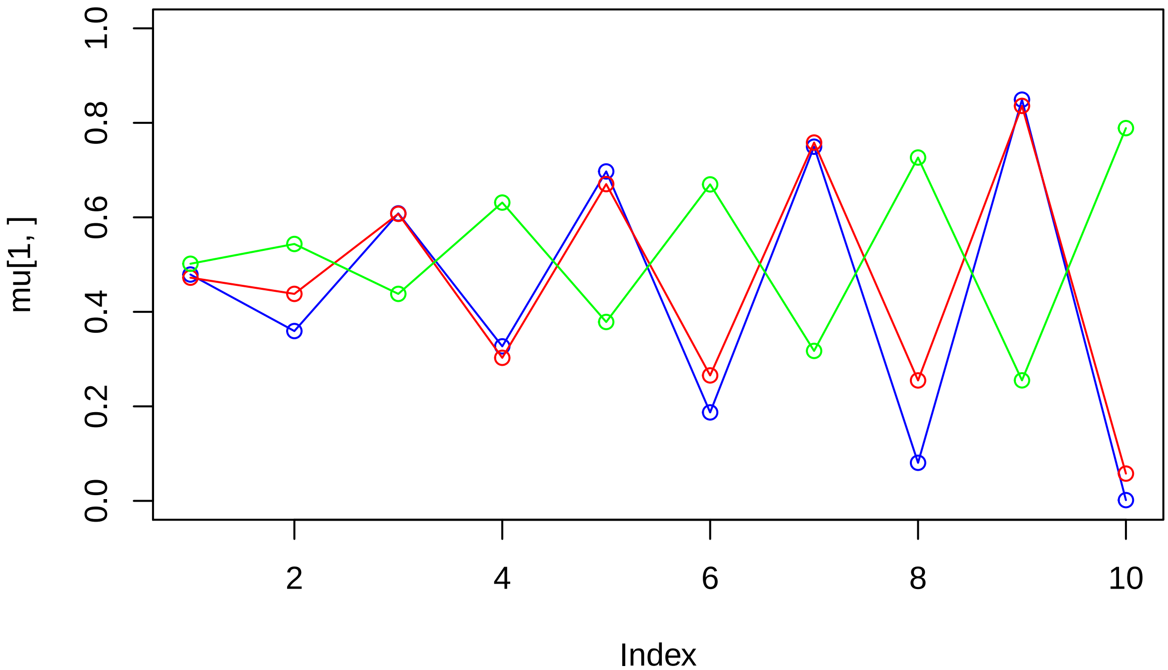

```
## iteration:  26 log likelihood:  -417.5328
```



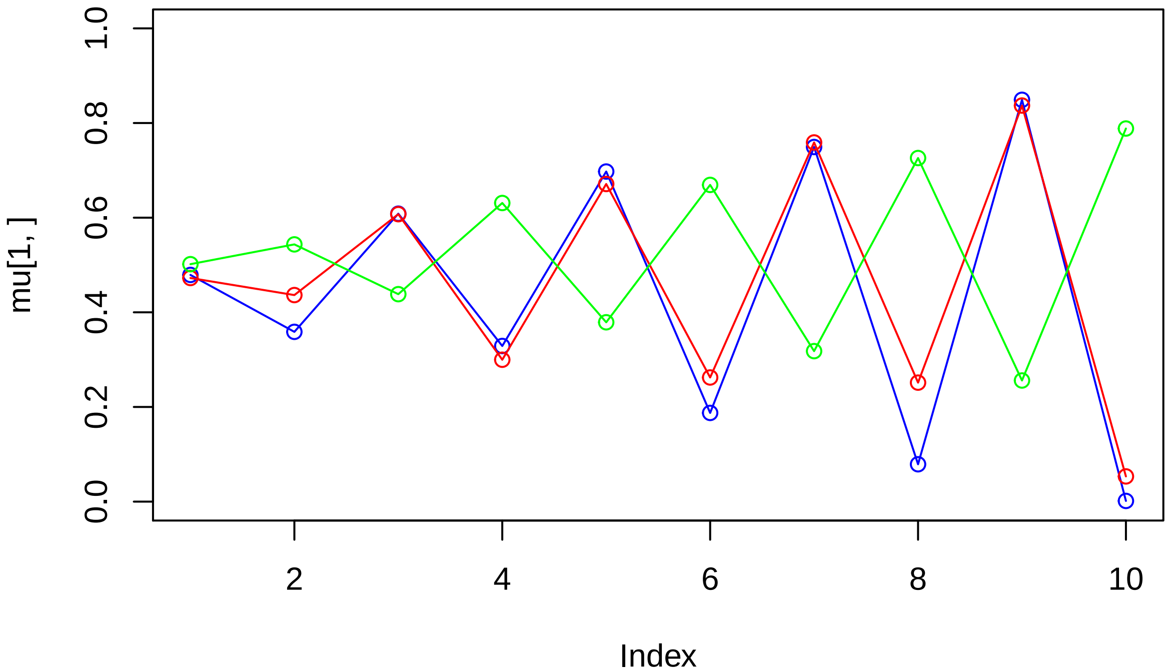

```
## iteration:  27 log likelihood:  -417.9753
```



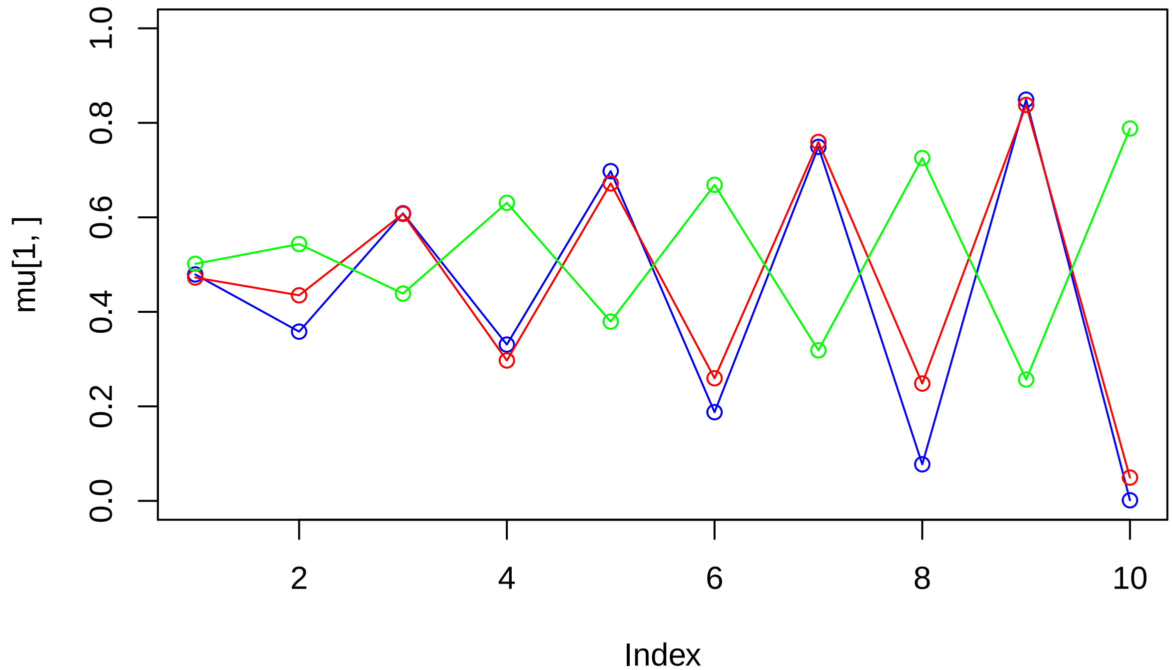

```
## iteration:  28 log likelihood:  -418.3836
```



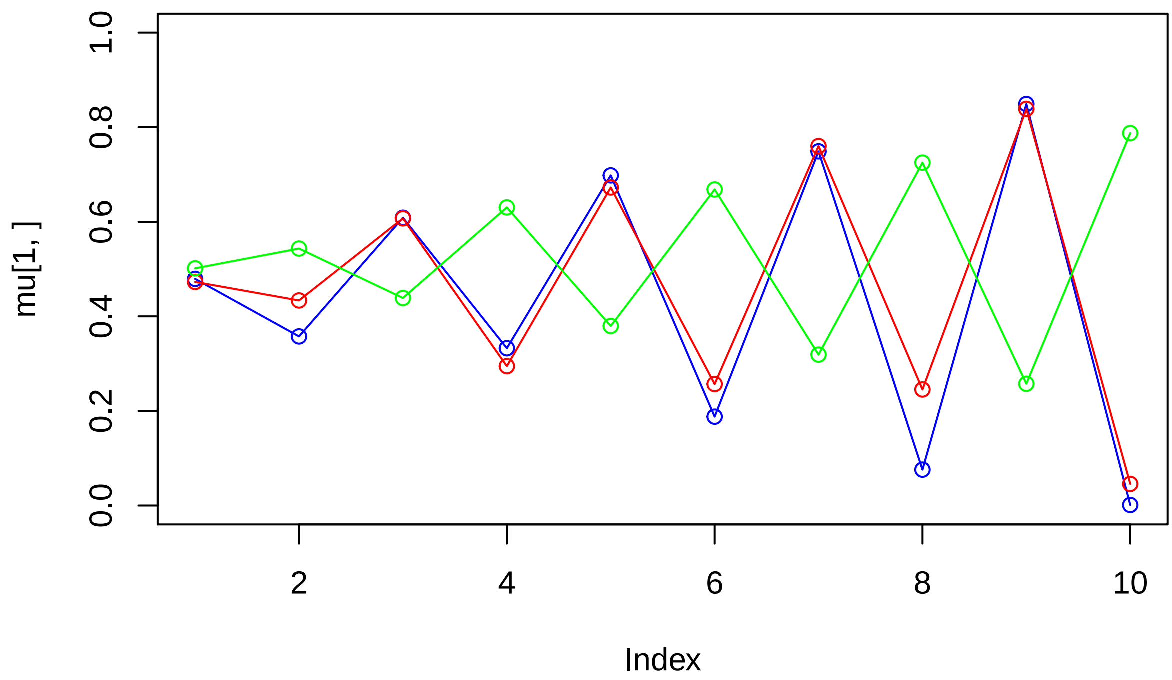

```
## iteration:  29 log likelihood:  -418.7601
```



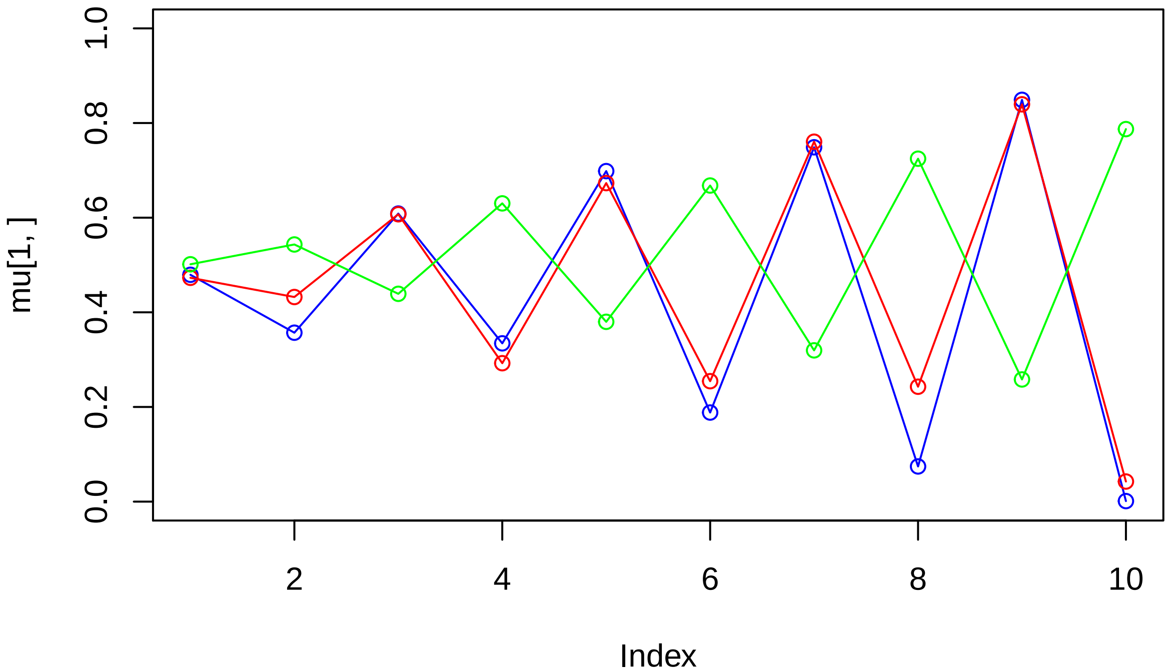

```
## iteration:  30 log likelihood:  -419.1074
```



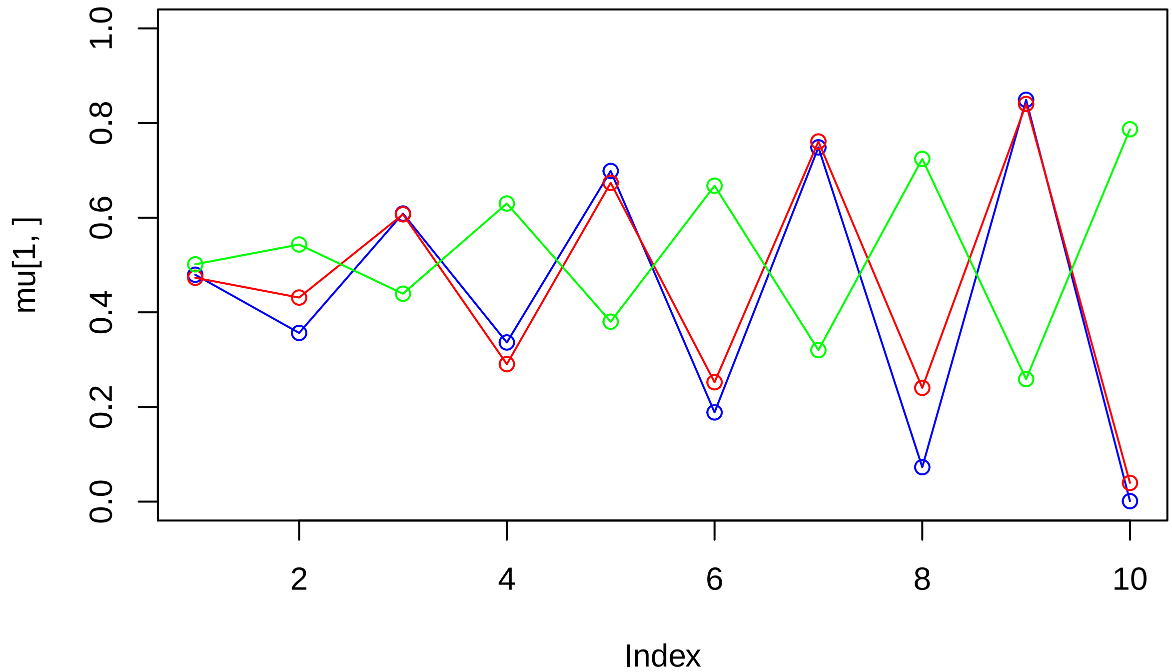

```
## iteration:  31 log likelihood:  -419.4277
```



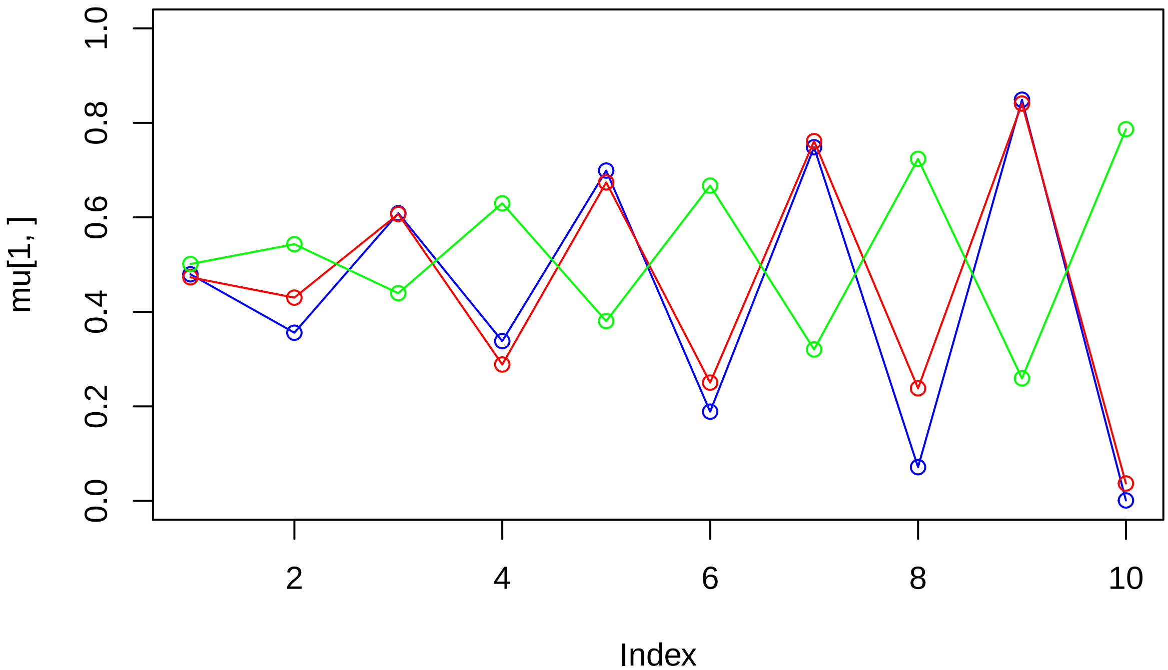

```
## iteration:  32 log likelihood:  -419.7229
```



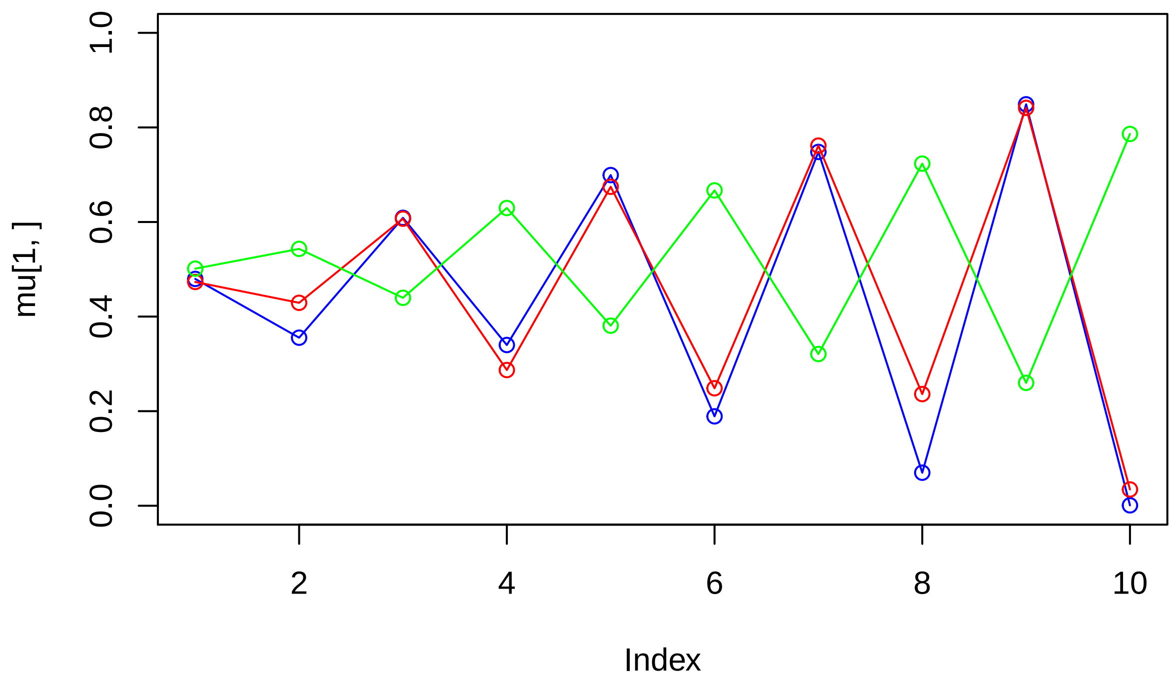

```
## iteration:  33 log likelihood:  -419.995
```



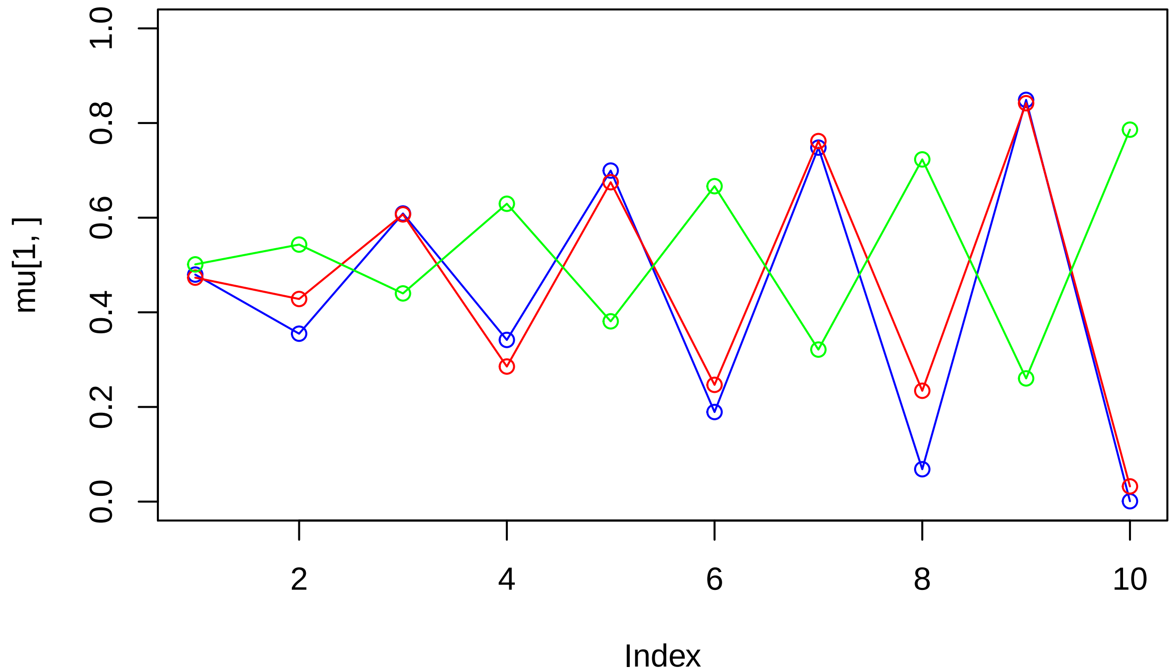

```
## iteration:  34 log likelihood:  -420.2457
```



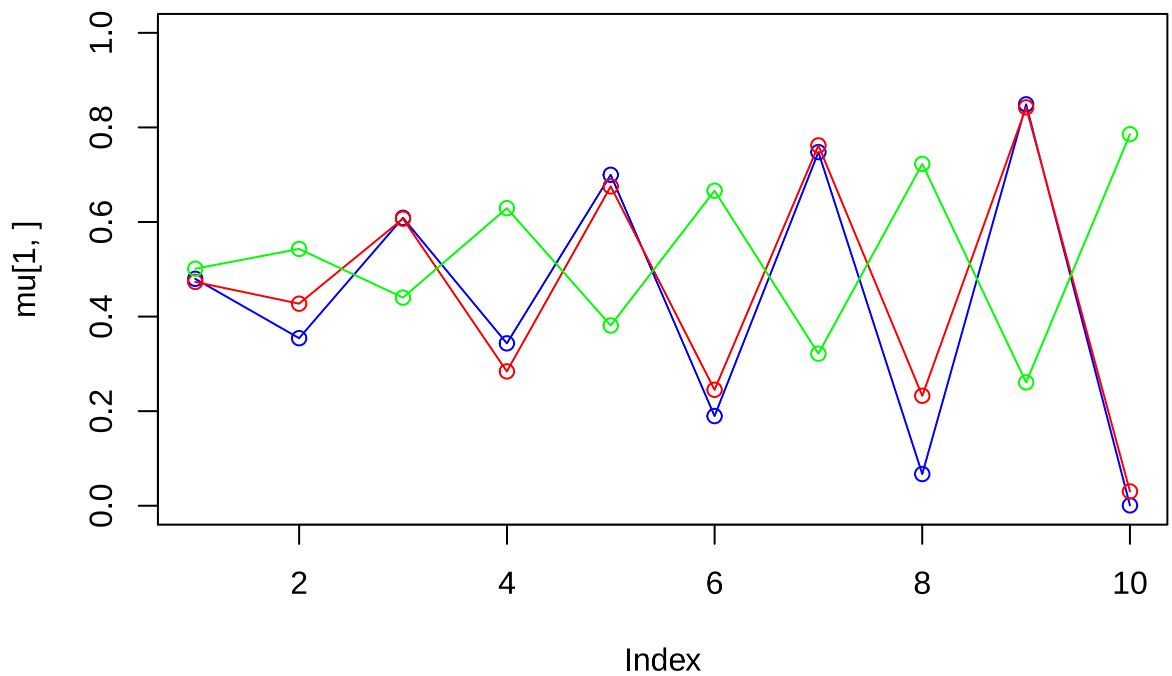

```
## iteration:  35 log likelihood:  -420.4767
```



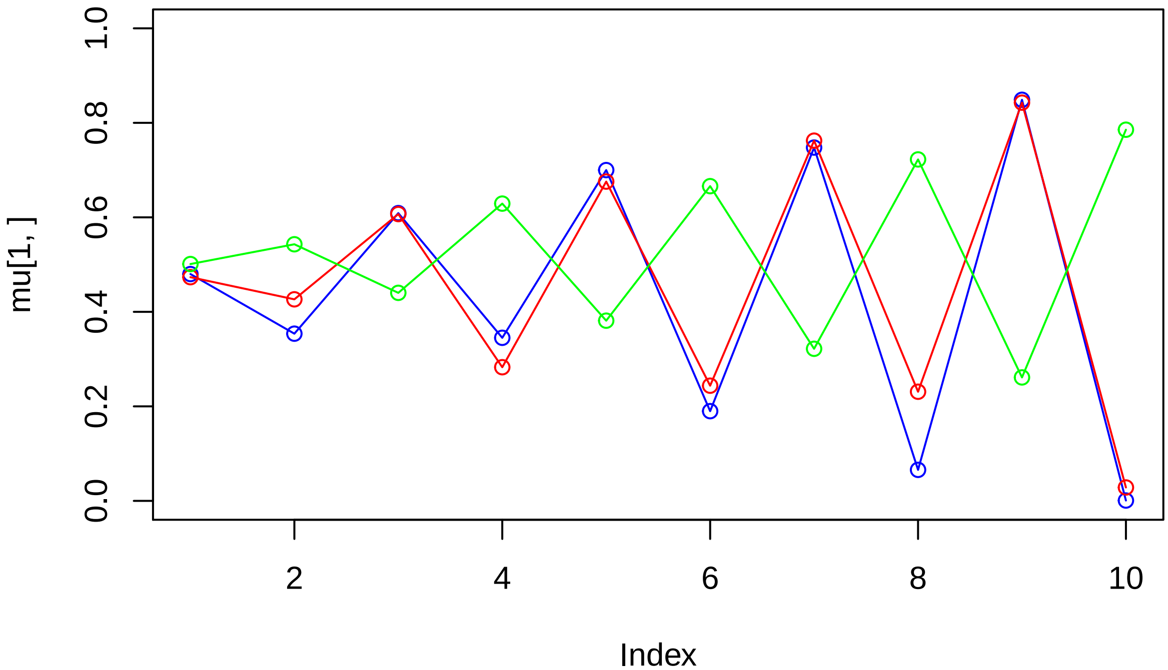

```
## iteration:  36 log likelihood:  -420.6895
```



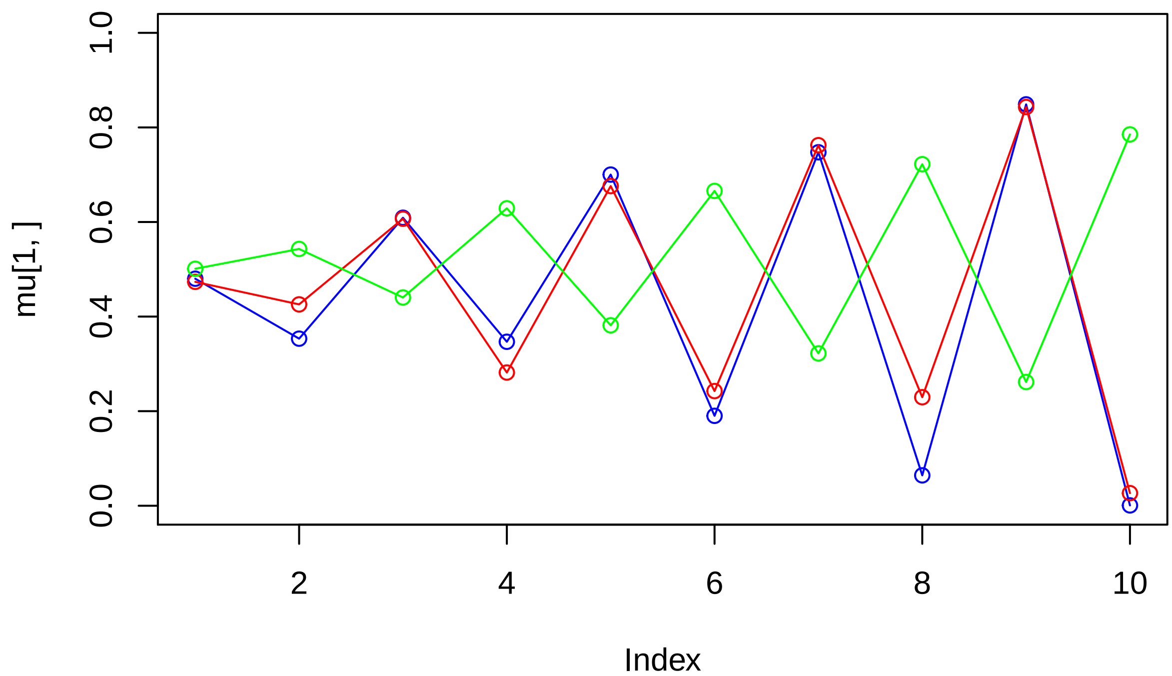

```
## iteration:  37 log likelihood:  -420.8856
```



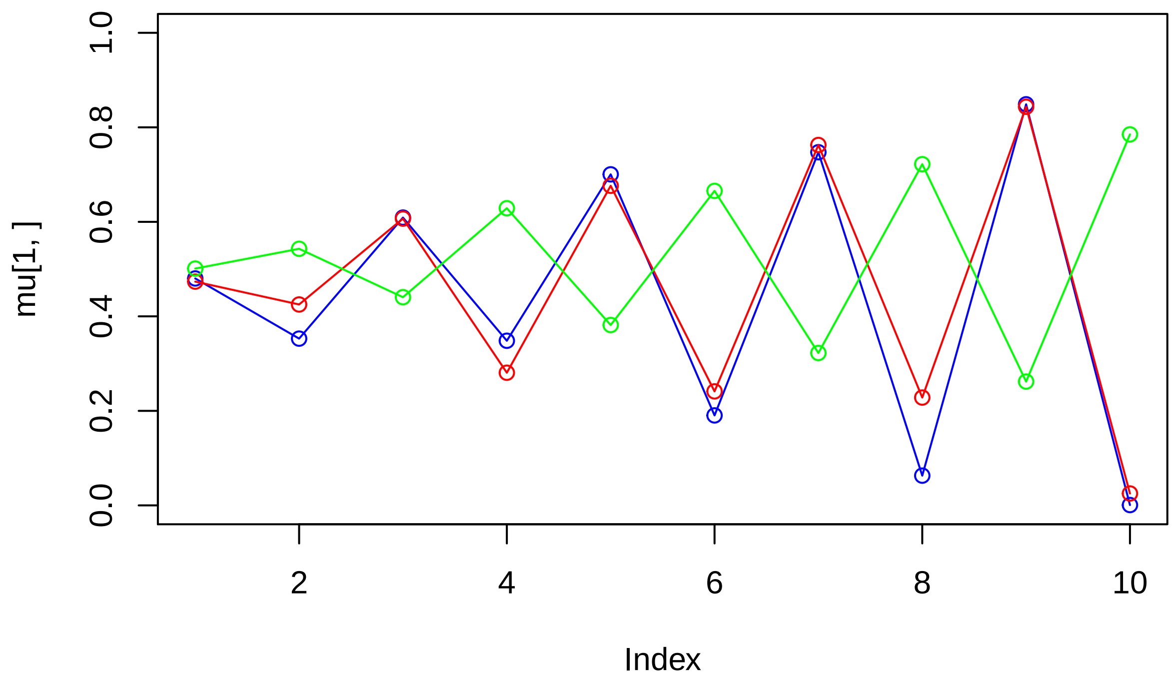

```
## iteration:  38 log likelihood:  -421.0663
```



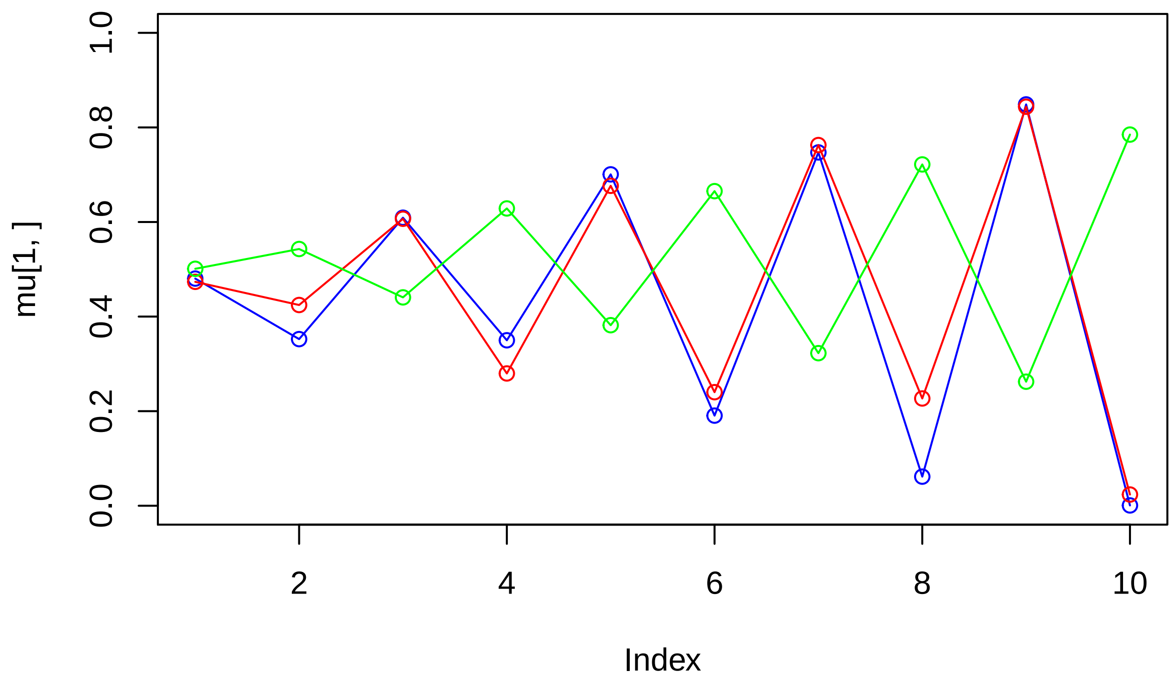

```
## iteration:  39 log likelihood:  -421.2329
```



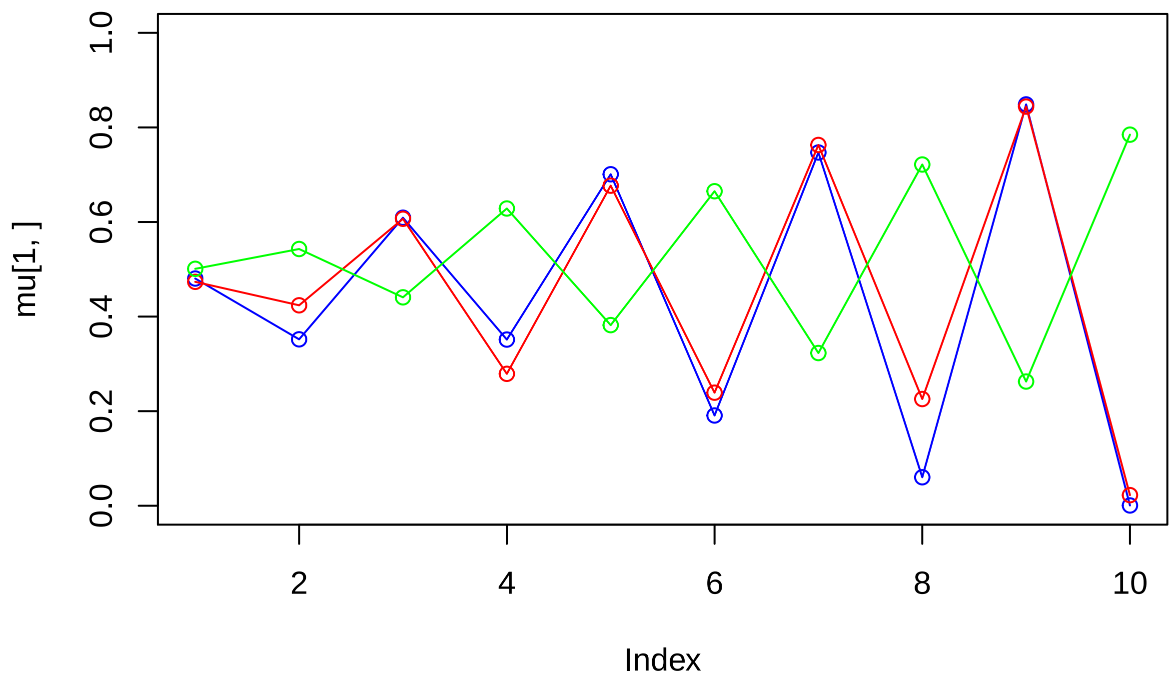

```
## iteration:  40 log likelihood:  -421.3865
```



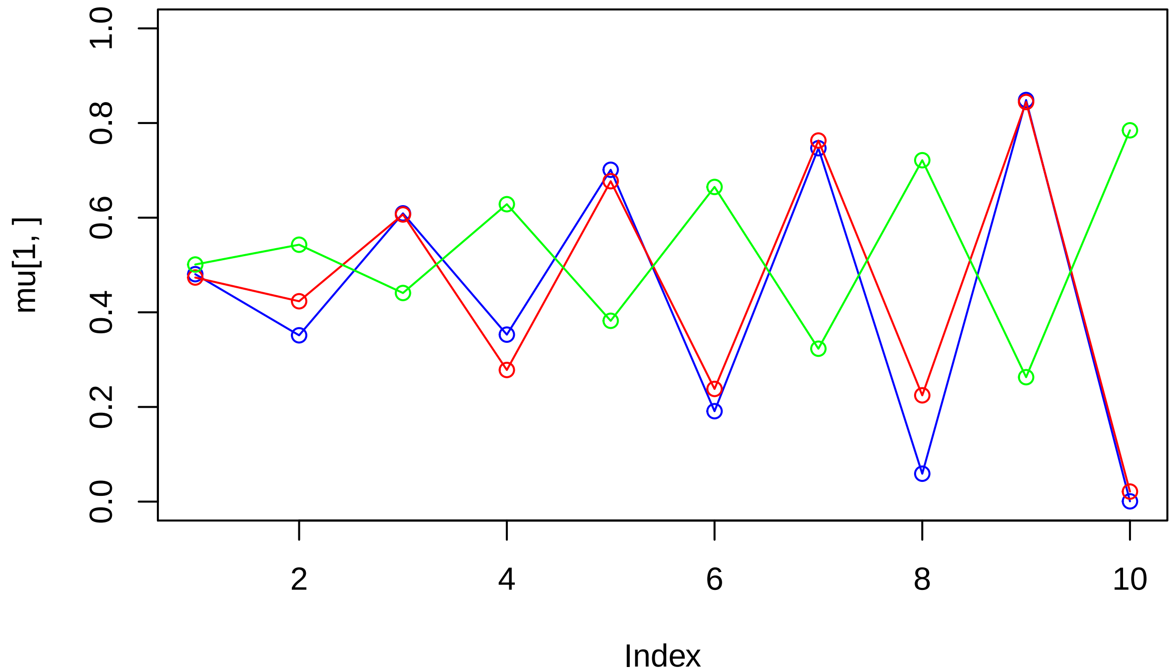

```
## iteration:  41 log likelihood:  -421.5282
```



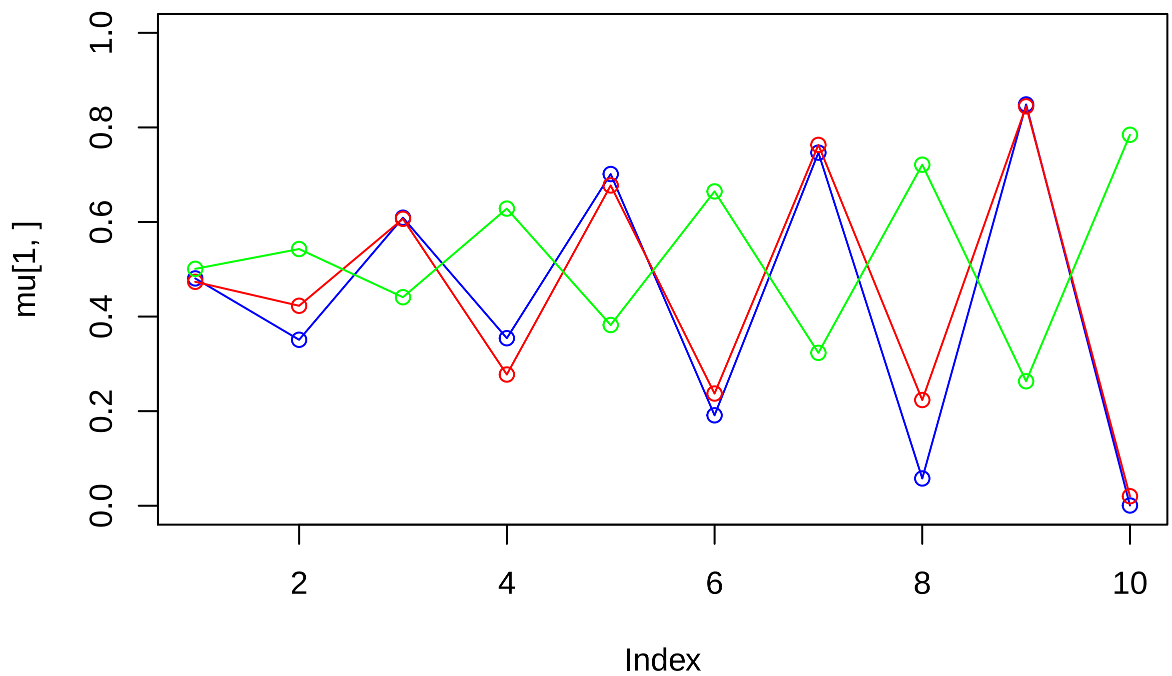

```
## iteration:  42 log likelihood:  -421.659
```



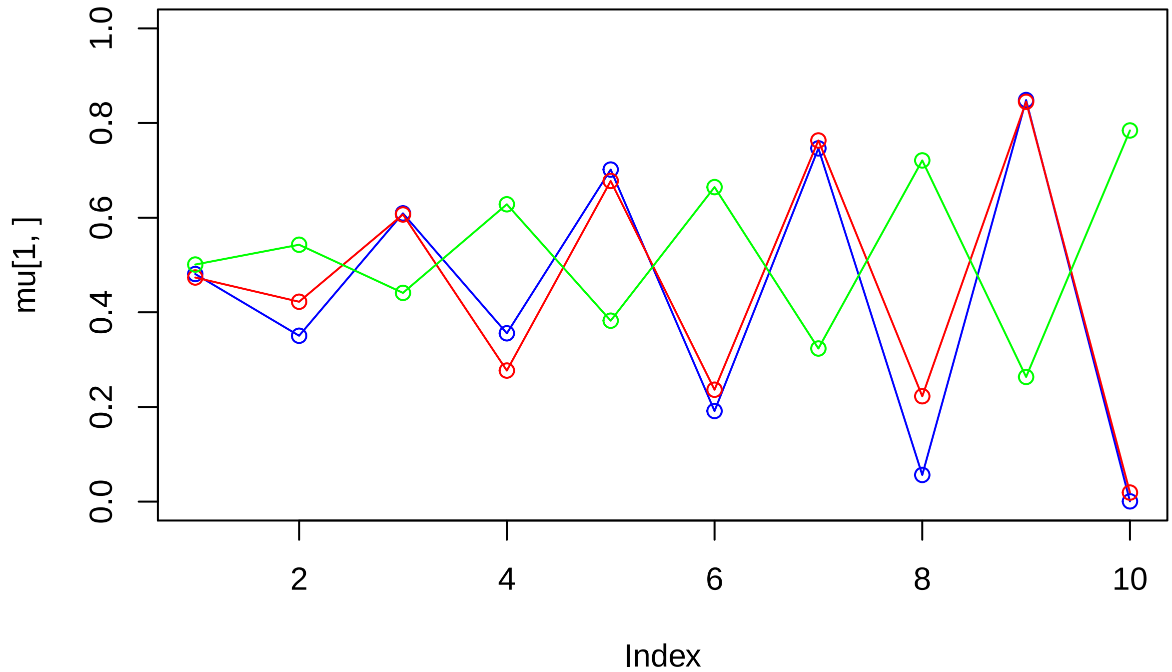

```
## iteration:  43 log likelihood:  -421.7797
```



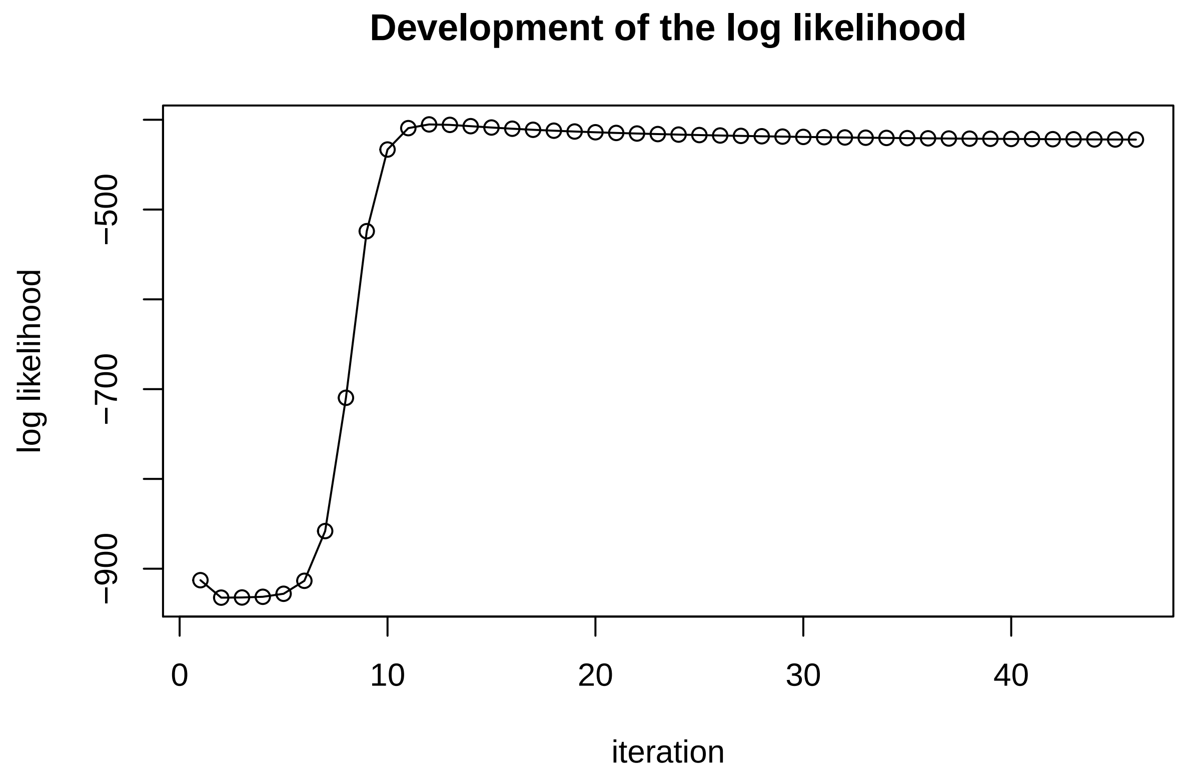

```
## iteration:  44 log likelihood:  -421.8913
```




```
## iteration:  45 log likelihood:  -421.9945
```



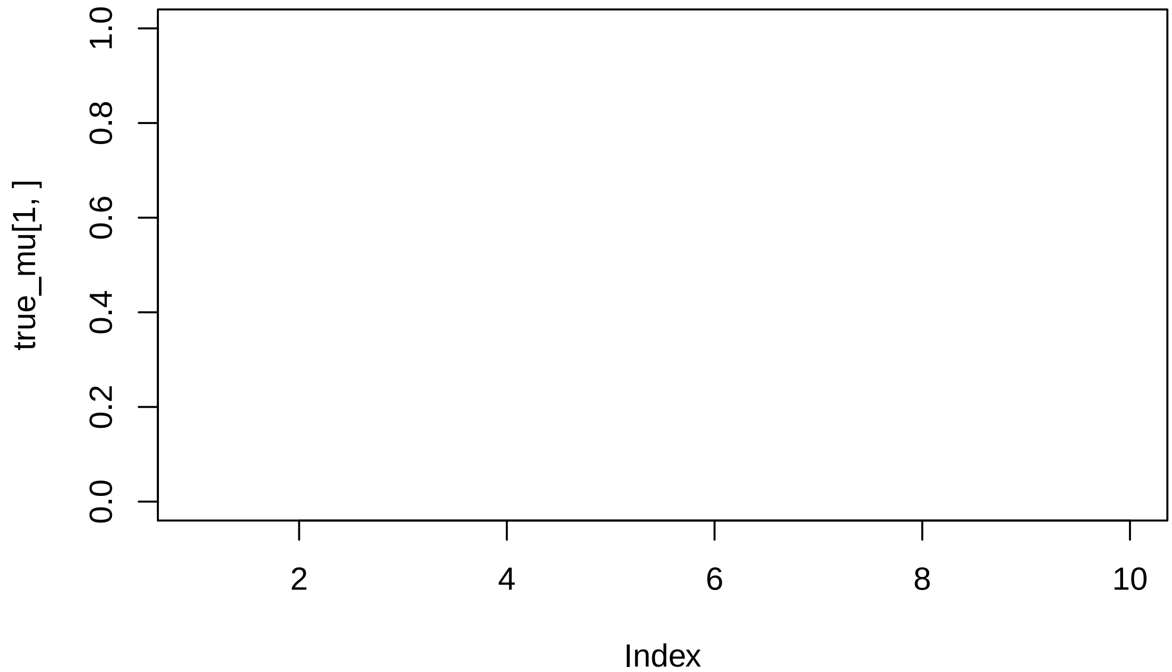

```
## iteration:  46 log likelihood:  -422.09
```



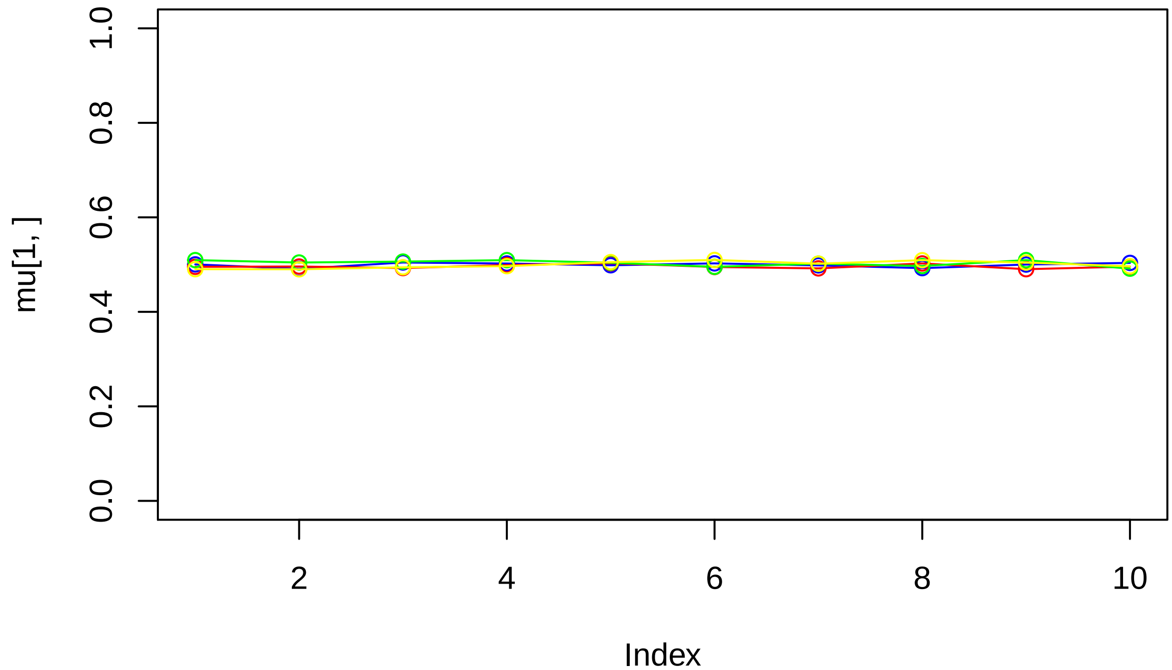

```
## $pi
## [1] 0.1679717 0.2034249 0.6286034
##
## $mu
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4808697 0.3505033 0.6091318 0.3556548 0.7016525 0.1914725 0.7465112
## [2,] 0.4735293 0.4223595 0.6067582 0.2768902 0.6775124 0.2364292 0.7631736
## [3,] 0.5009515 0.5428016 0.4410625 0.6282717 0.3823068 0.6645565 0.3235088
##            [,8]      [,9]        [,10]
## [1,] 0.05638549 0.8485479 0.0005534402
## [2,] 0.22264183 0.8448195 0.0190935069
## [3,] 0.72102367 0.2634581 0.7843147843
##
## $logLikelihoodDevelopment
## NULL
```

4. $\mathbf{K = 4}$

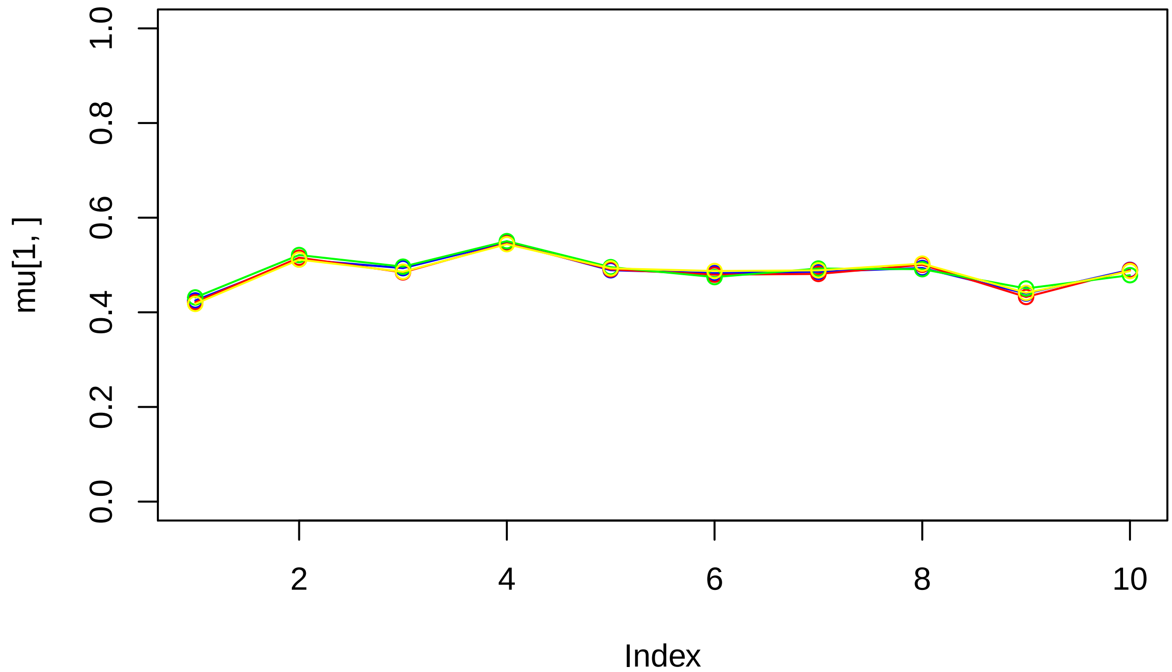
```r
myem(K=4)
```



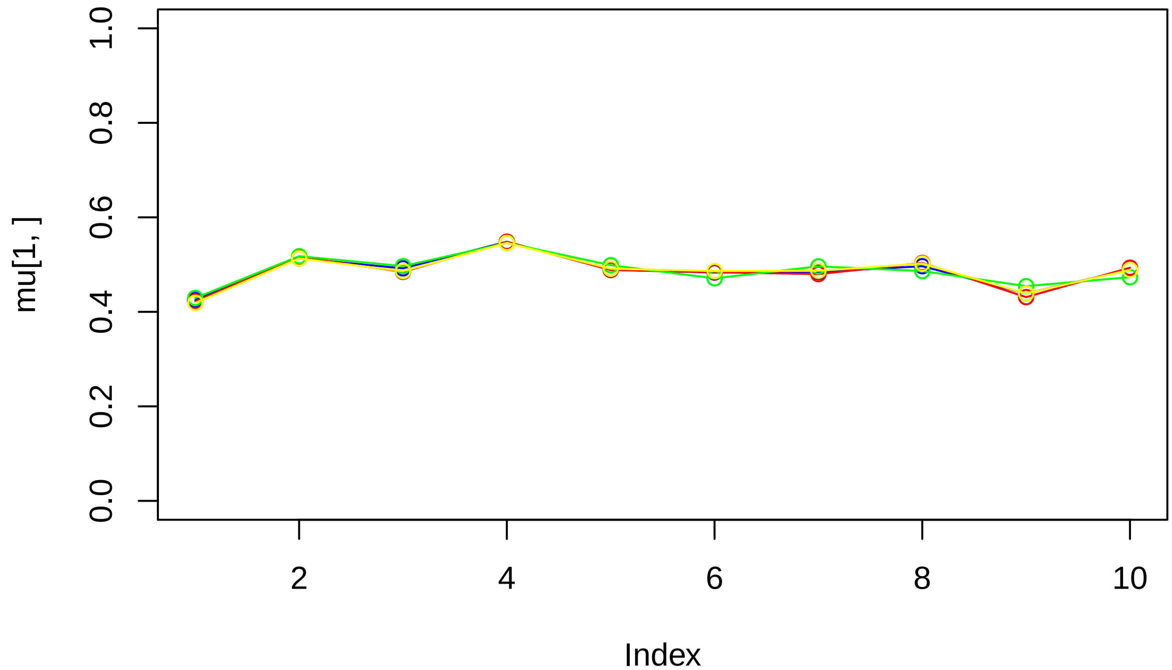
true_mu[1,]
1.0
0.8
0.6
0.4
0.2
0.0
2
4
6
8
10
Index



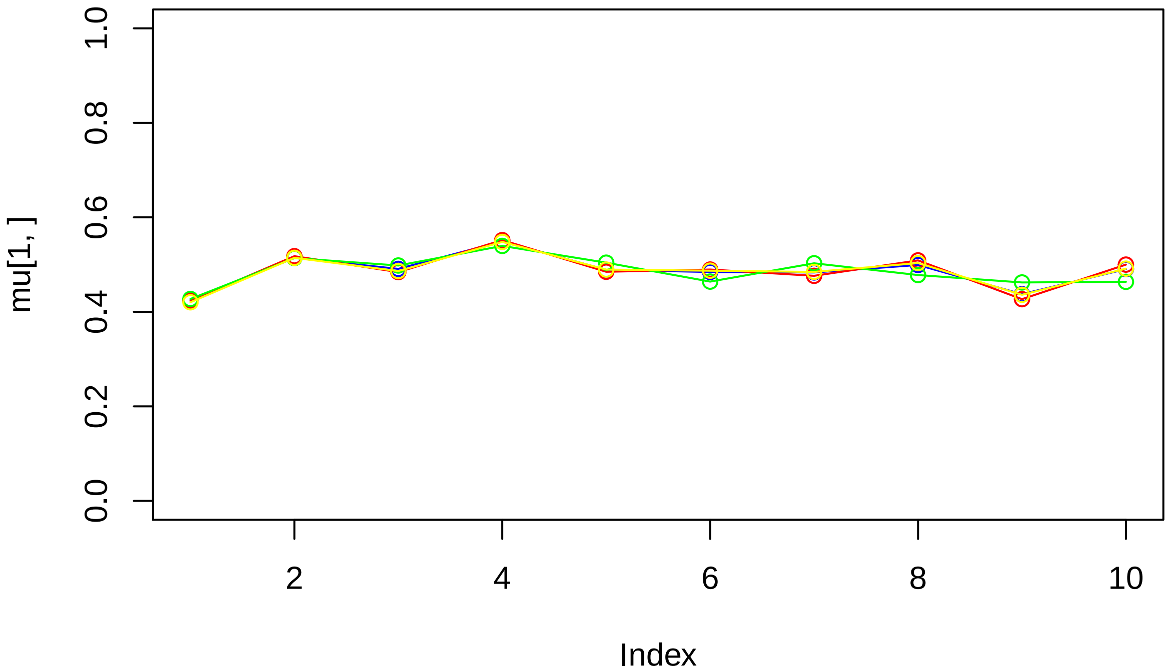

```
## iteration:  1 log likelihood:  -800.5436
```



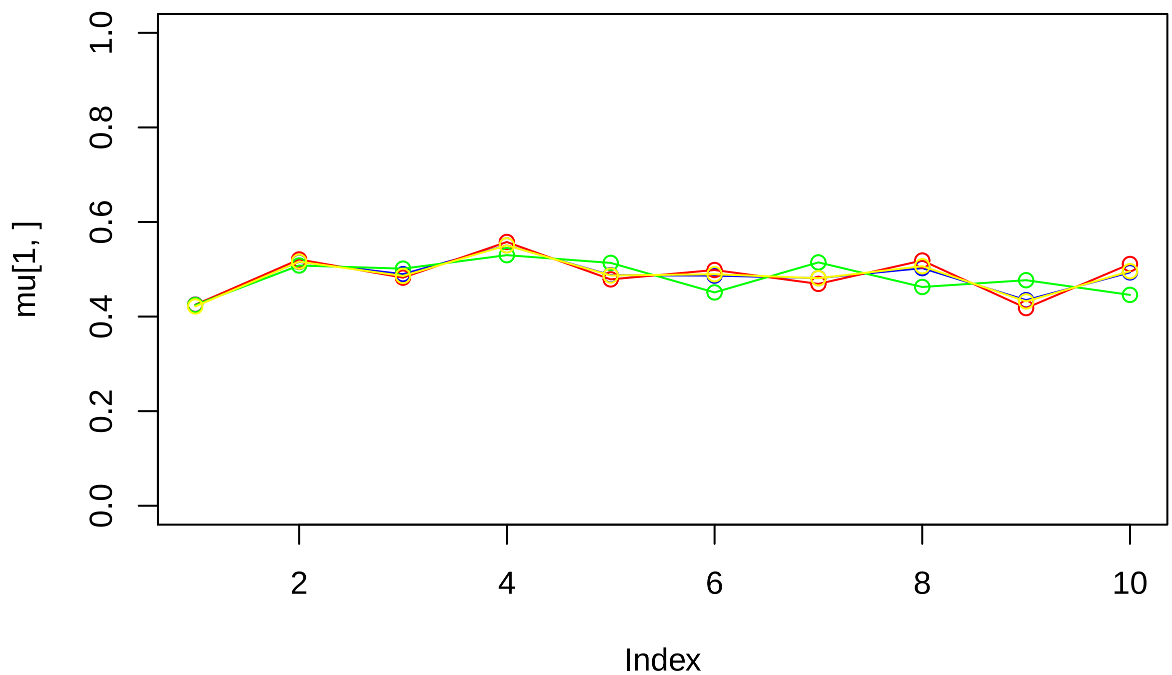

```
## iteration:  2 log likelihood:  -842.949
```



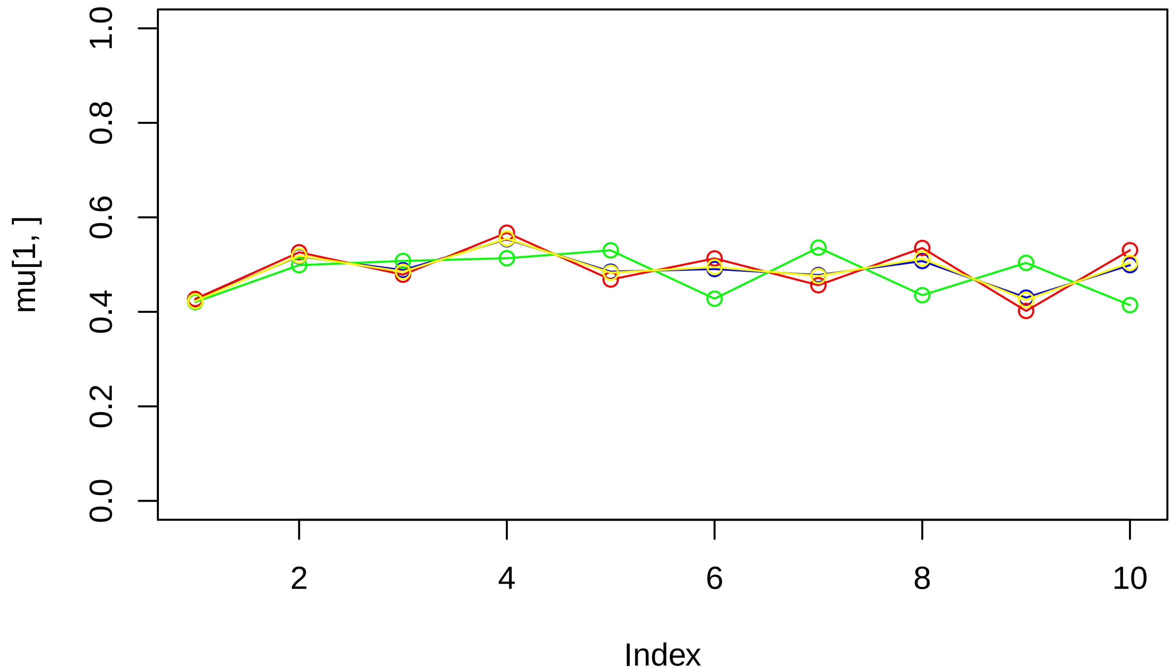

```
## iteration:  3 log likelihood:  -842.6806
```



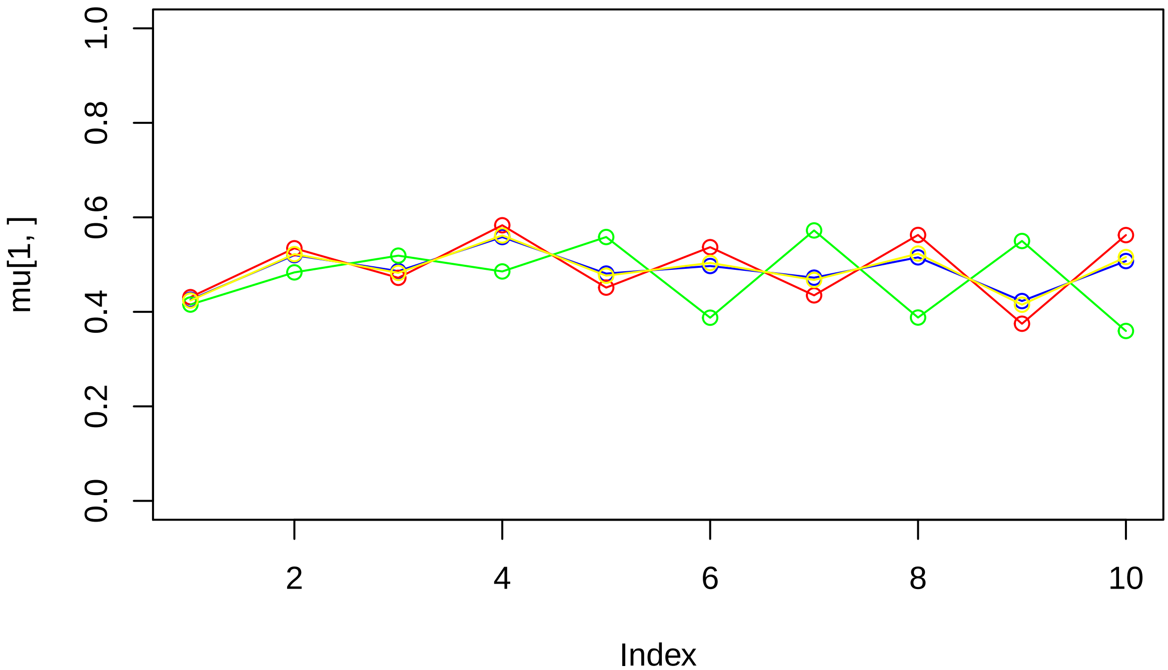

```
## iteration:  4 log likelihood:  -841.7499
```



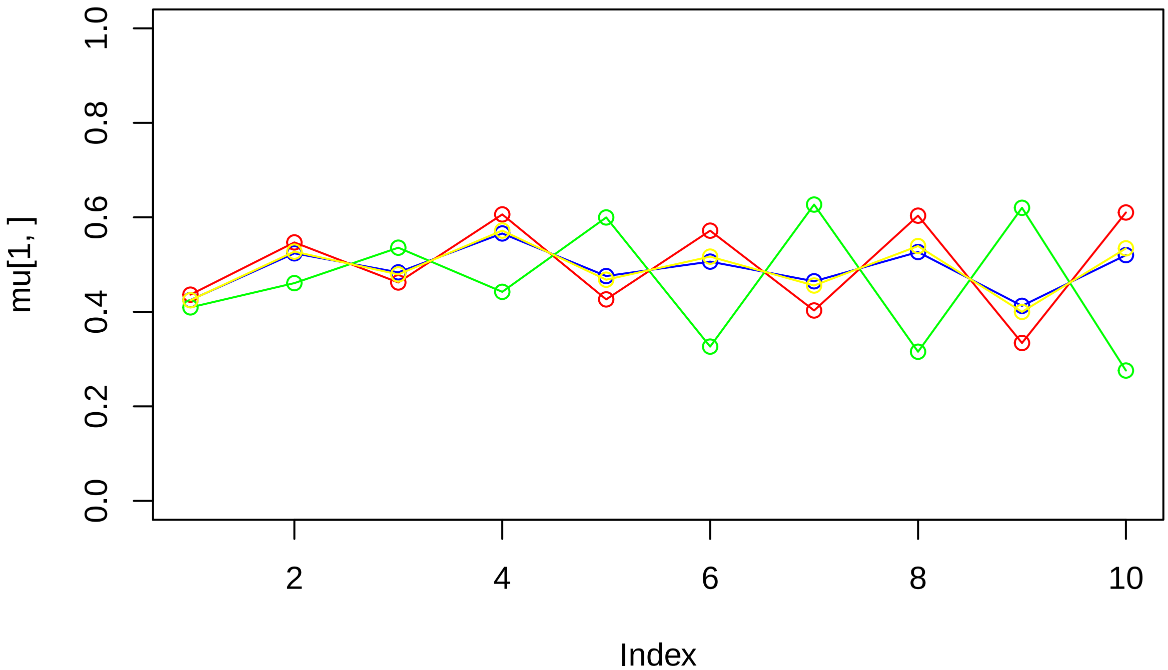

```
## iteration:  5 log likelihood:  -838.7414
```



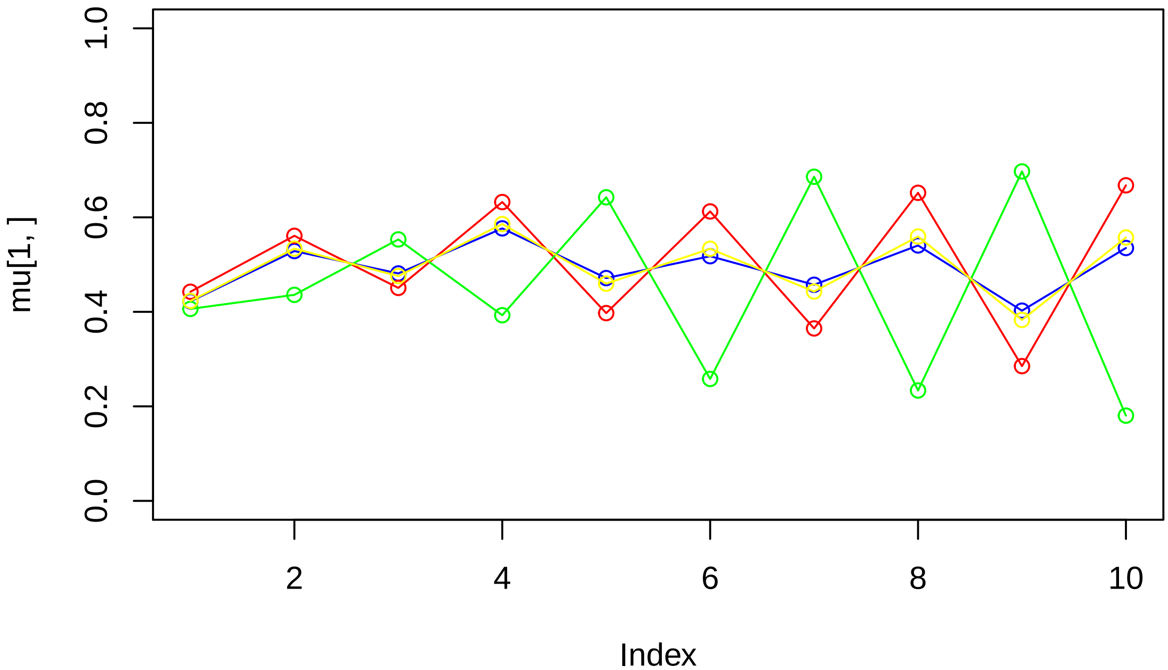

```
## iteration:  6 log likelihood:  -829.4624
```



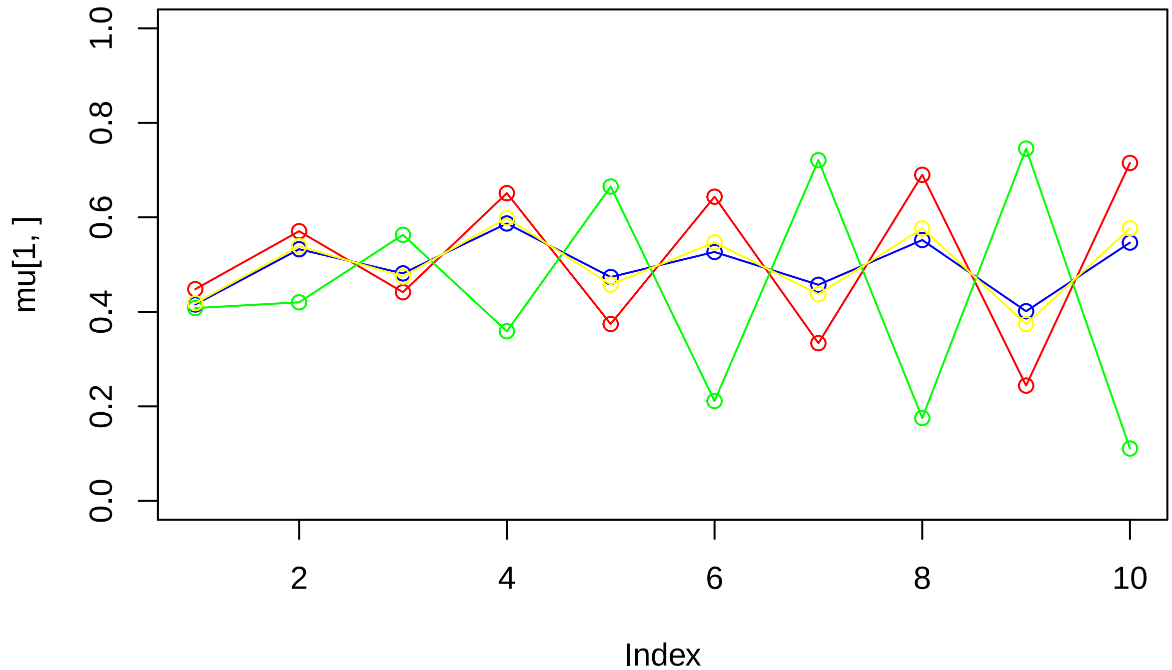

```
## iteration:  7 log likelihood:  -803.3592
```



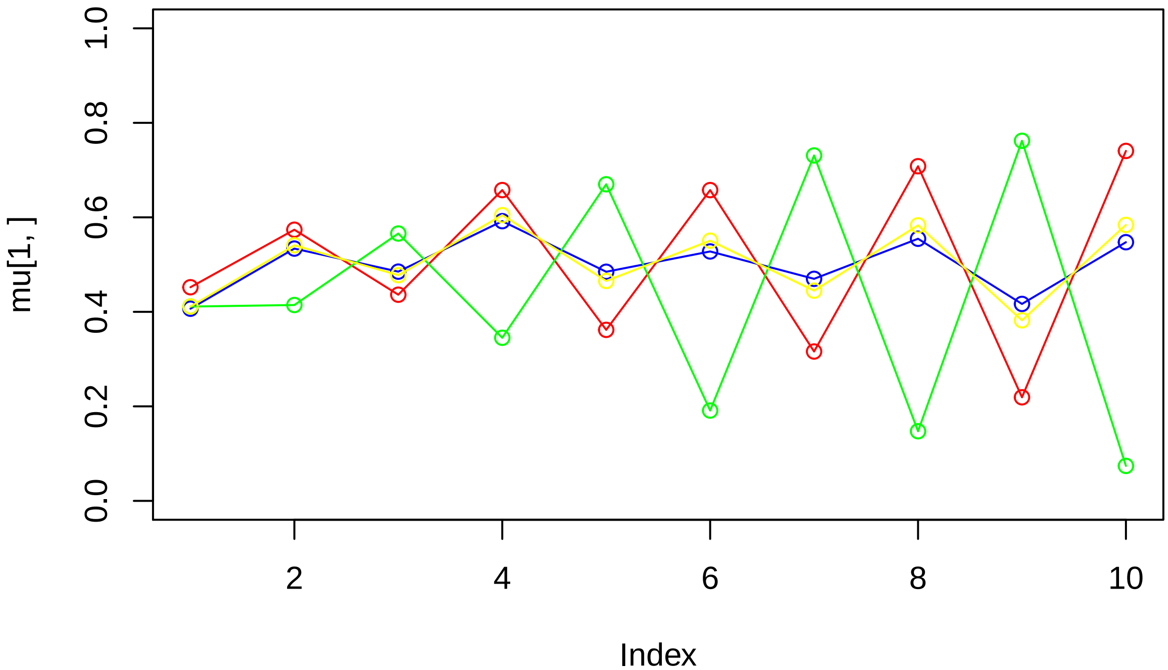

```
## iteration:  8 log likelihood:  -744.3623
```



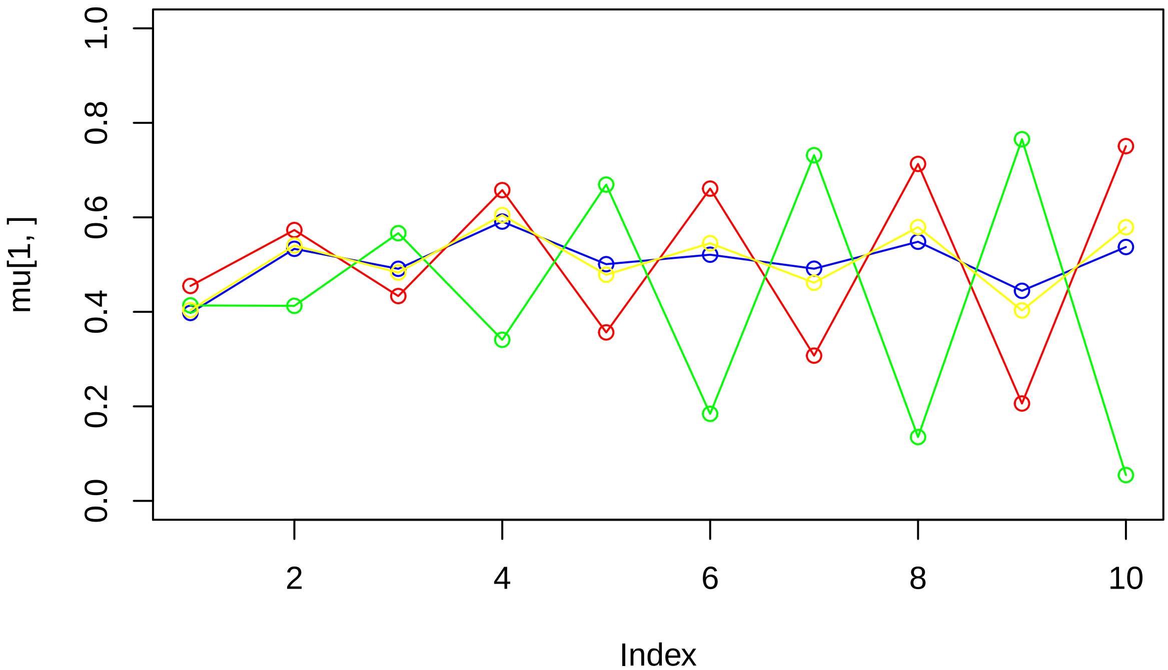

```
## iteration:  9 log likelihood:  -658.0191
```



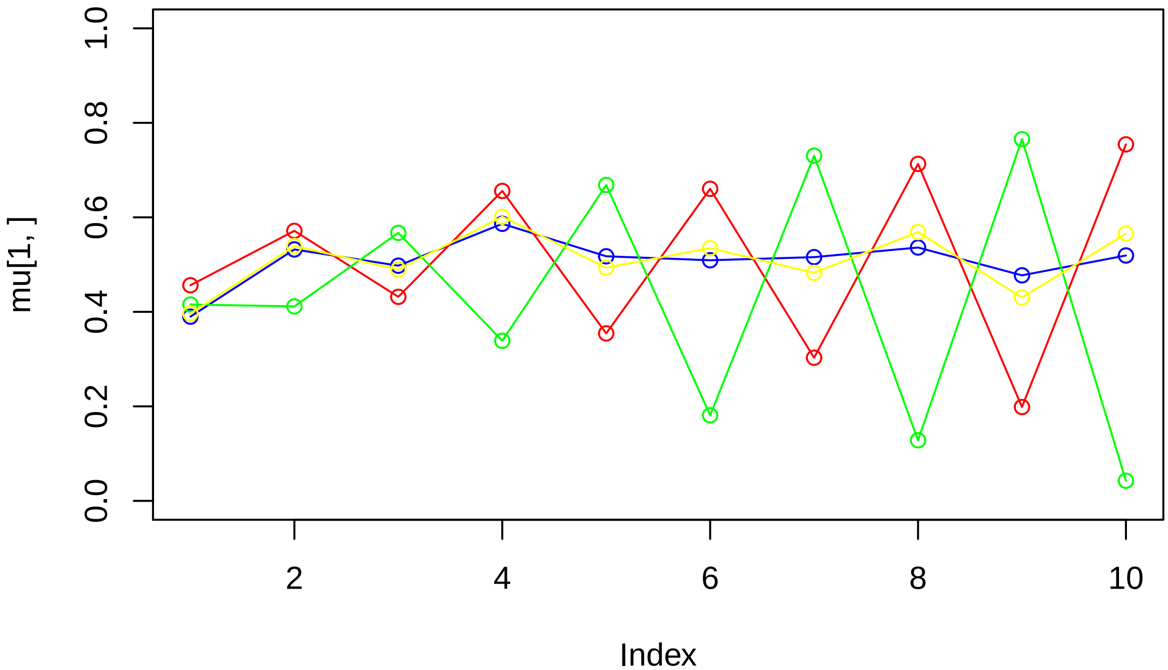

```
## iteration:  10 log likelihood:  -588.2999
```



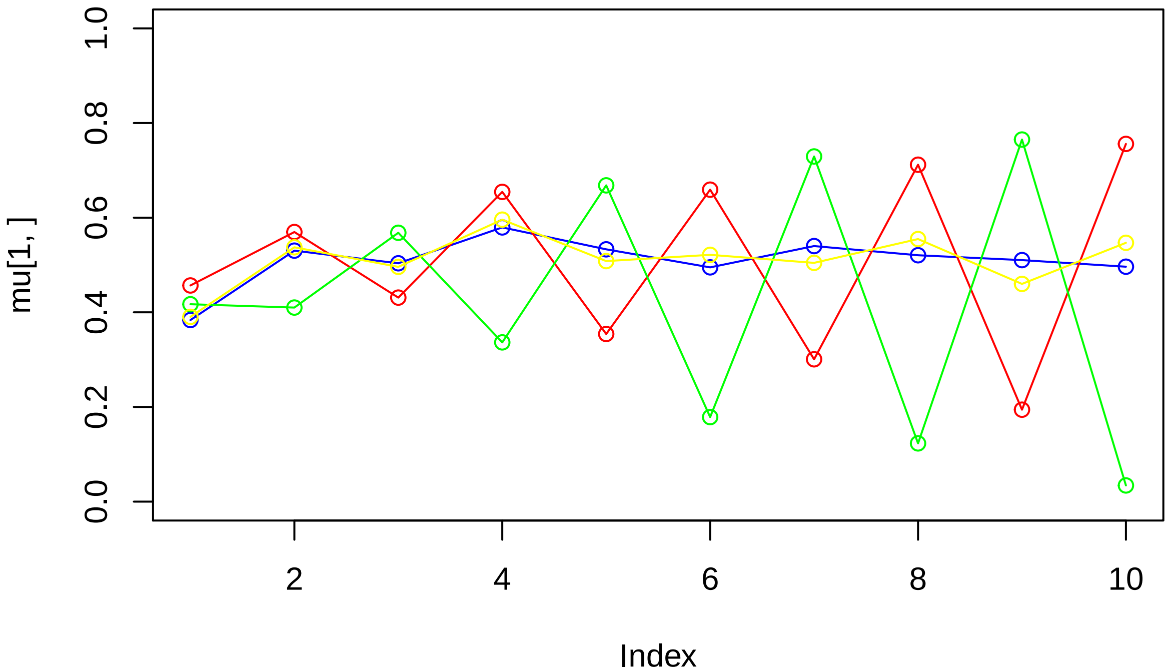

```
## iteration:  11 log likelihood:  -553.5615
```



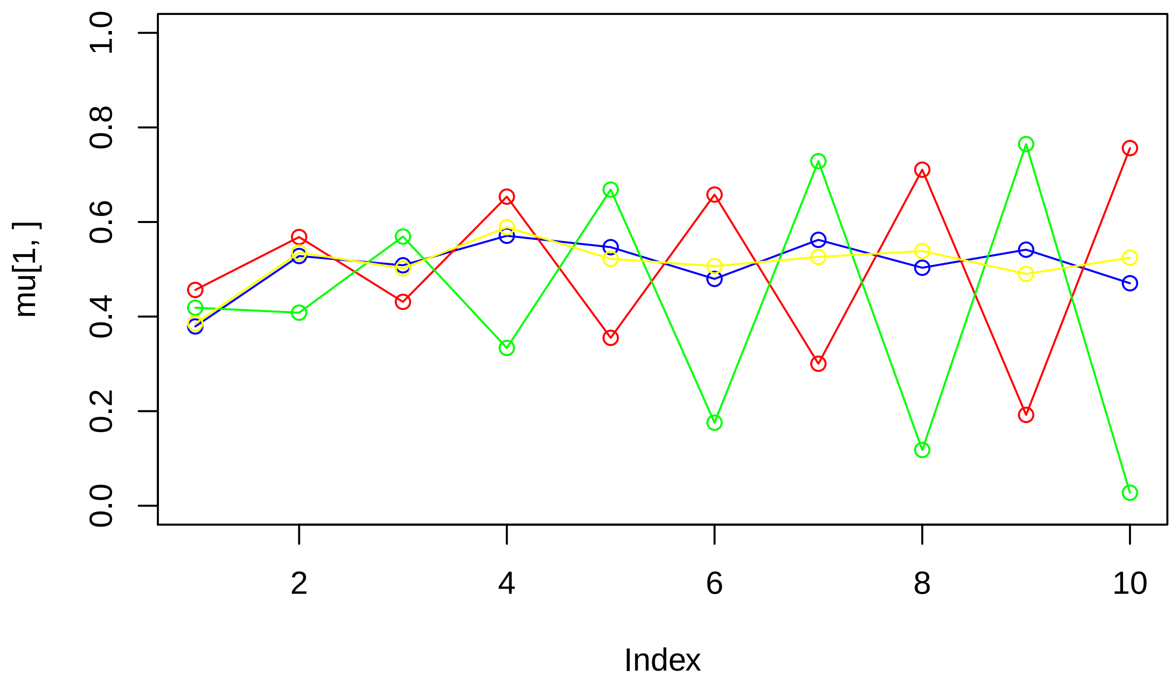

```
## iteration:  12 log likelihood:  -538.8823
```



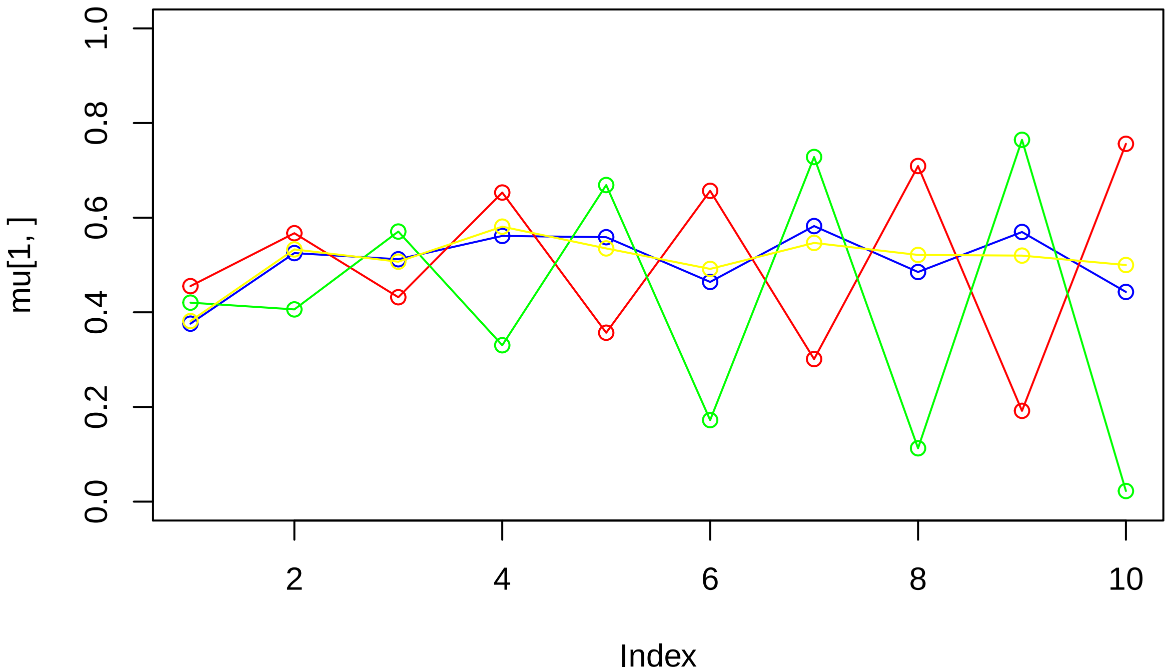

```
## iteration:  13 log likelihood:  -531.9182
```



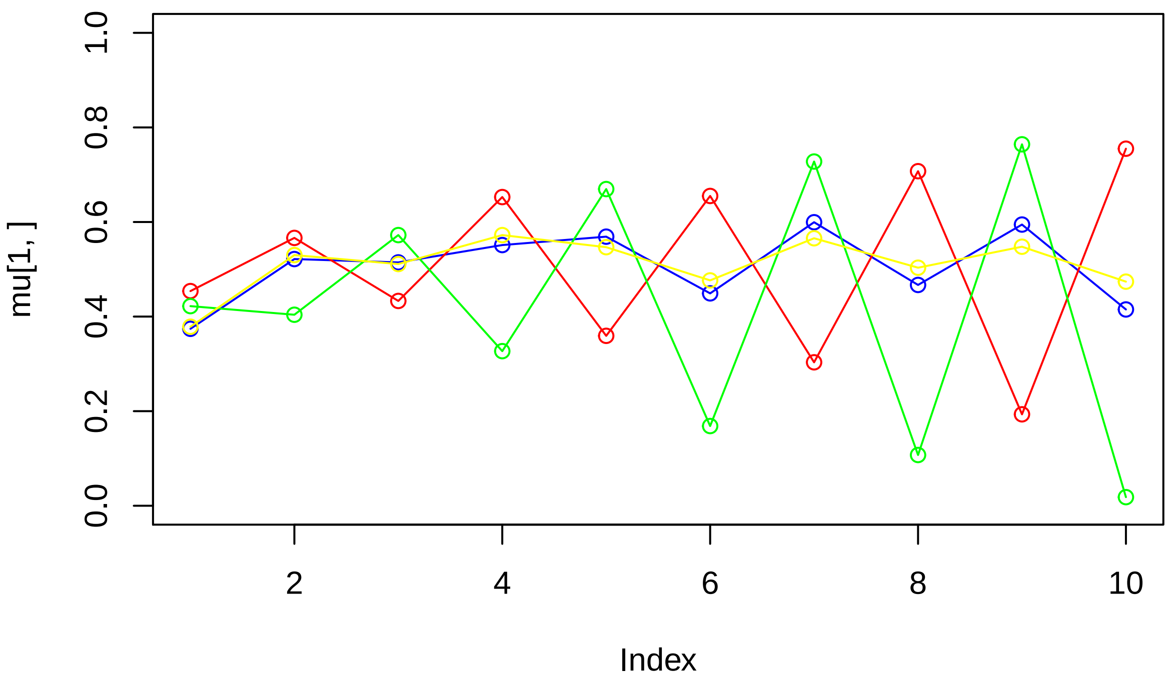

```
## iteration:  14 log likelihood:  -527.7567
```



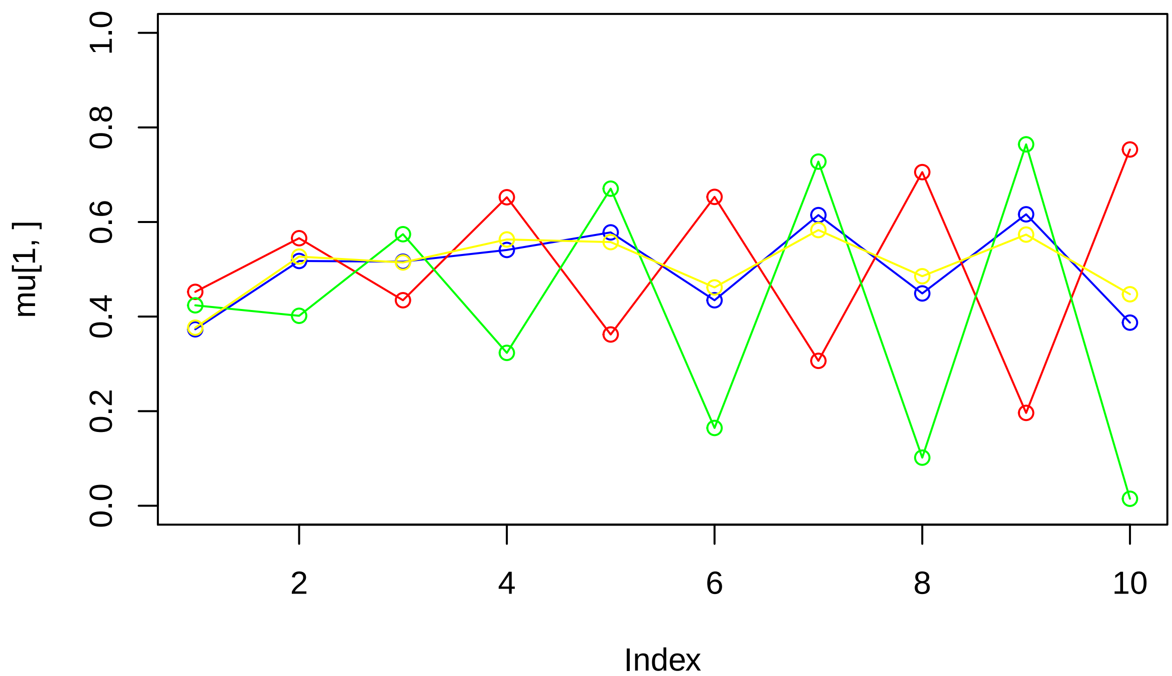

```
## iteration:  15 log likelihood:  -524.8526
```



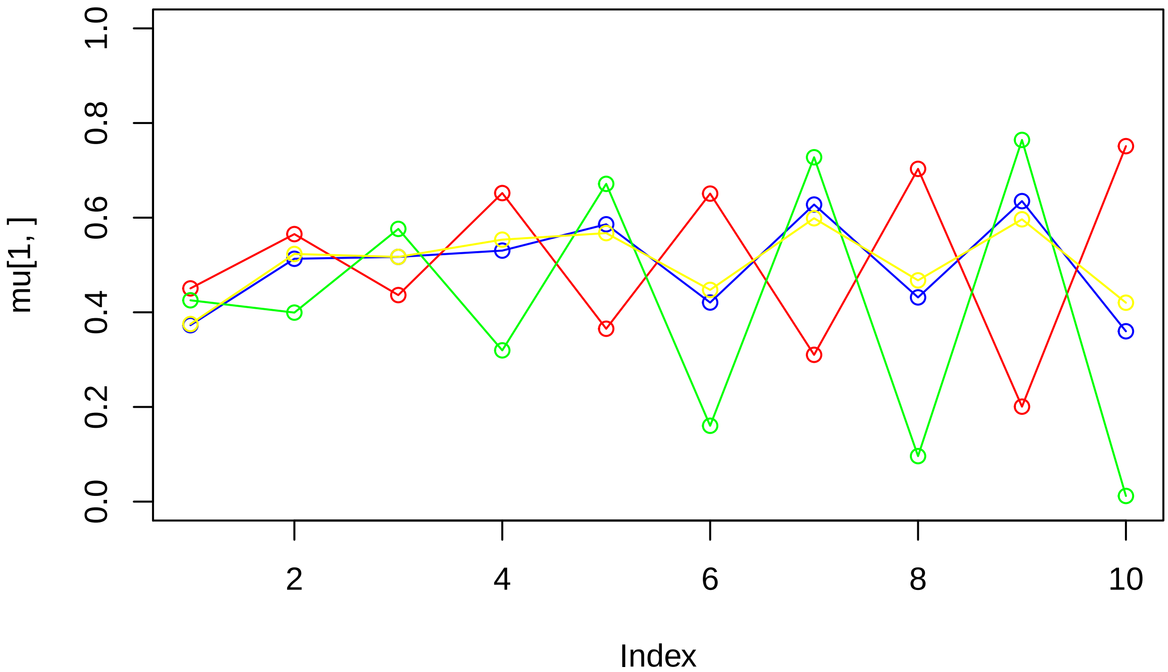

```
## iteration:  16 log likelihood:  -522.7751
```



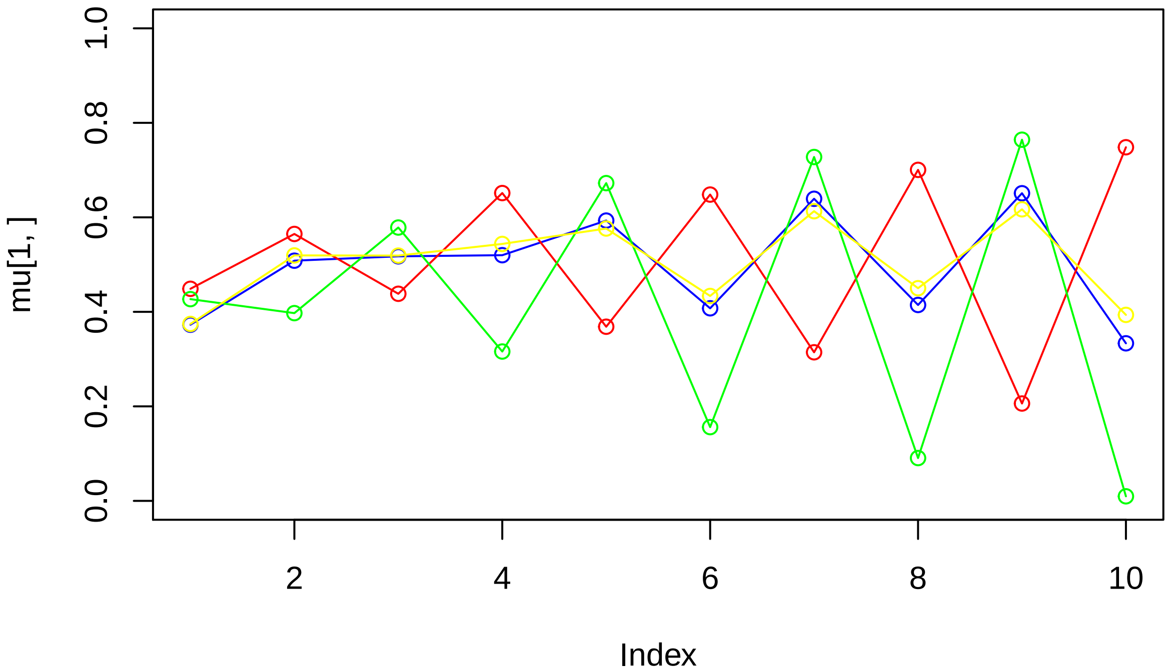

```
## iteration:  17 log likelihood:  -521.3929
```



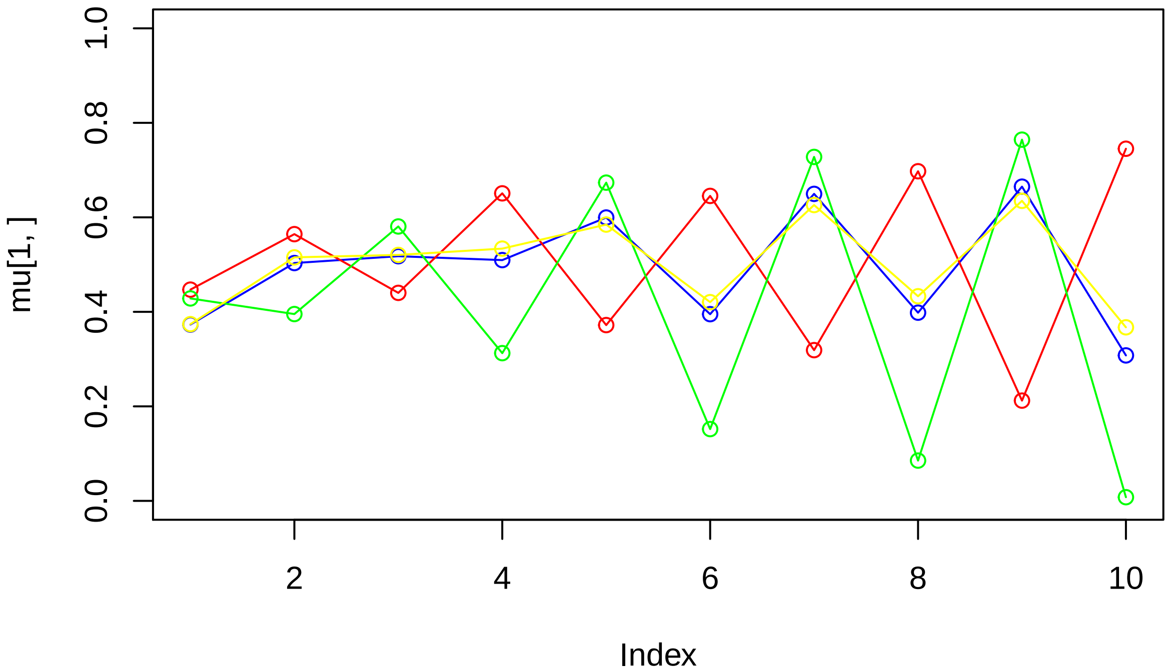

```
## iteration:  18 log likelihood:  -520.6263
```



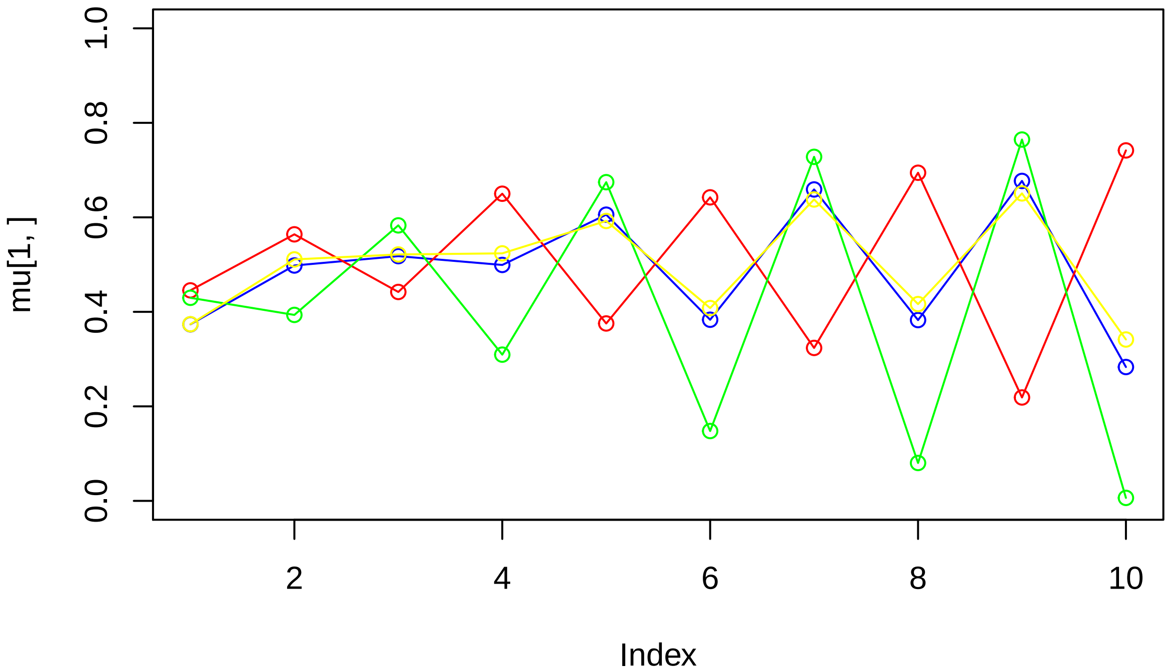

```
## iteration:  19 log likelihood:  -520.391
```



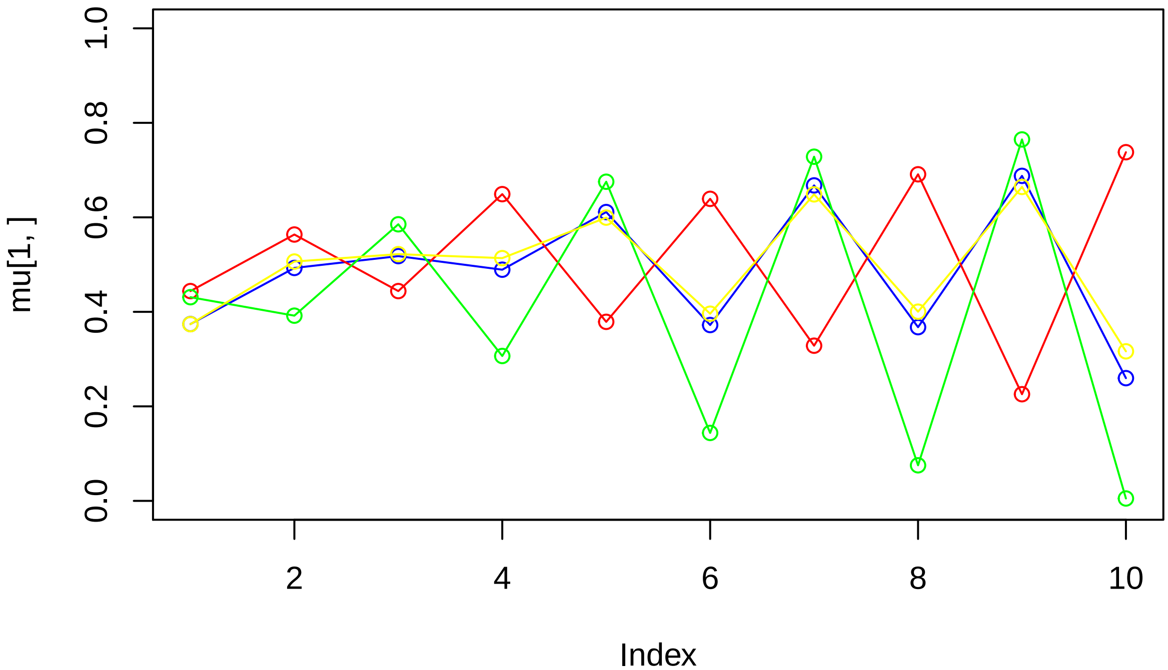

```
## iteration:  20 log likelihood:  -520.5983
```



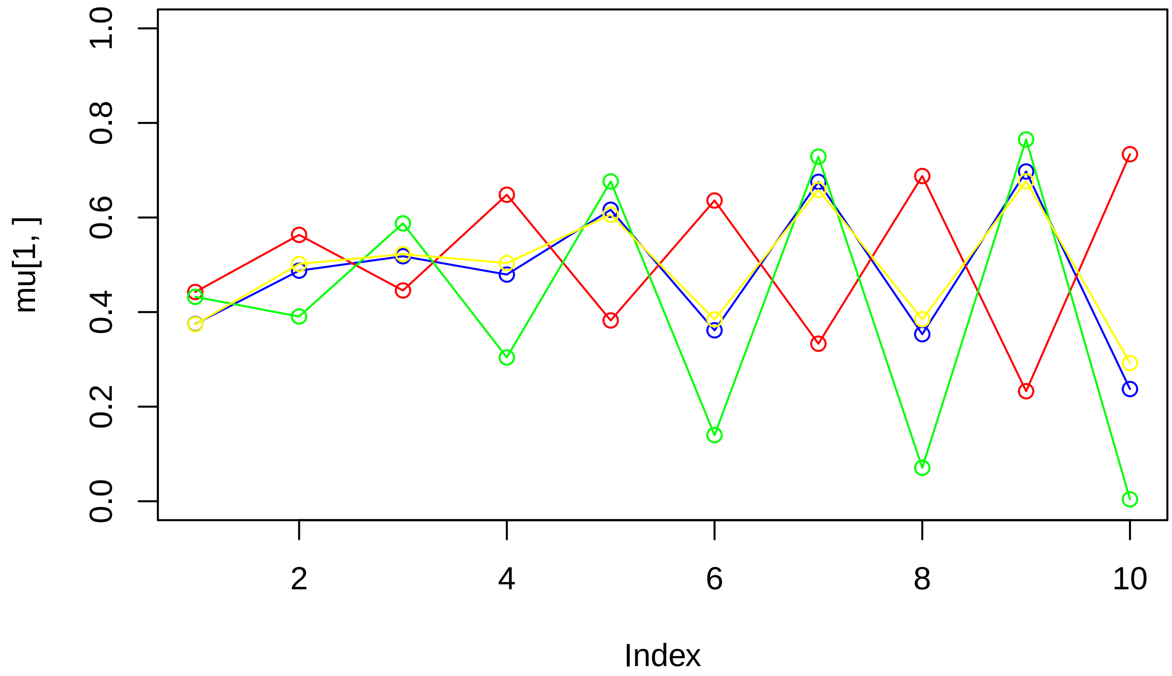

```
## iteration:  21 log likelihood:  -521.1652
```



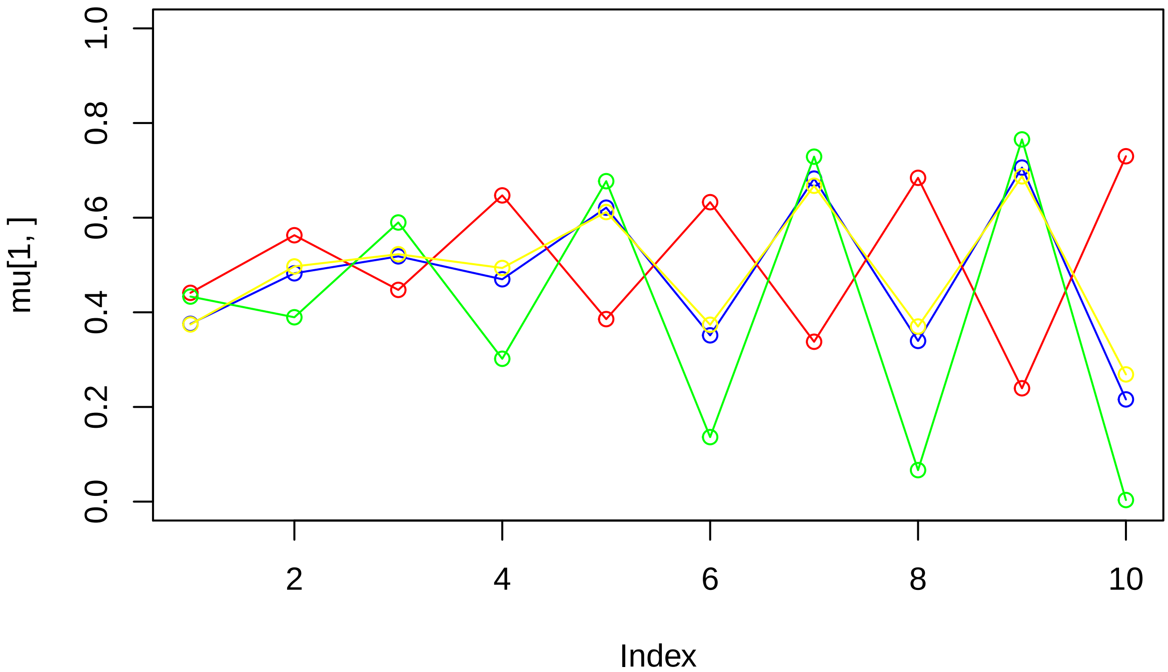

```
## iteration:  22 log likelihood:  -522.0204
```



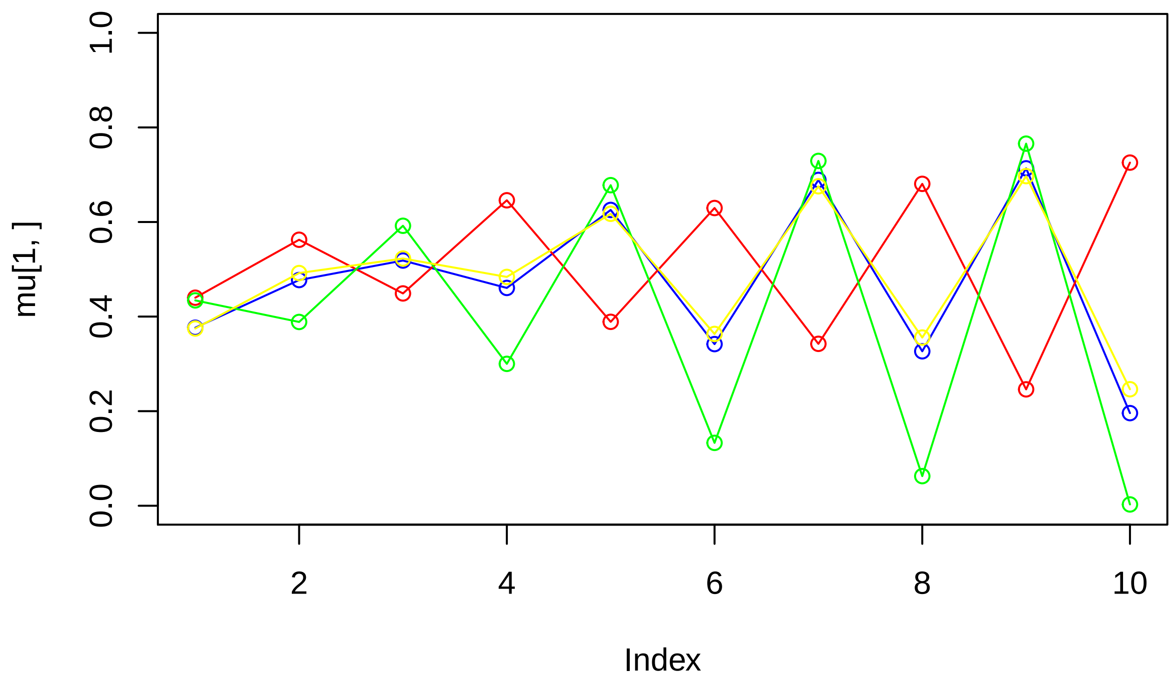

```
## iteration:  23 log likelihood:  -523.1059
```



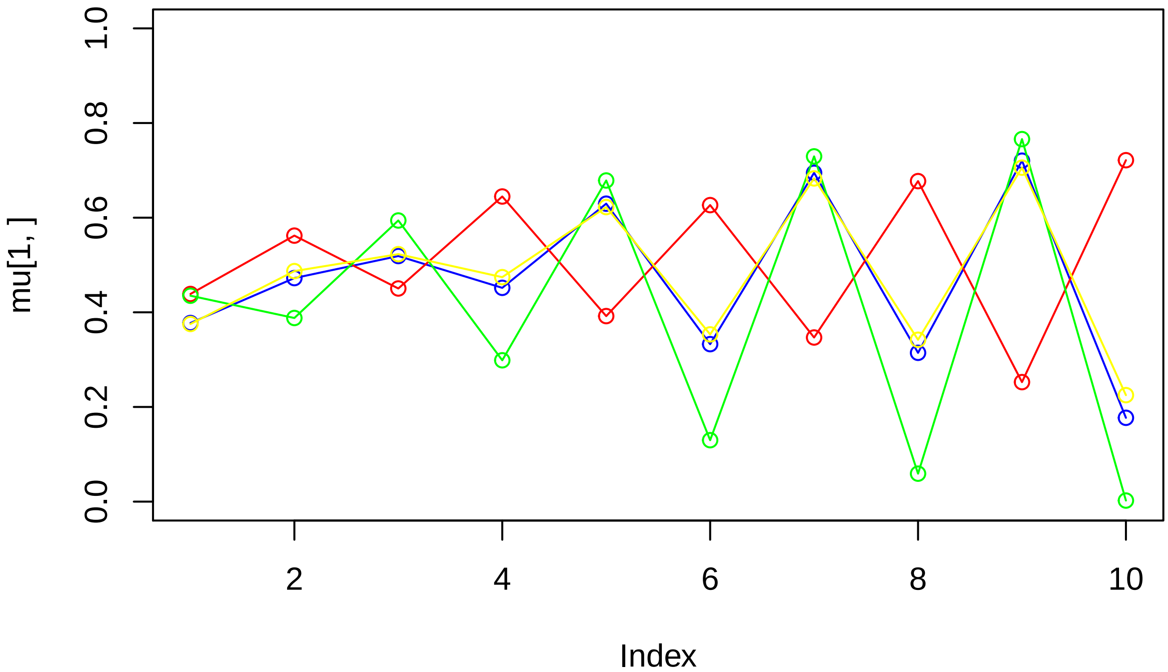

```
## iteration:  24 log likelihood:  -524.3754
```



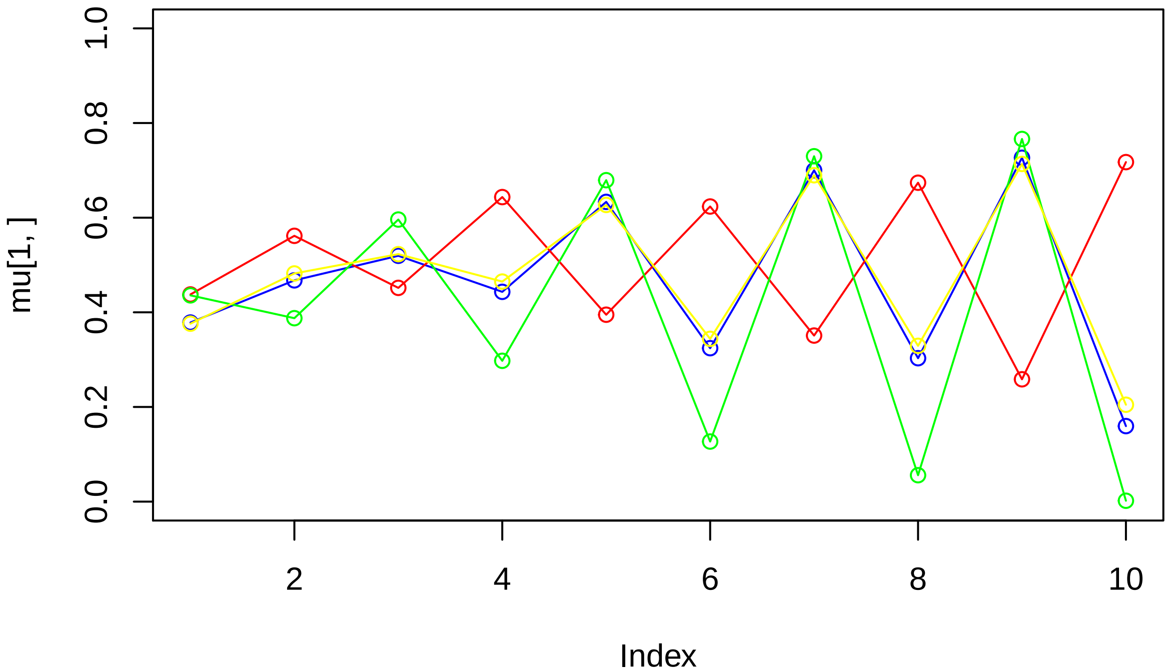

```
## iteration:  25 log likelihood:  -525.7912
```



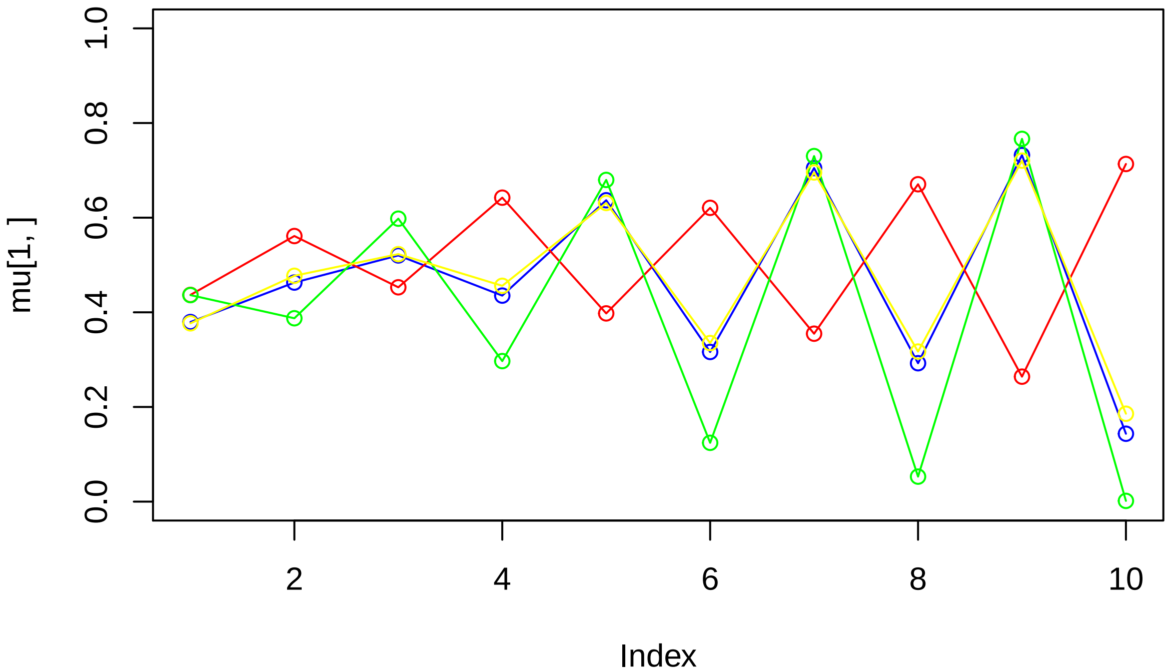

```
## iteration:  26 log likelihood:  -527.3207
```



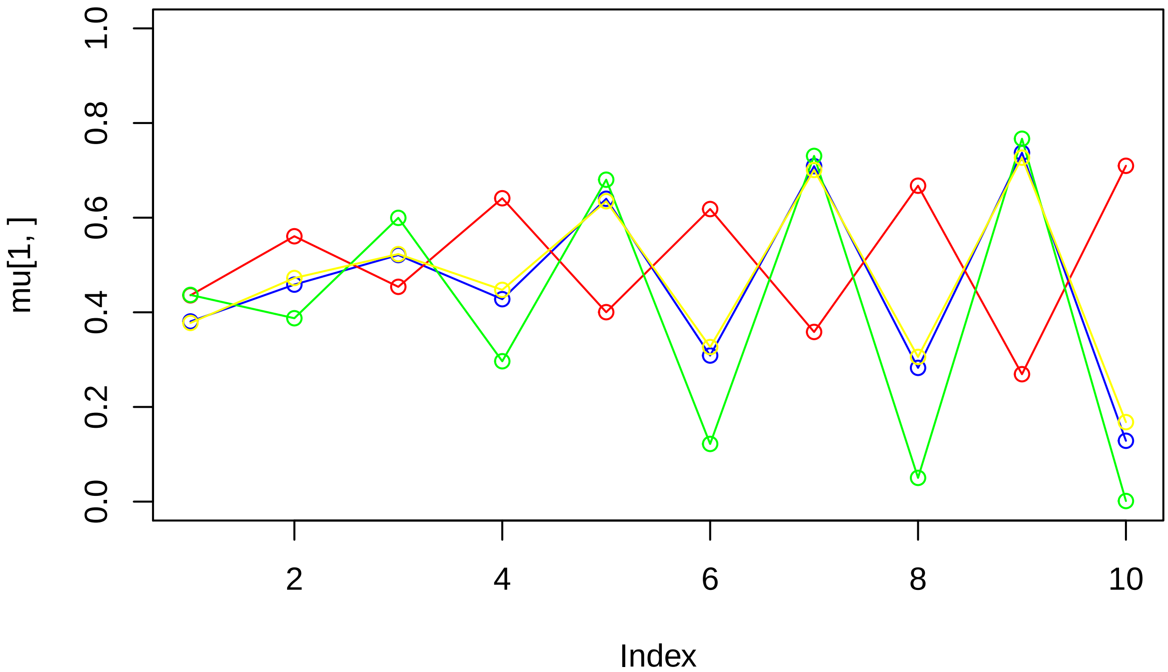

```
## iteration:  27 log likelihood:  -528.9346
```



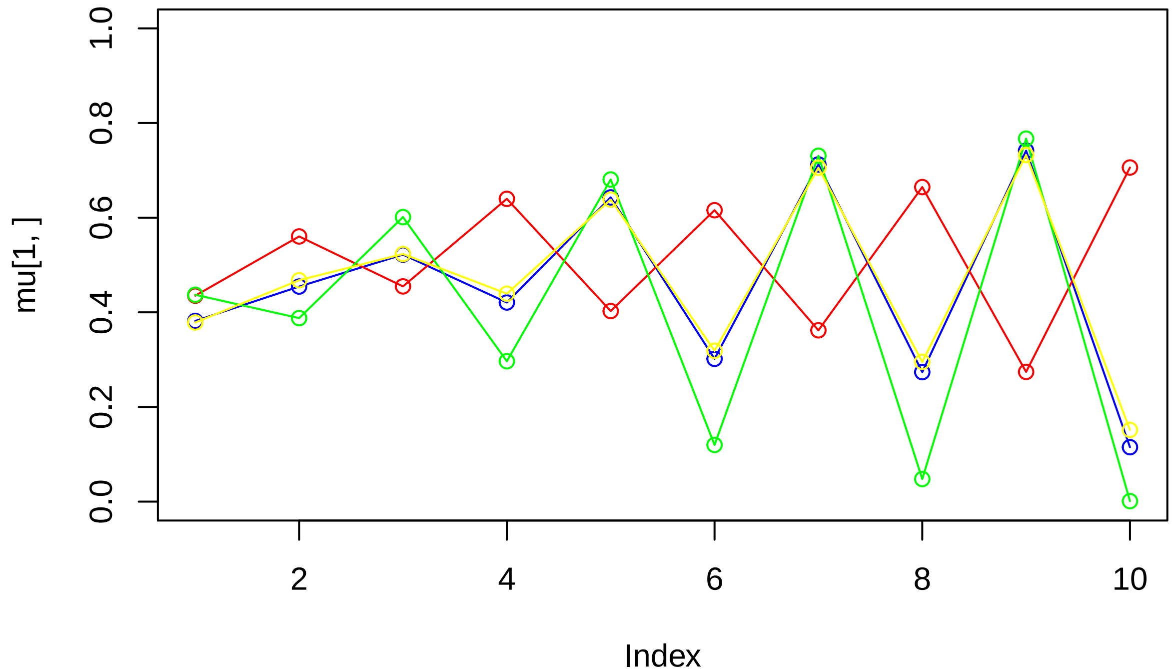

```
## iteration:  28 log likelihood:  -530.6046
```



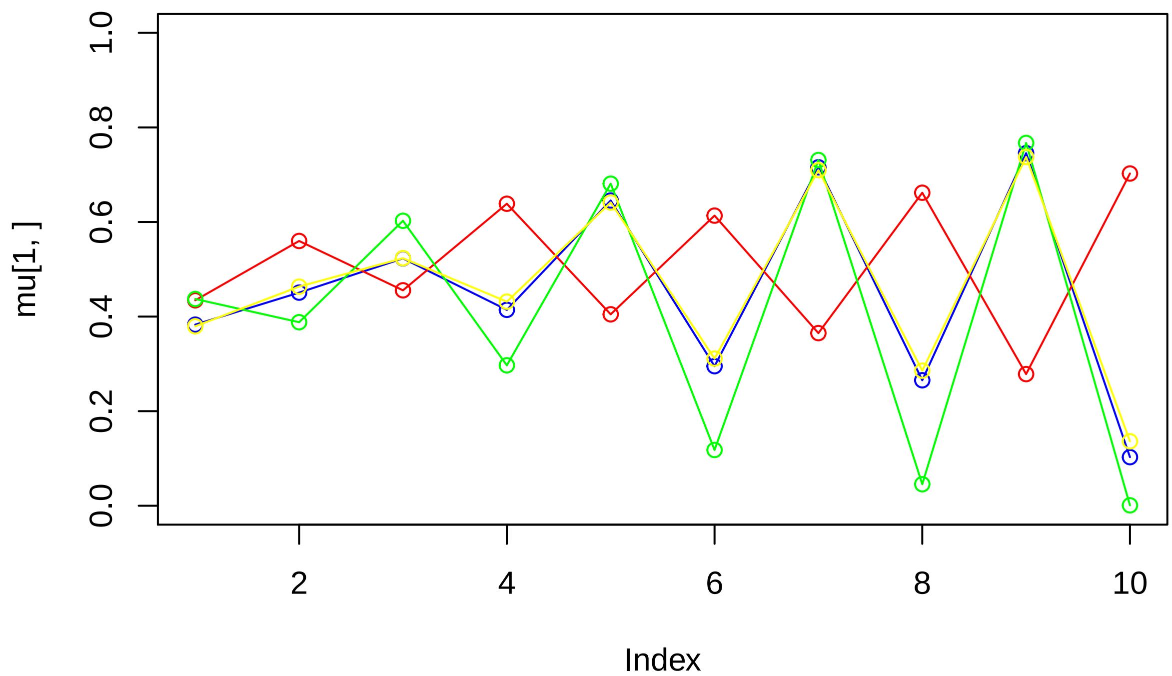

```
## iteration:  29 log likelihood:  -532.304
```



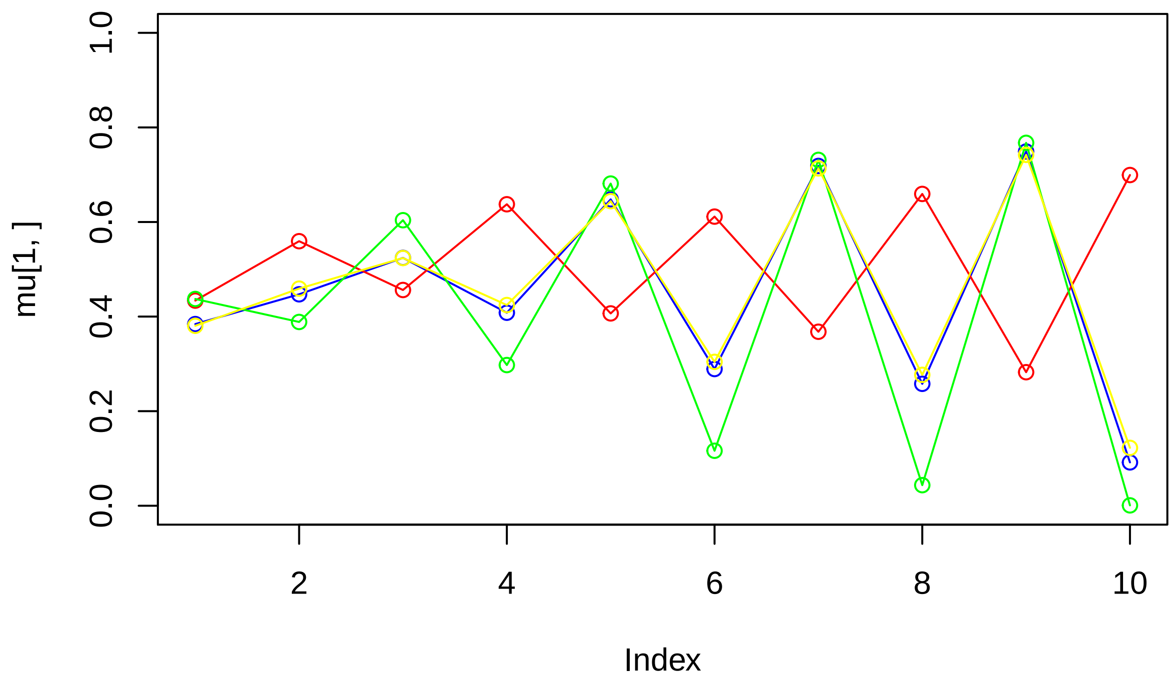

```
## iteration:  30 log likelihood:  -534.0069
```



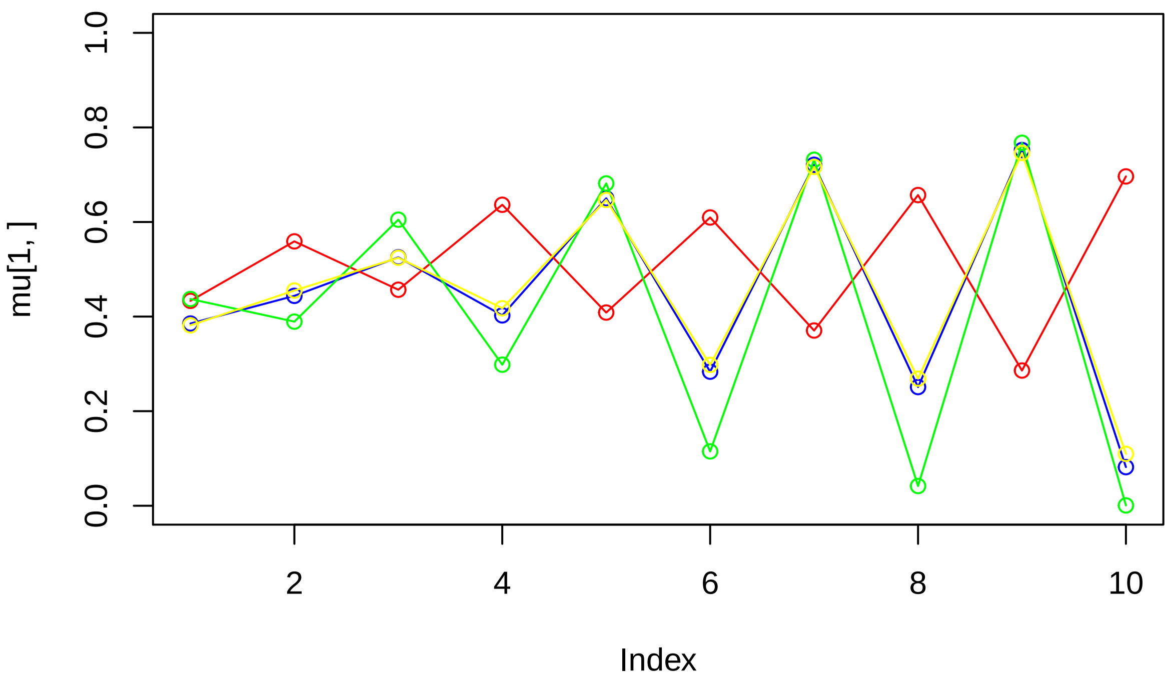

```
## iteration:  31 log likelihood:  -535.6895
```



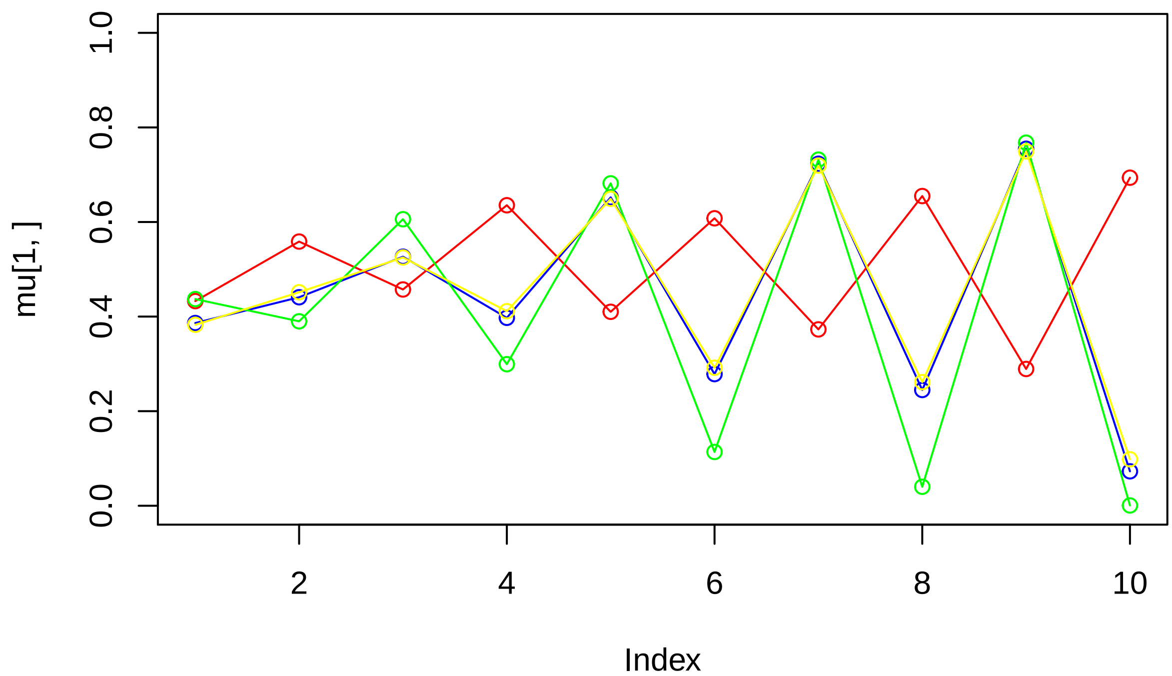

```
## iteration:  32 log likelihood:  -537.3305
```



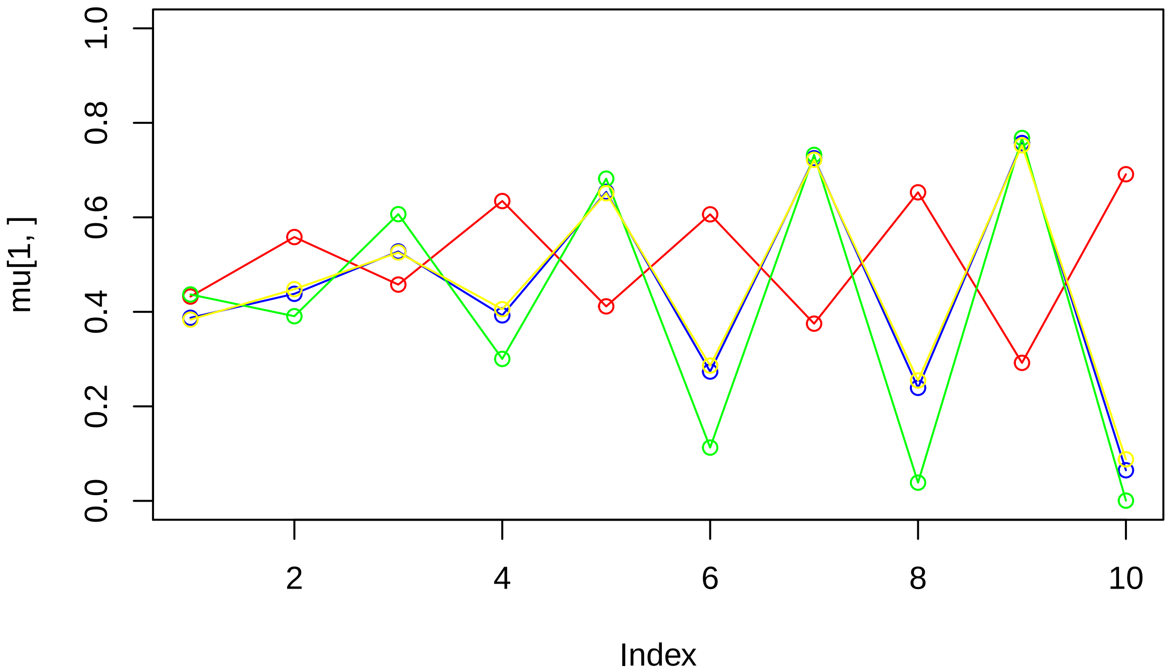

```
## iteration:  33 log likelihood:  -538.912
```



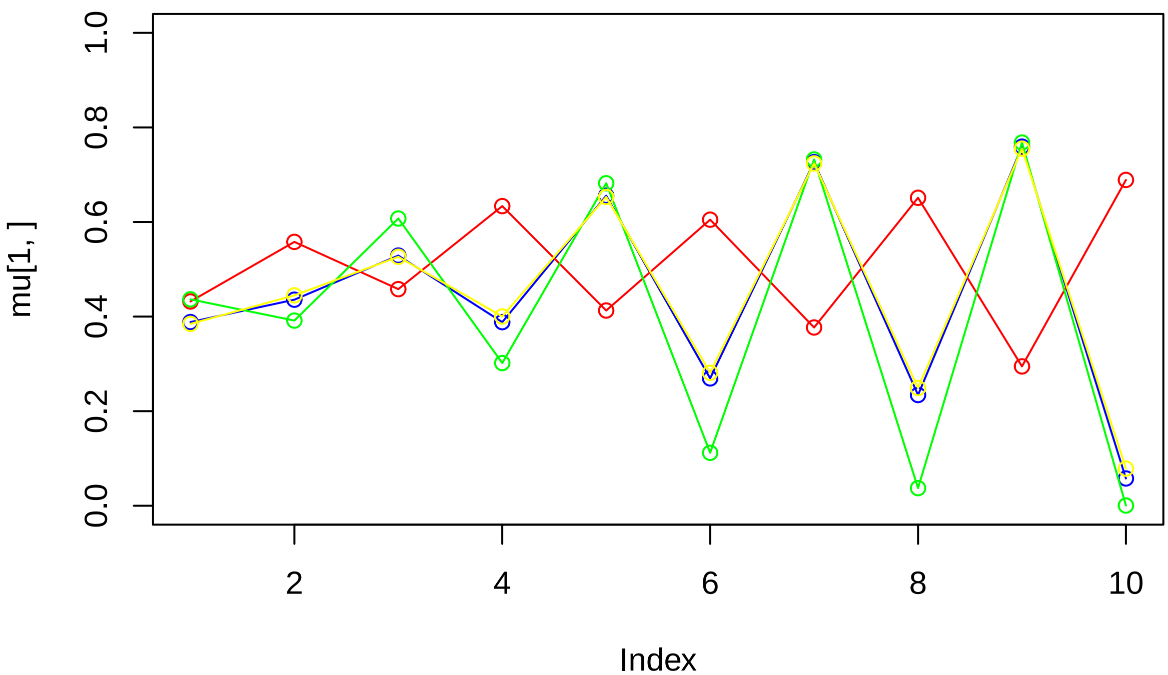

```
## iteration:  34 log likelihood:  -540.4198
```



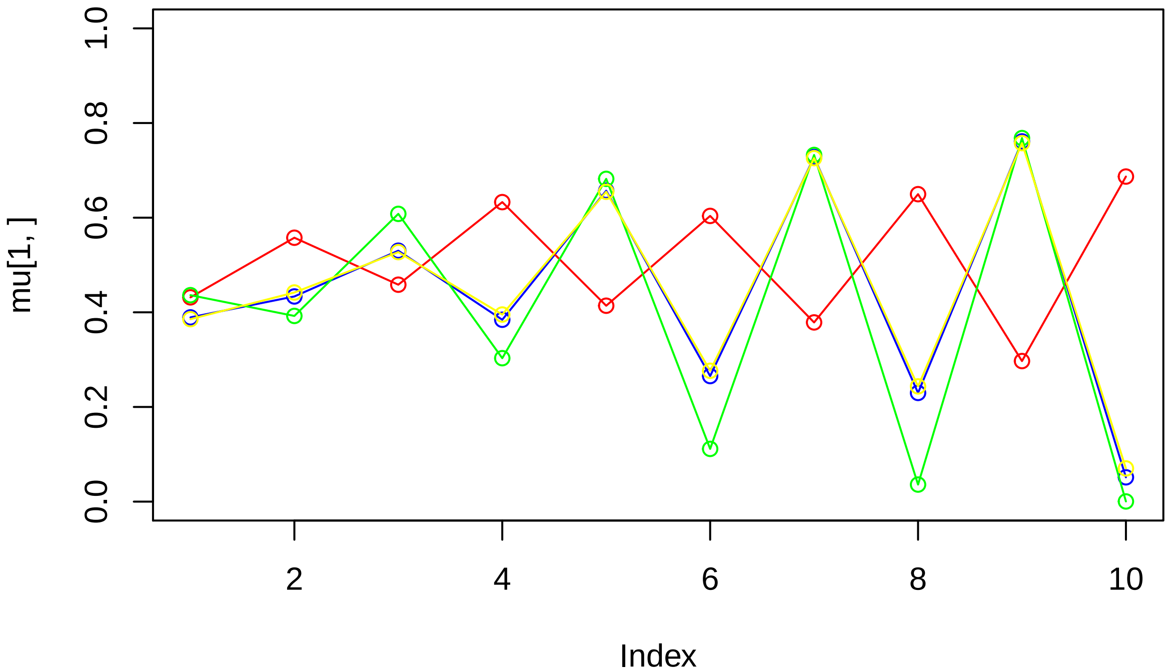

```
## iteration:  35 log likelihood:  -541.8433
```



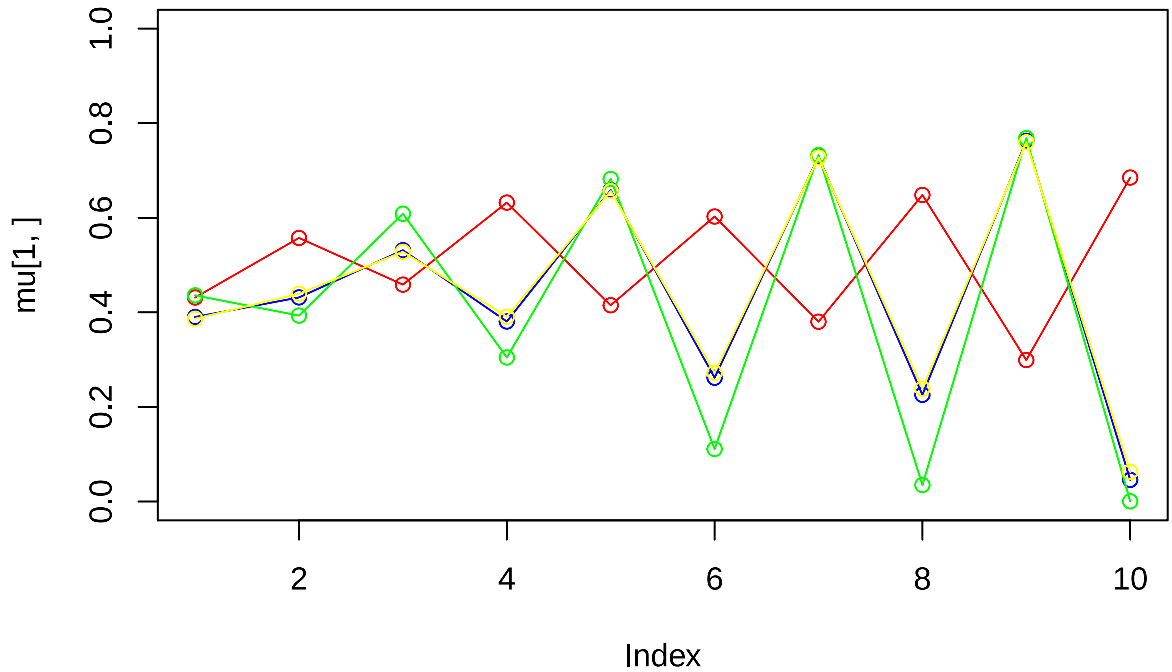

```
## iteration:  36 log likelihood:  -543.1756
```



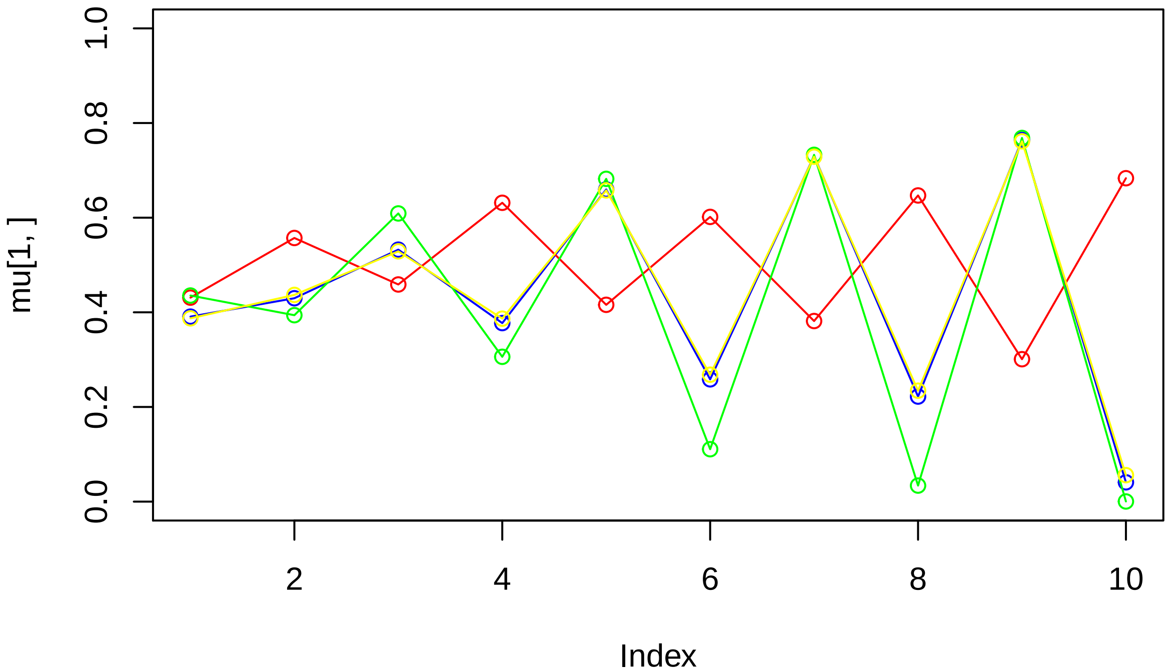

```
## iteration:  37 log likelihood:  -544.4133
```



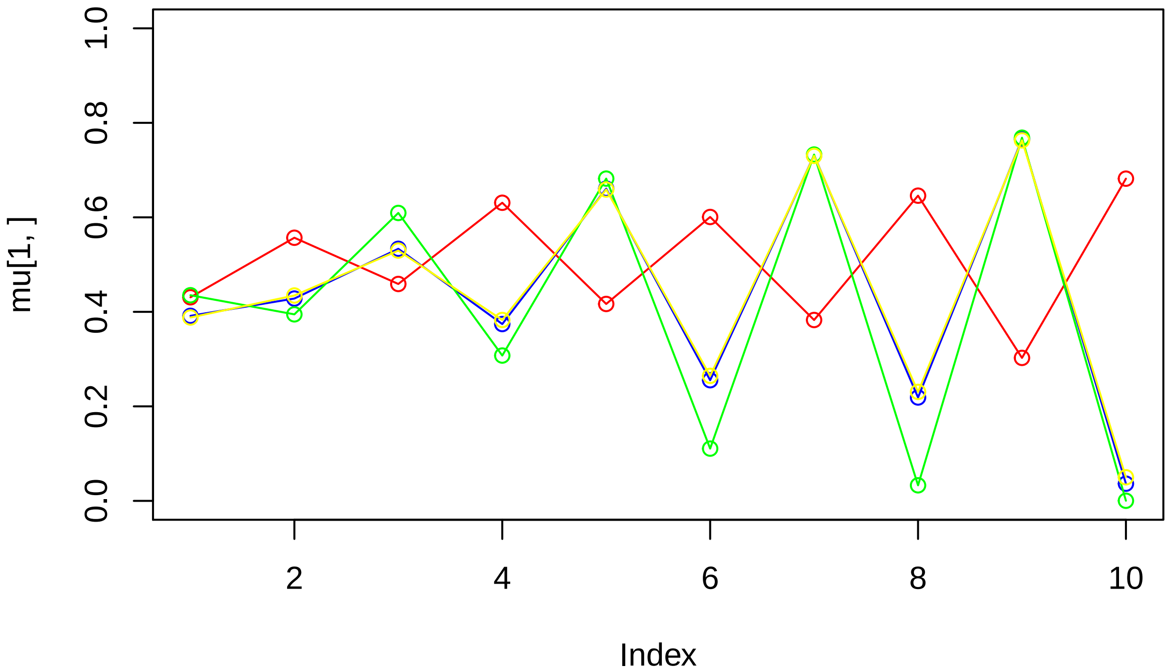

```
## iteration:  38 log likelihood:  -545.5555
```



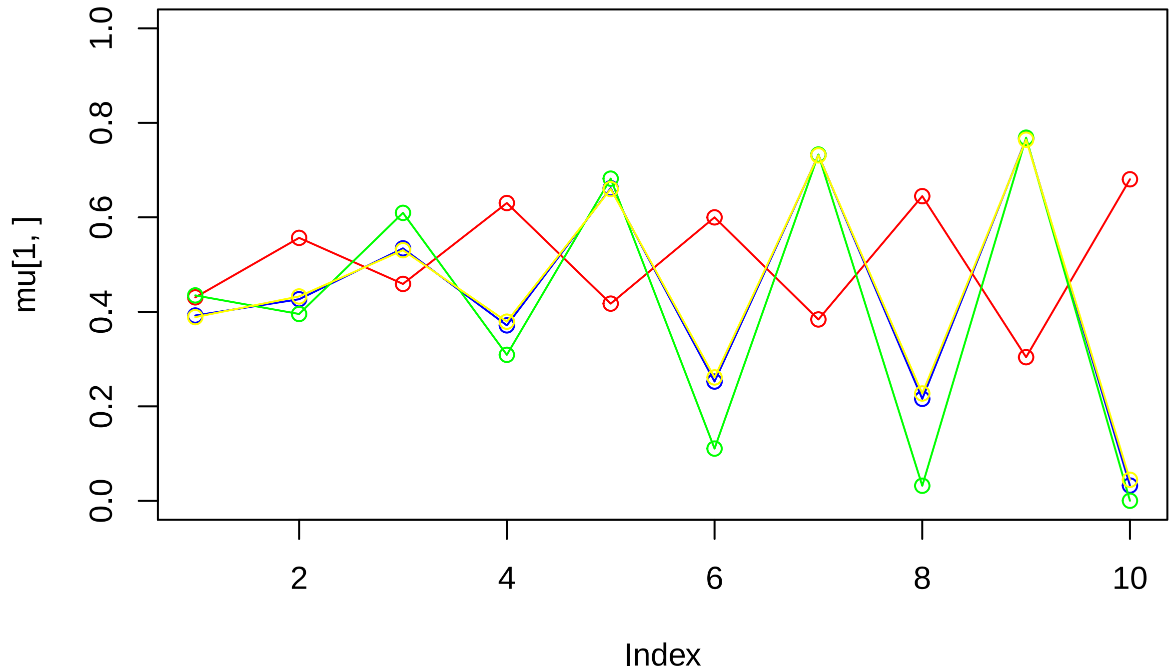

```
## iteration:  39 log likelihood:  -546.6036
```



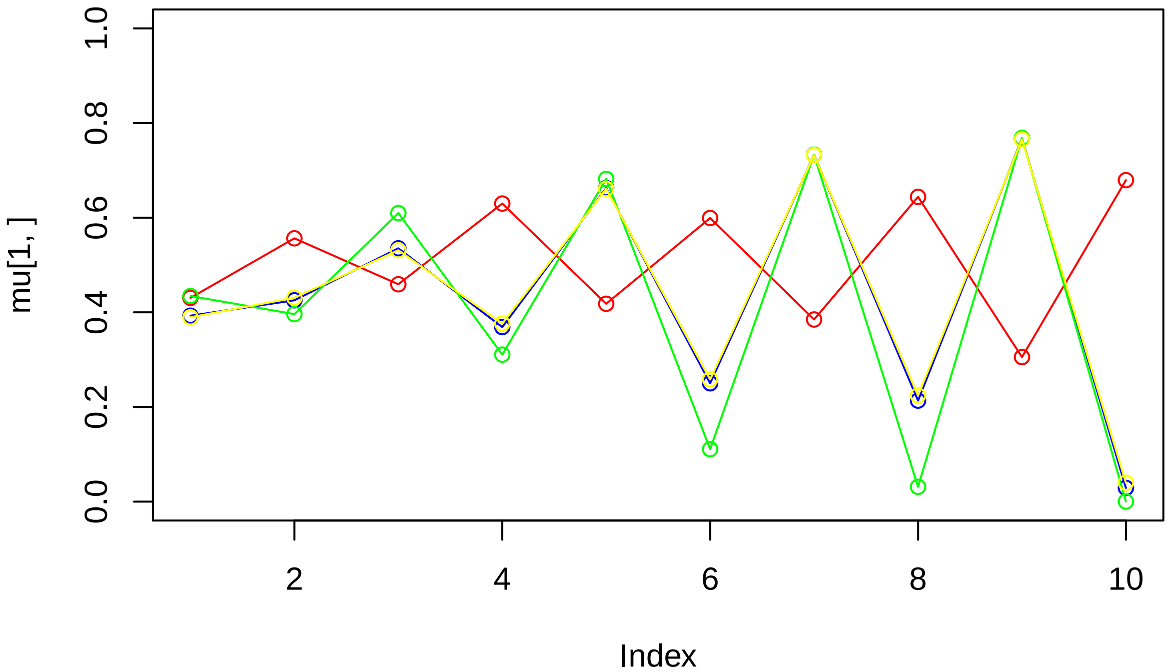

```
## iteration:  40 log likelihood:  -547.5609
```



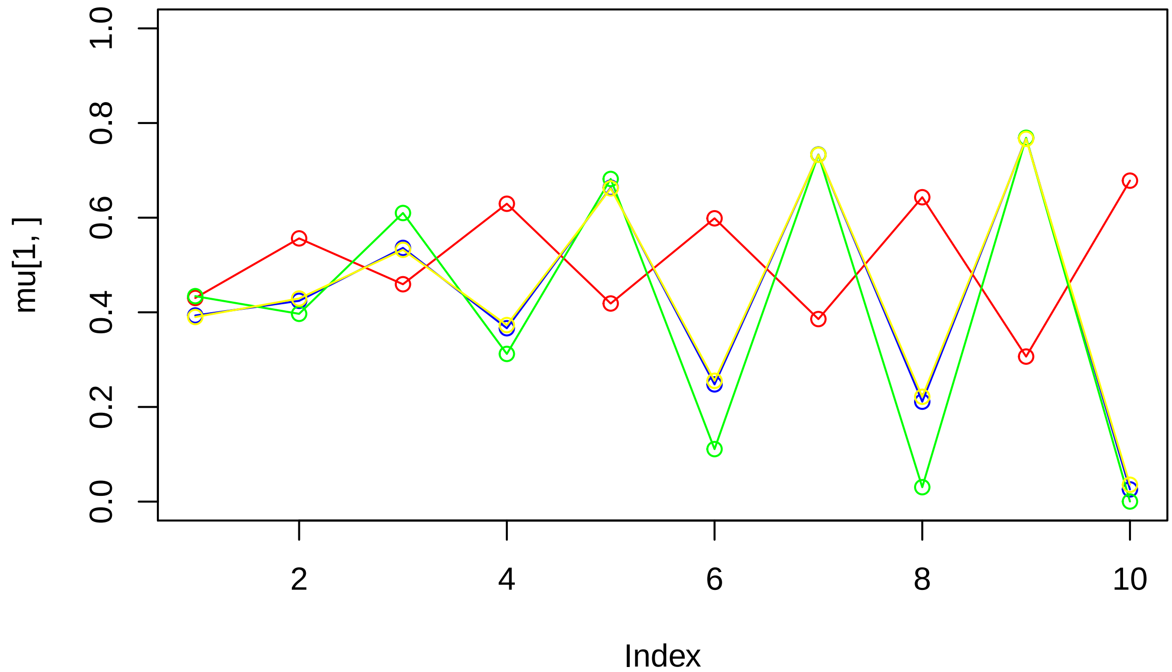

```
## iteration:  41 log likelihood:  -548.4315
```



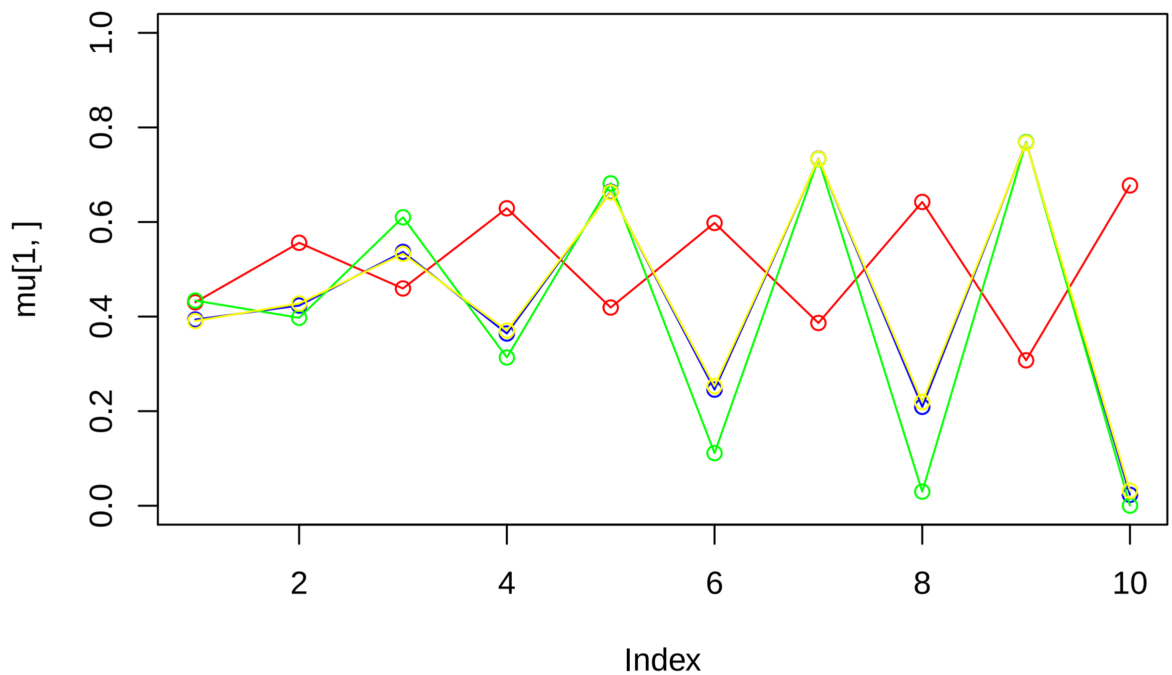

```
## iteration:  42 log likelihood:  -549.2208
```



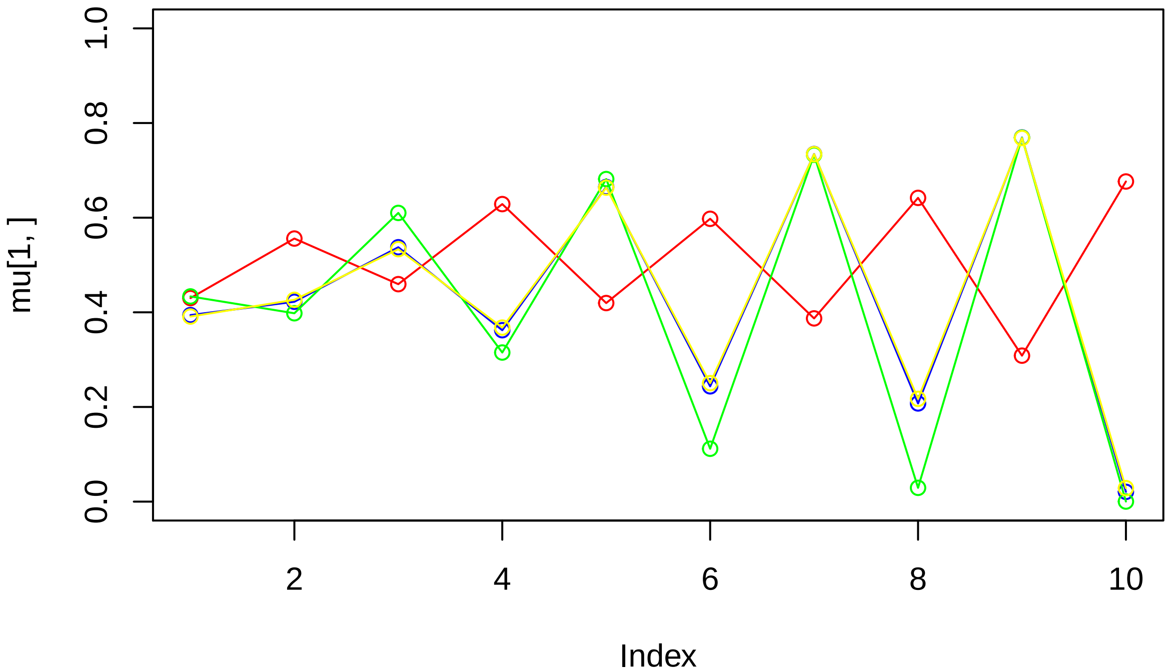

```
## iteration:  43 log likelihood:  -549.9344
```



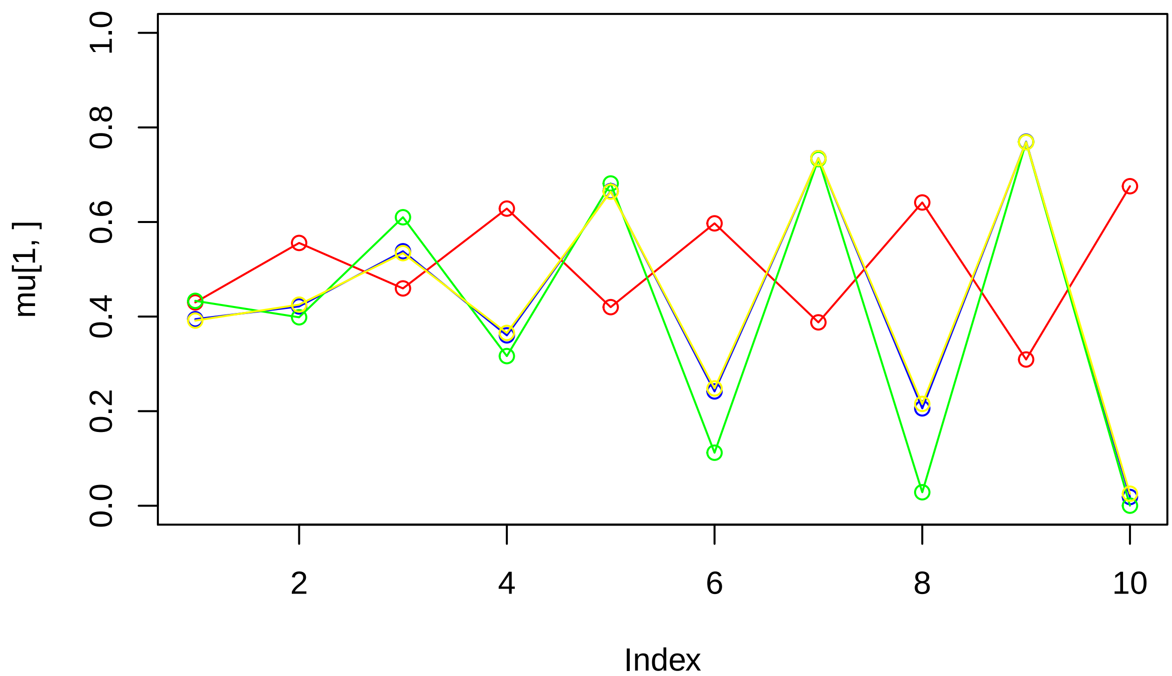

```
## iteration:  44 log likelihood:  -550.5781
```



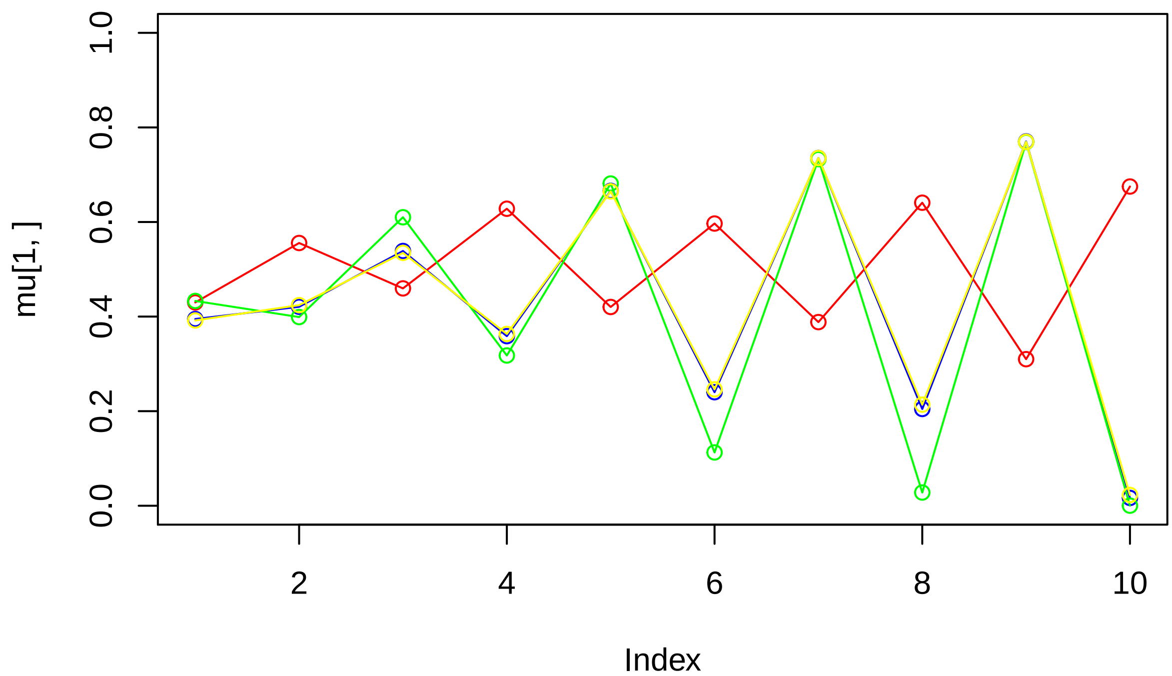

```
## iteration:  45 log likelihood:  -551.1577
```



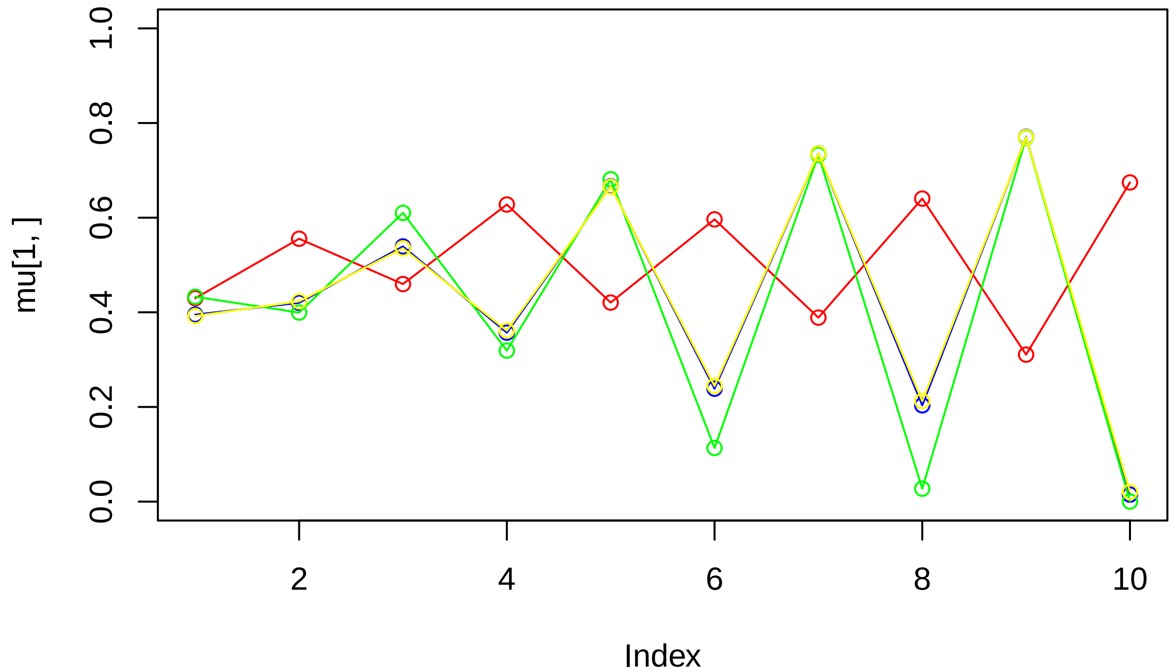

```
## iteration:  46 log likelihood:  -551.6789
```



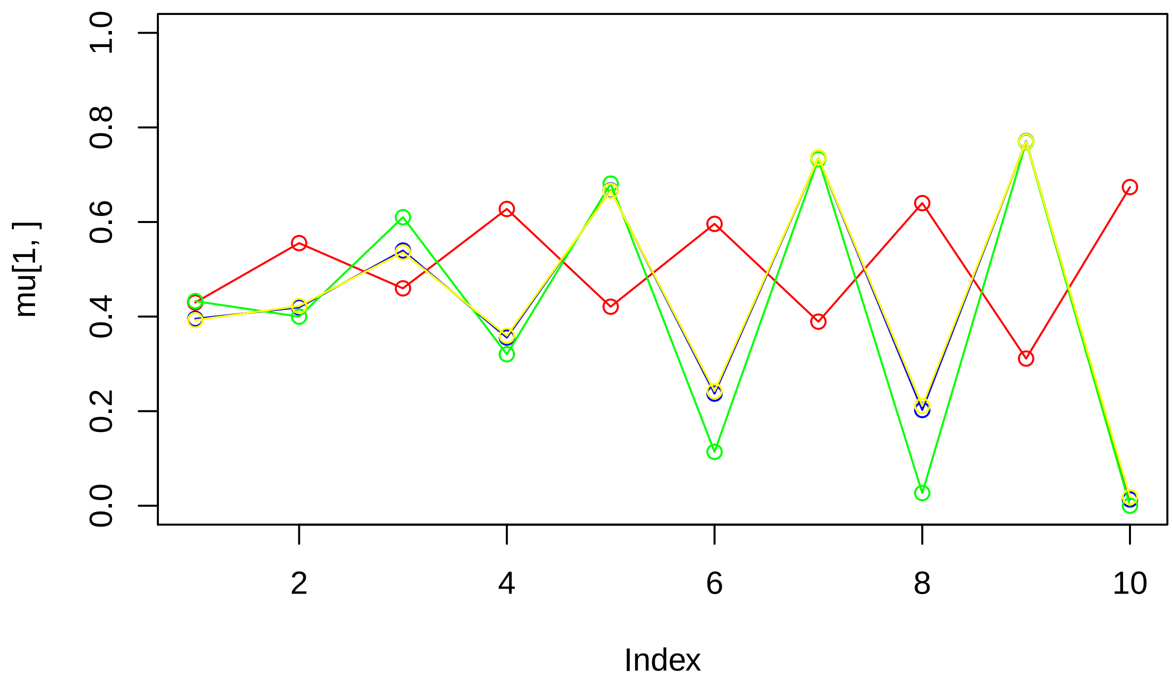

```
## iteration:  47 log likelihood:  -552.1471
```



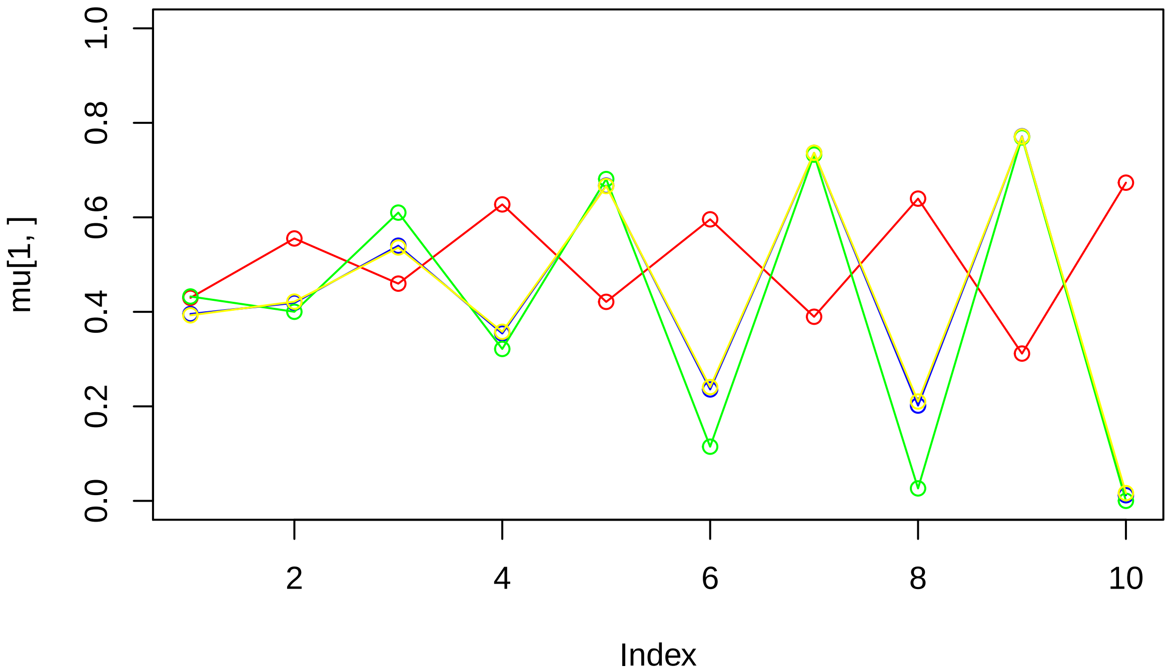

```
## iteration:  48 log likelihood:  -552.5674
```



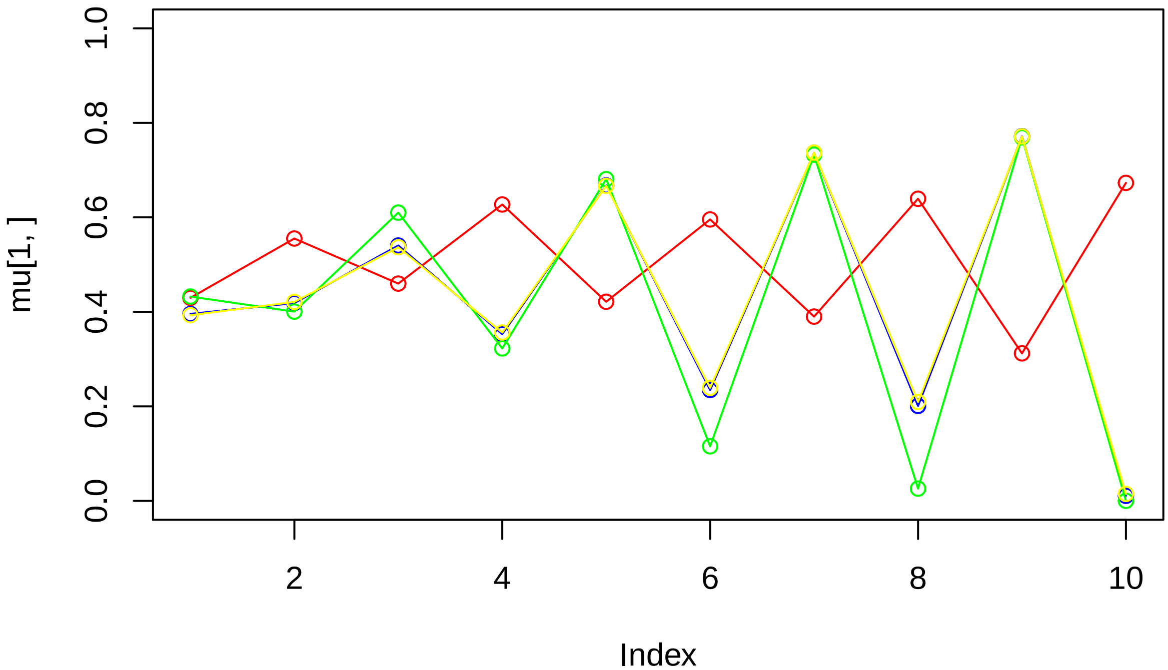

```
## iteration:  49 log likelihood:  -552.9443
```



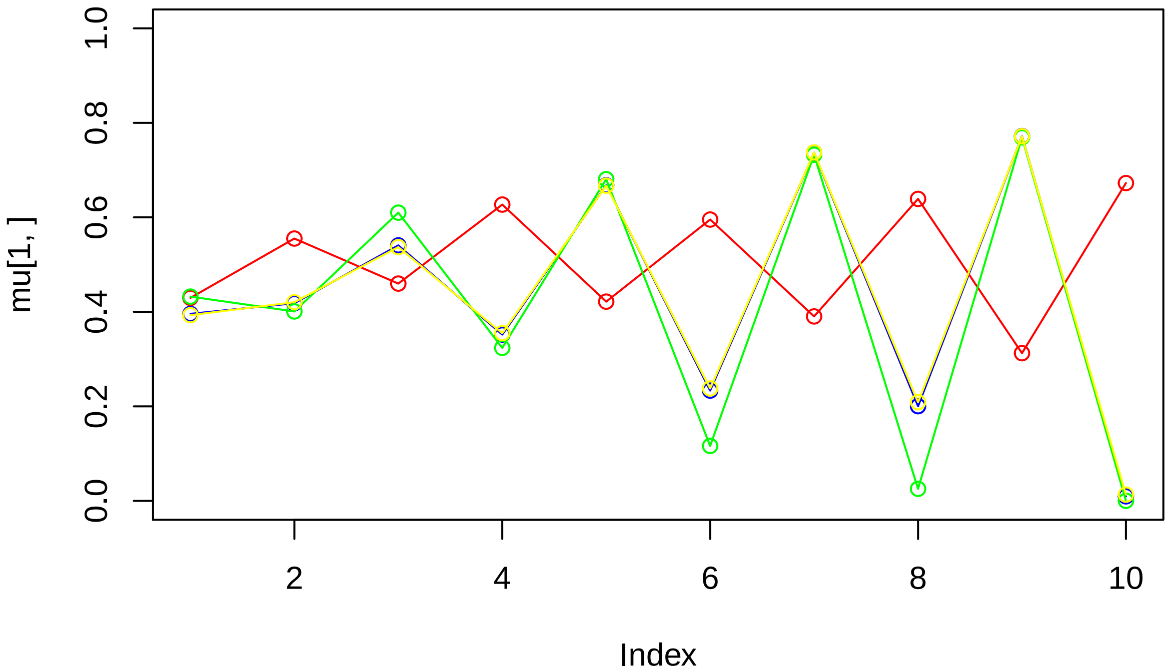

```
## iteration:  50 log likelihood:  -553.2824
```



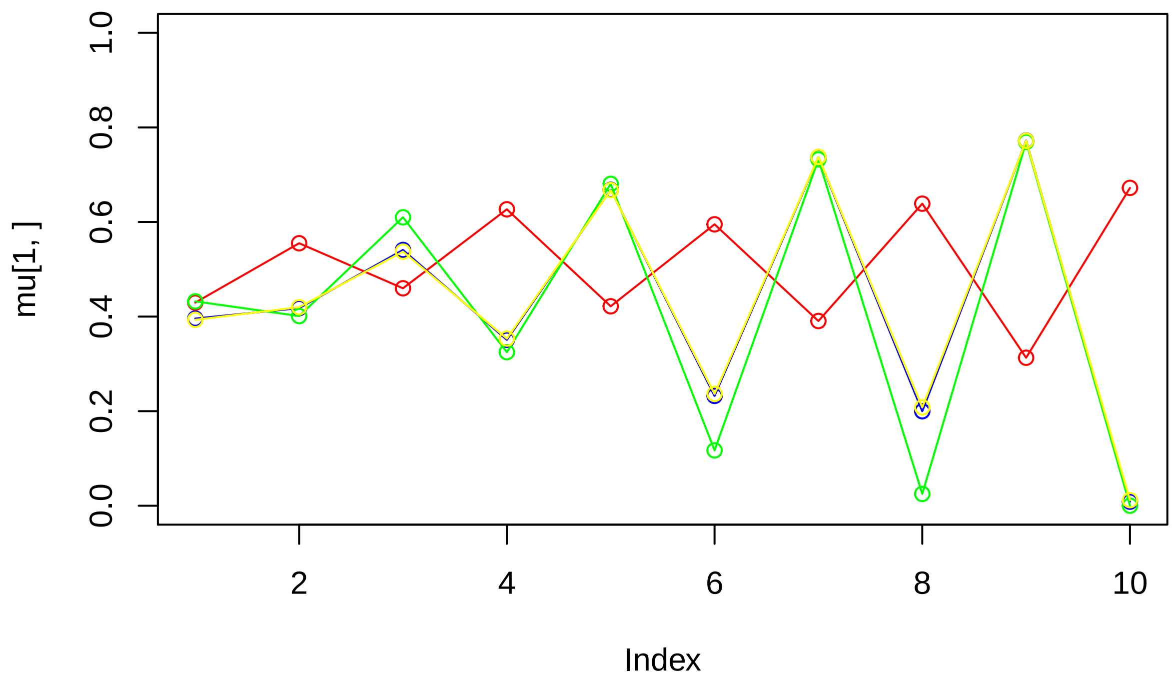

```
## iteration:  51 log likelihood:  -553.5855
```



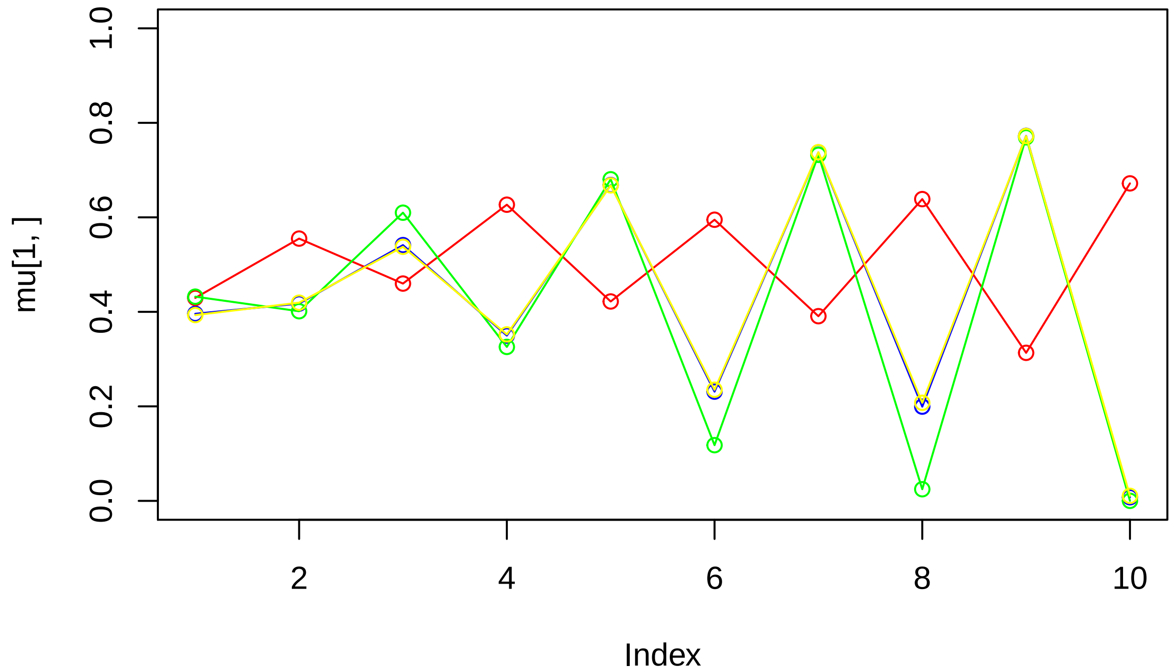

```
## iteration:  52 log likelihood:  -553.8573
```



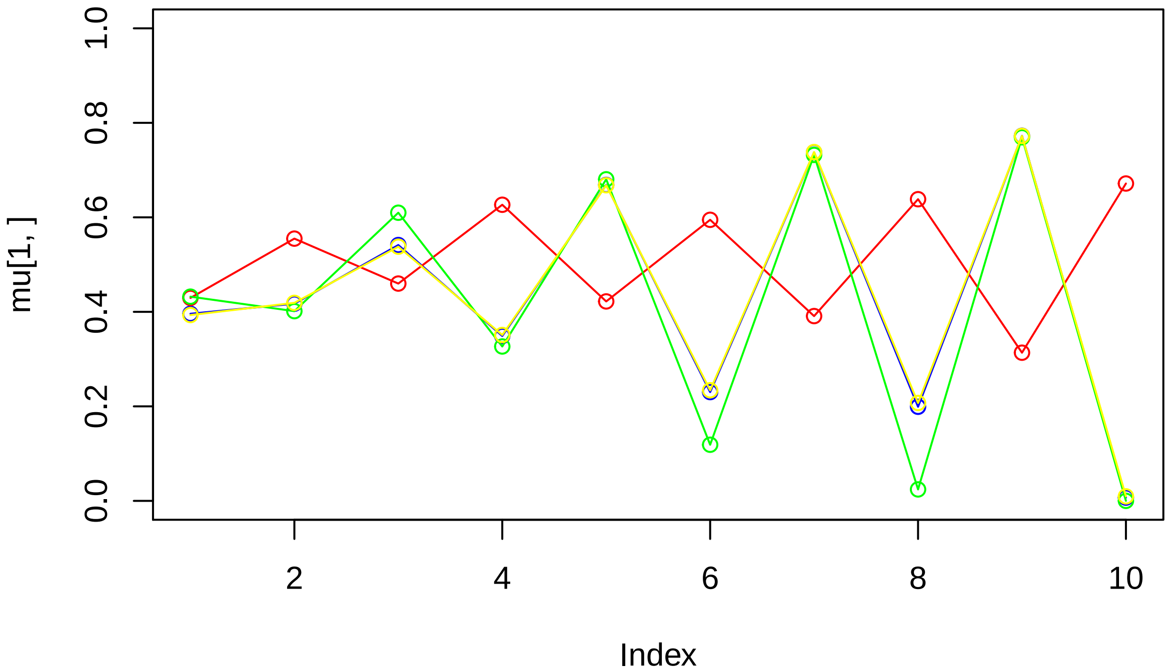

```
## iteration:  53 log likelihood:  -554.101
```



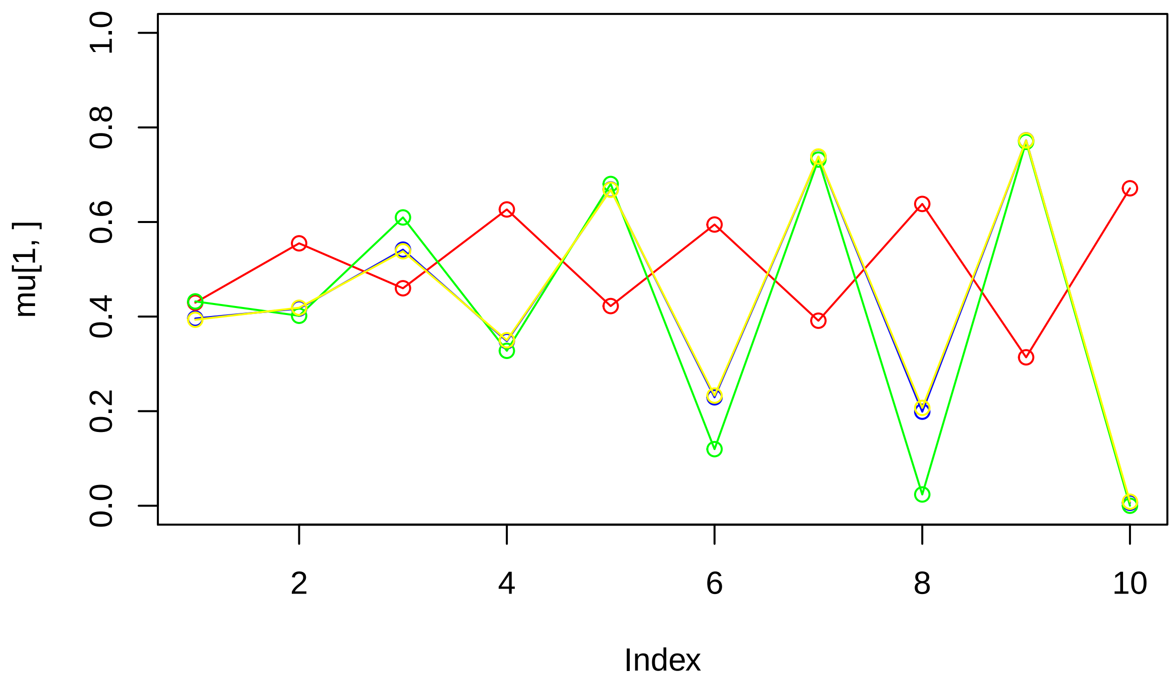

```
## iteration:  54 log likelihood:  -554.3194
```



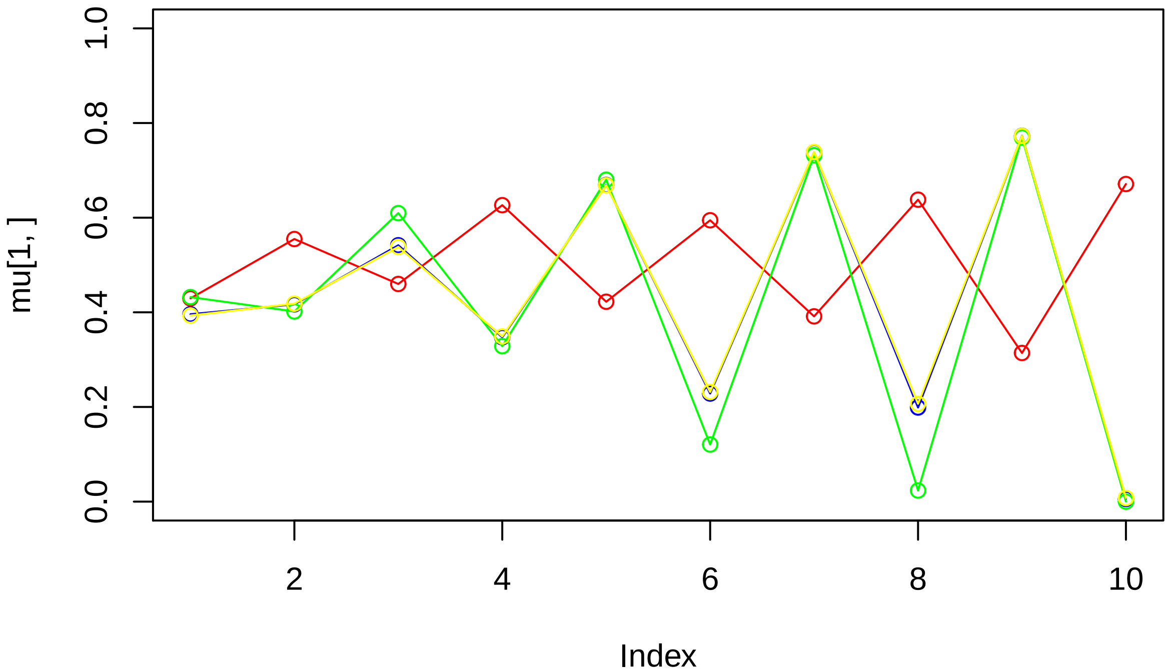

```
## iteration:  55 log likelihood:  -554.5153
```



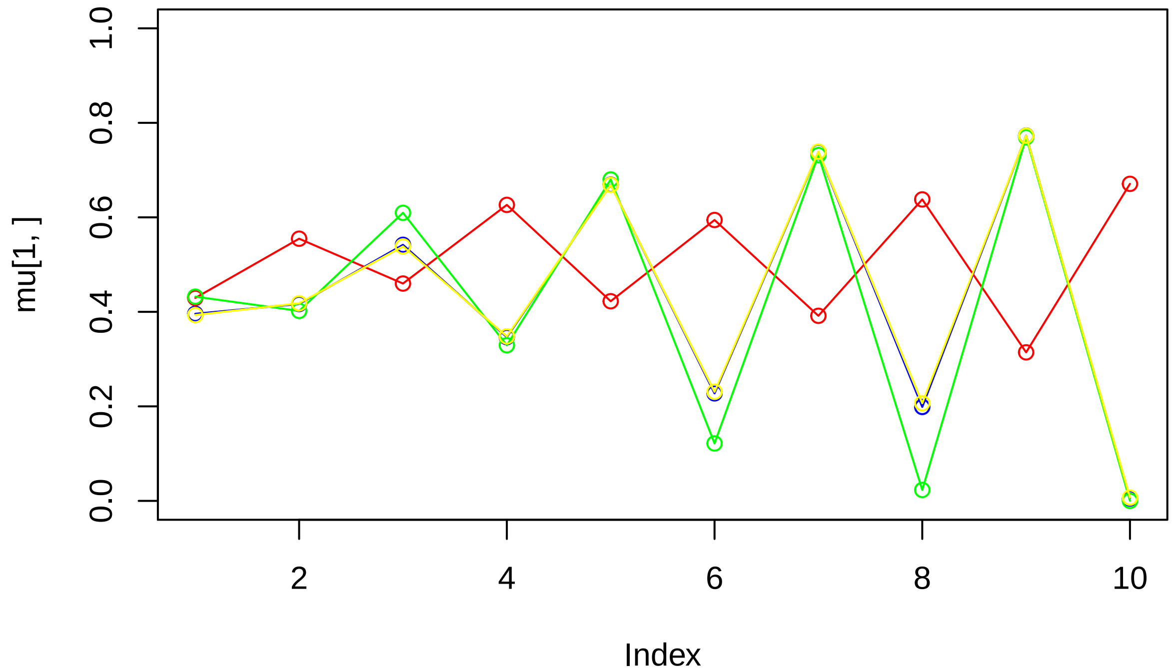

```
## iteration:  56 log likelihood:  -554.691
```



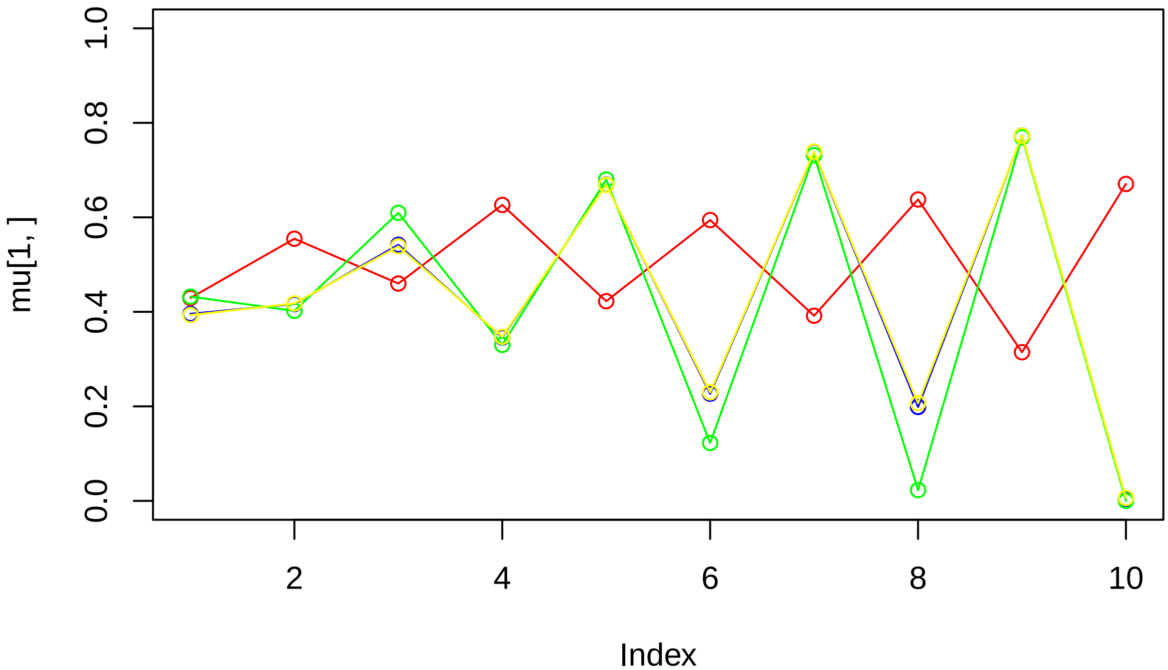

```
## iteration:  57 log likelihood:  -554.8485
```



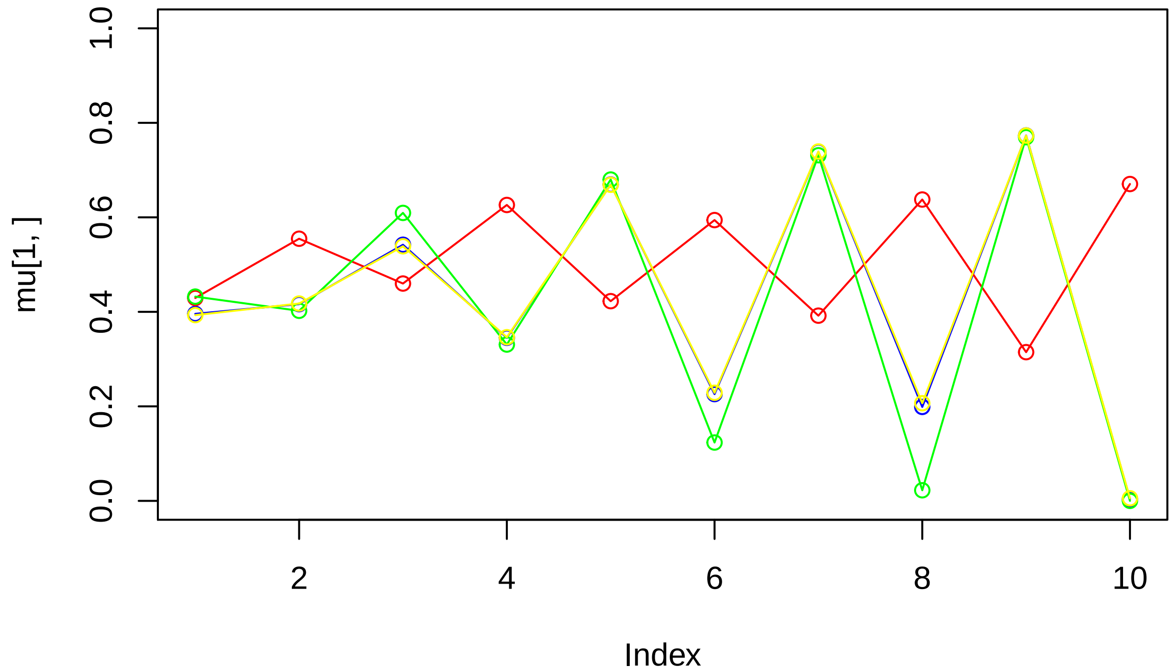

```
## iteration:  58 log likelihood:  -554.9898
```



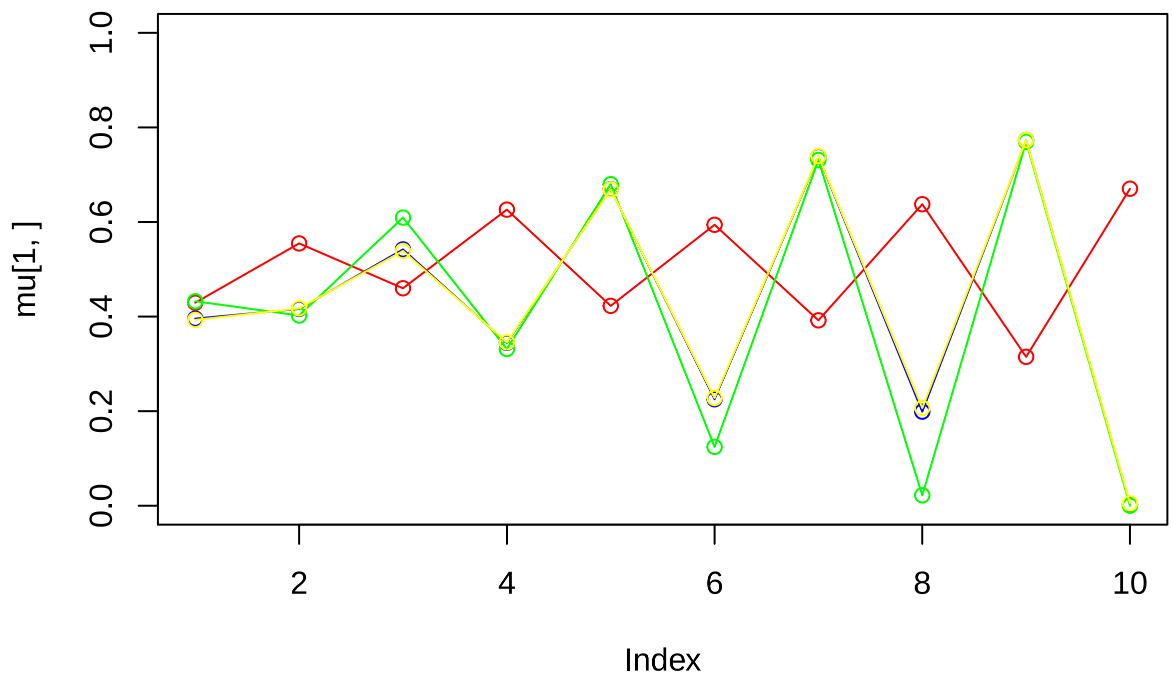

```
## iteration:  59 log likelihood:  -555.1165
```



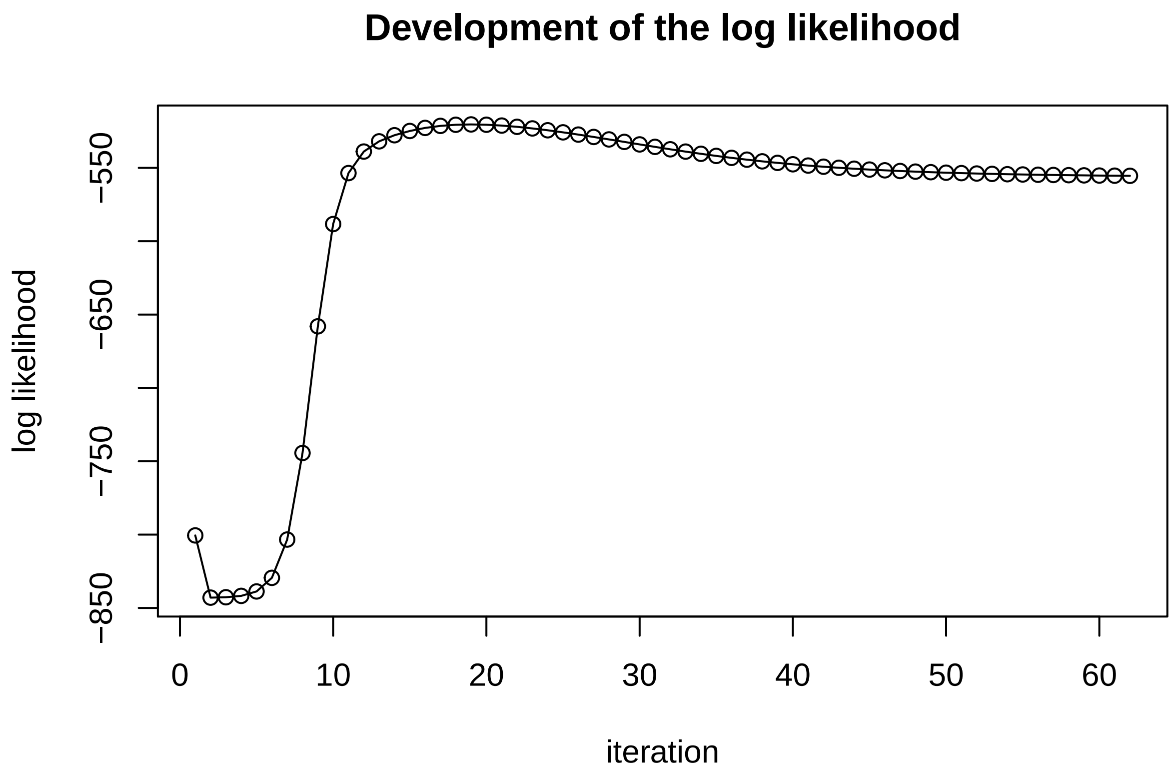

```
## iteration:  60 log likelihood:  -555.2301
```




```
## iteration:  61 log likelihood:  -555.3319
```




```
## iteration:  62 log likelihood:  -555.4231
```




```
## $pi
## [1] 0.06812071 0.72393758 0.11442851 0.09351320
##
## $mu
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.3956838 0.4162506 0.5420280 0.3444983 0.6696118 0.2251983 0.7389032
## [2,] 0.4293539 0.5547107 0.4599340 0.6261408 0.4227076 0.5941305 0.3920563
## [3,] 0.4323433 0.4023143 0.6093482 0.3315033 0.6799272 0.1244291 0.7312642
## [4,] 0.3929703 0.4174015 0.5388154 0.3455370 0.6690888 0.2278577 0.7396138
##            [,8]      [,9]        [,10]
## [1,] 0.19895705 0.7733978 0.0036278205
## [2,] 0.63760909 0.3148516 0.6703401502
## [3,] 0.02206754 0.7696255 0.0000374818
## [4,] 0.20673621 0.7733195 0.0049635197
##
## $logLikelihoodDevelopment
## NULL
```

Analysis:

EM is an iterative expectation maximumation technique. The way this works is for a given mixed distribution we guess the components of the data. This is done by first guessing the number of components and then randomly initializing the parameters of the said distribution (Mean, Varience).

Sometimes the data do not follow any known probability distribution but a mixture of known distributions such as:

$$p(x) = \sum_{k=1}^{K} p(k).p(x|k)$$

where p(x|k) are called mixture components and p(k) are called mixing coefficients: where p(k) is denoted by

$$\pi_k$$

With the following conditions

$$0 \le \pi_k \le 1$$

and

$$\sum_k \pi_k = 1$$

We are also given that the mixture model follows a Bernoulli distribution, for bernoulli we know that

$$Bern(x|\mu_k) = \prod_i \mu_{ki}^{x_i}.(1-\mu_{ki})^{(1-x_i)}$$

The EM algorithm for an Bernoulli mixed model is:

Set pi and mu to some initial values Repeat until pi and mu do not change E-step: Compute p(z|x) for all k and n M-step: Set pi^k to pi^k(ML) from likehood estimate, do the same to mu

M step:

$$p(z_{nk}|x_n, \mu, \pi) = Z = \frac{\pi_k p(x_n|\mu_k)}{\sum_k p(x_n|\mu_k)}$$

E step:

$$\pi_k^{ML} = \frac{\sum_N p(z_{nk}|x_n, \mu, \pi)}{N}$$

$$\mu_{ki}^{ML} = \frac{\sum_n x_{ni} p(z_{nk}|x_n, \mu, \pi)}{\sum_n p(z_{nk}|x_n, \mu, \pi)}$$

The maximum likehood of E step is:

$$\log_e p(X|\mu, \pi) = \sum_{n=1}^{N} \log_e \sum_{k=1}^{K} .\pi_k.p(x_n|\mu_k)$$

Summarising:

When K becomes too less or too many, our model starts to overfit the distribution and

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
if (!require("pacman")) install.packages("pacman")
pacman::p_load(mboost, randomForest, ggplot2)

options("jtools-digits" = 2, scipen = 999)
```

```r
# Loading packages and importing files ####
sp <- read.csv2("spambase.data", header = FALSE, sep = ",", stringsAsFactors = FALSE)
num_sp <- data.frame(data.matrix(sp))
num_sp$V58 <- factor(num_sp$V58)
# shuffling data and dividing into train and test ####
n <- dim(num_sp)[1]
ncol <- dim(num_sp)[2]
set.seed(1234567890)
id <- sample(1:n, floor(n*(2/3)))
train <- num_sp[id,]
test <- num_sp[-id,]
# Adaboost
ntree <- c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
error <- c()

for (i in seq(from = 10, to = 100, by = 10)){
bb <- blackboost(V58 ~., data = train, control = boost_control(mstop = i), family = AdaExp())
bb_predict <- predict(bb, newdata = test, type = c("class"))
confusion_bb <- table(test$V58, bb_predict)
miss_class_bb <- (confusion_bb[1,2] + confusion_bb[2,1])/nrow(test)
error[(i/10)] <- miss_class_bb
}

error_df <- data.frame(cbind(ntree, error))
# Random forest ####
ntree_rf <- c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
error_rf <- c()

for (i in seq(from = 10, to = 100, by = 10)){
rf <- randomForest(V58 ~., data = train, ntree= 10)
rf_predict <- predict(rf, newdata = test, type = c("class"))
confusion_rf <- table(test$V58, rf_predict)
miss_class_rf <- (confusion_rf[1,2] + confusion_rf[2,1])/nrow(test)
error_rf[i/10] <- miss_class_rf
}

error_df_rf <- data.frame(cbind(ntree_rf, error_rf))

df <- cbind(error_df, error_df_rf)
df <- df[, -3]

plot_final <- ggplot(df, aes(ntree)) +
  geom_line(aes(y=error, color = "Adaboost")) +
  geom_line(aes(y=error_rf, color = "Random forest"))

plot_final <- plot_final + ggtitle("Error rate vs number of trees")
plot_final
em_loop = function(K) {
# Initializing data
set.seed(1234567890)
max_it = 100 # max number of EM iterations
min_change = 0.1 # min change in log likelihood between two consecutive EM iterations
N = 1000 # number of training points
```

```
D = 10 # number of dimensions
x = matrix(nrow=N, ncol = D) # training data
true_pi = vector(length = K) # true mixing coefficients
true_mu = matrix(nrow = K, ncol = D) # true conditional distributions
true_pi = c(rep(1/K, K))
if (K == 2) {
true_mu[1,] = c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,] = c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
plot(true_mu[1,], type = "o", xlab = "dimension", col = "blue",
ylim = c(0,1), main = "True")
points(true_mu[2,], type="o", xlab = "dimension", col = "red",
main = "True")
} else if (K == 3) {
true_mu[1,] = c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,] = c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
true_mu[3,] = c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
plot(true_mu[1,], type = "o", xlab = "dimension", col = "blue", ylim=c(0,1),
main = "True")
points(true_mu[2,], type = "o", xlab = "dimension", col = "red",
main = "True")
points(true_mu[3,], type = "o", xlab = "dimension", col = "green",
main = "True")
} else {
true_mu[1,] = c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,] = c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
true_mu[3,] = c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
true_mu[4,] = c(0.3,0.5,0.5,0.7,0.5,0.5,0.5,0.5,0.4,0.5)
plot(true_mu[1,], type = "o", xlab = "dimension", col = "blue",
ylim = c(0,1), main = "True")
points(true_mu[2,], type = "o", xlab = "dimension", col = "red",
main = "True")
points(true_mu[3,], type = "o", xlab = "dimension", col = "green",
main = "True")
points(true_mu[4,], type = "o", xlab = "dimension", col = "yellow",
main = "True")
}
z = matrix(nrow = N, ncol = K) # fractional component assignments
pi = vector(length = K) # mixing coefficients
mu = matrix(nrow = K, ncol = D) # conditional distributions
llik = vector(length = max_it) # log likelihood of the EM iterations
# Producing the training data
for(n in 1:N) {
k = sample(1:K, 1, prob=true_pi)
for(d in 1:D) {
x[n,d] = rbinom(1, 1, true_mu[k,d])
}
}
# Random initialization of the paramters
pi = runif(K, 0.49, 0.51)
pi = pi / sum(pi)
for(k in 1:K) {
mu[k,] = runif(D, 0.49, 0.51)
}
```

```
#EM algorithm
for(it in 1:max_it) {
# Plotting mu
# Defining plot title
title = paste0("Iteration", it)
if (K == 2) {
plot(mu[1,], type = "o", xlab = "dimension", col = "blue", ylim = c(0,1), main = title)
points(mu[2,], type = "o", xlab = "dimension", col = "red", main = title)
} else if (K == 3) {
plot(mu[1,], type = "o", xlab = "dimension", col = "blue", ylim = c(0,1), main = title)
points(mu[2,], type = "o", xlab = "dimension", col = "red", main = title)
points(mu[3,], type = "o", xlab = "dimension", col = "green", main = title)
} else {
plot(mu[1,], type = "o", xlab = "dimension", col = "blue", ylim = c(0,1), main = title)
points(mu[2,], type = "o", xlab = "dimension", col = "red", main = title)
points(mu[3,], type = "o", xlab = "dimension", col = "green", main = title)
points(mu[4,], type = "o", xlab = "dimension", col = "yellow", main = title)
}
Sys.sleep(0.5)
# E-step: Computation of the fractional component assignments
for (n in 1:N) {
# Creating empty matrix (column 1:K = p_x_given_k; column K+1 = p(x|all k)
p_x = matrix(data = c(rep(1,K), 0), nrow = 1, ncol = K+1)
# Calculating p(x|k) and p(x|all k)
for (k in 1:K) {
# Calculating p(x|k)
for (d in 1:D) {
p_x[1,k] = p_x[1,k] * (mu[k,d]^x[n,d]) * (1-mu[k,d])^(1-x[n,d])
}
p_x[1,k] = p_x[1,k] * pi[k] # weighting with pi[k]
# Calculating p(x|all k) (denominator)
p_x[1,K+1] = p_x[1,K+1] + p_x[1,k]
}
#Calculating z for n and all k
for (k in 1:K) {
z[n,k] = p_x[1,k] / p_x[1,K+1]
}
}
#Log likelihood computation
for (n in 1:N) {
for (k in 1:K) {
log_term = 0
for (d in 1:D) {
log_term = log_term + x[n,d] * log(mu[k,d]) + (1-x[n,d]) * log(1-mu[k,d])
}
llik[it] = llik[it] + z[n,k] * (log(pi[k]) + log_term)
}
}
cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
flush.console()
# Stop if the log likelihood has not changed significantly
if (it != 1) {
if (abs(llik[it] - llik[it-1]) < min_change) {
```

```
break
}
}
#M-step: ML parameter estimation from the data and fractional component assignments
# Updating pi
for (k in 1:K) {
pi[k] = sum(z[,k])/N
}
#Updating mu
for (k in 1:K) {
mu[k,] = 0
for (n in 1:N) {
    mu[k,] = mu[k,] + x[n,] * z[n,k]
}
mu[k,] = mu[k,] / sum(z[,k])
}
}
#Printing pi, mu and development of log likelihood at the end
return(list(
pi = pi,
mu = mu,
logLikelihoodDevelopment = plot(llik[1:it],
type = "o",
main = "Development of the log likelihood",
xlab = "iteration",
ylab = "log likelihood")
))
}

em_loop(2)
em_loop(3)
em_loop(4)
myem <- function(K){
  set.seed(1234567890)

max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
N=1000 # number of training points
D=10 # number of dimensions
x <- matrix(nrow=N, ncol=D) # training data
true_pi <- vector(length = K) # true mixing coefficients
true_mu <- matrix(nrow=K, ncol=D) # true conditional distributions
true_pi=c(rep(1/3, K))

if(K == 2){
  plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
  points(true_mu[2,], type="o", col="red")

  true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
  true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
}else if(K == 3){
    plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
    points(true_mu[2,], type="o", col="red")
```

```
    points(true_mu[3,], type="o", col="green")

  true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
  true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
  true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
}else {
    plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
    points(true_mu[2,], type="o", col="red")
    points(true_mu[3,], type="o", col="green")
    points(true_mu[4,], type="o", col="yellow")

    true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
    true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
    true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
    true_mu[4,] = c(0.3,0.5,0.5,0.7,0.5,0.5,0.5,0.5,0.4,0.5)}

# Producing the training data
for(n in 1:N) {
k <- sample(1:K,1,prob=true_pi)
for(d in 1:D) {
x[n,d] <- rbinom(1,1,true_mu[k,d])
}
}

z <- matrix(nrow=N, ncol=K) # fractional component assignments
pi <- vector(length = K) # mixing coefficients
mu <- matrix(nrow=K, ncol=D) # conditional distributions
llik <- vector(length = max_it) # log likelihood of the EM iterations
# Random initialization of the paramters
pi <- runif(K,0.49,0.51)
pi <- pi / sum(pi)


for(k in 1:K) {
mu[k,] <- runif(D,0.49,0.51)
}

for(it in 1:max_it) {

if(K == 2){
  plot(mu[1,], type="o", col="blue", ylim=c(0,1))
  points(mu[2,], type="o", col="red")
}else if(K == 3){
  plot(mu[1,], type="o", col="blue", ylim=c(0,1))
  points(mu[2,], type="o", col="red")
  points(mu[3,], type="o", col="green")
}else{
    plot(mu[1,], type="o", col="blue", ylim=c(0,1))
    points(mu[2,], type="o", col="red")
    points(mu[3,], type="o", col="green")
    points(mu[4,], type="o", col="yellow")}
```

```r
Sys.sleep(0.5)
# E-step: Computation of the fractional component assignments

for(k in 1:K)
prod <- exp(x %*% log(t(mu))) * exp((1-x) %*% t(1-mu))

num = matrix(rep(pi,N), ncol = K, byrow = TRUE) * prod
dem = rowSums(num)
poster = num/dem

#Log likelihood computation.
llik[it] = sum(log(dem))
# Your code here
cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
flush.console()
# Stop if the lok likelihood has not changed significantly
if( it != 1){
if(abs(llik[it] - llik[it-1]) < min_change){break}
}
#M-step: ML parameter estimation from the data and fractional component assignments
# Your code here
num_pi = colSums(poster)
pi = num_pi/N
mu = (t(poster) %*% x)/num_pi
}

#Printing pi, mu and development of log likelihood at the end
return(list(
pi = pi,
mu = mu,
logLikelihoodDevelopment = plot(llik[1:it],
type = "o",
main = "Development of the log likelihood",
xlab = "iteration",
ylab = "log likelihood")
))
}
myem(K=2)
myem(K=3)
myem(K=4)
```