machine learning(732A99) lab1 Block 2

Anubhav Dikshit(anudi287)

4 December 2018

Contents

Loading The Libraries

```
if (!require("pacman")) install.packages("pacman")
pacman::p_load(mboost, randomForest, dplyr, ggplot2)

options(scipen = 999)
```

1.Your task is to evaluate the performance of Adaboost classification trees and random forests on the spam data. Specifically, provide a plot showing the error rates when the number of trees considered are 10,20,..,100. To estimate the error rates, use 2/3 of the data for training and 1/3 as hold-out test data.

Loading Input files

```
spam_data <- read.csv(file = "spambase.data", header = FALSE)
colnames(spam_data)[58] <- "Spam"
spam_data$Spam <- factor(spam_data$Spam, levels = c(0,1), labels = c("0", "1"))
```

Splitting into Train and Test with 66% and 33% ratio.

```
set.seed(12345)
n =  NROW(spam_data)
id = sample(1:n, floor(n*(2/3)))
train = spam_data[id,]
test = spam_data[-id,]
```

Trainning the Model

Adaboost with varying depth

```
final_result <- NULL
for(i in seq(from = 10, to = 100, by = 10)){

ada_model <- mboost::blackboost(Spam~.,
                                 data = train,
                                 family = AdaExp(),
                               control=boost_control(mstop=i))

forest_model <- randomForest(Spam~., data = train, ntree = i)


prediction_function <- function(model, data){
  predicted <- predict(model, newdata = data, type = c("class"))
  predict_correct <- ifelse(data$Spam == predicted, 1, 0)
  score <- sum(predict_correct)/NROW(data)
  return(score)
}


train_ada_model_predict <- predict(ada_model, newdata = train, type = c("class"))
test_ada_model_predict <- predict(ada_model, newdata = test, type = c("class"))
train_forest_model_predict <- predict(forest_model, newdata = train, type = c("class"))
test_forest_model_predict <- predict(forest_model, newdata = test, type = c("class"))
```

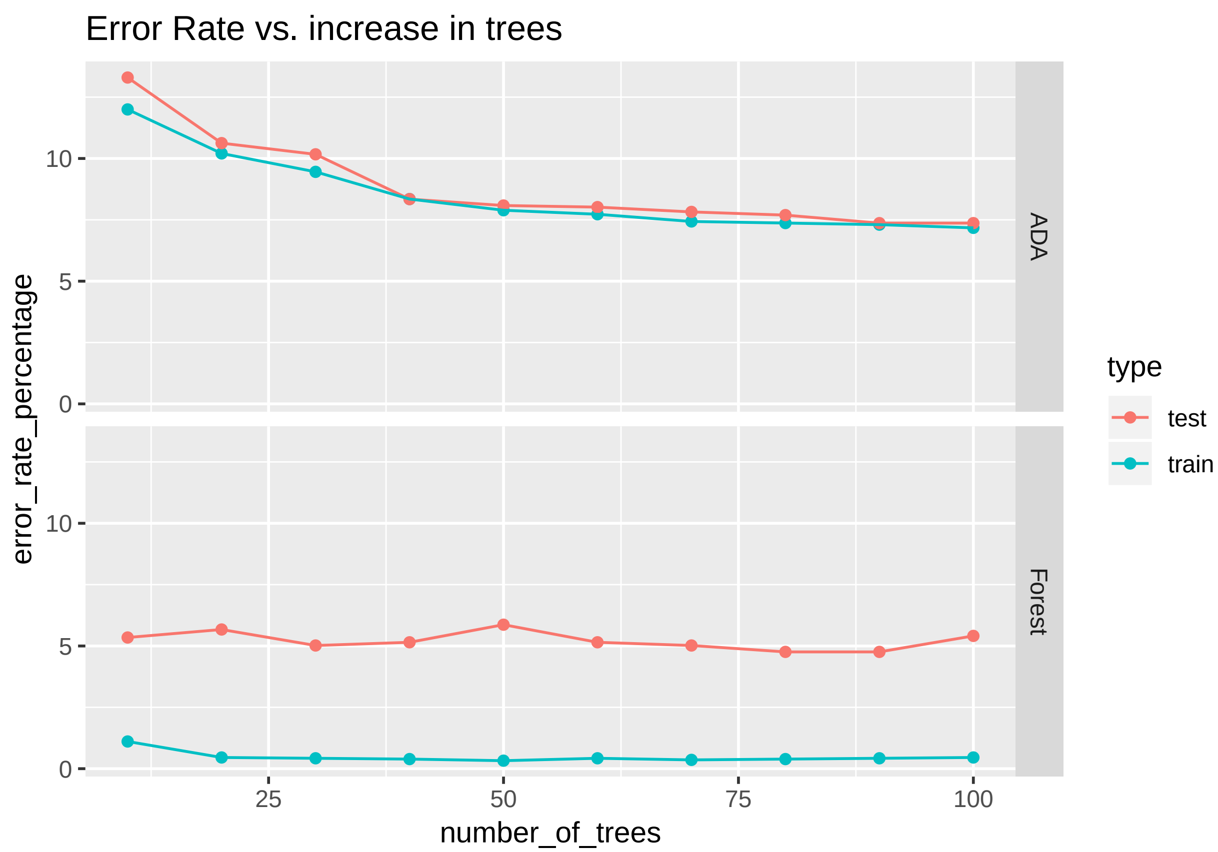
```r
test_predict_correct <- ifelse(test$Spam == test_forest_model_predict, 1, 0)
train_predict_correct <- ifelse(train$Spam == train_forest_model_predict, 1, 0)


train_ada_score <-  prediction_function(ada_model, train)
test_ada_score <-  prediction_function(ada_model, test)
train_forest_score <-  prediction_function(forest_model, train)
test_forest_score <-  prediction_function(forest_model, test)

iteration_result <- data.frame(number_of_trees = i,
                               accuracy = c(train_ada_score,
                                            test_ada_score,
                                            train_forest_score,
                                            test_forest_score),
                               type  = c("train", "test", "train", "test"),
                               model = c("ADA", "ADA",  "Forest", "Forest"))


final_result <- rbind(iteration_result, final_result)
}

final_result$error_rate_percentage <- 100*(1 - final_result$accuracy)
ggplot(data = final_result, aes(x = number_of_trees,
                                y = error_rate_percentage,
                                group = type, color = type)) +
  geom_point() +
  geom_line() +
  ggtitle("Error Rate vs. increase in trees") + facet_grid(rows = vars(model))
```



Analysis:

From the plots we can clearly see that ADA boosted methods uses more trees(~50) to reduce the test error, while randomforest achieves saturation in short number of trees(~10). We also see that random forest achieves less error than ADA tree for both tree and test cases.

2 Your task is to implement the EM algorithm for mixtures of multivariate Bernoulli distributions. Please use the template in the next page to solve the assignment. Then, use your implementation to show what happens when your mixture models has too few and too many components, i.e. set K = 2,3,4 and compare results. Please provide a short explanation as well.

Description of the EM algorithm

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

$$0 \leq \pi_k \leq 1$$

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:

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

Code

To compare the results for K = 2,3,4, the em_loop-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
```

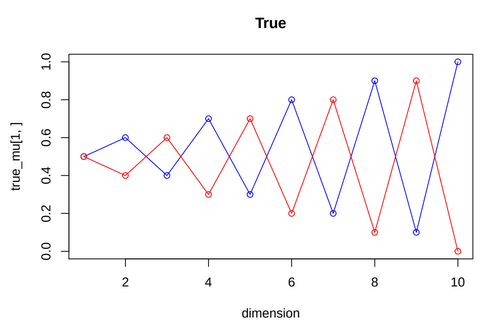
```r
x = matrix(nrow=N, ncol = D) # training data
true_pi = vector(length = K) # true mixing coefficients
true_mu = matrix(nrow = K, ncol = D) # true conditional distributions
true_pi = c(rep(1/K, K))
if (K == 2) {
true_mu[1,] = c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,] = c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
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
```

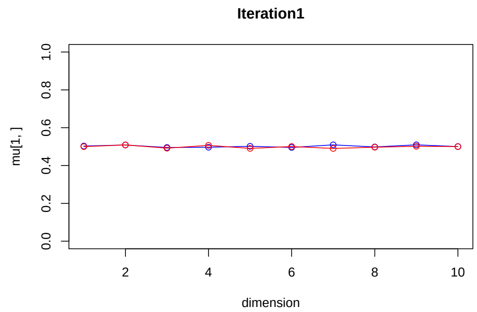
```
for(it in 1:max_it) {
# Plotting mu
# Defining plot title
title = paste0("Iteration", it)
if (K == 2) {
plot(mu[1,], type = "o", xlab = "dimension", col = "blue", ylim = c(0,1), main = title)
points(mu[2,], type = "o", xlab = "dimension", col = "red", main = title)
} else if (K == 3) {
plot(mu[1,], 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")
))
}
```

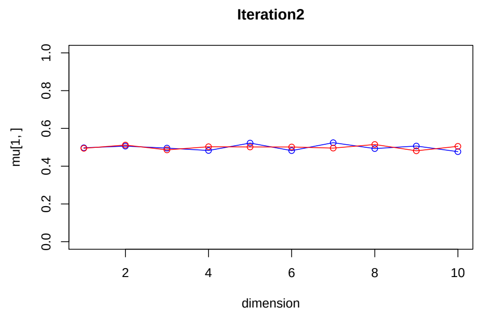
K=2

```
em_loop(2)
```

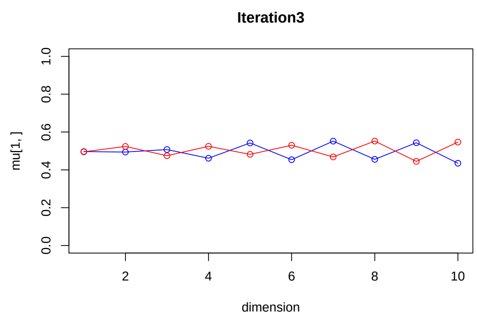




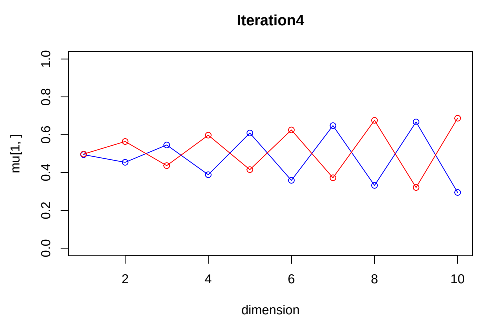

```
## iteration:  1 log likelihood:  -7623.897
```



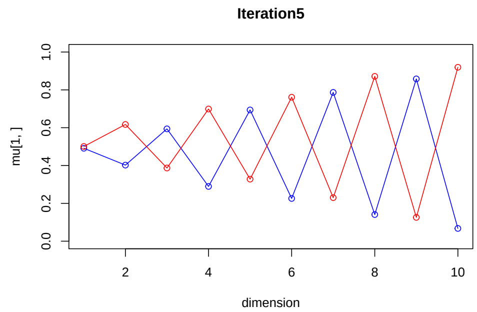

```
## iteration:  2 log likelihood:  -7610.745
```



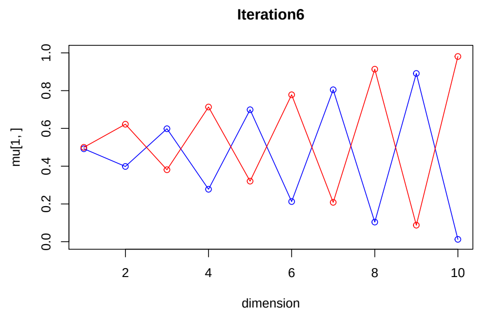

```
## iteration:  3 log likelihood:  -7463.445
```



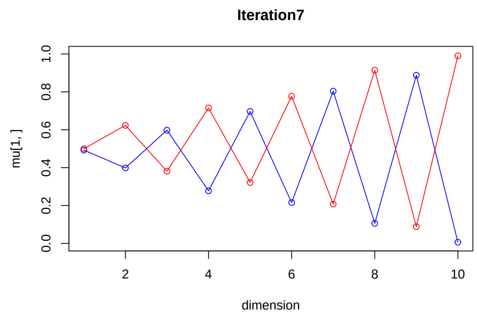

```
## iteration:  4 log likelihood:  -6575.121
```



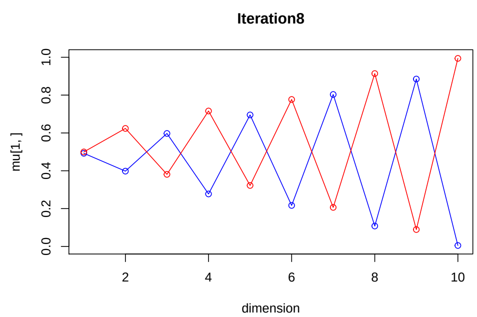
iteration: 5 log likelihood: -5731.559



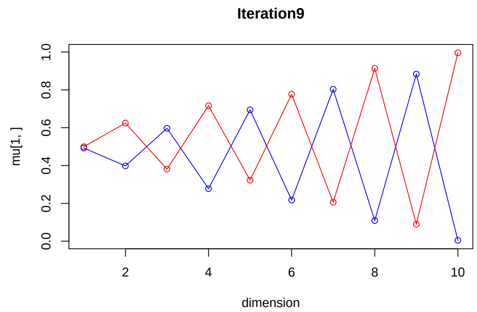
iteration: 6 log likelihood: -5656.174



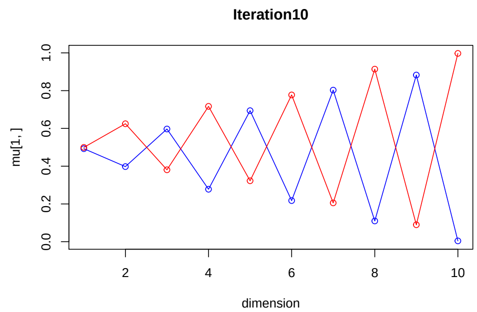
iteration: 7 log likelihood: -5648.904



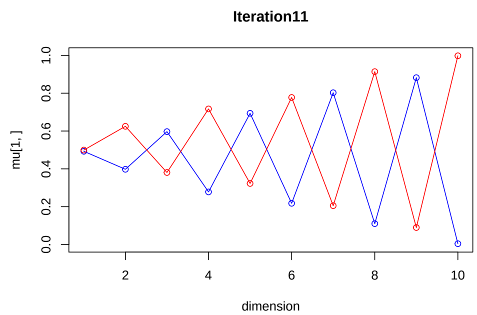
iteration: 8 log likelihood: -5646.139



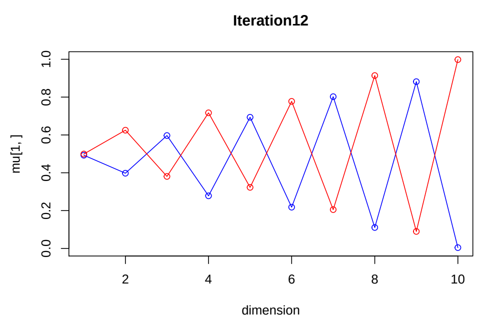
iteration: 9 log likelihood: -5644.608



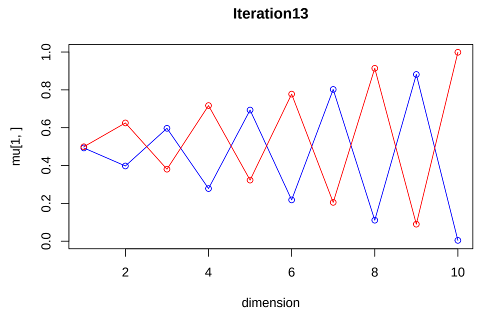
iteration: 10 log likelihood: -5643.615



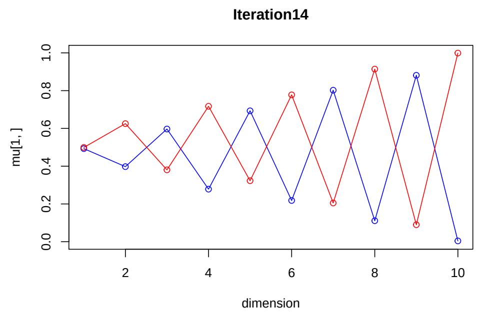
iteration: 11 log likelihood: -5642.913



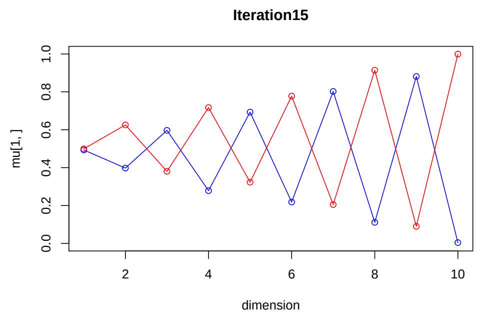
iteration: 12 log likelihood: -5642.386



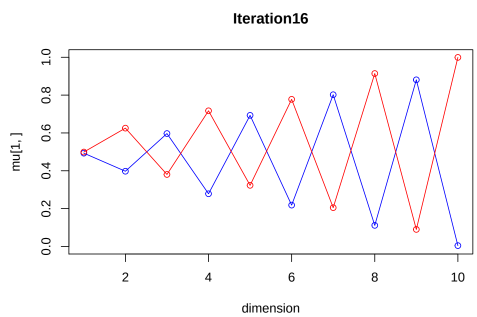
iteration: 13 log likelihood: -5641.977



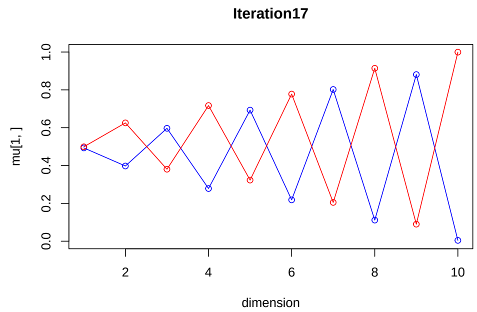
iteration: 14 log likelihood: -5641.649



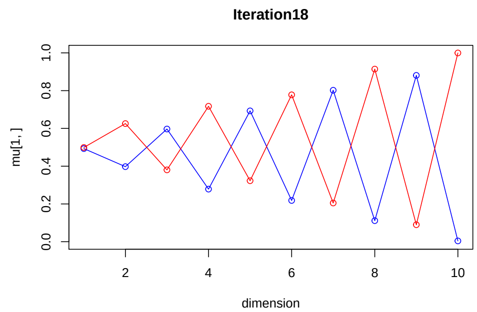
iteration: 15 log likelihood: -5641.382



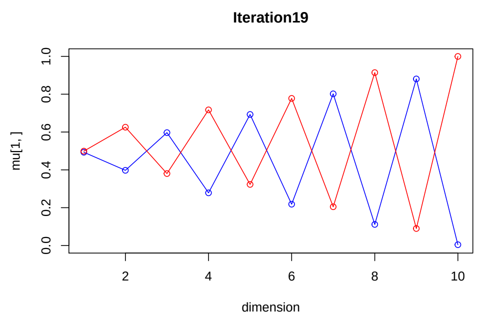
iteration: 16 log likelihood: -5641.161



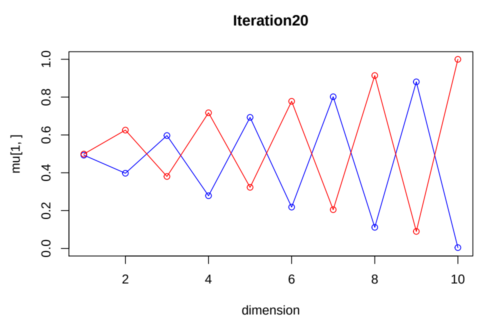
iteration: 17 log likelihood: -5640.975



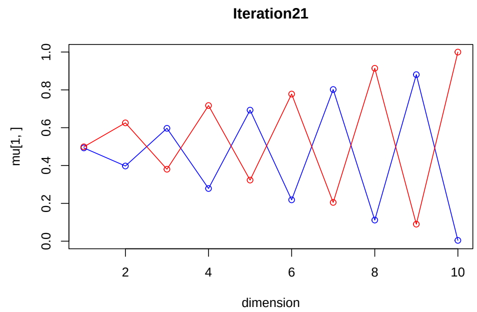
iteration: 18 log likelihood: -5640.819



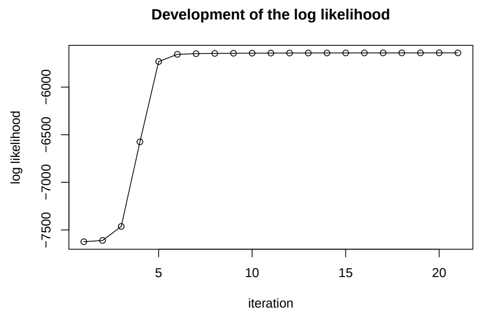
iteration: 19 log likelihood: -5640.685



iteration: 20 log likelihood: -5640.571



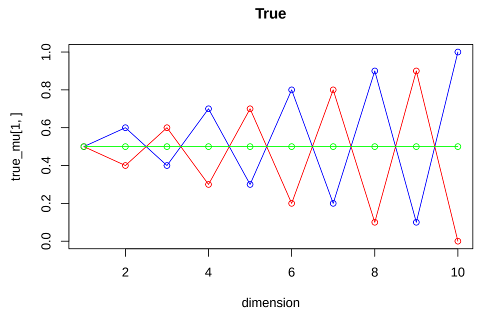

```
## iteration:  21 log likelihood:  -5640.473
```

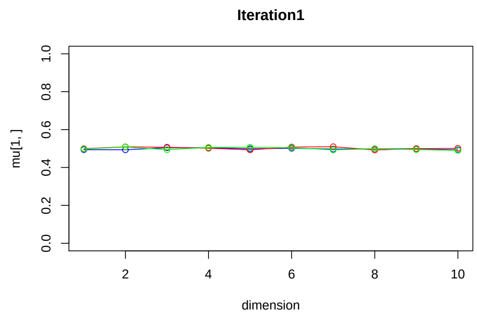



```
## $pi
## [1] 0.5110531 0.4889469
##
## $mu
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4931735 0.3974606 0.5967811 0.2785480 0.6927917 0.2184957 0.8018491
## [2,] 0.4989543 0.6255823 0.3804363 0.7171478 0.3230343 0.7778699 0.2049559
##           [,8] 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

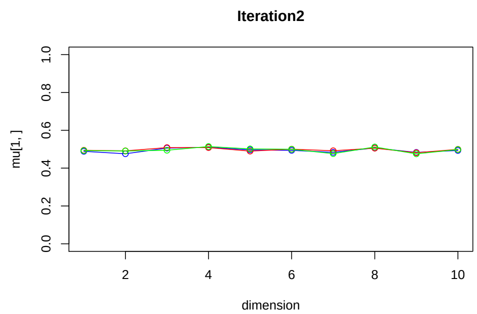
K=3

```
em_loop(3)
```

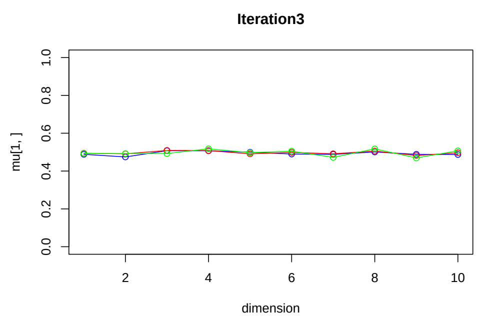




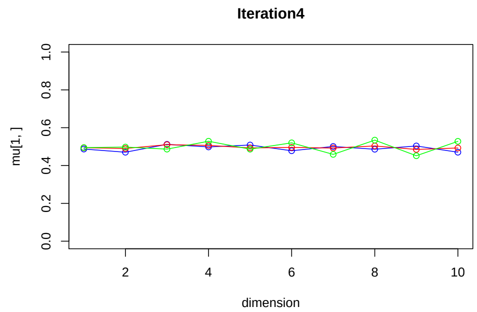

```
## iteration:  1 log likelihood:  -8029.723
```



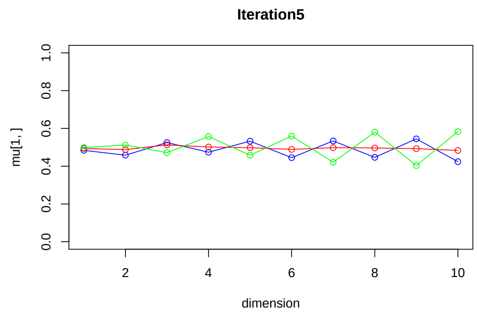

```
## iteration:  2 log likelihood:  -8027.183
```



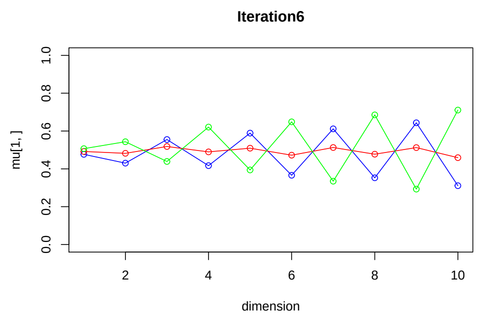

```
## iteration:  3 log likelihood:  -8024.696
```



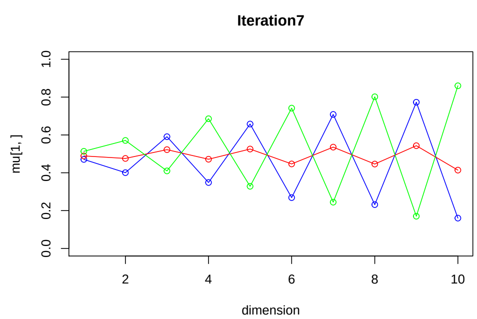

```
## iteration:  4 log likelihood:  -8005.631
```



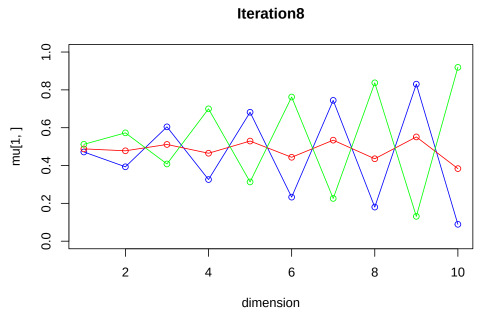

```
## iteration:  5 log likelihood:  -7877.606
```



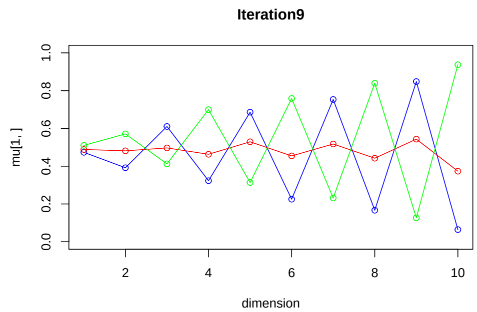

```
## iteration:  6 log likelihood:  -7403.513
```



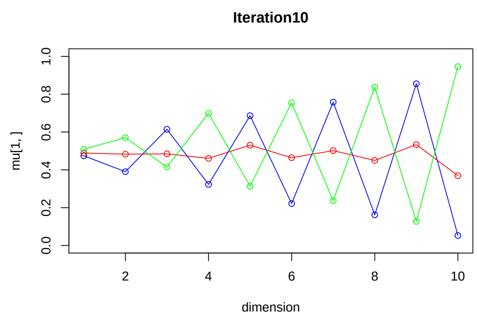

```
## iteration:  7 log likelihood:  -6936.919
```



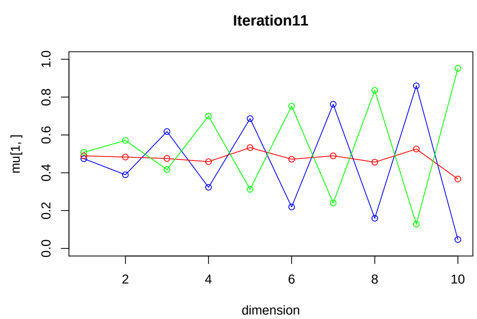

```
## iteration:  8 log likelihood:  -6818.582
```



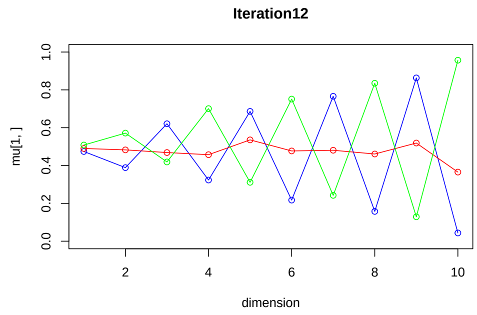

```
## iteration:  9 log likelihood:  -6791.377
```



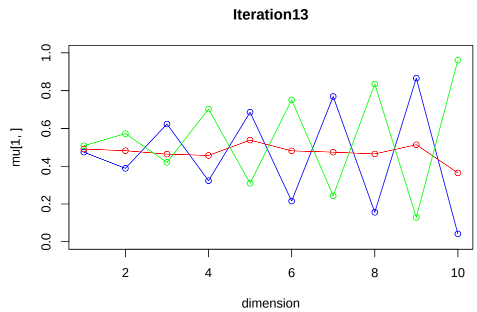

```
## iteration:  10 log likelihood:  -6780.713
```



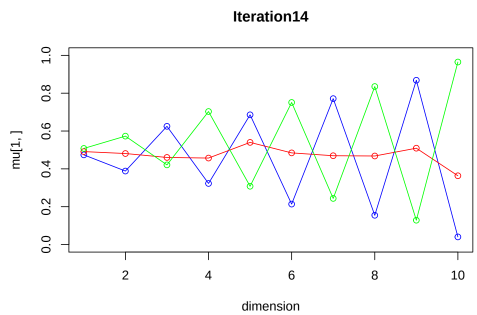

```
## iteration:  11 log likelihood:  -6774.958
```



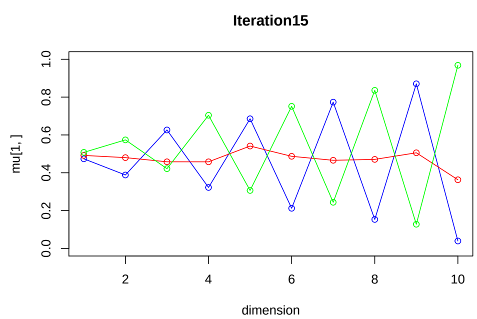

```
## iteration:  12 log likelihood:  -6771.261
```



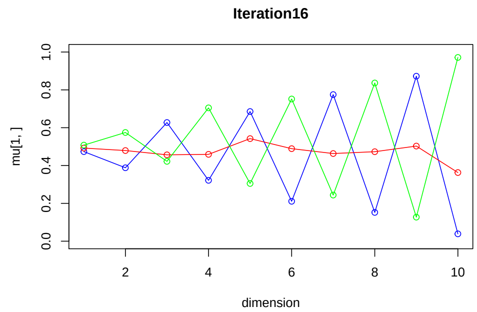

```
## iteration:  13 log likelihood:  -6768.606
```



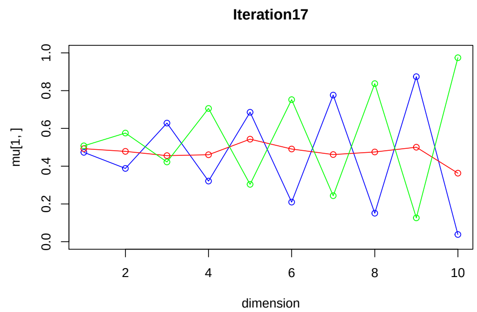

```
## iteration:  14 log likelihood:  -6766.535
```



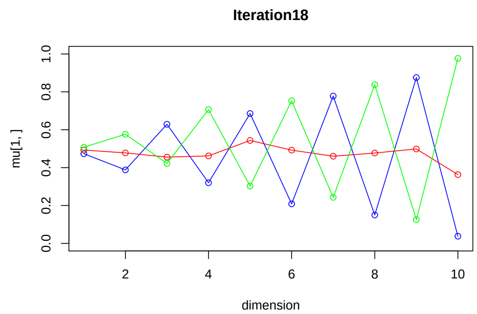

```
## iteration:  15 log likelihood:  -6764.815
```



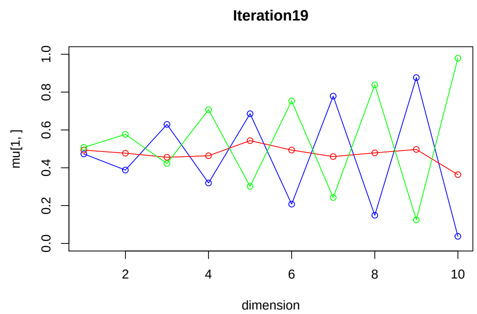

```
## iteration:  16 log likelihood:  -6763.316
```



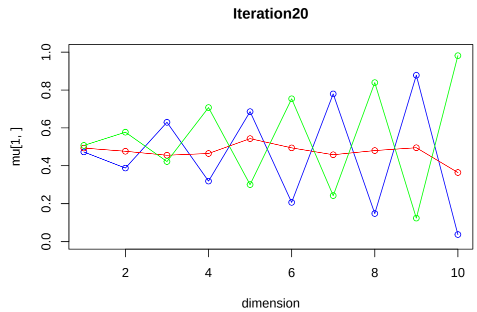

```
## iteration:  17 log likelihood:  -6761.967
```



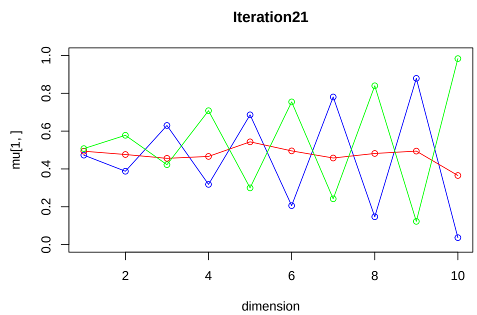

```
## iteration:  18 log likelihood:  -6760.727
```



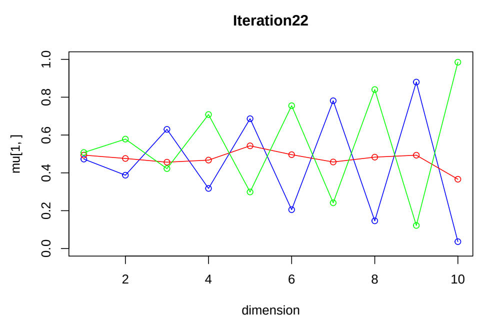

```
## iteration:  19 log likelihood:  -6759.572
```



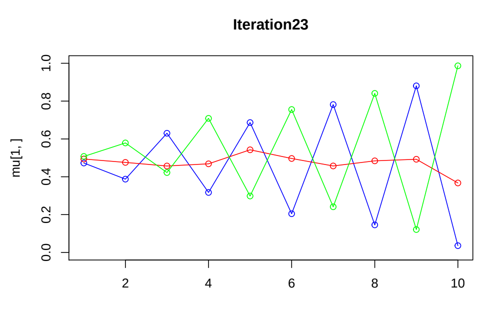
iteration: 20 log likelihood: -6758.491



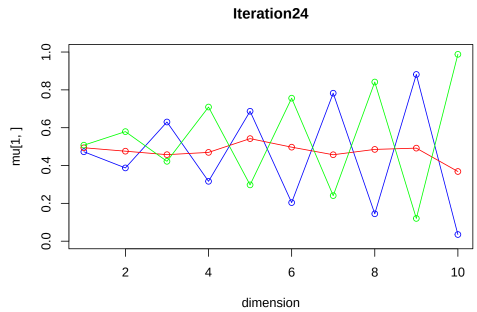
iteration: 21 log likelihood: -6757.475



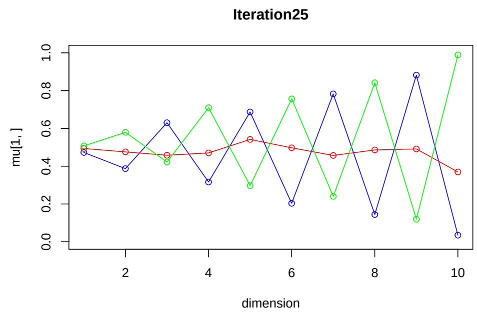
iteration: 22 log likelihood: -6756.521



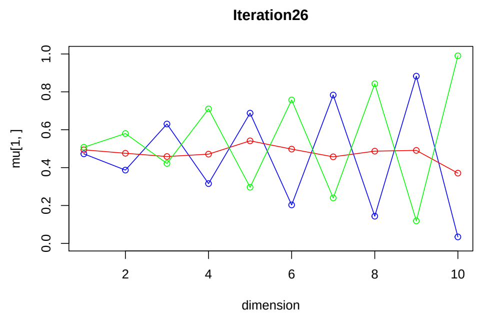
iteration: 23 log likelihood: -6755.625



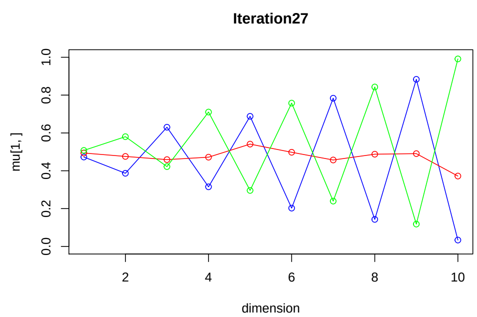
iteration: 24 log likelihood: -6754.784



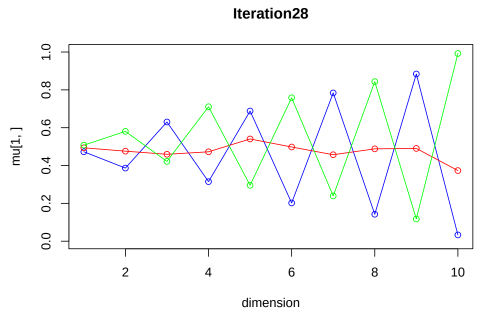
iteration: 25 log likelihood: -6753.996



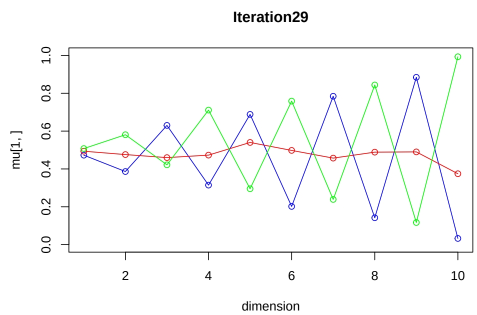
iteration: 26 log likelihood: -6753.26



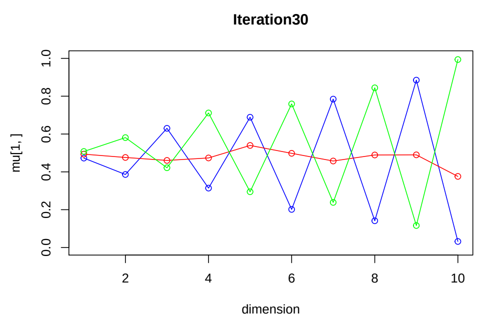
iteration: 27 log likelihood: -6752.571



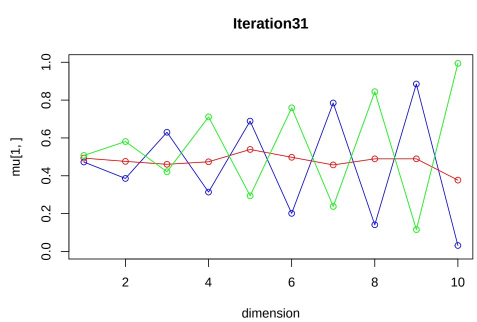

```
## iteration:  28 log likelihood:  -6751.928
```



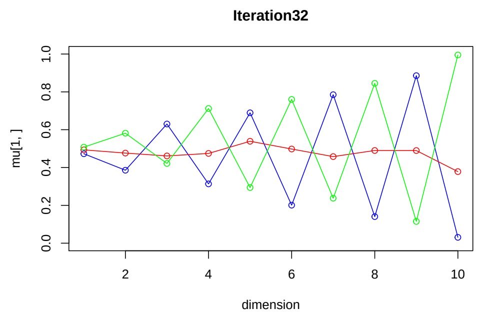

```
## iteration:  29 log likelihood:  -6751.328
```



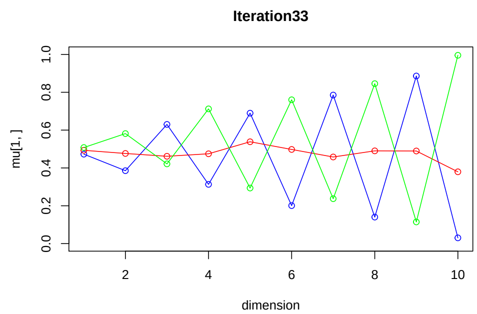

```
## iteration:  30 log likelihood:  -6750.768
```



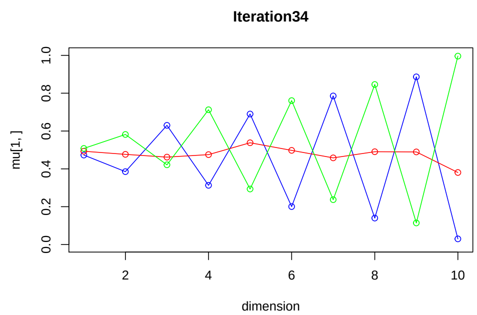

```
## iteration:  31 log likelihood:  -6750.246
```



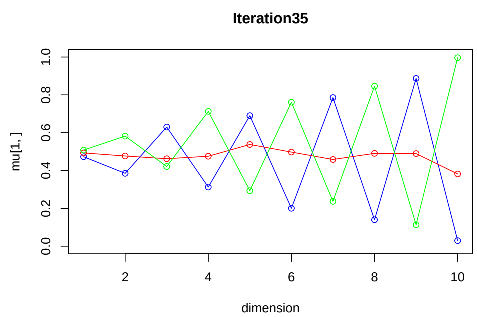

```
## iteration:  32 log likelihood:  -6749.758
```



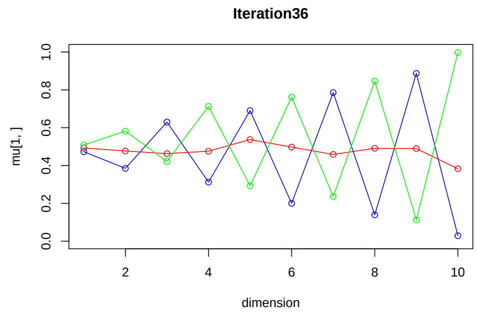

```
## iteration:  33 log likelihood:  -6749.304
```



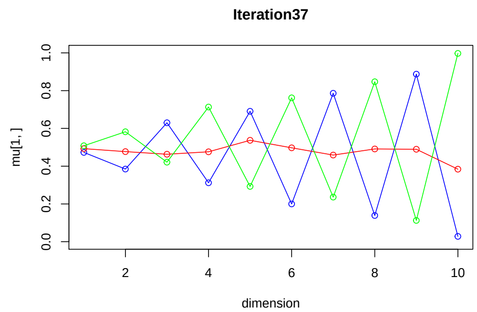

```
## iteration:  34 log likelihood:  -6748.88
```



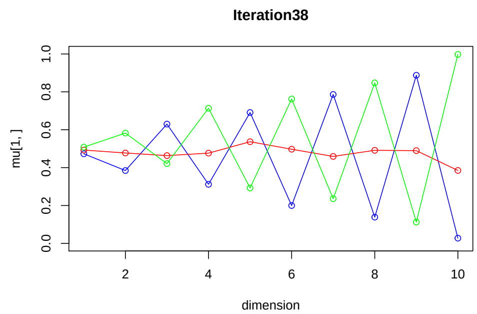

```
## iteration:  35 log likelihood:  -6748.484
```



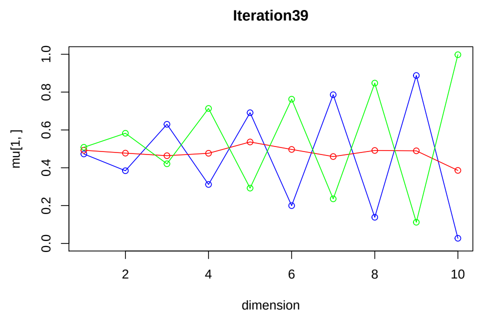

```
## iteration:  36 log likelihood:  -6748.114
```



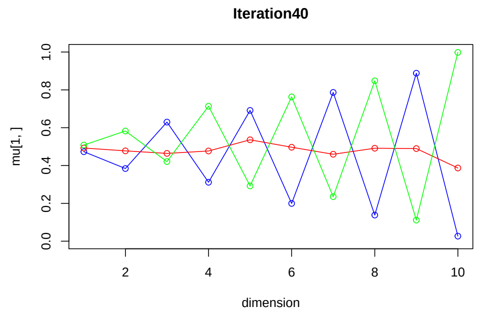

```
## iteration:  37 log likelihood:  -6747.767
```



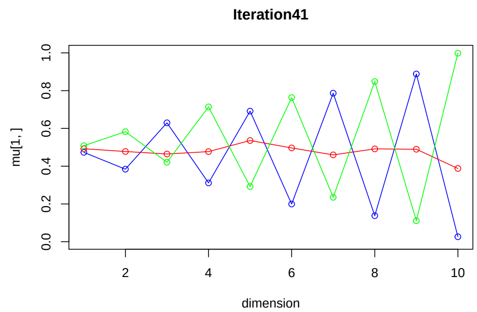

```
## iteration:  38 log likelihood:  -6747.444
```



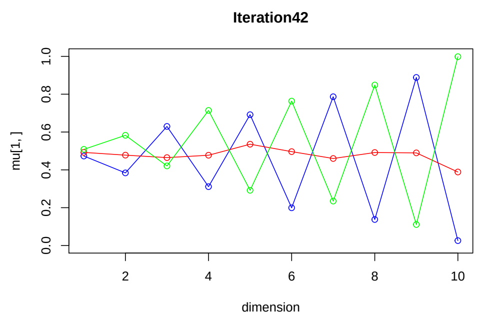

```
## iteration:  39 log likelihood:  -6747.14
```



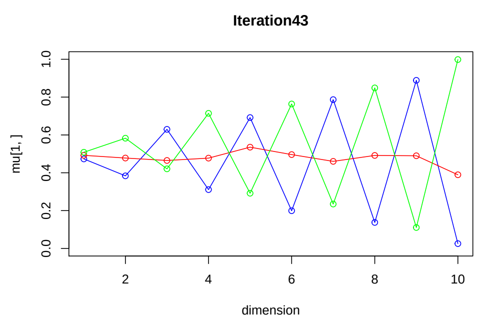

```
## iteration:  40 log likelihood:  -6746.856
```



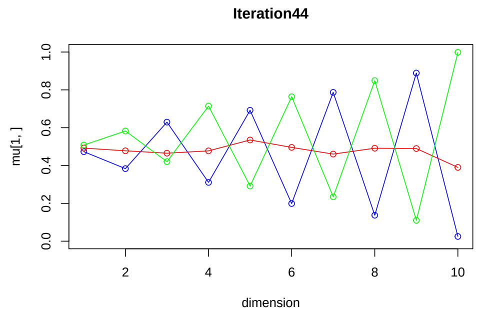

```
## iteration:  41 log likelihood:  -6746.589
```



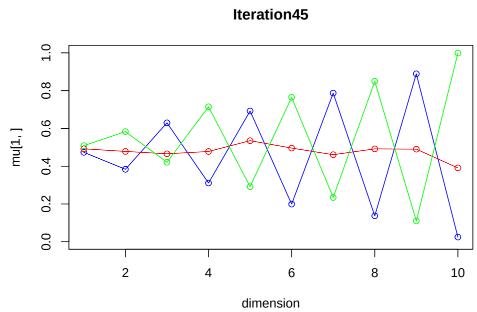

```
## iteration:  42 log likelihood:  -6746.338
```



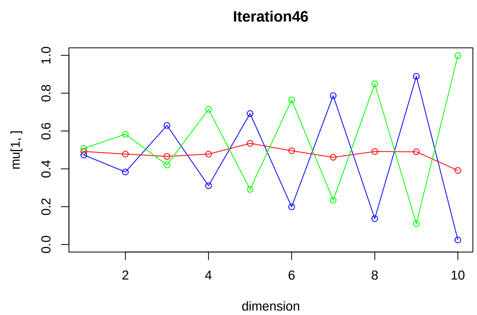

```
## iteration:  43 log likelihood:  -6746.102
```



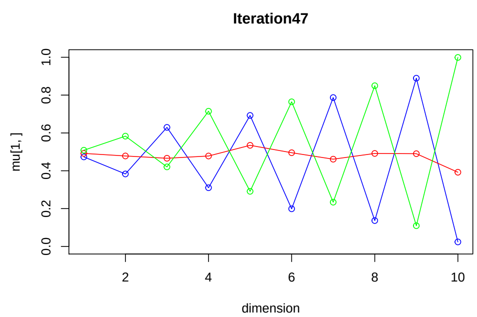

```
## iteration:  44 log likelihood:  -6745.88
```



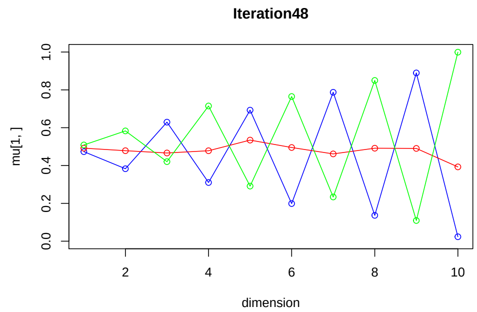

```
## iteration:  45 log likelihood:  -6745.67
```



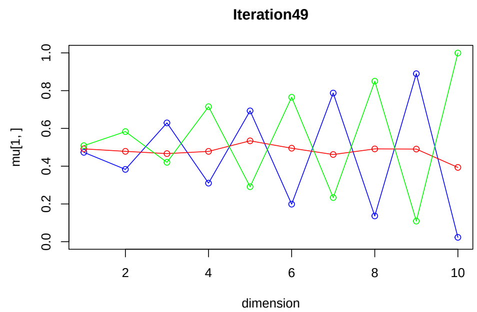

```
## iteration:  46 log likelihood:  -6745.472
```



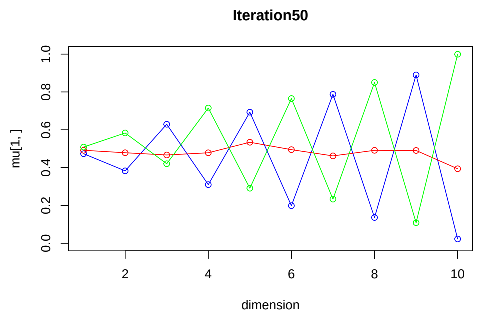

```
## iteration:  47 log likelihood:  -6745.285
```



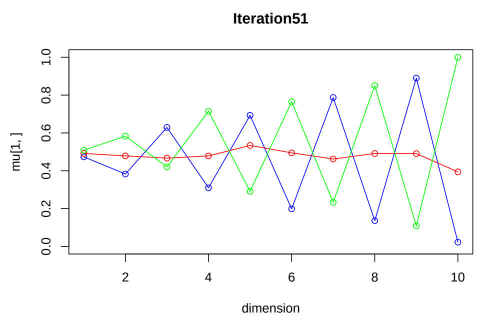

```
## iteration:  48 log likelihood:  -6745.108
```



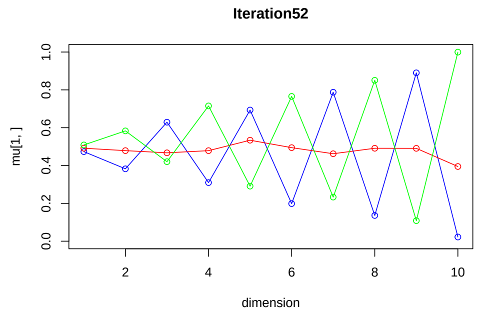

```
## iteration:  49 log likelihood:  -6744.939
```



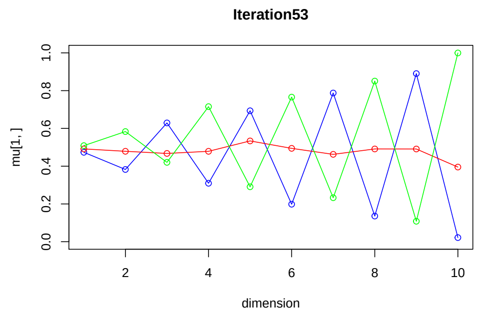

```
## iteration:  50 log likelihood:  -6744.78
```



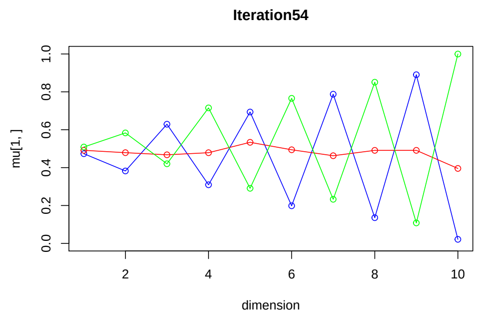

```
## iteration:  51 log likelihood:  -6744.627
```



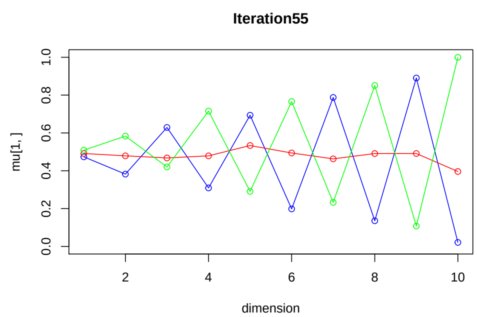
iteration: 52 log likelihood: -6744.483



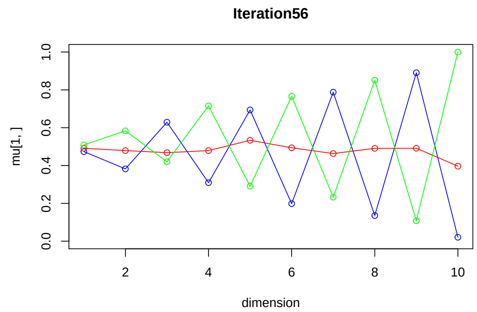
iteration: 53 log likelihood: -6744.344



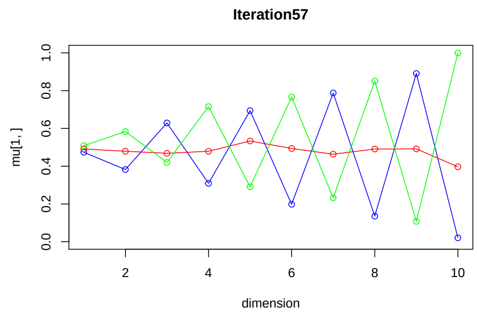
iteration: 54 log likelihood: -6744.212



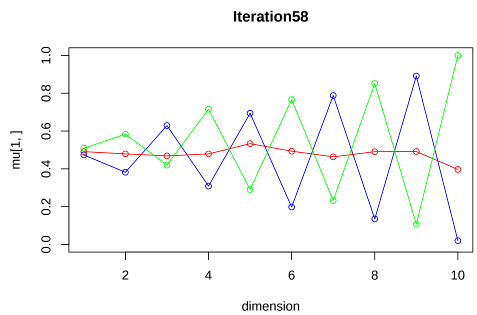
iteration: 55 log likelihood: -6744.086



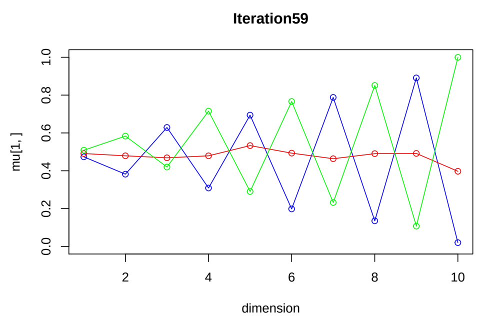

```
## iteration:  56 log likelihood:  -6743.964
```



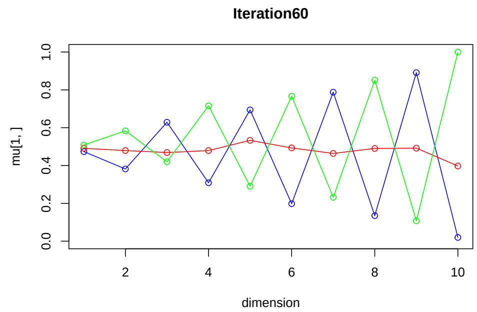

```
## iteration:  57 log likelihood:  -6743.848
```



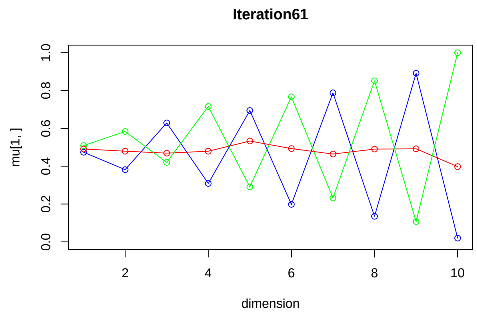

```
## iteration:  58 log likelihood:  -6743.736
```



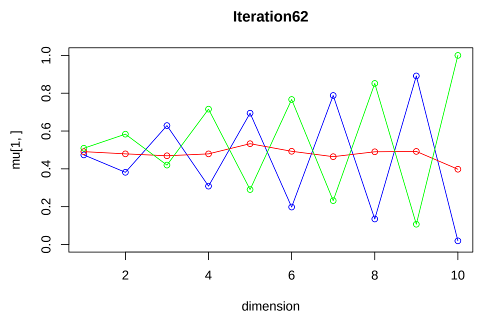

```
## iteration:  59 log likelihood:  -6743.628
```



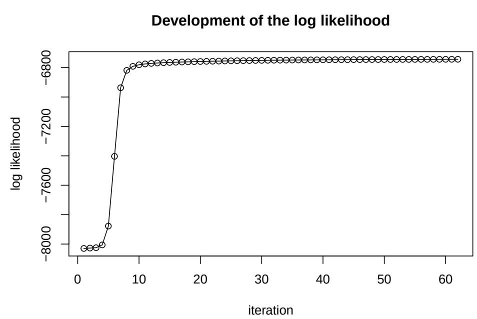

```
## iteration:  60 log likelihood:  -6743.524
```




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




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

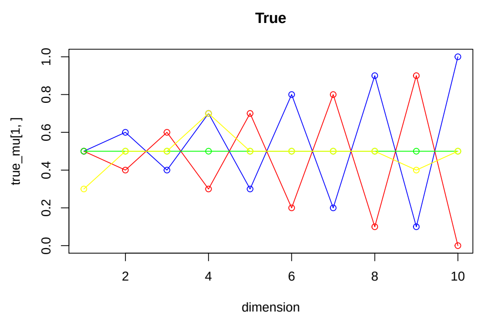


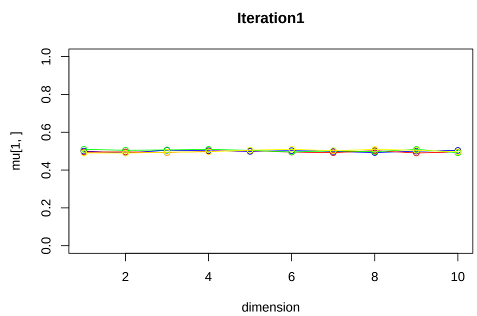

```
## $pi
## [1] 0.3259592 0.3044579 0.3695828
##
```

```
## $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
```

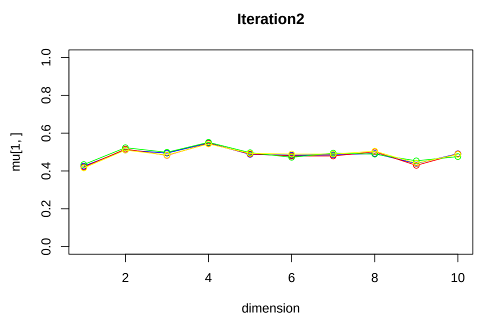
K=4

```r
em_loop(4)
```

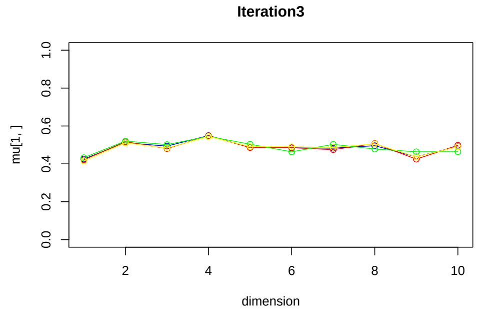




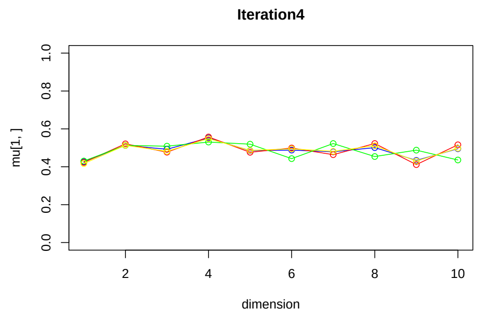

```
## iteration:  1 log likelihood:  -8316.904
```



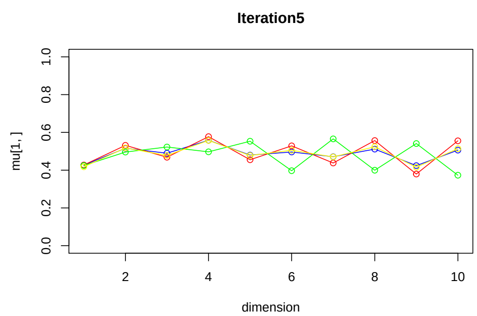
iteration: 2 log likelihood: -8291.114



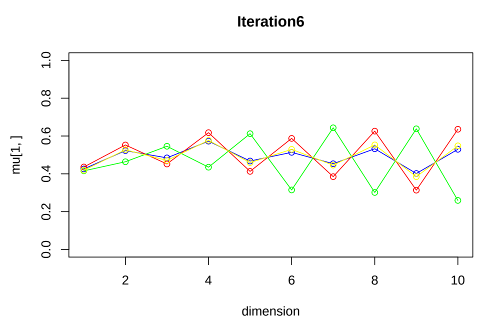
iteration: 3 log likelihood: -8286.966



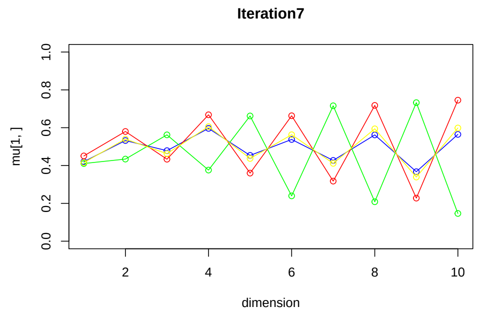
iteration: 4 log likelihood: -8264.806



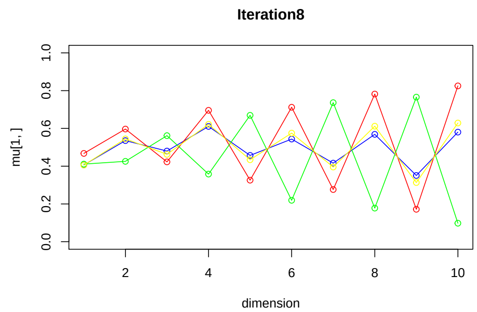
iteration: 5 log likelihood: -8161.19



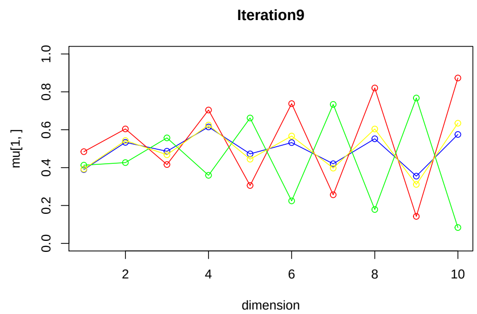
iteration: 6 log likelihood: -7868.89



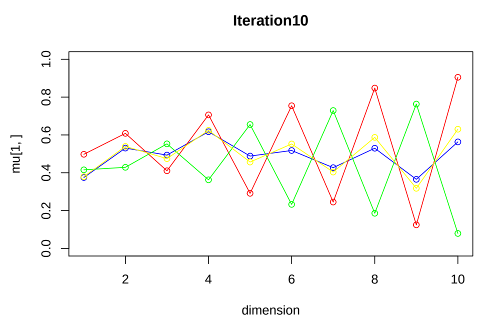
iteration: 7 log likelihood: -7570.873



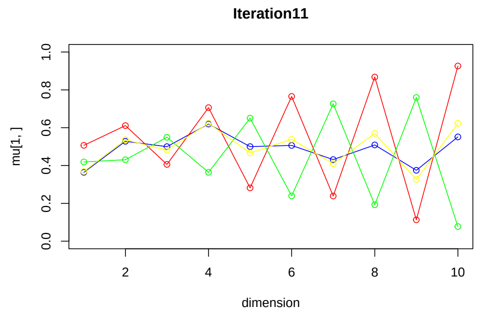
iteration: 8 log likelihood: -7445.719



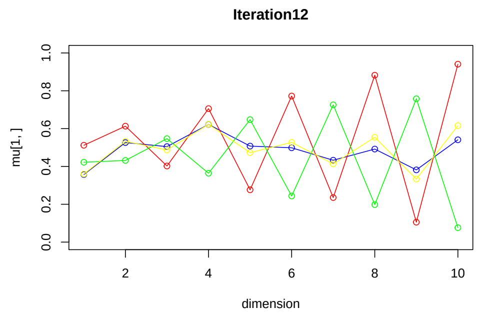
iteration: 9 log likelihood: -7389.741



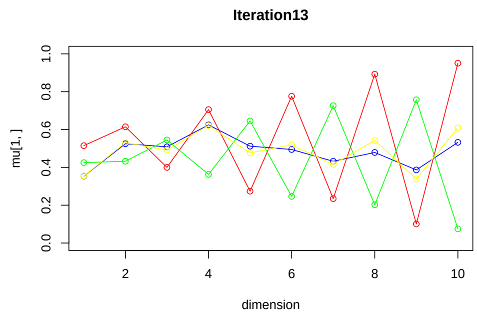
iteration: 10 log likelihood: -7356.803



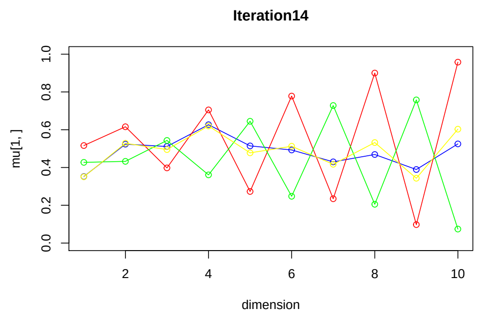

```
## iteration:  11 log likelihood:  -7337.208
```



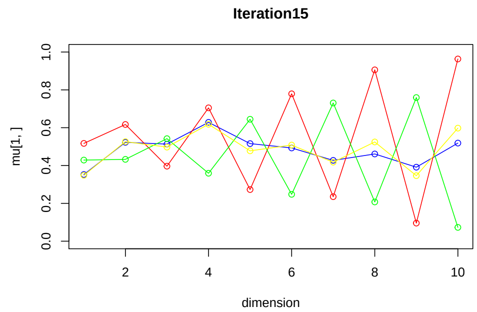

```
## iteration:  12 log likelihood:  -7326.118
```



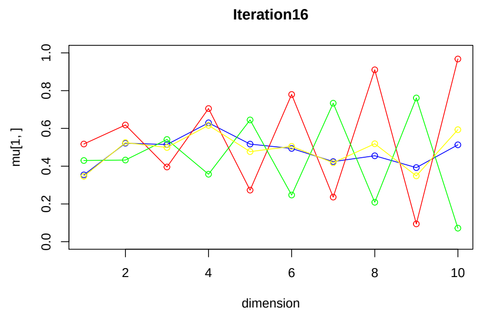

```
## iteration:  13 log likelihood:  -7319.998
```



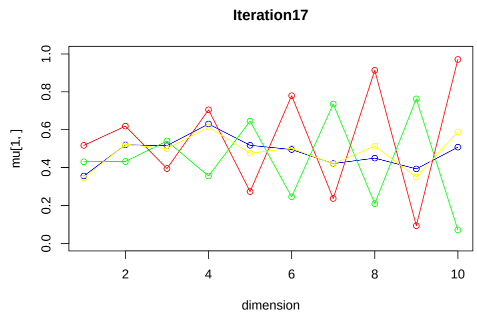

```
## iteration:  14 log likelihood:  -7316.6
```



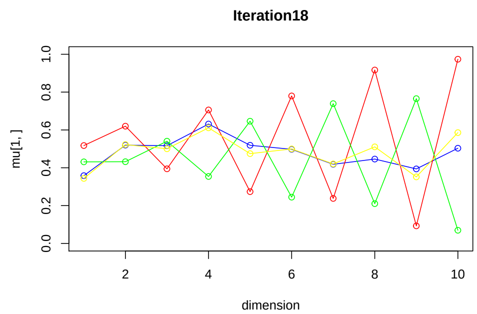
iteration: 15 log likelihood: -7314.666



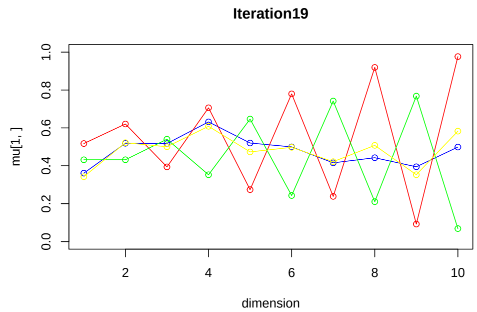
iteration: 16 log likelihood: -7313.528



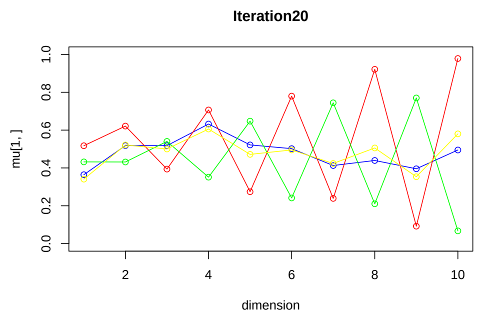
iteration: 17 log likelihood: -7312.829



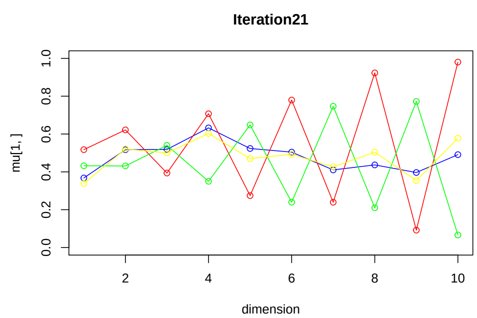
iteration: 18 log likelihood: -7312.367



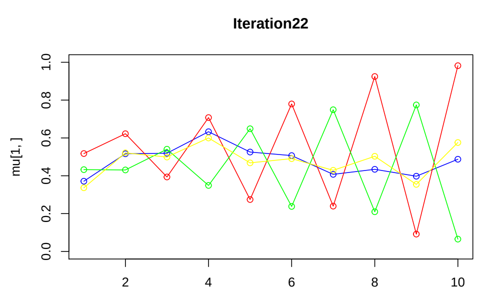
iteration: 19 log likelihood: -7312.024



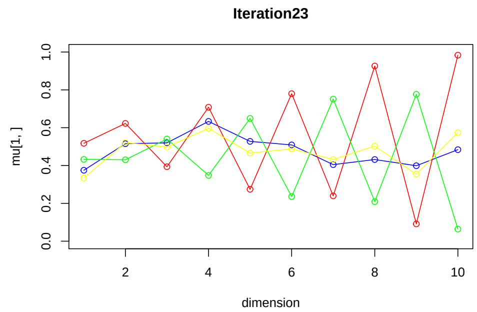
iteration: 20 log likelihood: -7311.723



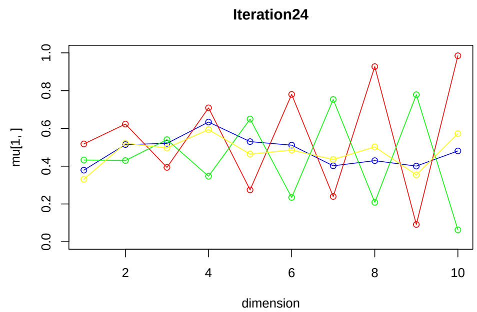
iteration: 21 log likelihood: -7311.407



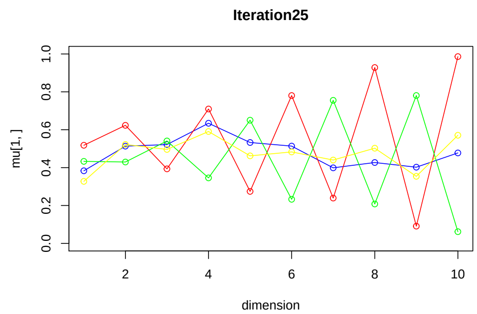
iteration: 22 log likelihood: -7311.036



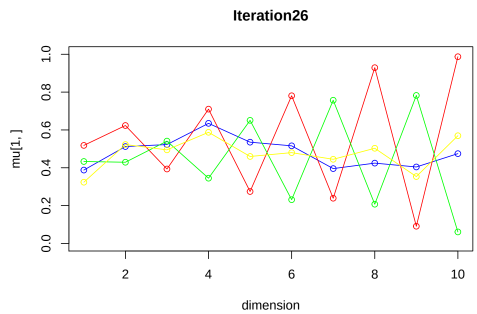
iteration: 23 log likelihood: -7310.574



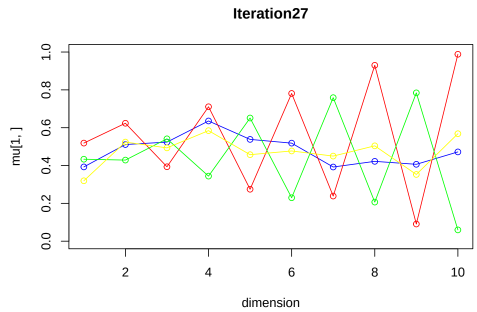
iteration: 24 log likelihood: -7309.988



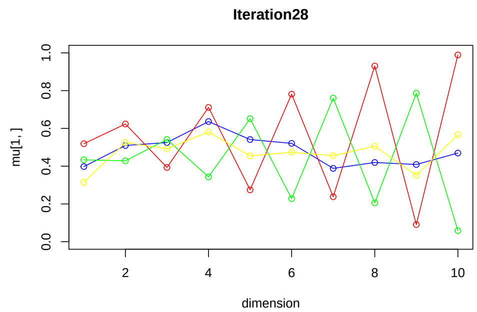
iteration: 25 log likelihood: -7309.248



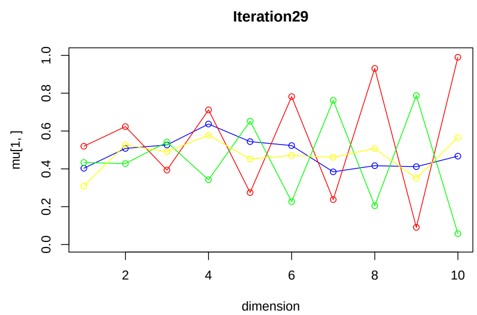
iteration: 26 log likelihood: -7308.322



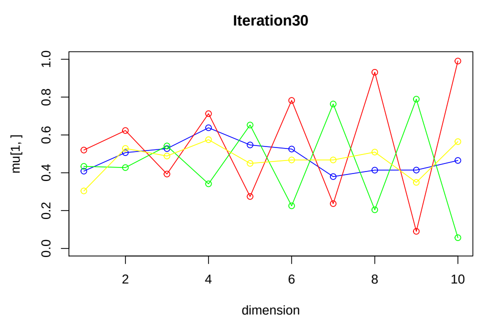

```
## iteration:  27 log likelihood:  -7307.185
```



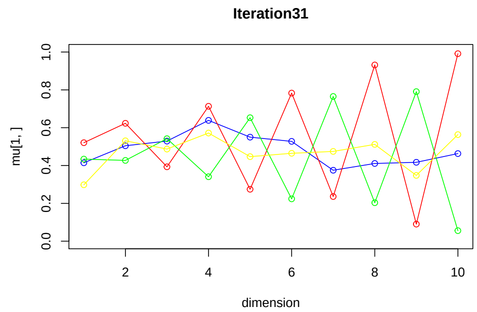

```
## iteration:  28 log likelihood:  -7305.809
```



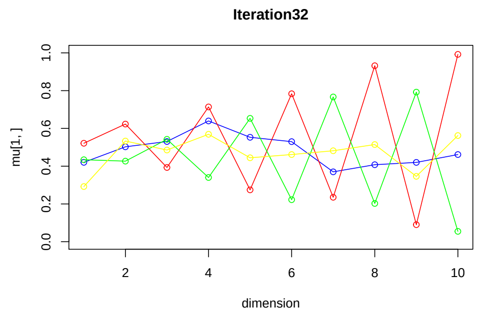

```
## iteration:  29 log likelihood:  -7304.176
```



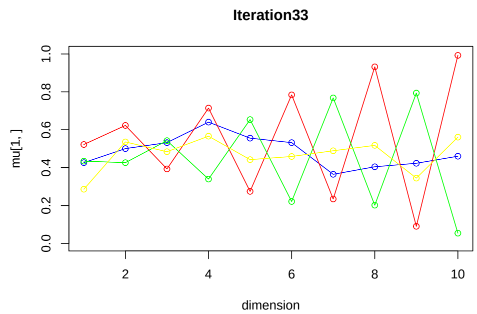

```
## iteration:  30 log likelihood:  -7302.273
```



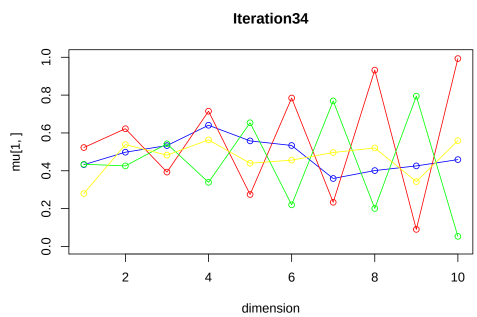

```
## iteration:  31 log likelihood:  -7300.1
```



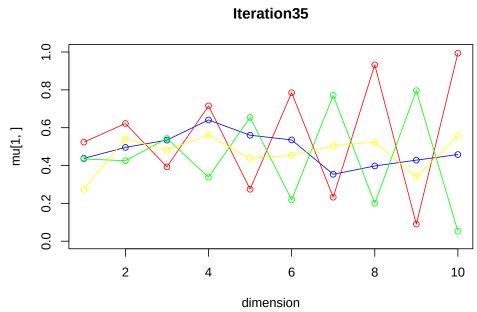

```
## iteration:  32 log likelihood:  -7297.671
```



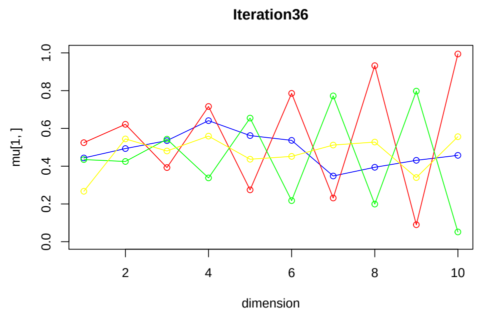

```
## iteration:  33 log likelihood:  -7295.014
```



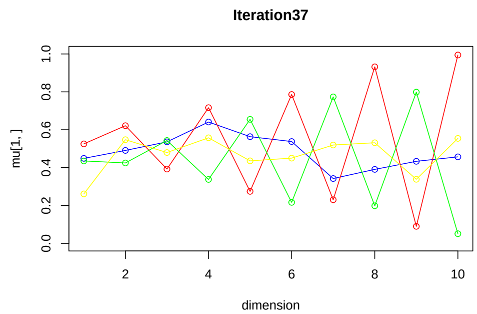

```
## iteration:  34 log likelihood:  -7292.171
```



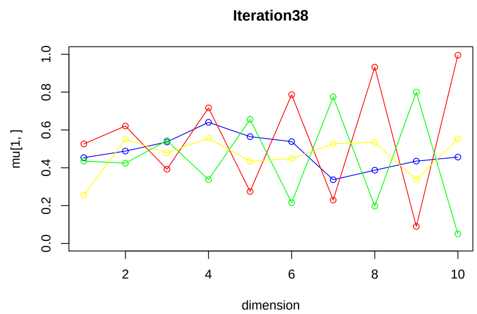

```
## iteration:  35 log likelihood:  -7289.196
```



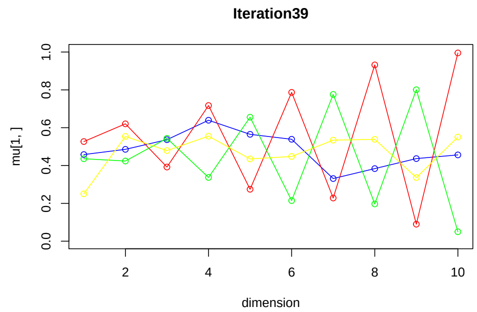

```
## iteration:  36 log likelihood:  -7286.15
```



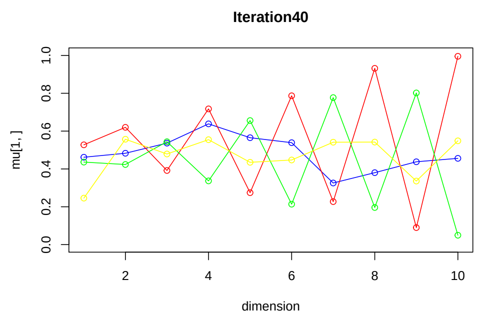

```
## iteration:  37 log likelihood:  -7283.093
```



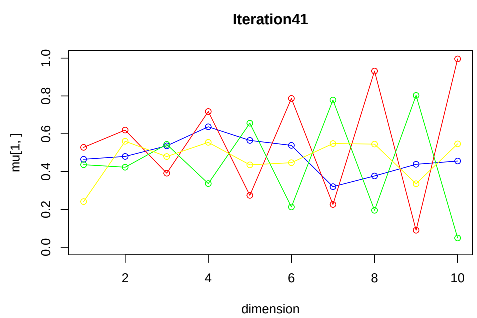

```
## iteration:  38 log likelihood:  -7280.079
```



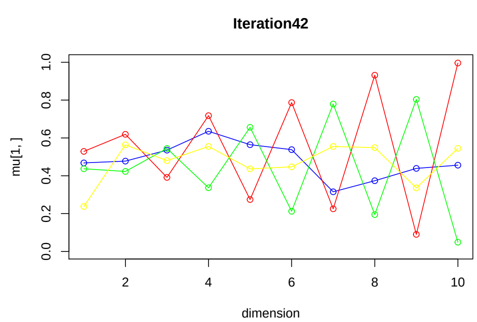

```
## iteration:  39 log likelihood:  -7277.151
```



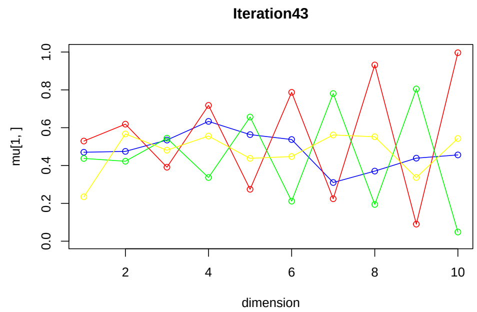

```
## iteration:  40 log likelihood:  -7274.34
```



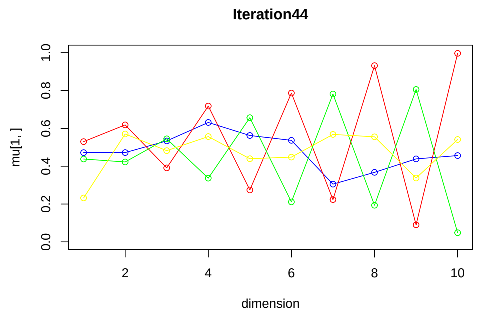

```
## iteration:  41 log likelihood:  -7271.66
```



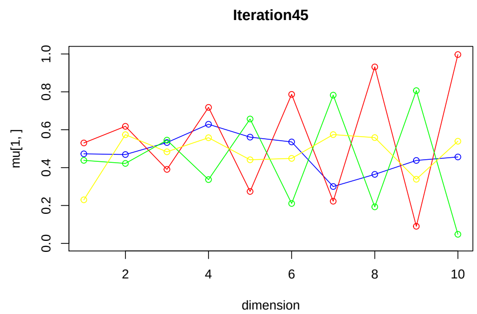

```
## iteration:  42 log likelihood:  -7269.116
```



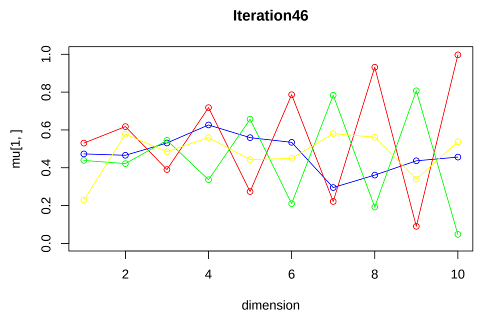

```
## iteration:  43 log likelihood:  -7266.7
```



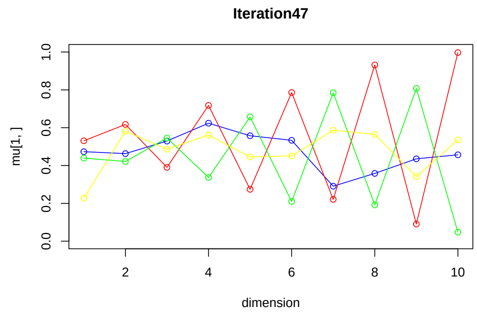

```
## iteration:  44 log likelihood:  -7264.398
```



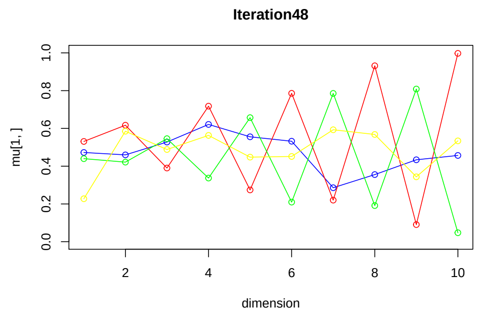

```
## iteration:  45 log likelihood:  -7262.189
```



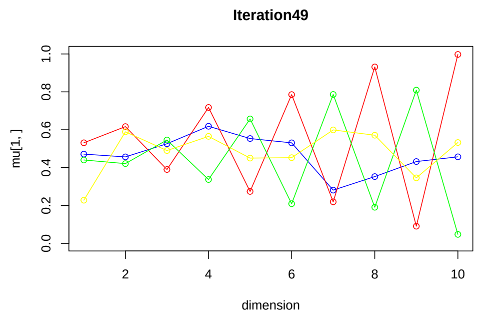

```
## iteration:  46 log likelihood:  -7260.051
```



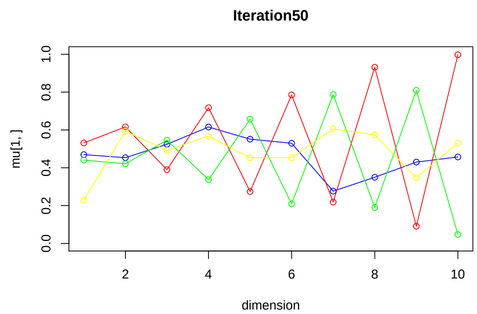

```
## iteration:  47 log likelihood:  -7257.96
```



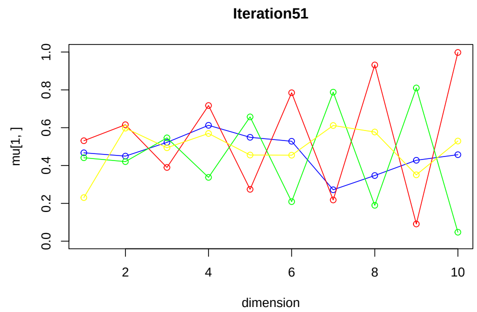

```
## iteration:  48 log likelihood:  -7255.892
```



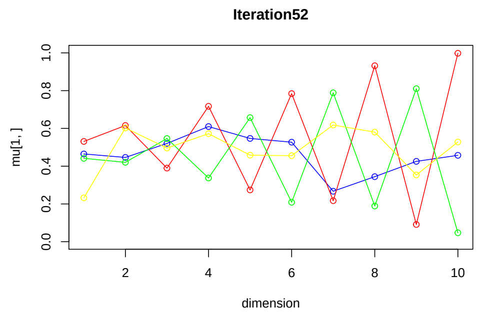

```
## iteration:  49 log likelihood:  -7253.824
```



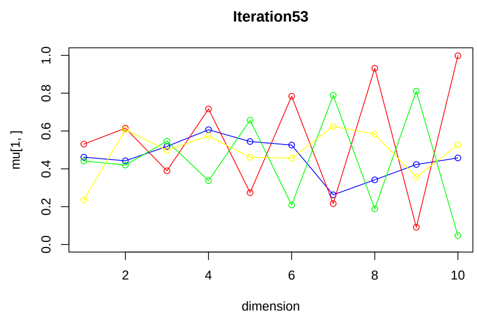

```
## iteration:  50 log likelihood:  -7251.733
```



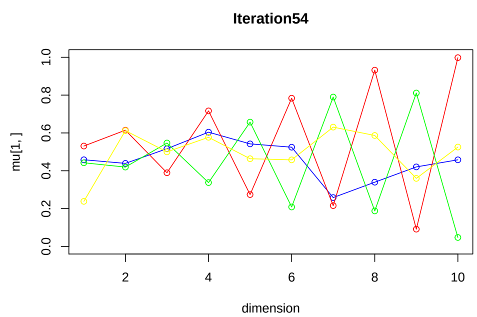

```
## iteration:  51 log likelihood:  -7249.603
```



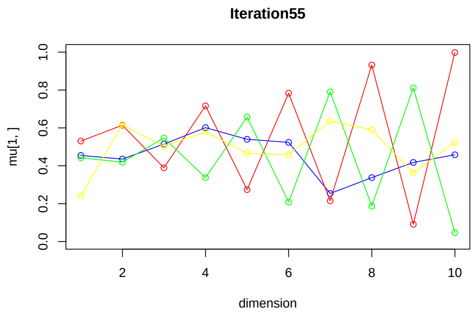

```
## iteration:  52 log likelihood:  -7247.419
```



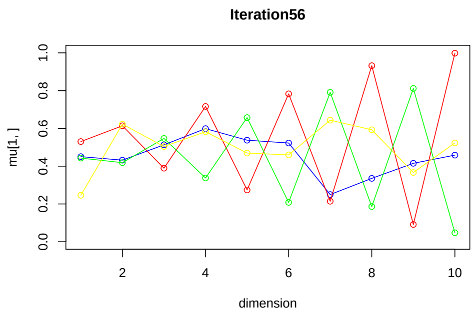

```
## iteration:  53 log likelihood:  -7245.17
```



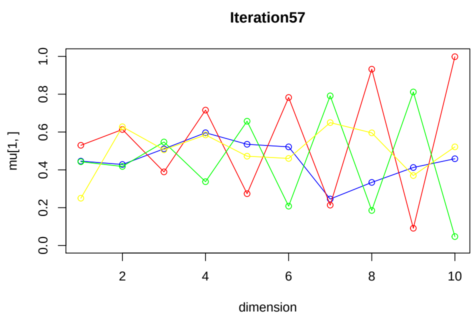

```
## iteration:  54 log likelihood:  -7242.853
```



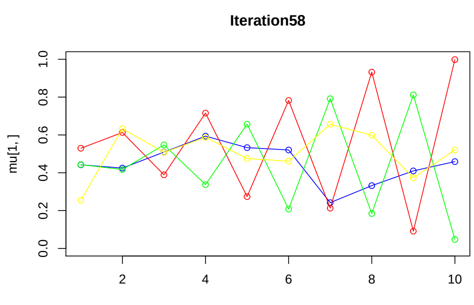

```
## iteration:  55 log likelihood:  -7240.472
```



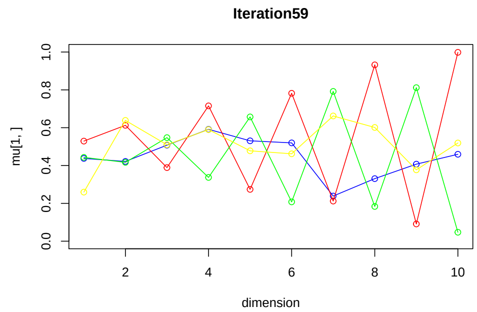

```
## iteration:  56 log likelihood:  -7238.038
```



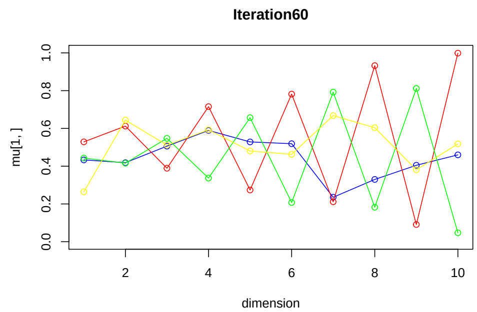

```
## iteration:  57 log likelihood:  -7235.571
```



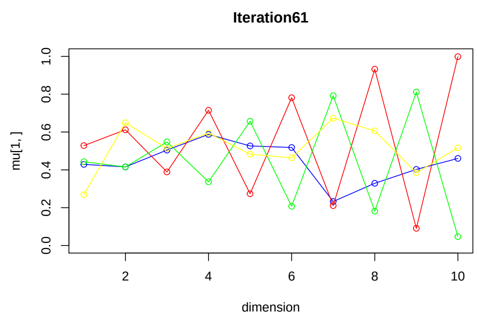

```
## iteration:  58 log likelihood:  -7233.095
```



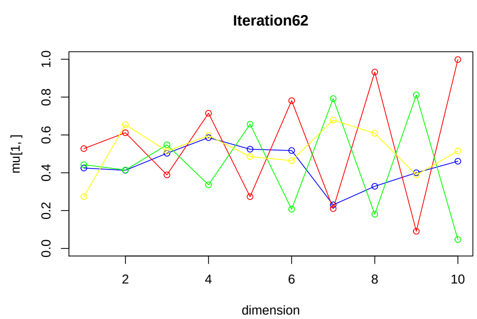

```
## iteration:  59 log likelihood:  -7230.64
```



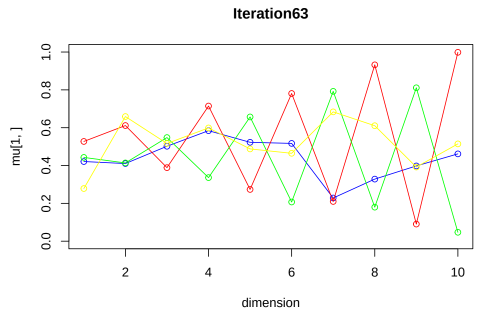

```
## iteration:  60 log likelihood:  -7228.239
```



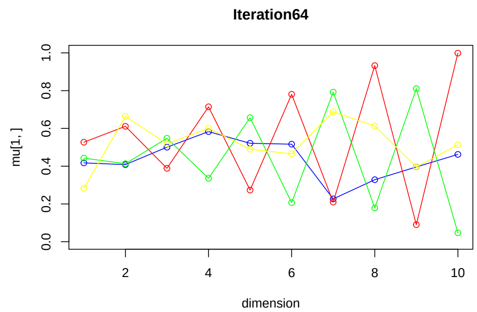

```
## iteration:  61 log likelihood:  -7225.925
```



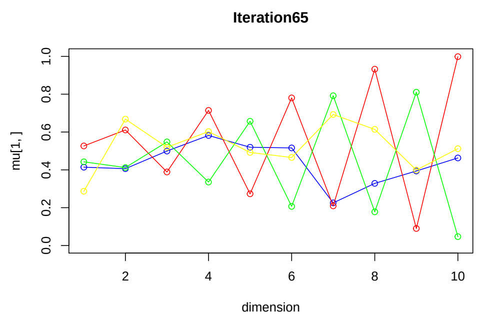

```
## iteration:  62 log likelihood:  -7223.725
```



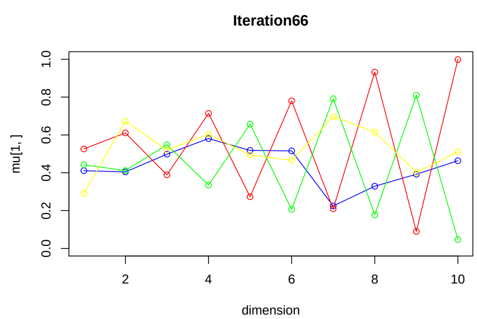
iteration: 63 log likelihood: -7221.663



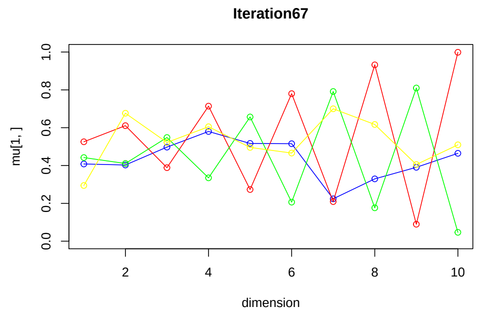
iteration: 64 log likelihood: -7219.755



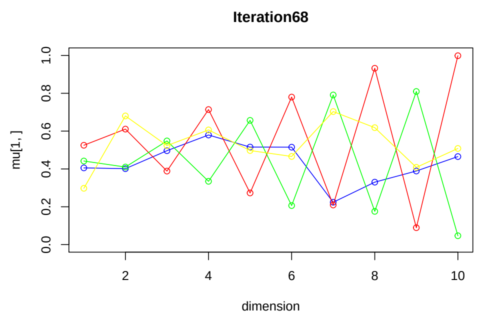
iteration: 65 log likelihood: -7218.01



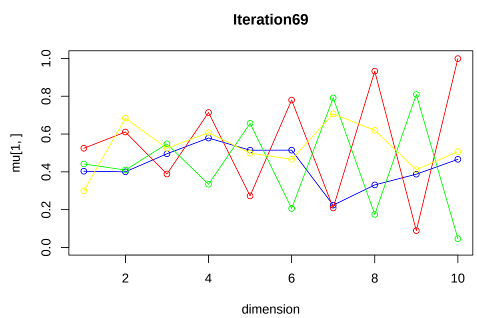
iteration: 66 log likelihood: -7216.431



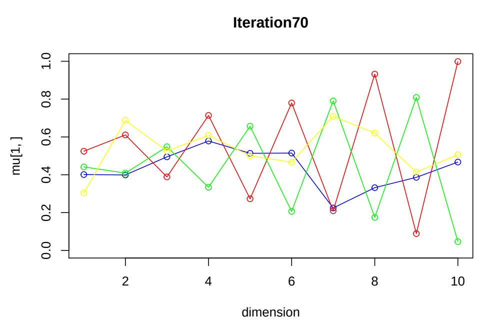
iteration: 67 log likelihood: -7215.013



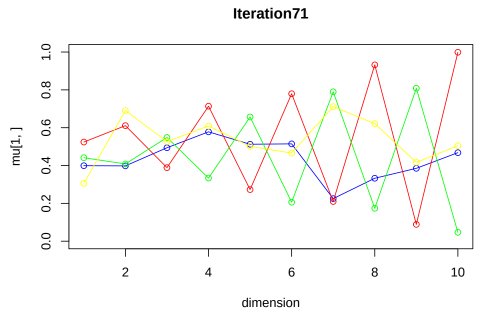
iteration: 68 log likelihood: -7213.748



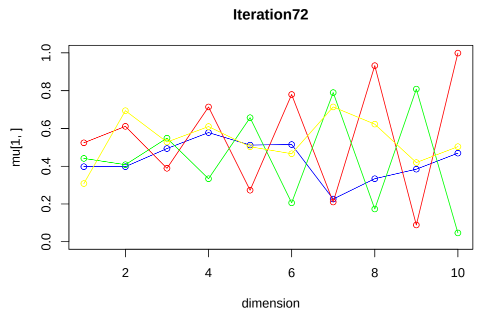
iteration: 69 log likelihood: -7212.621



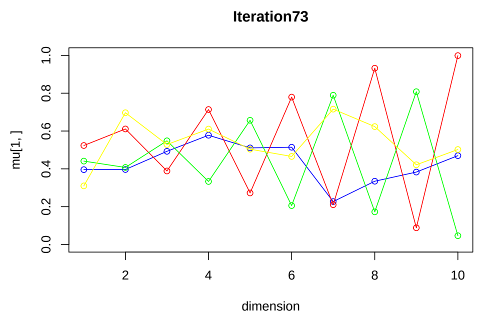
iteration: 70 log likelihood: -7211.62



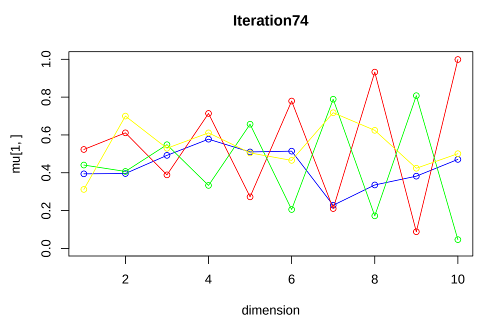
iteration: 71 log likelihood: -7210.727



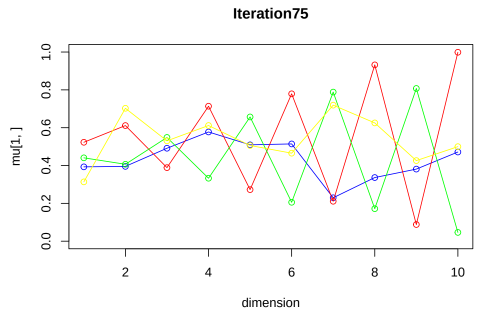
iteration: 72 log likelihood: -7209.929



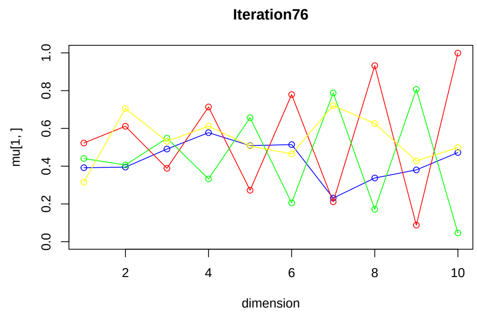
iteration: 73 log likelihood: -7209.208



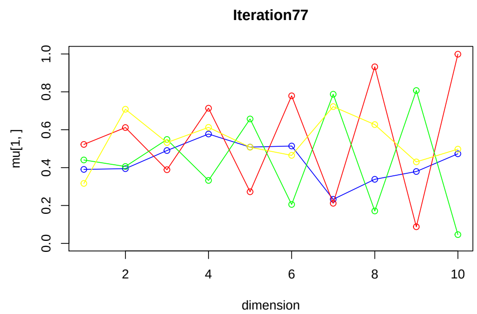
iteration: 74 log likelihood: -7208.552



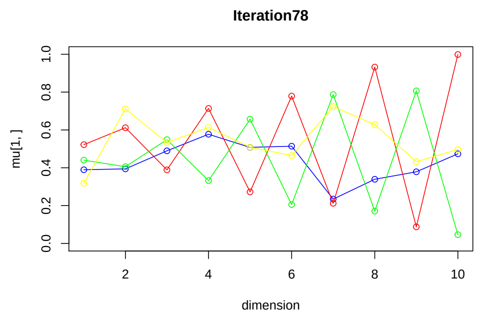
iteration: 75 log likelihood: -7207.946



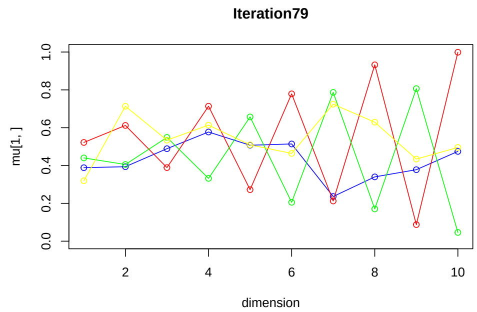
iteration: 76 log likelihood: -7207.38



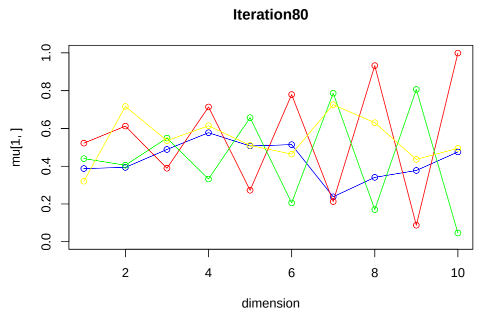
iteration: 77 log likelihood: -7206.844



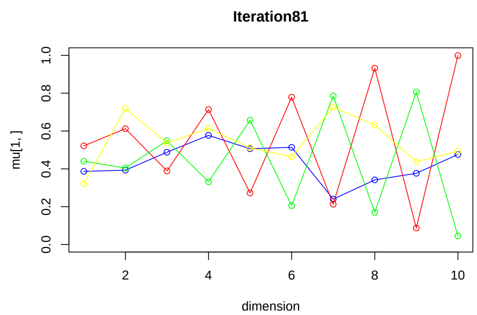
iteration: 78 log likelihood: -7206.327



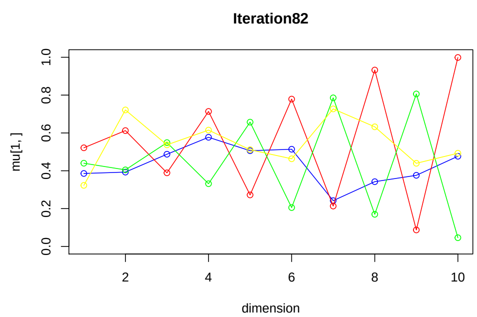
iteration: 79 log likelihood: -7205.824



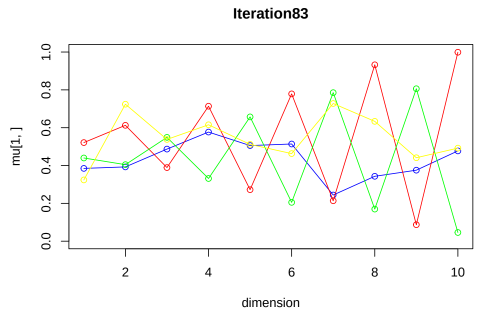
iteration: 80 log likelihood: -7205.326



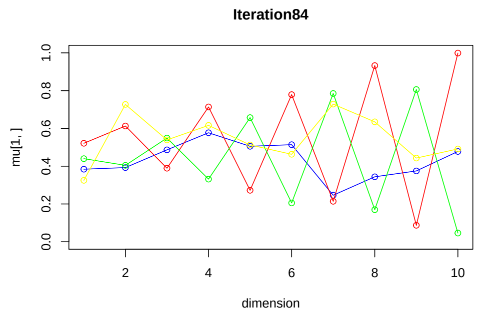
iteration: 81 log likelihood: -7204.829



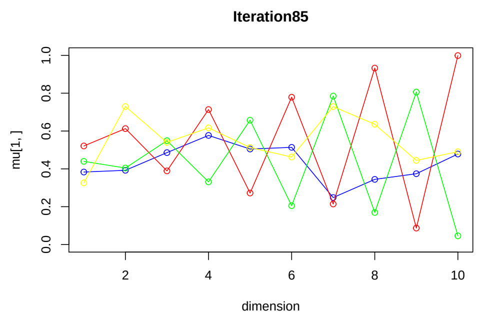
iteration: 82 log likelihood: -7204.327



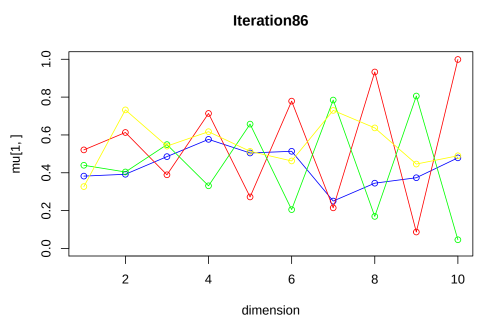

```
## iteration:  83 log likelihood:  -7203.816
```



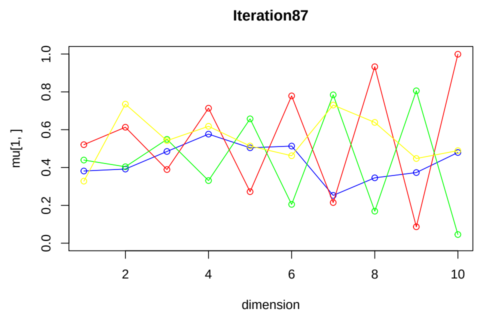

```
## iteration:  84 log likelihood:  -7203.294
```



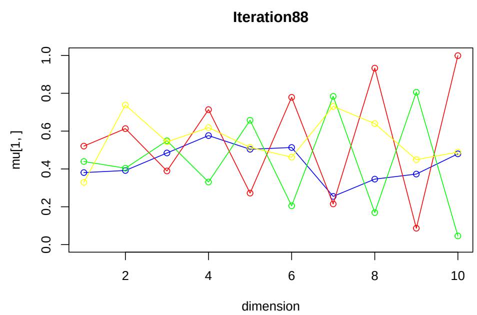

```
## iteration:  85 log likelihood:  -7202.756
```



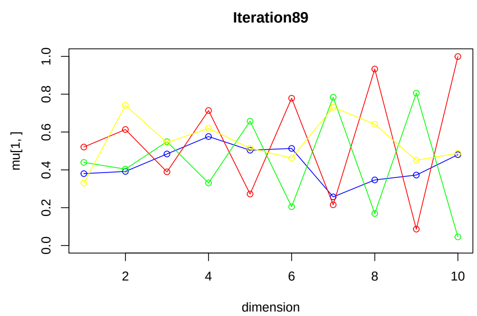

```
## iteration:  86 log likelihood:  -7202.201
```



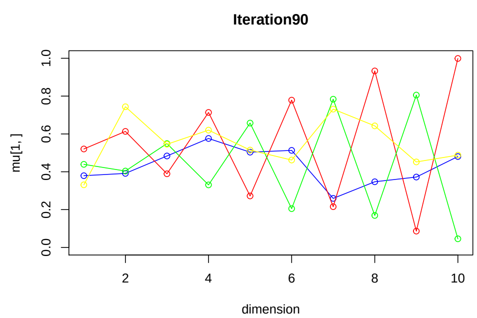
iteration: 87 log likelihood: -7201.627



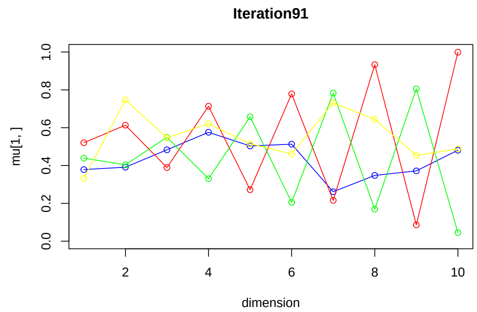
iteration: 88 log likelihood: -7201.032



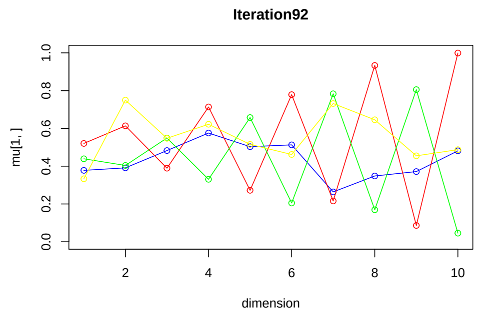
iteration: 89 log likelihood: -7200.414



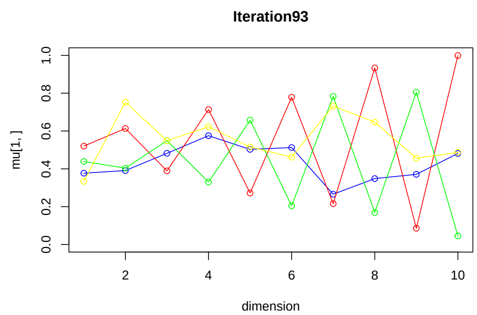
iteration: 90 log likelihood: -7199.773



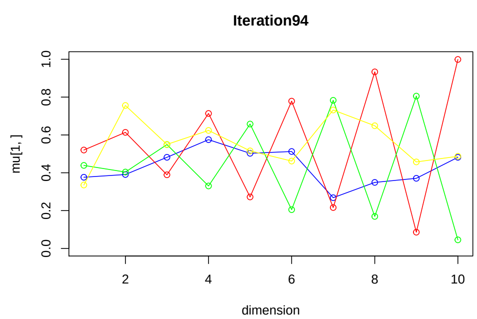

```
## iteration:  91 log likelihood:  -7199.107
```



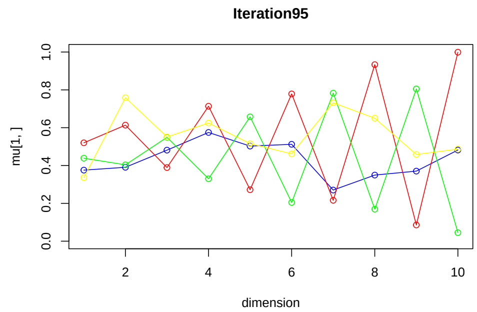

```
## iteration:  92 log likelihood:  -7198.416
```



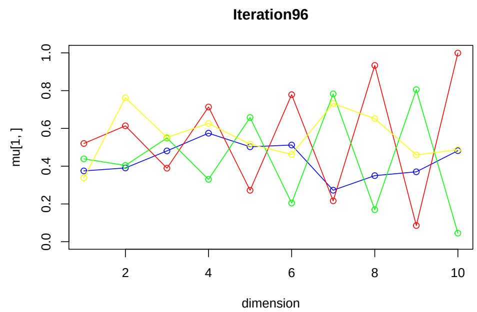

```
## iteration:  93 log likelihood:  -7197.7
```



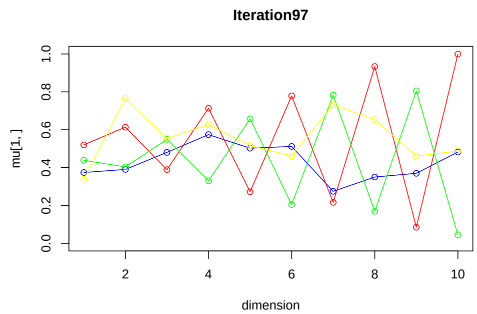

```
## iteration:  94 log likelihood:  -7196.957
```



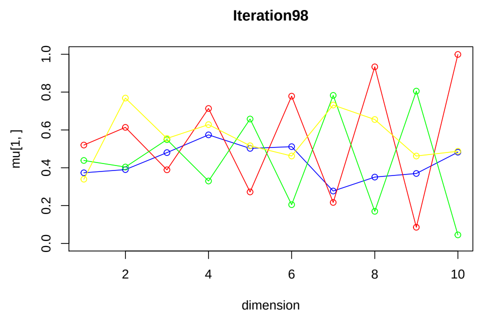
iteration: 95 log likelihood: -7196.188



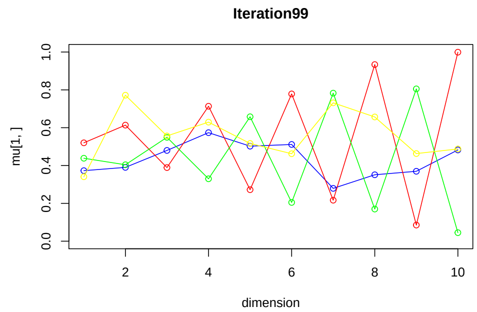
iteration: 96 log likelihood: -7195.392



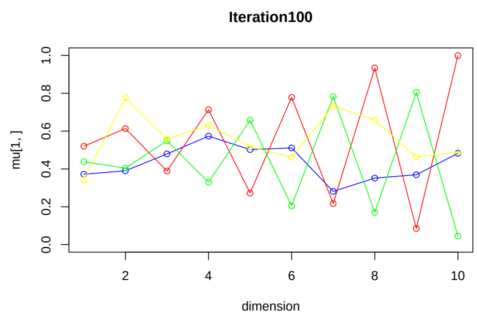
iteration: 97 log likelihood: -7194.57



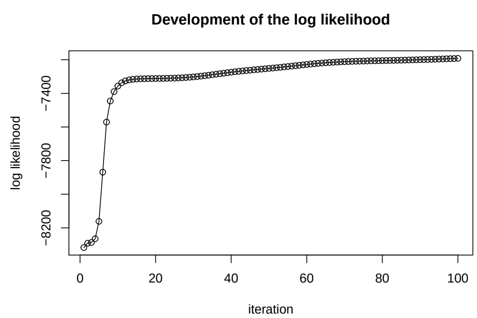
iteration: 98 log likelihood: -7193.722




```
## 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
```

Analysis

Comparing the final plots for each of the cases, it becomes clear that when the mixture model has more components (K = 4), the EM algorithm does not perform as accurate as for fewer components (K = 2 or K = 3). The segregation between each component gets diluted as the components get higher.

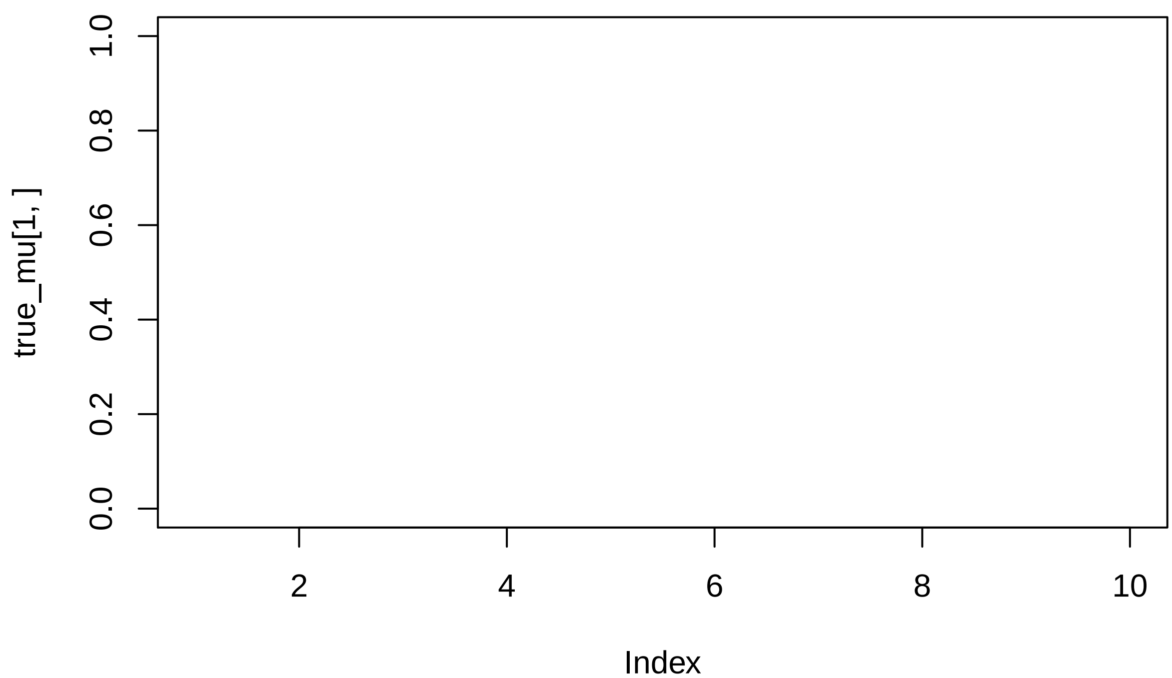
EM algo with matrix

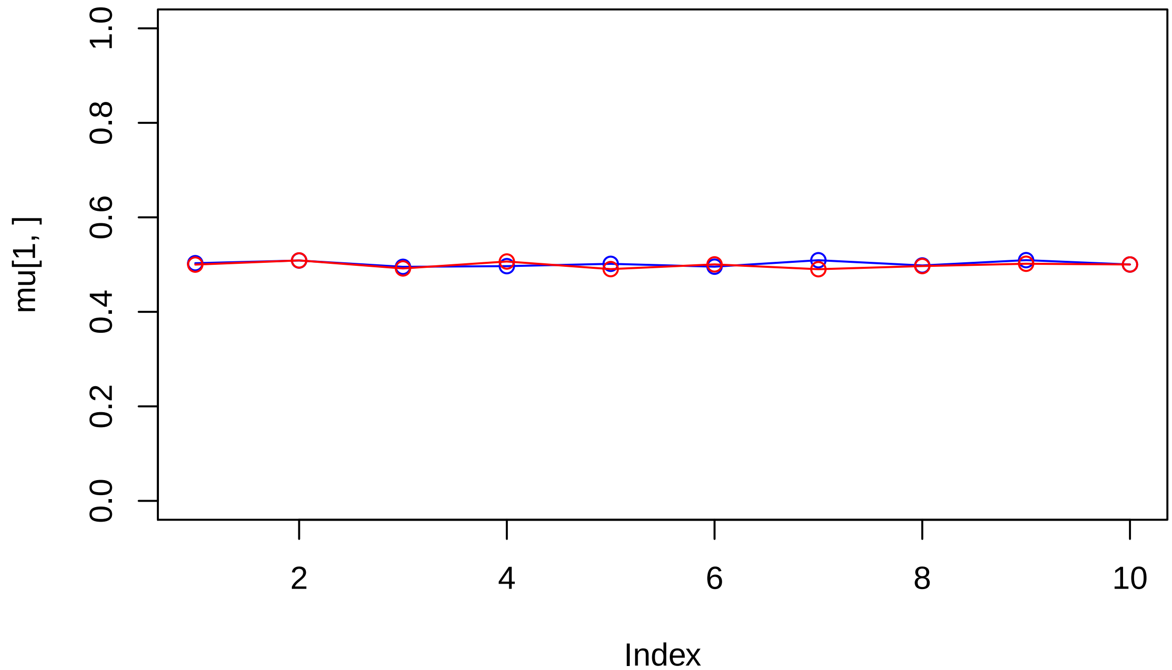
```r
em_mat <- function(k){
set.seed(1234567890)
# max number of EM iterations
max_it <- 100
# min change in log likelihood between two consecutive EM iterations
min_change <- 0.1
#------------- Producing Training data and Initialization ----------------#
# number of training points
N <- 1000
# number of dimensions
D <- 10
# training data
x <- matrix(nrow=N, ncol=D)
# true mixing coefficients
true_pi <- vector(length = k)
true_pi <- rep(1/k, k)
# true conditional distributions
true_mu <- matrix(nrow = k, ncol = D)
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) {
l <- sample(1:k,1,prob=true_pi)
```

```
for(d in 1:D) {
x[n,d] <- rbinom(1,1,true_mu[l,d])
}
}
# fractional component assignments
z <- matrix(nrow = N, ncol = k)
# mixing coefficients
pi <- vector(length = k)
# conditional distributions
mu <- matrix(nrow = k, ncol = D)
# log likelihood of the EM iterations
llik <- vector(length = max_it)
# Random initialization of the paramters
pi <- runif(k,0.49,0.51)
pi <- pi / sum(pi)
for(i in 1:k) {
mu[i,] <- runif(D,0.49,0.51)
}
#----------------------- Iteration stage -----------------------#
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
# Updating z matrix
p_Xn_MUn <- exp(x %*% log(t(mu)) + (1 - x) %*% log(1 - t(mu)))
numerator <- matrix(rep(pi,N), ncol = k, byrow = TRUE) * p_Xn_MUn
denominator <- rowSums(numerator)
Z_nk <- numerator/denominator
# Updating pi
pi <- colSums(Z_nk)/N
# Updating mu
mu <- (t(Z_nk) %*% x)/colSums(Z_nk)
#Log likelihood computation.
llik[it] <- sum(Z_nk * ((x %*% log(t(mu)) + (1 - x) %*% log(1 - t(mu))
) + matrix(rep(pi,N), ncol = k, byrow = TRUE)))
cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
flush.console()
# Stop if the log likelihood has not changed significantly
if(it >= 2){
if((llik[it] - llik[it-1]) < min_change){break()}
}
#M-step: ML parameter estimation from the data and fractional component assignments
```

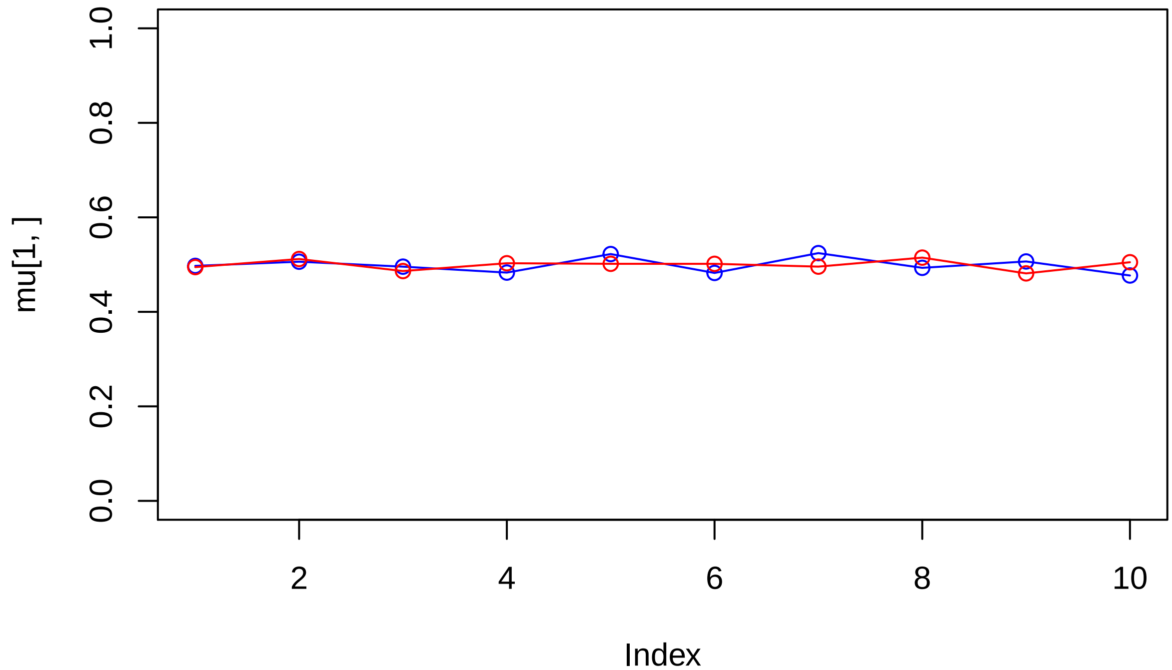
```r
# pi_ML
pi_ML <- pi
#mu_ML
mu_ML <- mu
}
#----------------------- output stage ----------------------#
df <- data.frame(Iteration = 1:length(llik[which(llik != 0.000)])
, log_likelihood = llik[which(llik != 0.000)])
plot <- ggplot(data = df) +
geom_point(mapping = aes(x = Iteration, y = log_likelihood),
color = 'black') +
geom_line(mapping = aes(x = Iteration, y = log_likelihood),
color = 'black', size = 1) +
ggtitle('Maximum likelihood vs Number of iterations') +
theme(plot.title = element_text(hjust = 0.5)) +
theme_light()
output <- list(pi_ML = pi_ML,
mu_ML = mu_ML,
plot = plot
)
output
}

EM_2 <- em_mat(2)
```

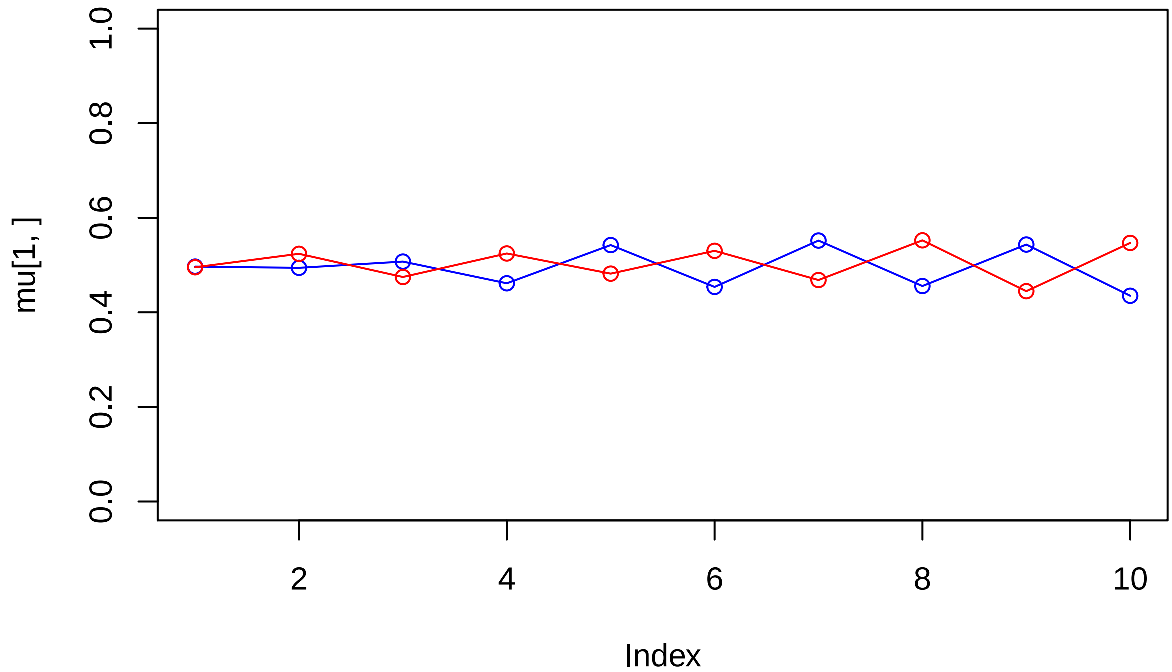




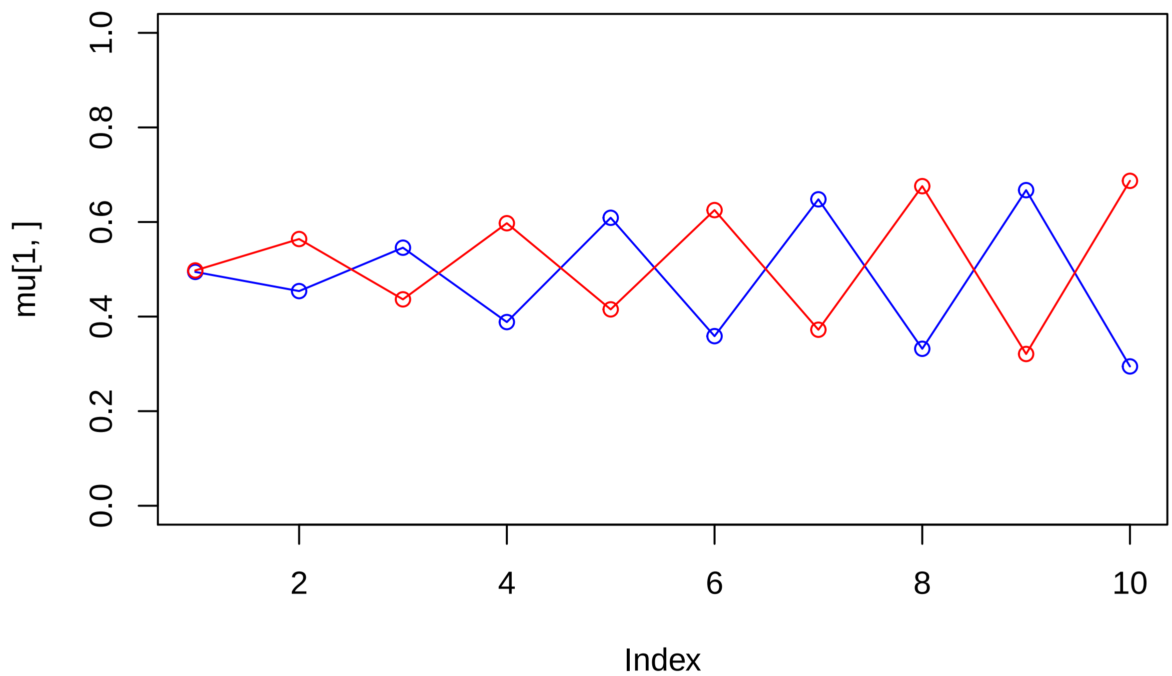

```
## iteration:  1 log likelihood:  -6428.122
```



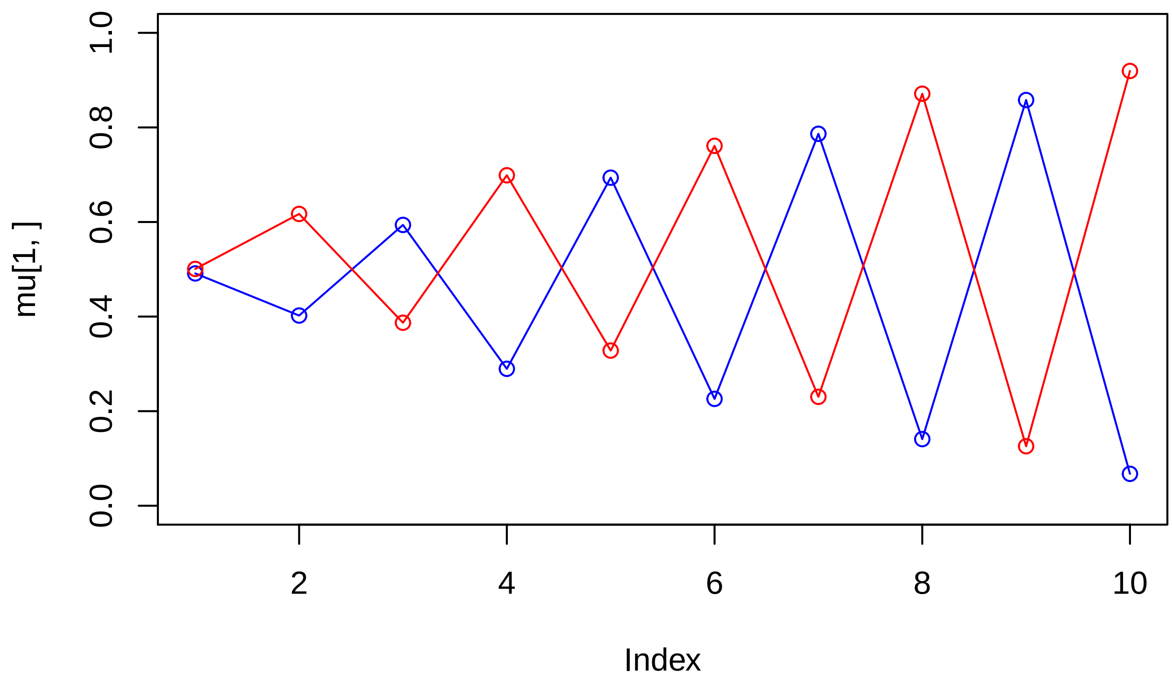

```
## iteration:  2 log likelihood:  -6403.127
```



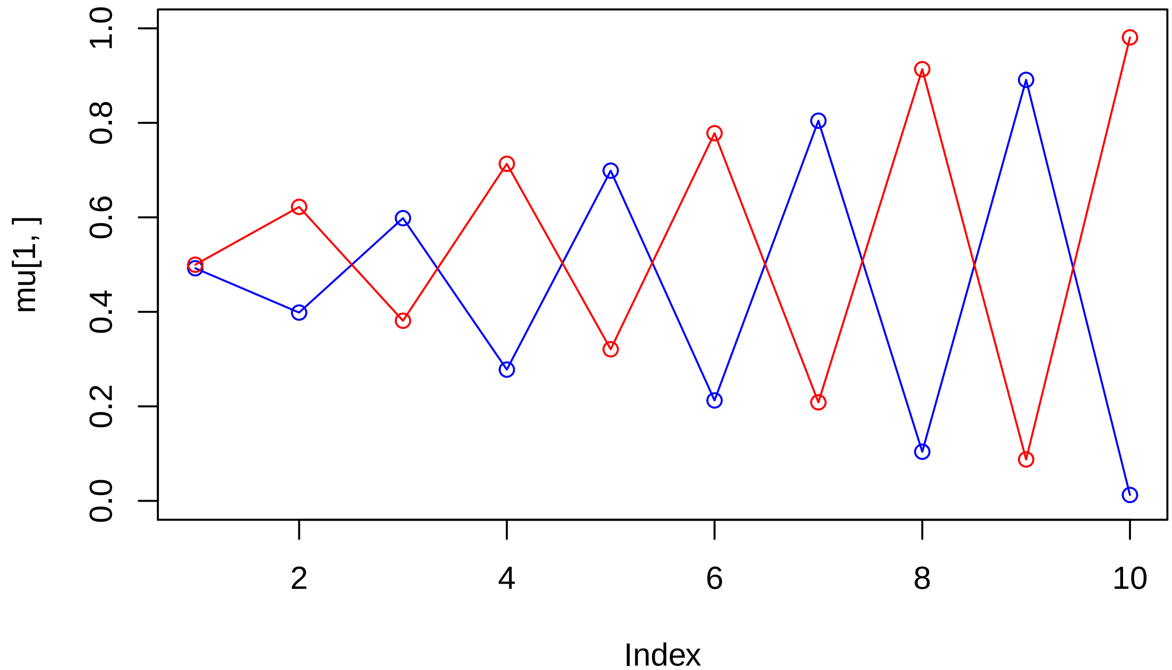

```
## iteration:  3 log likelihood:  -6101.854
```



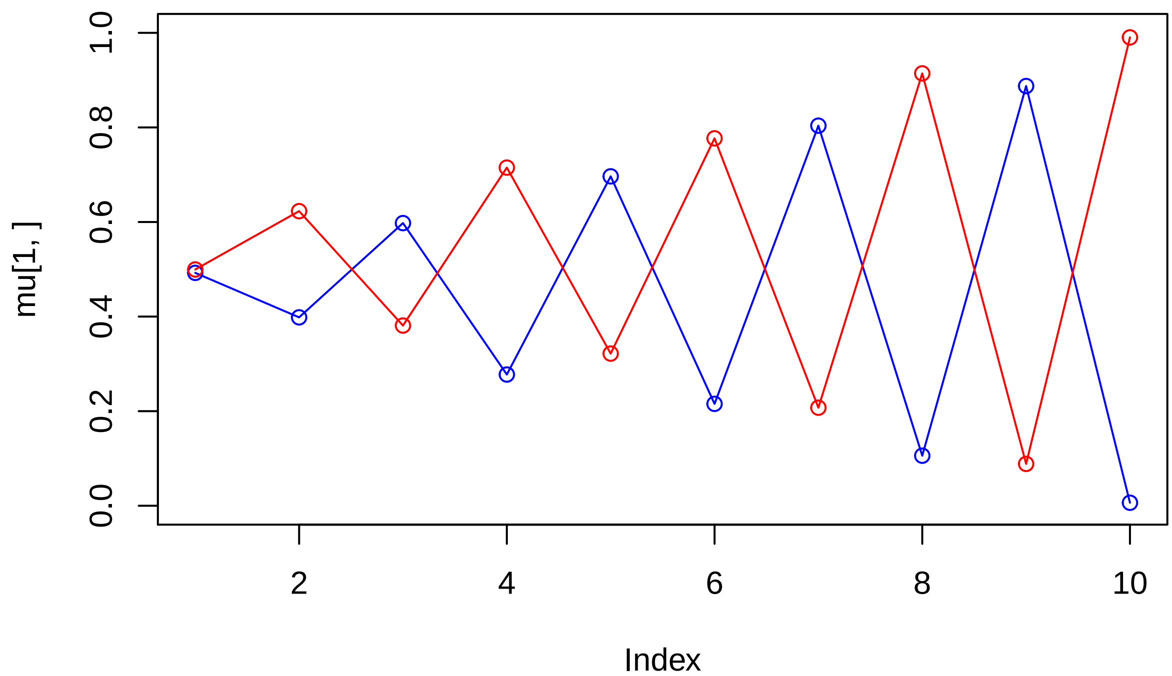

```
## iteration:  4 log likelihood:  -4889.04
```



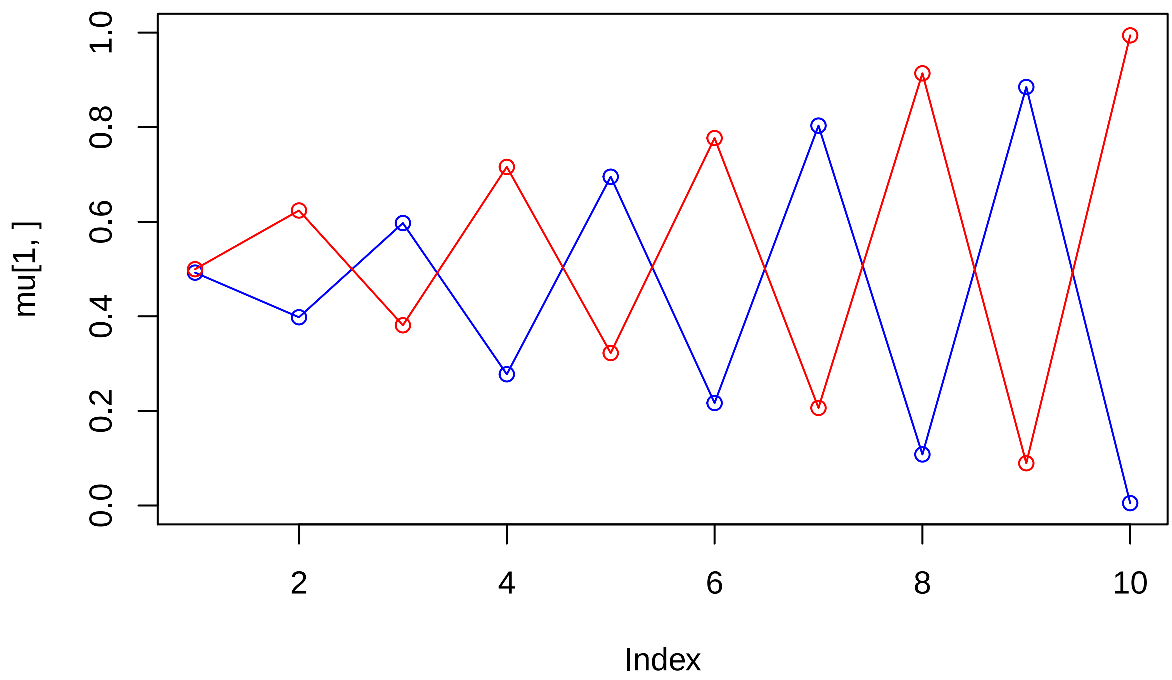

```
## iteration:  5 log likelihood:  -4486.667
```



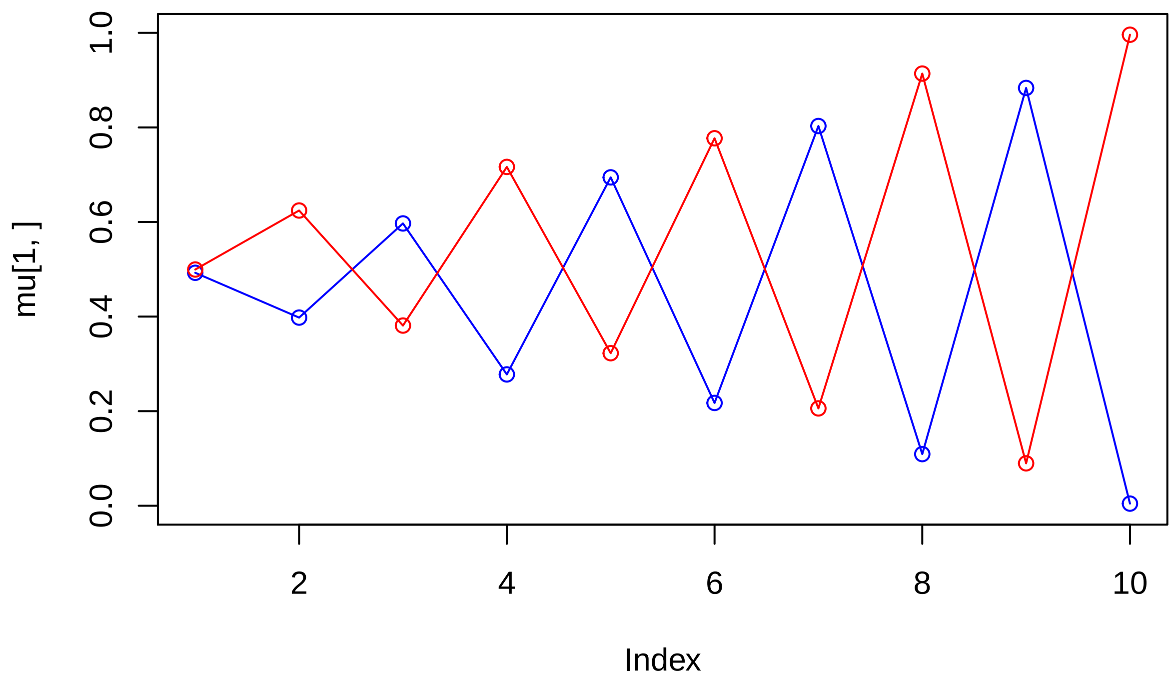

```
## iteration:  6 log likelihood:  -4460.54
```



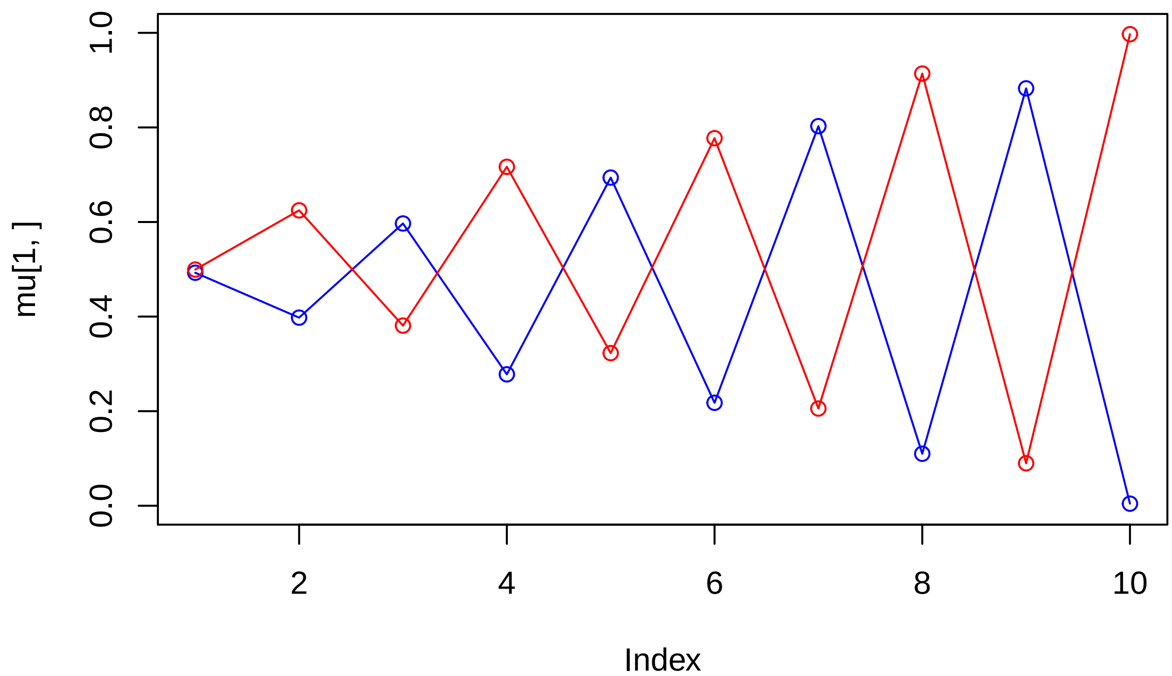

```
## iteration:  7 log likelihood:  -4455.219
```



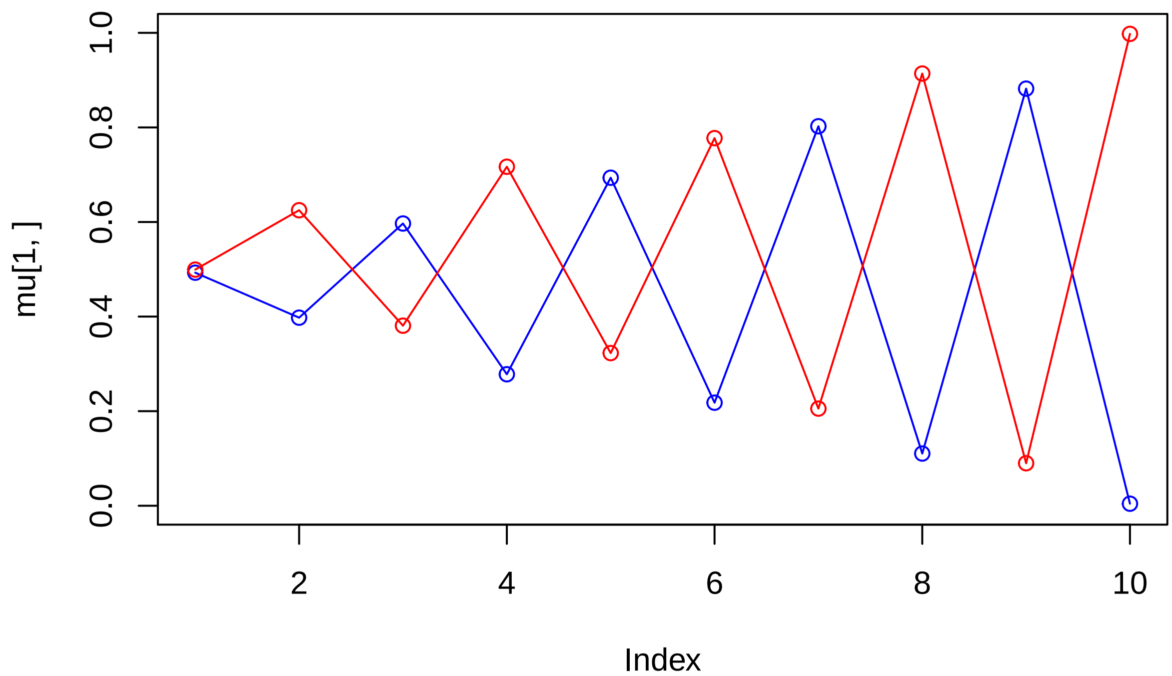

```
## iteration:  8 log likelihood:  -4452.797
```



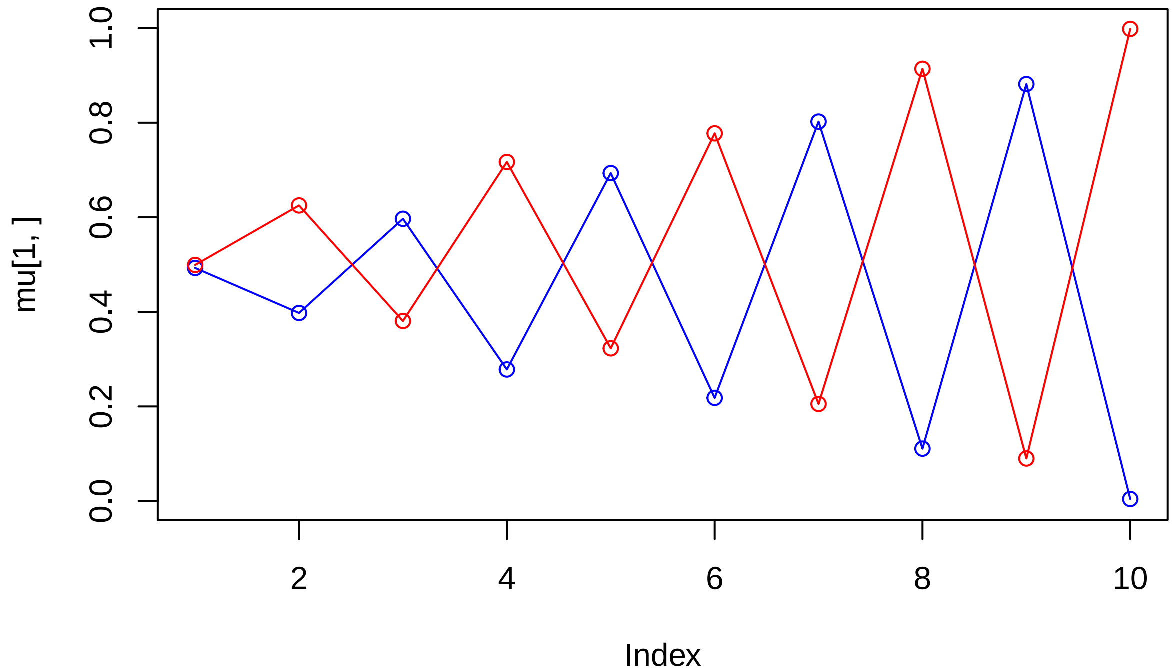

```
## iteration:  9 log likelihood:  -4451.369
```



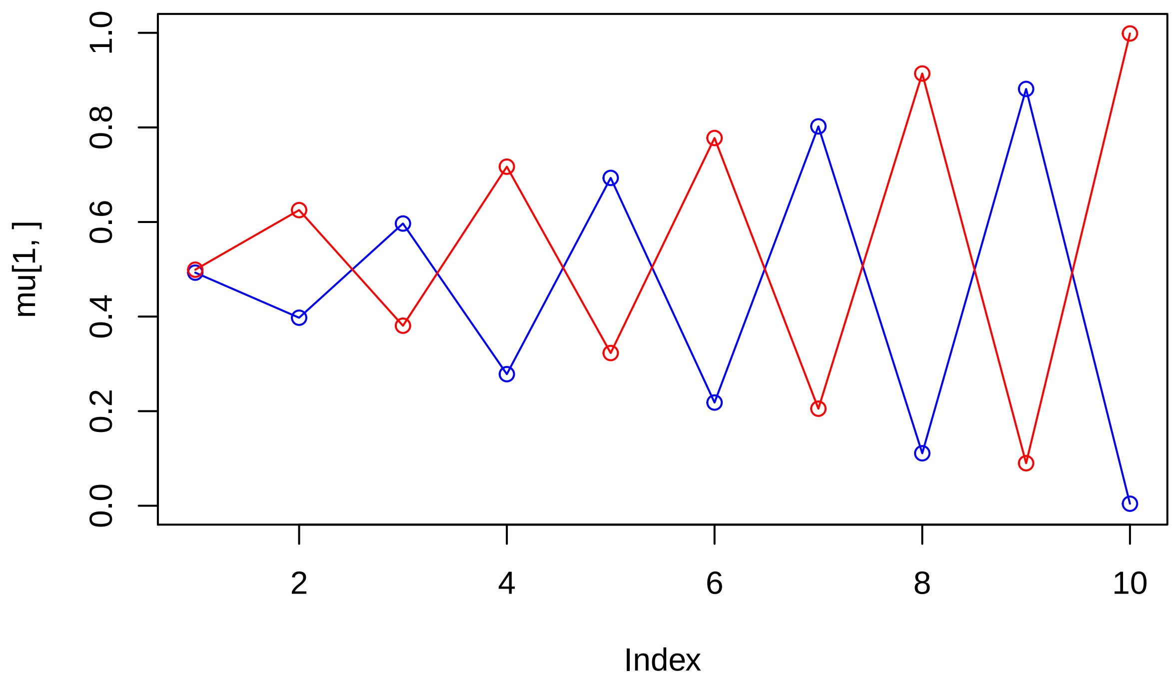

```
## iteration:  10 log likelihood:  -4450.418
```



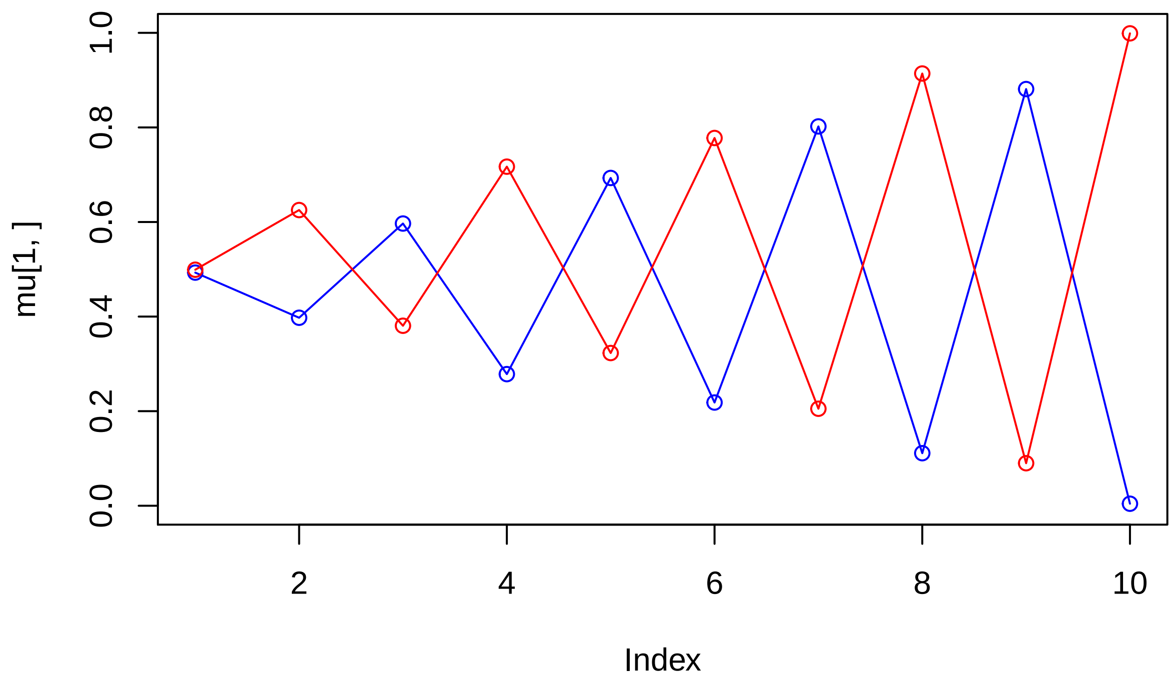

```
## iteration:  11 log likelihood:  -4449.735
```



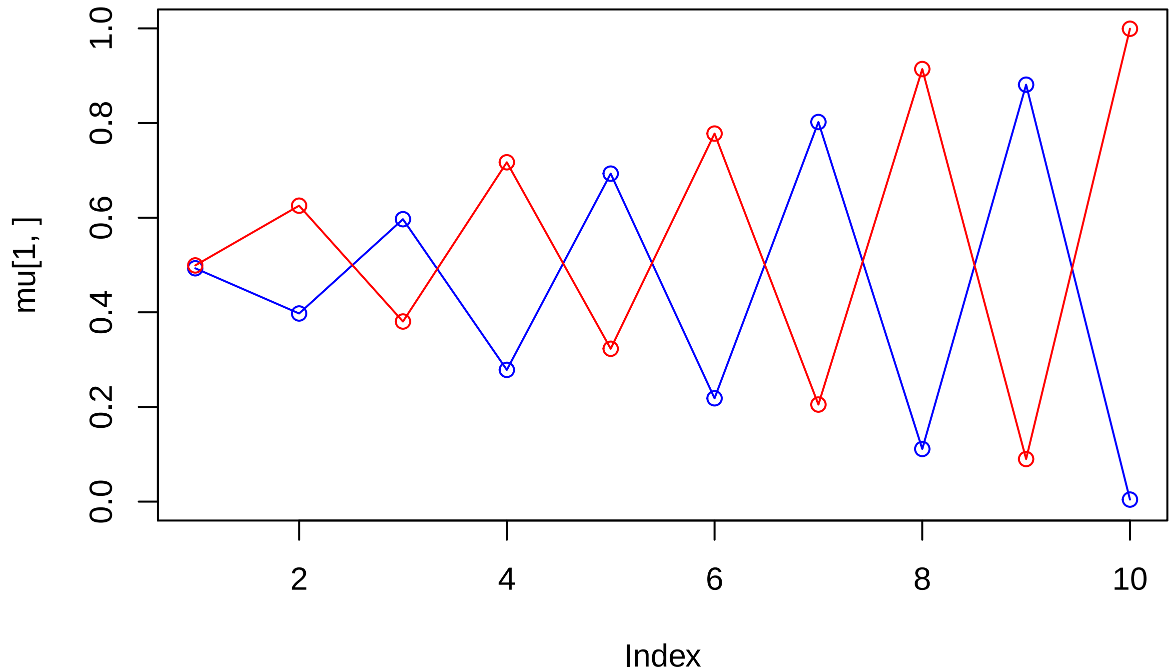

```
## iteration:  12 log likelihood:  -4449.219
```



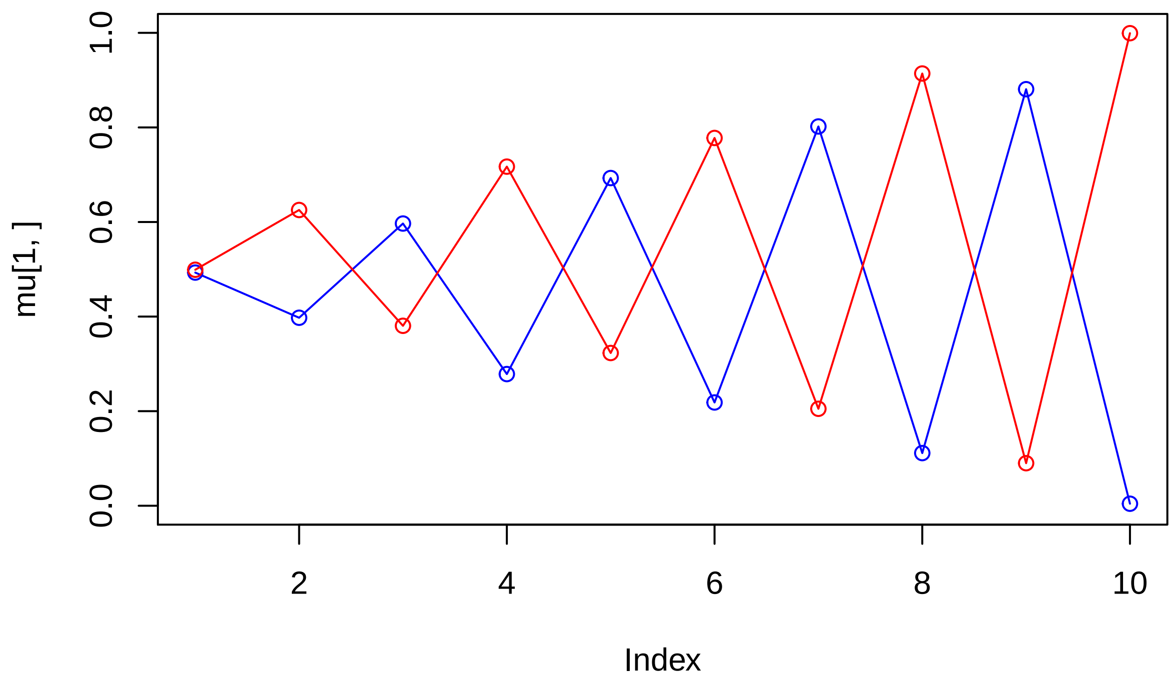

```
## iteration:  13 log likelihood:  -4448.816
```



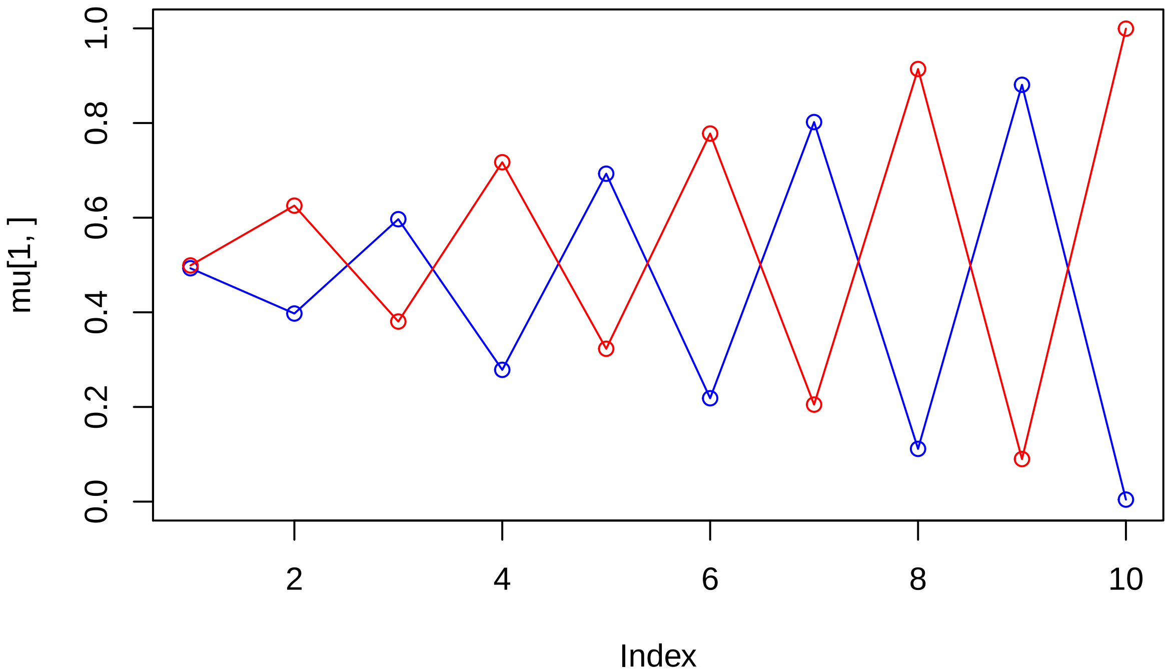

```
## iteration:  14 log likelihood:  -4448.492
```



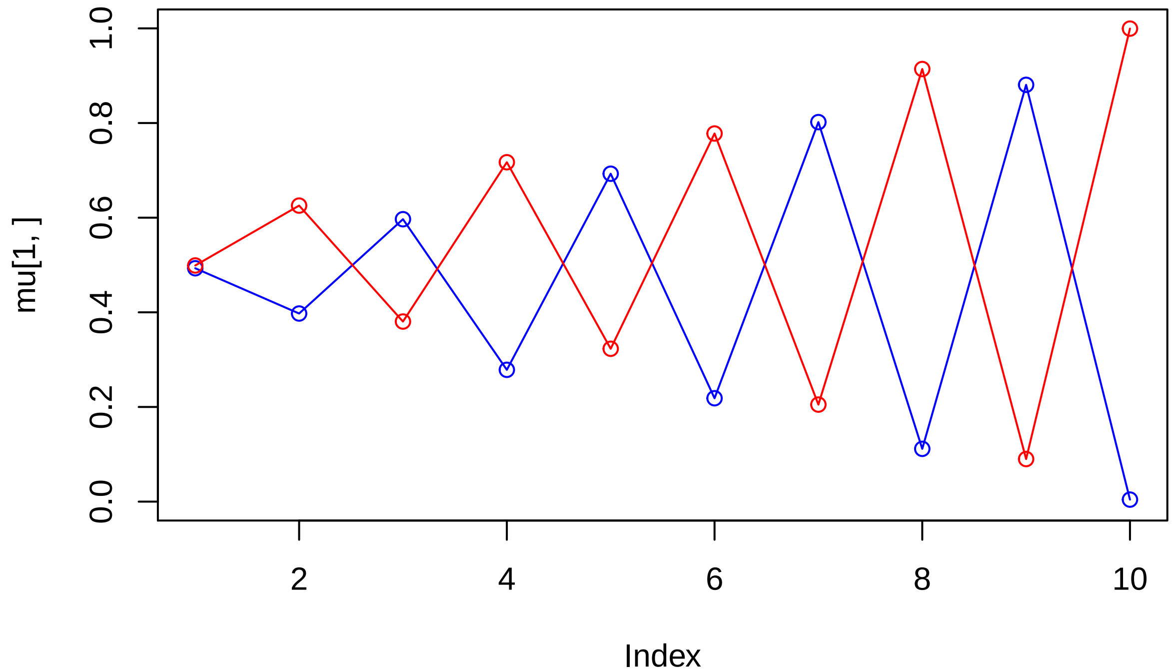

```
## iteration:  15 log likelihood:  -4448.227
```



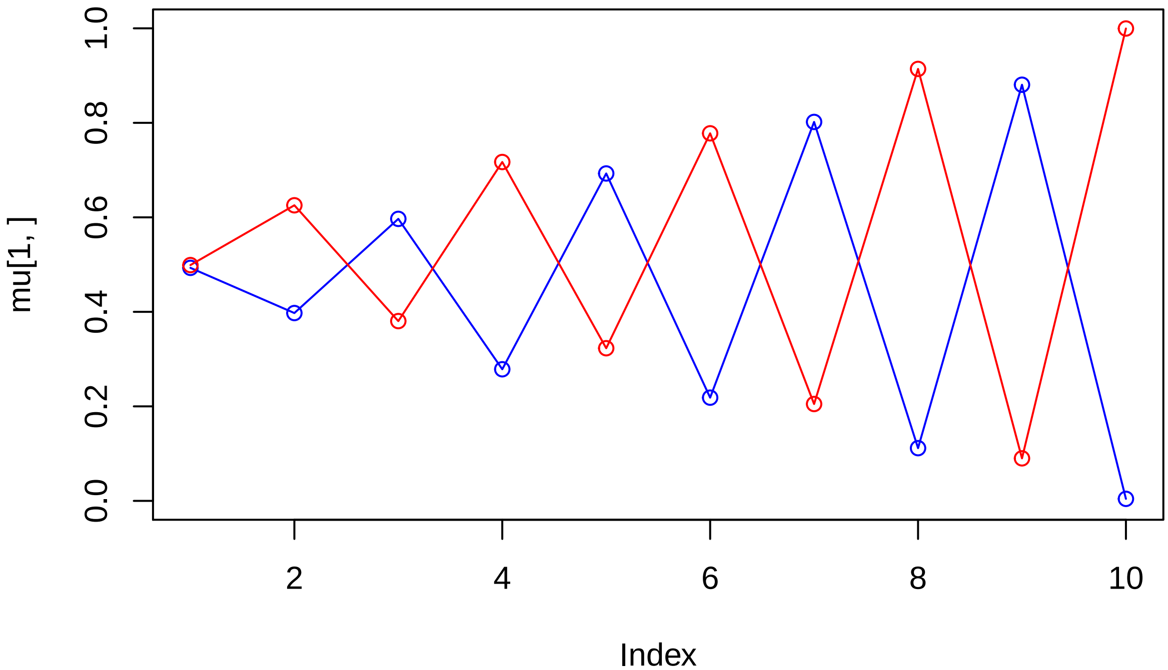

```
## iteration:  16 log likelihood:  -4448.008
```



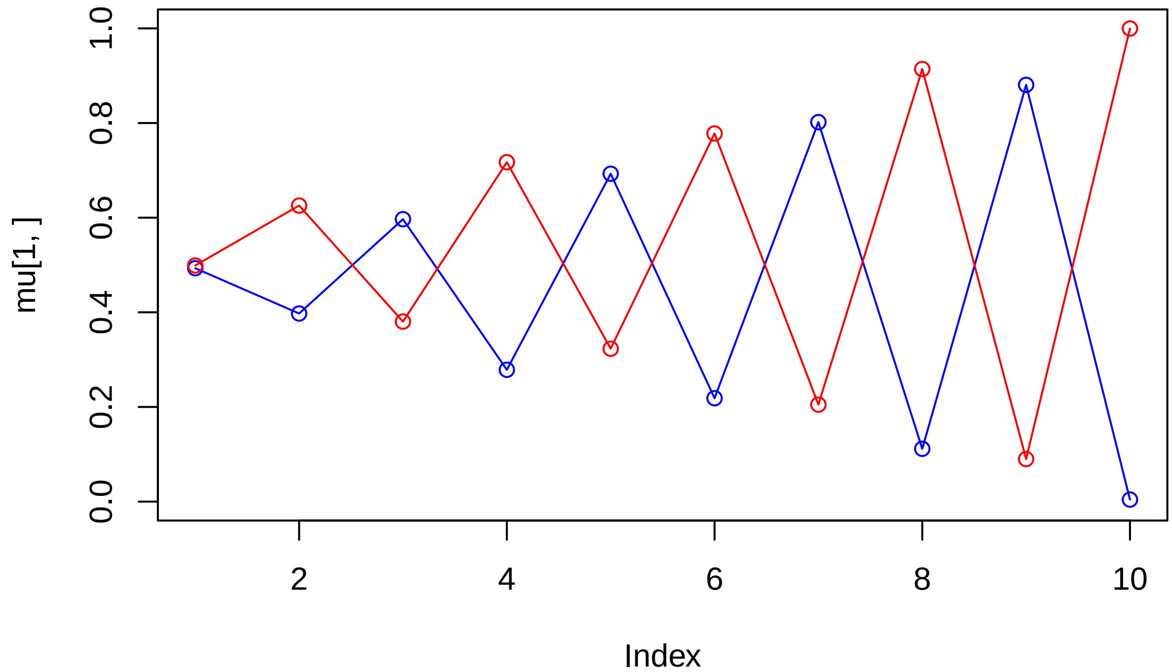

```
## iteration:  17 log likelihood:  -4447.824
```



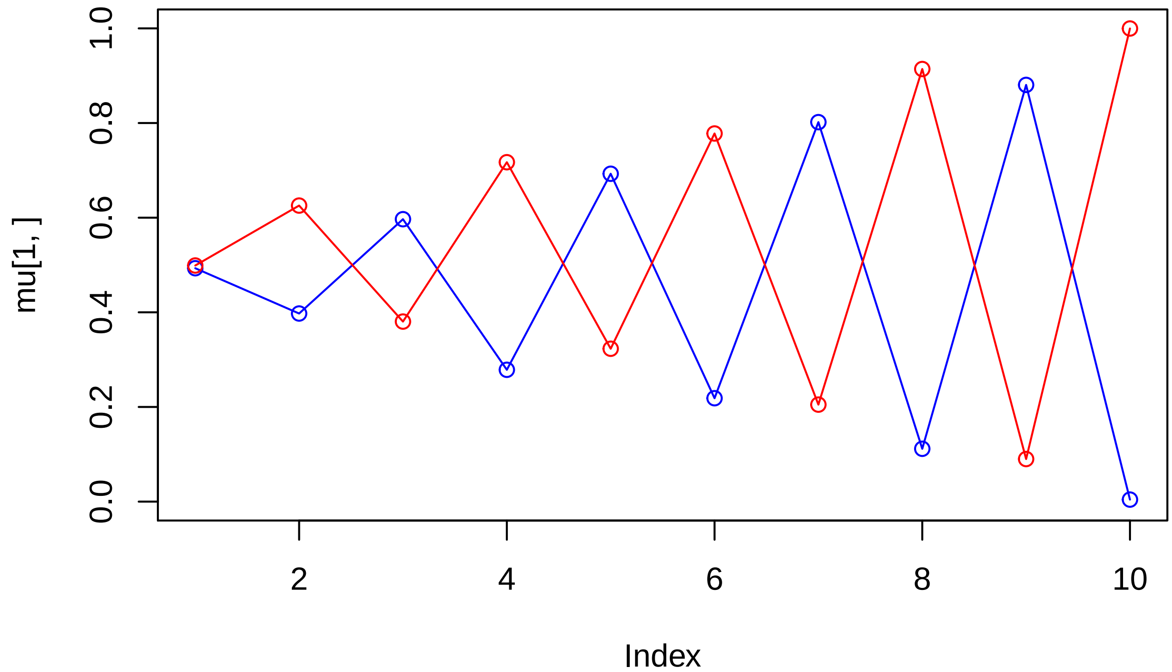

```
## iteration:  18 log likelihood:  -4447.668
```




```
## iteration:  19 log likelihood:  -4447.536
```

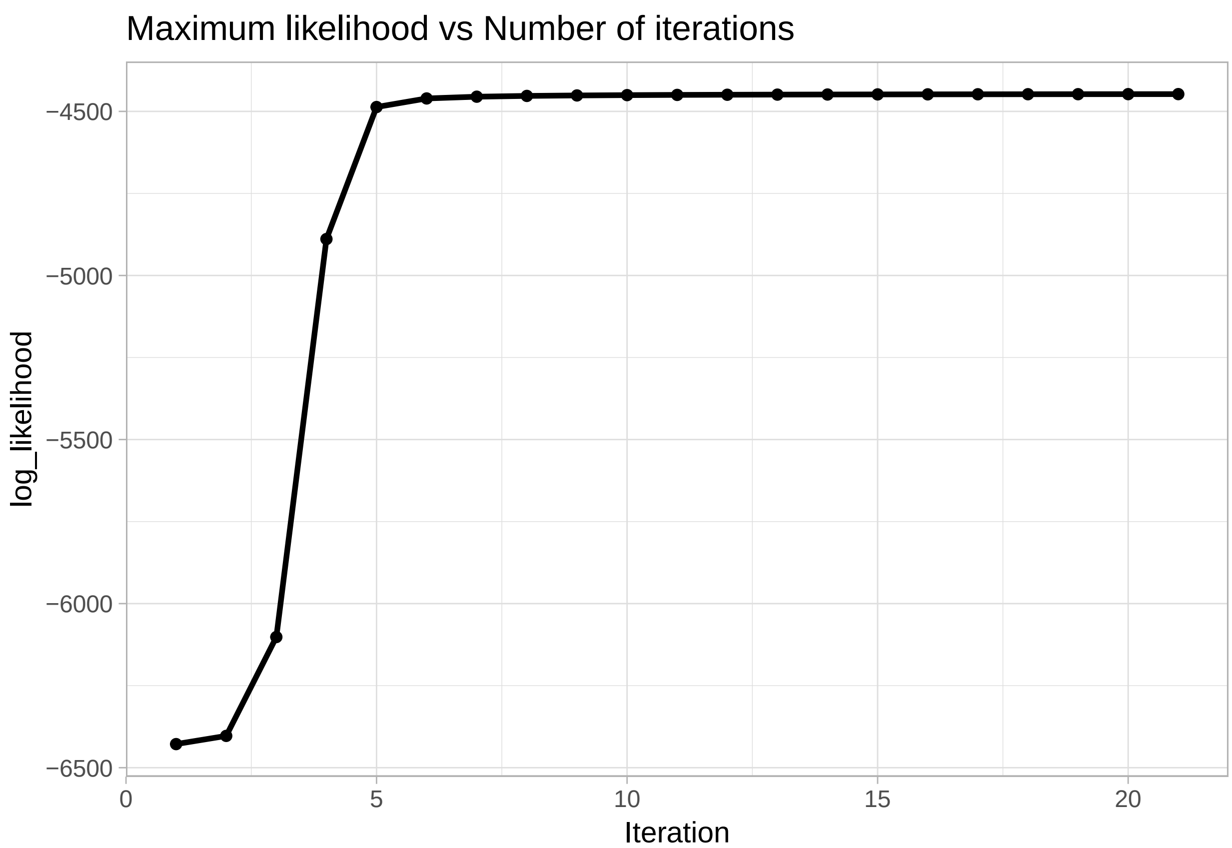



```
## iteration:  20 log likelihood:  -4447.422
```




```
## iteration:  21 log likelihood:  -4447.324
```

```
EM_2$plot
```



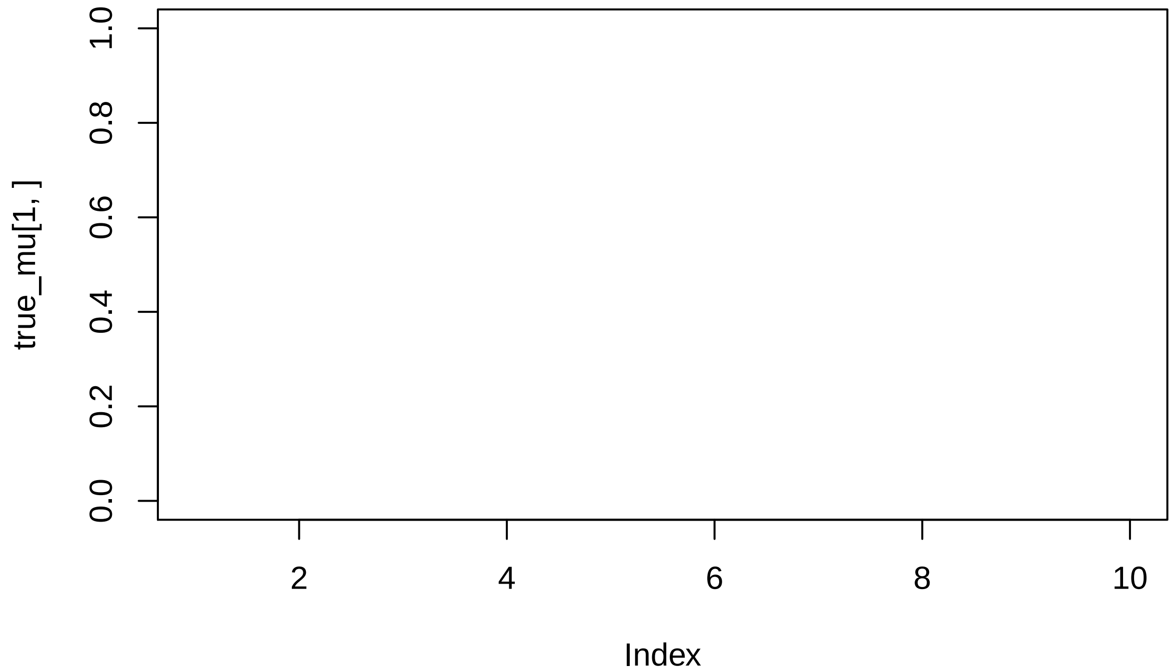

```
EM_2$pi_ML
```

```
## [1] 0.5110531 0.4889469
```

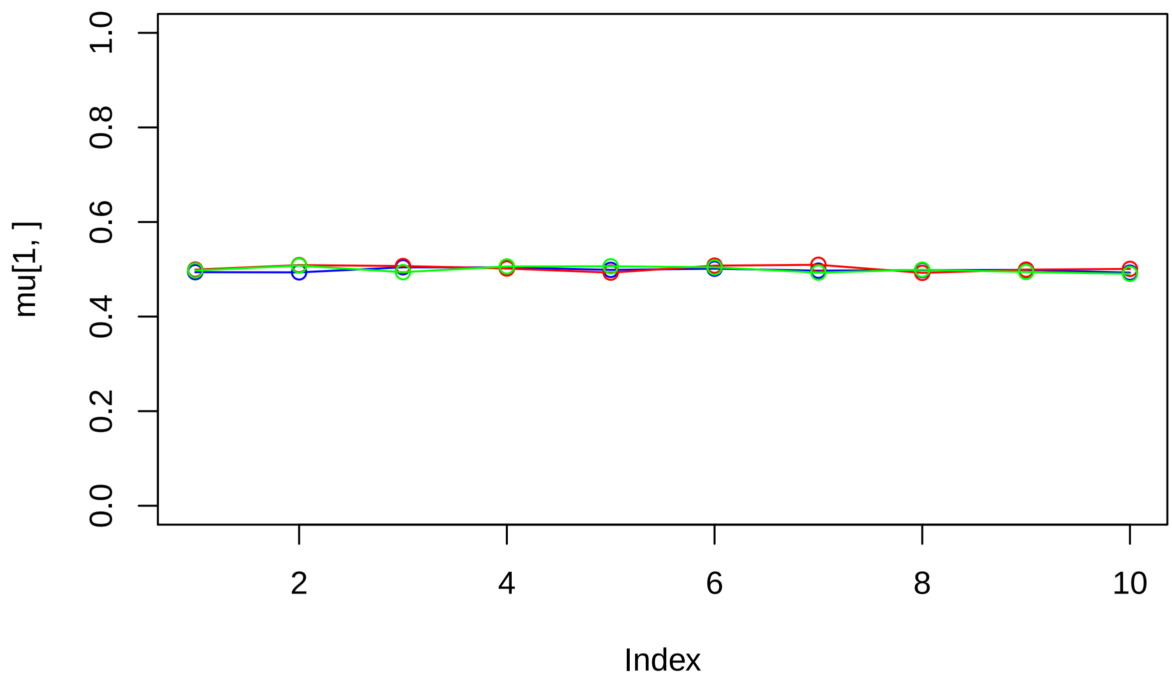
```
EM_2$mu_ML
```

```
##           [,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
```

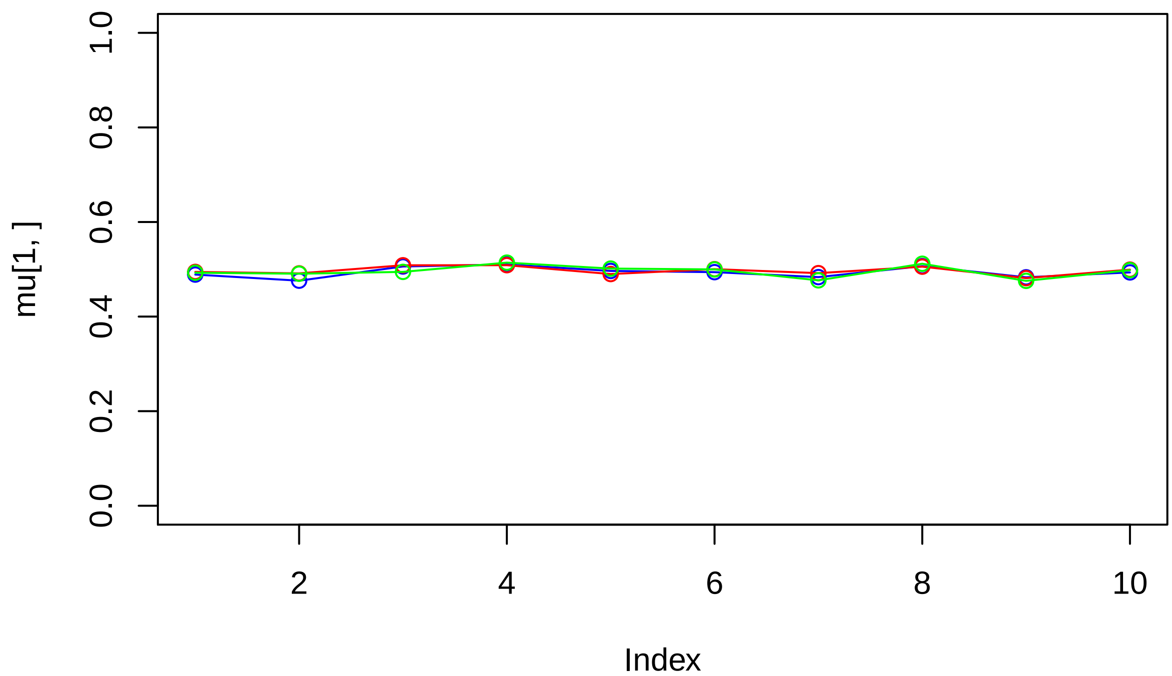
```
EM_3 <- em_mat(3)
```



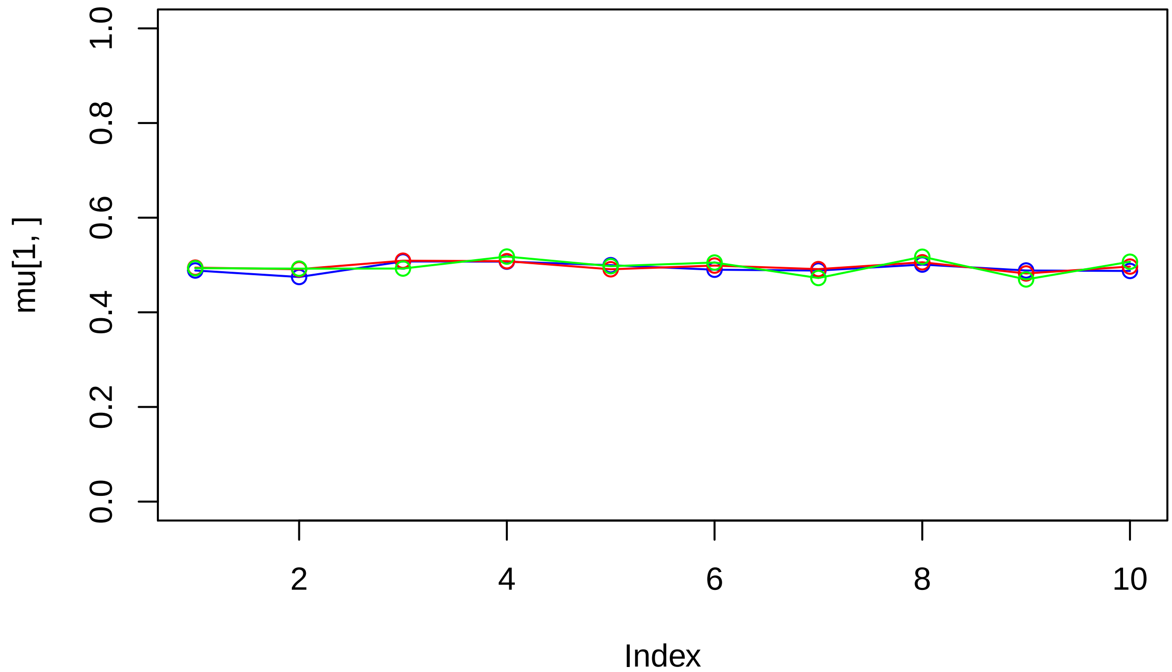
true_mu[1,]
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Index



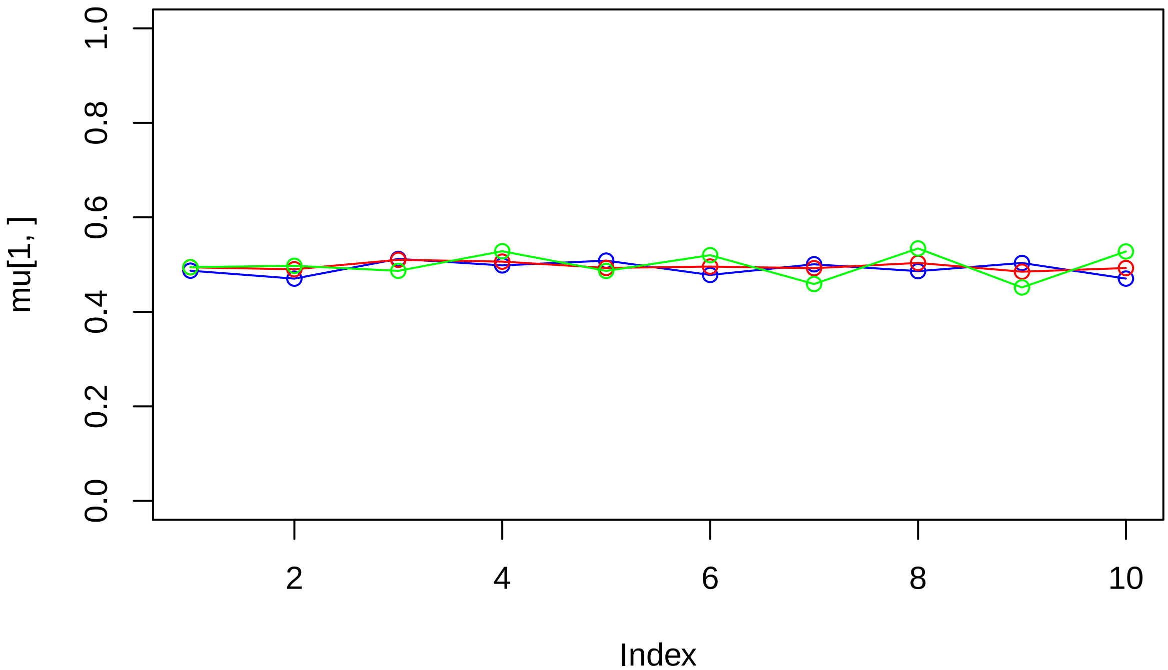

```
## iteration:  1 log likelihood:  -6595.475
```



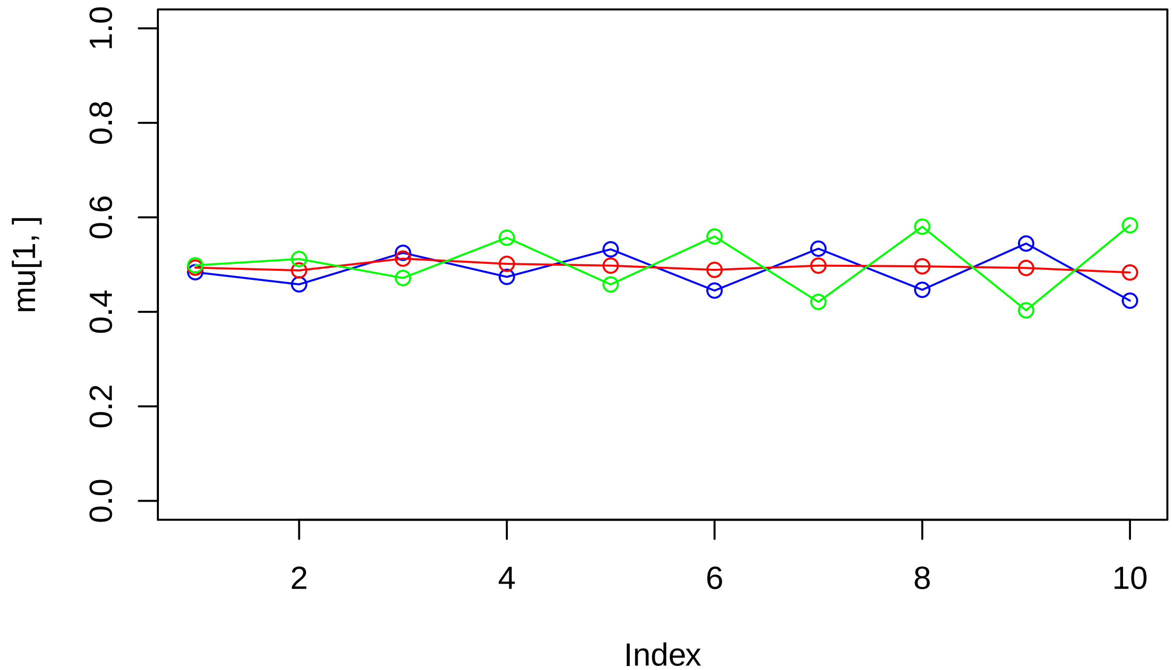

```
## iteration:  2 log likelihood:  -6594.971
```



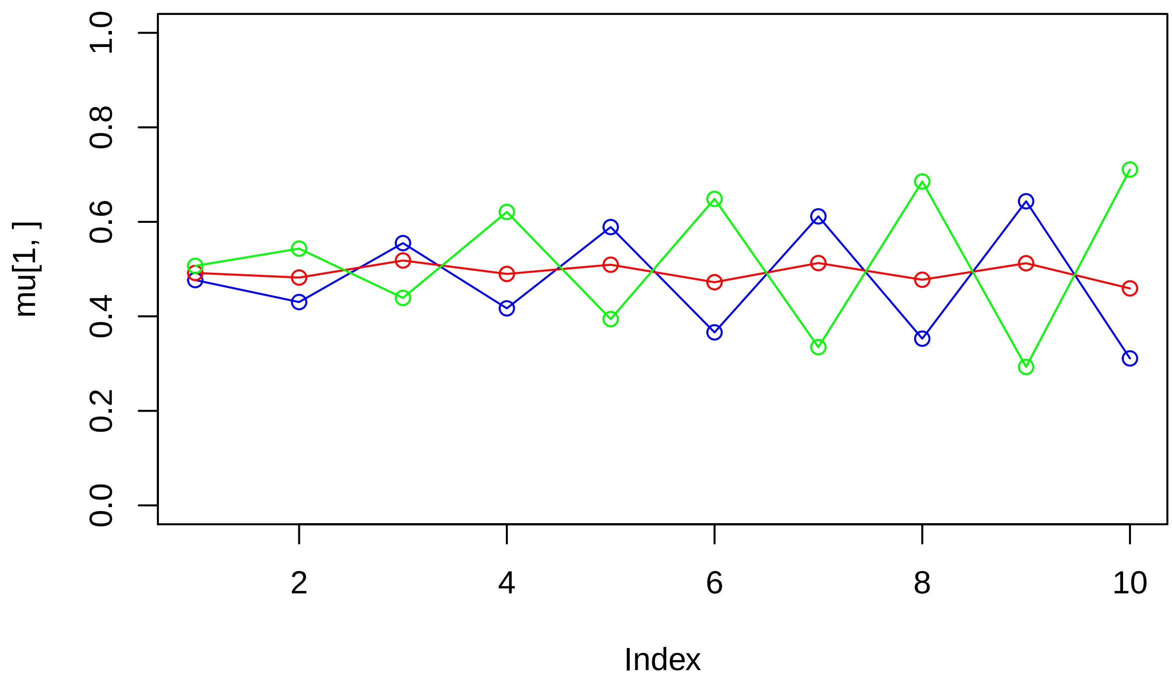

```
## iteration:  3 log likelihood:  -6590.744
```



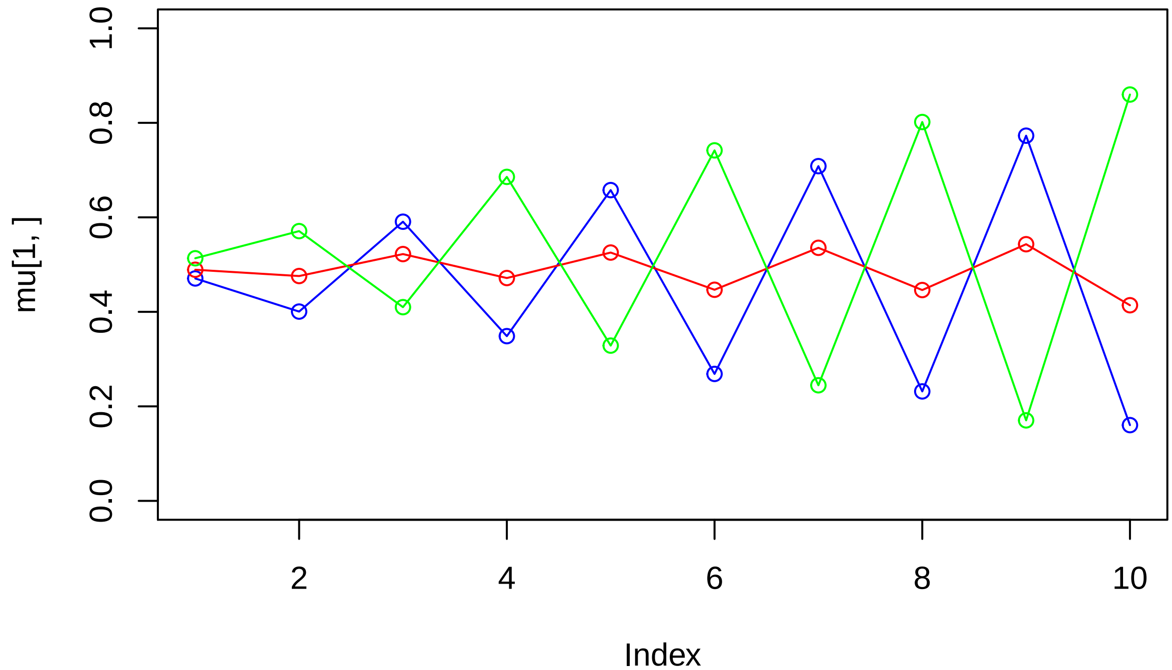

```
## iteration:  4 log likelihood:  -6558.775
```



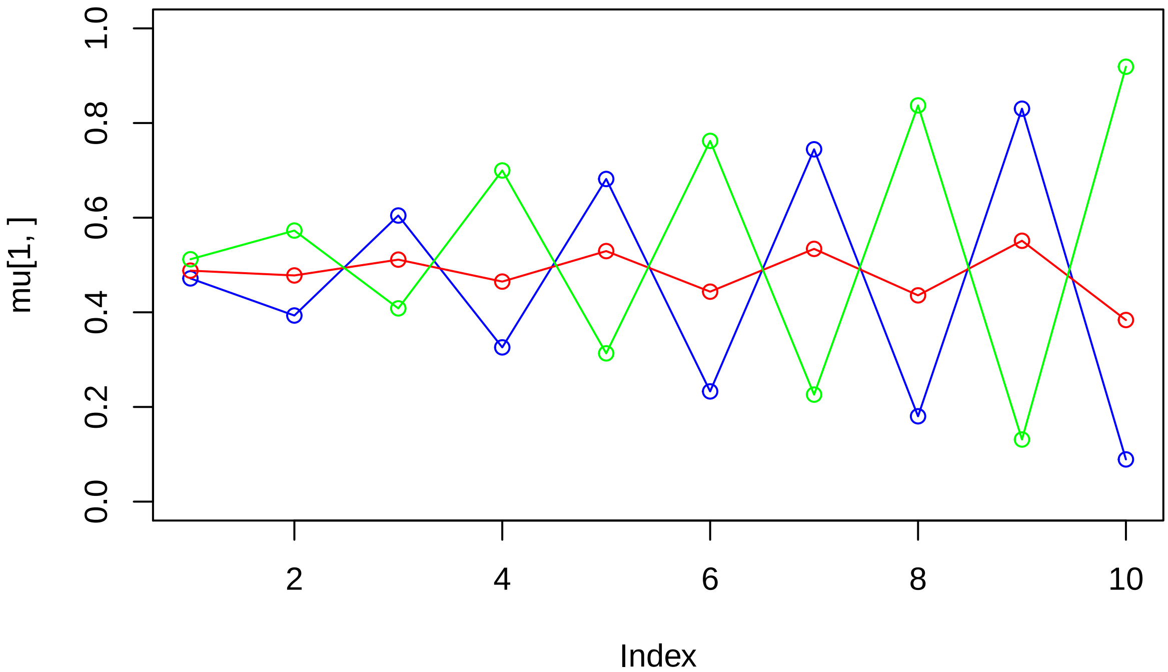

```
## iteration:  5 log likelihood:  -6359.678
```



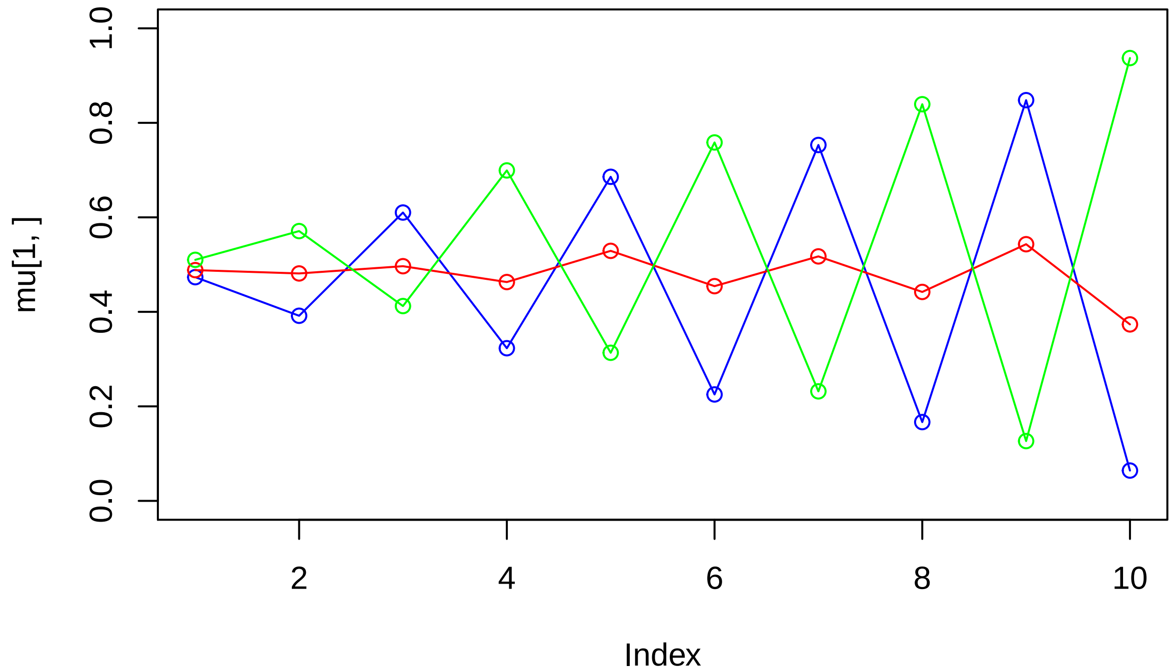

```
## iteration:  6 log likelihood:  -5818.144
```



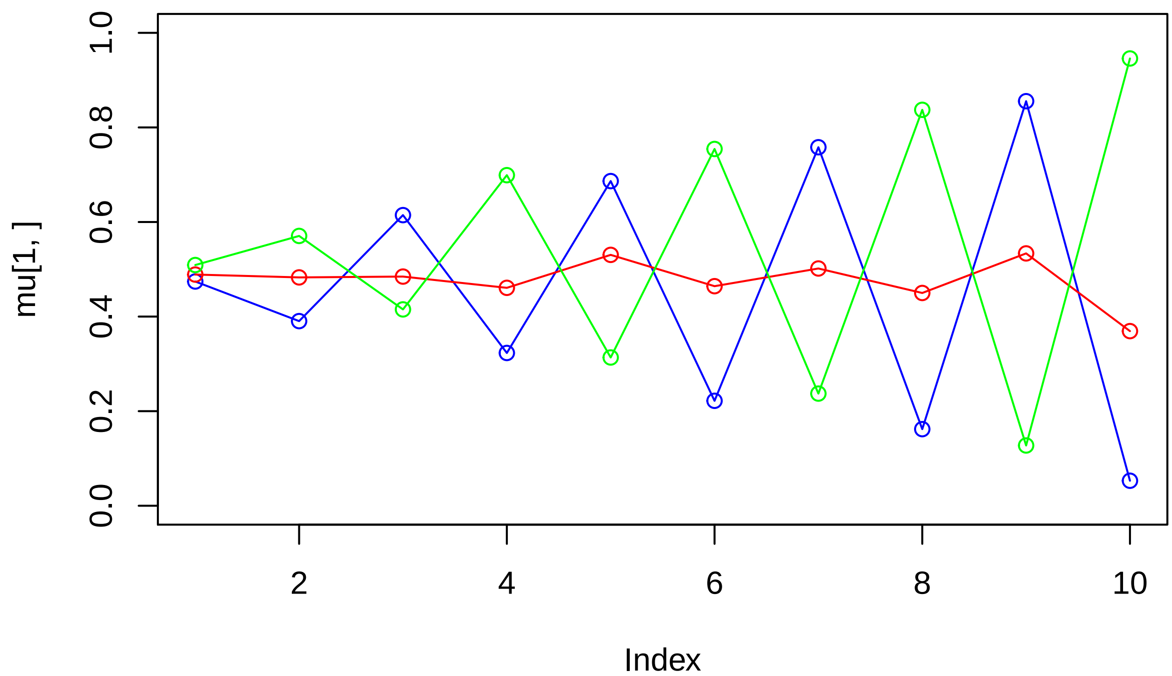

```
## iteration:  7 log likelihood:  -5482.541
```



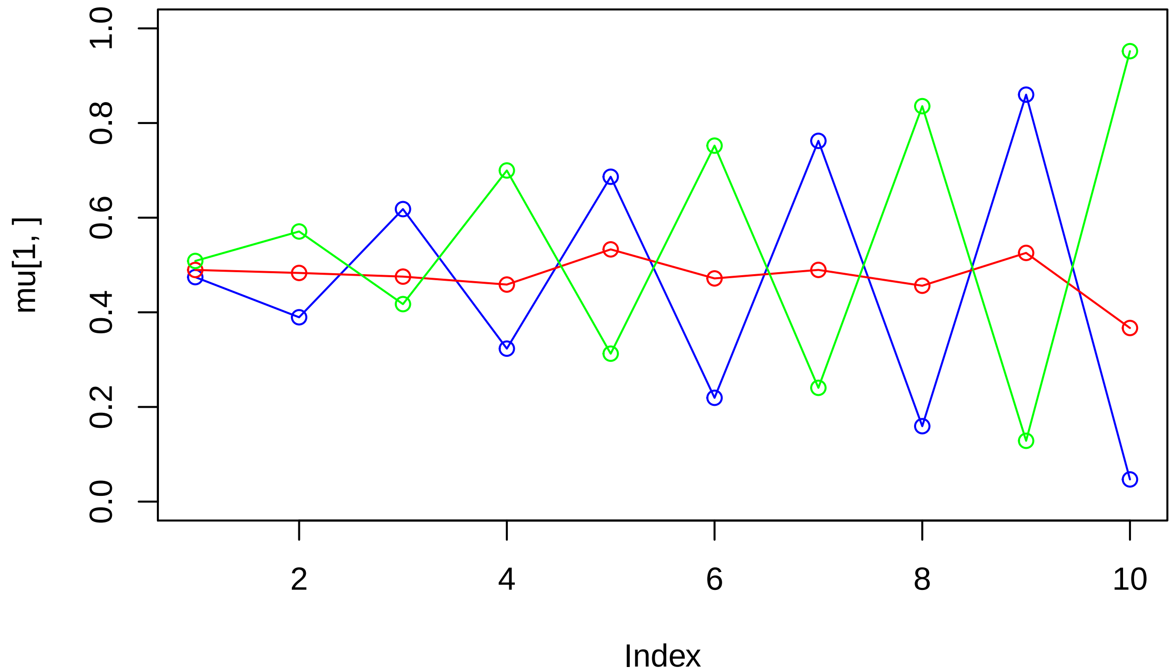

```
## iteration:  8 log likelihood:  -5397.642
```



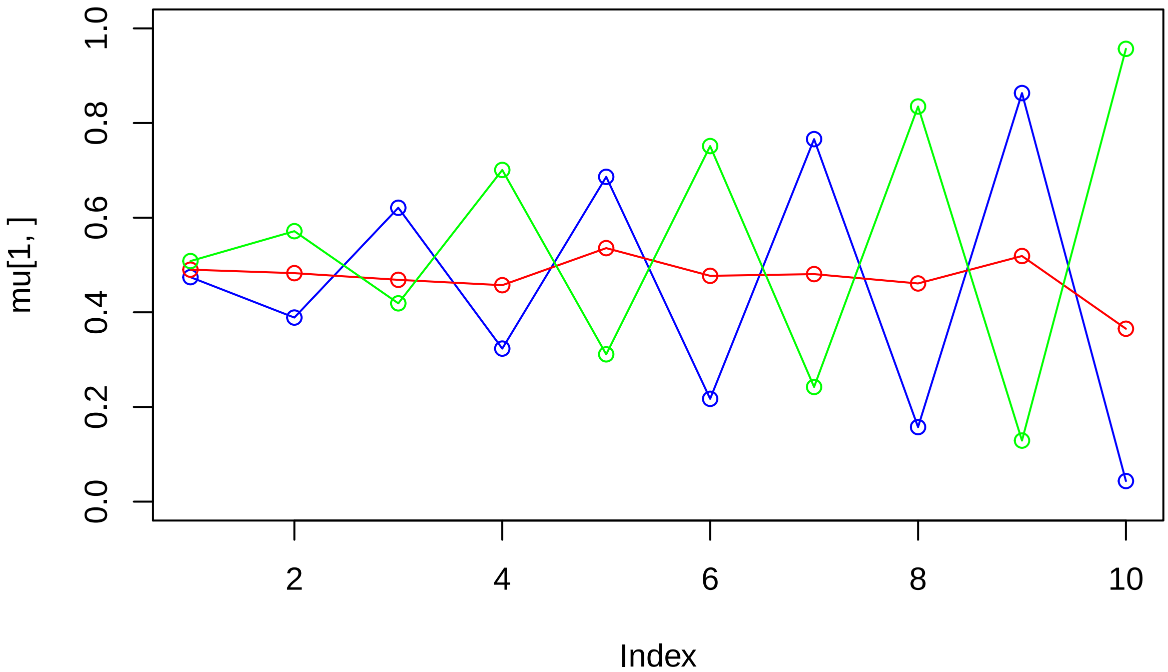

```
## iteration:  9 log likelihood:  -5373.097
```



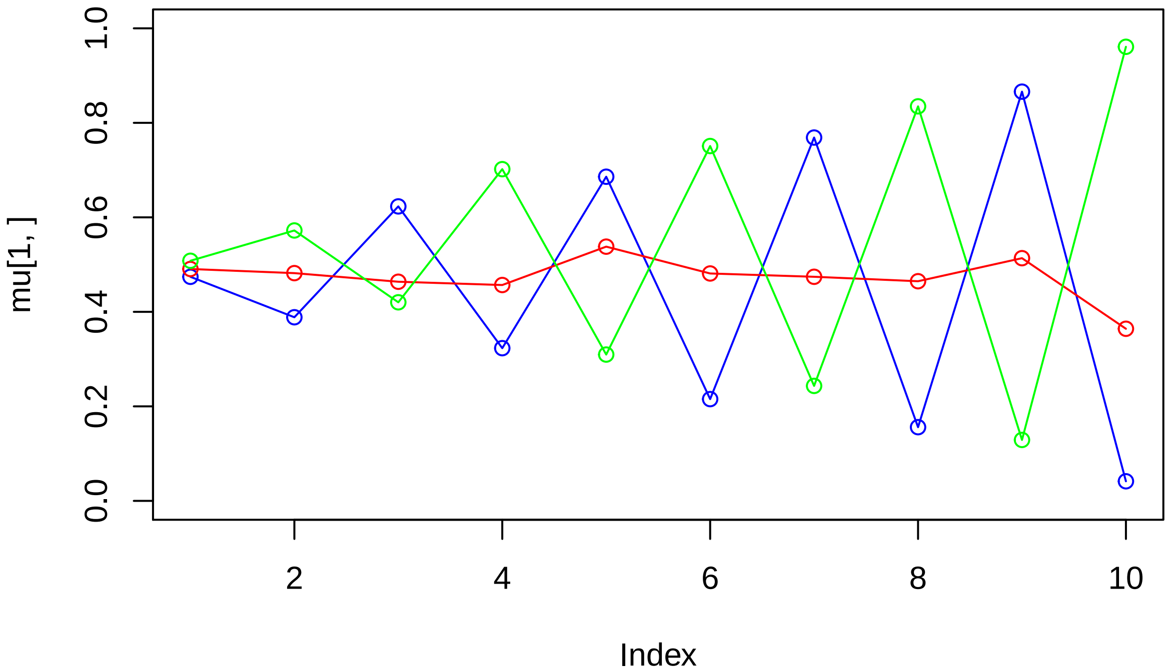

```
## iteration:  10 log likelihood:  -5362.128
```



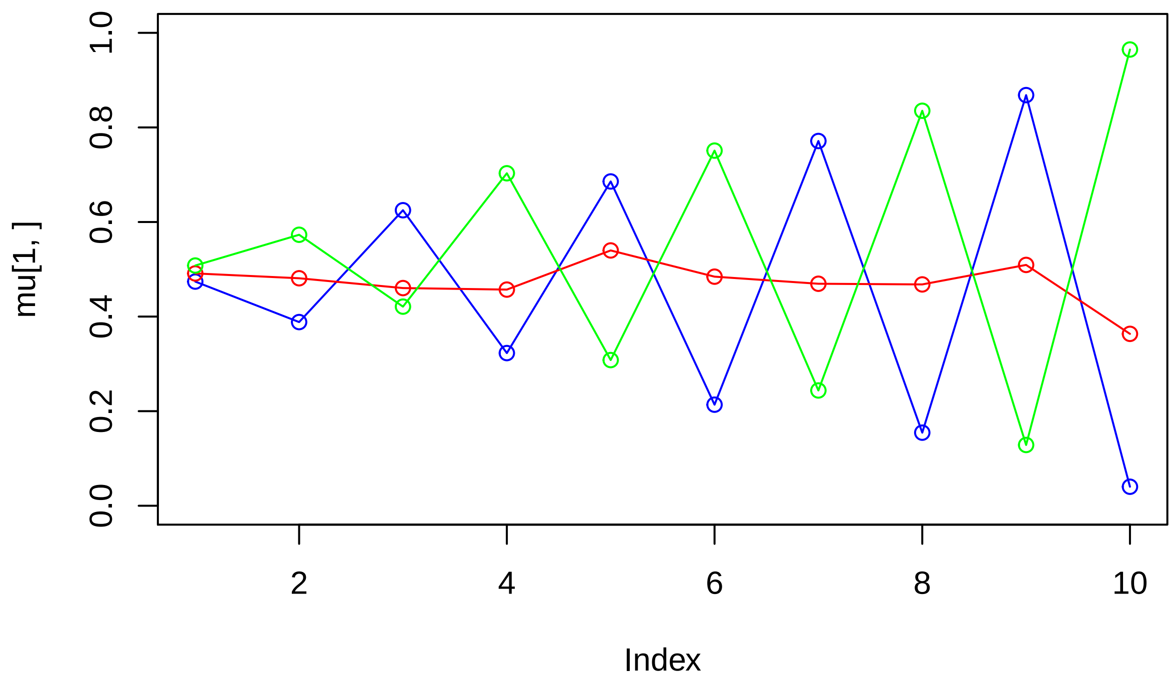

```
## iteration:  11 log likelihood:  -5355.483
```



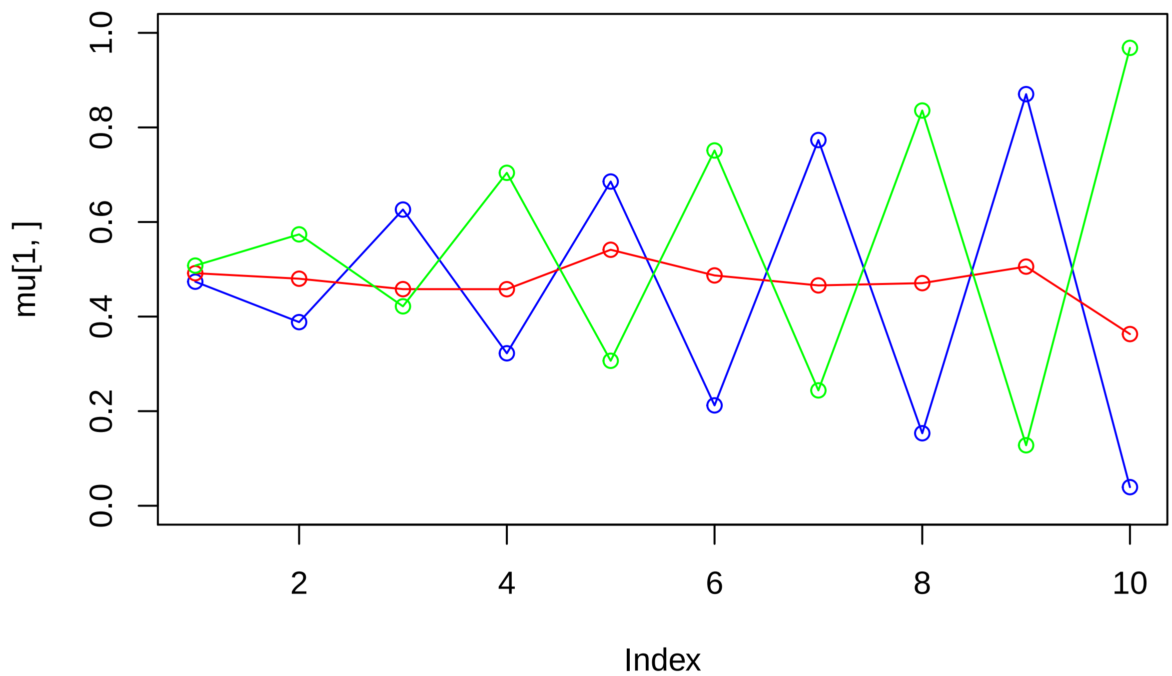

```
## iteration:  12 log likelihood:  -5350.765
```



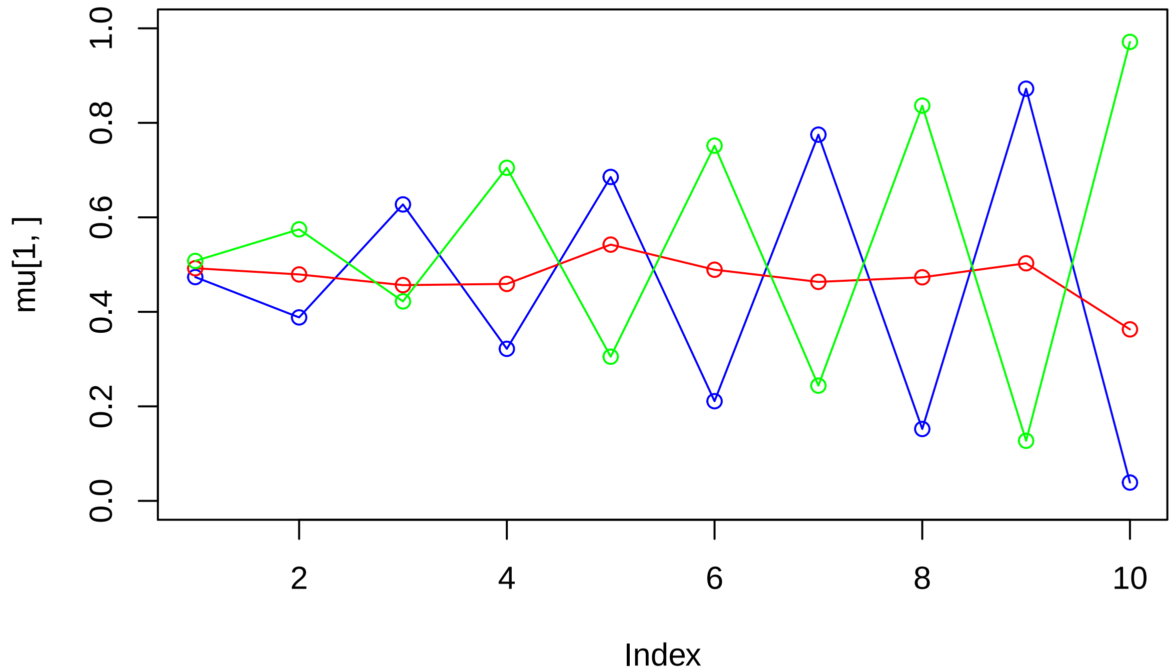

```
## iteration:  13 log likelihood:  -5347.103
```



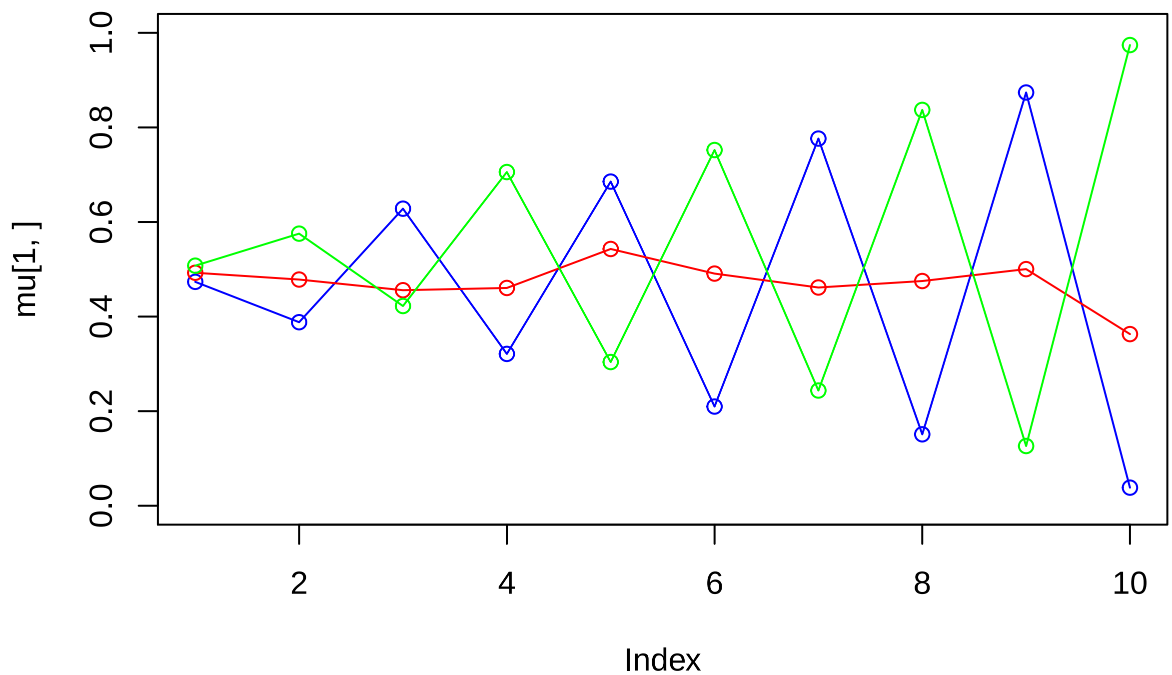

```
## iteration:  14 log likelihood:  -5344.094
```



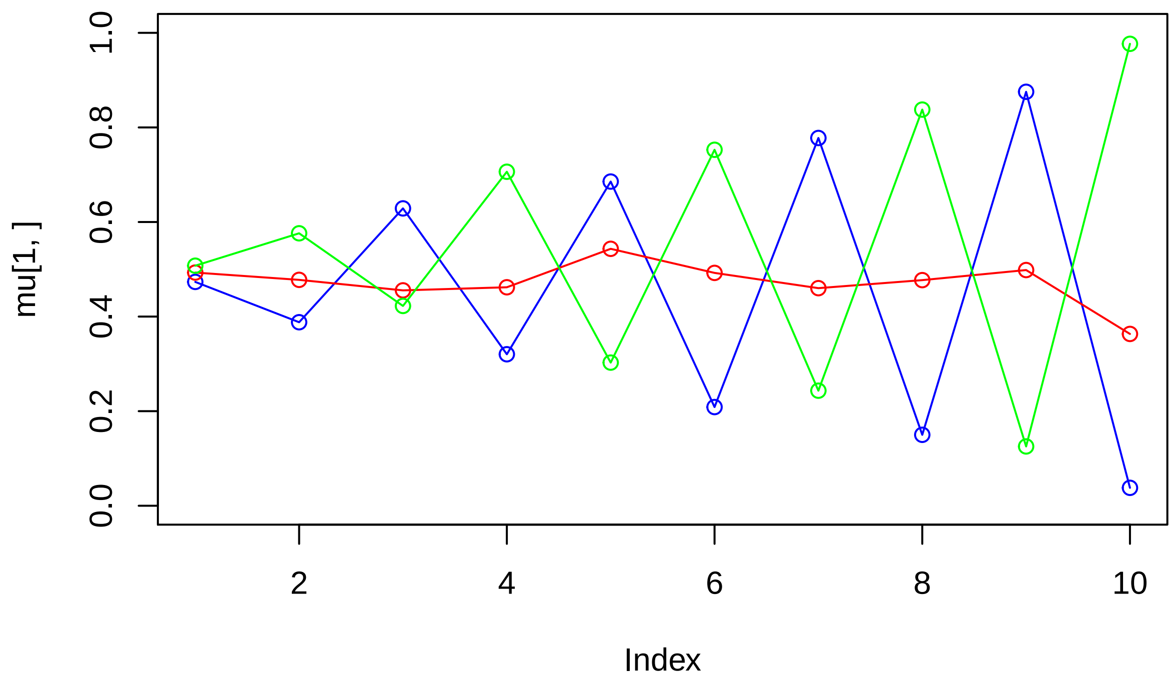

```
## iteration:  15 log likelihood:  -5341.521
```



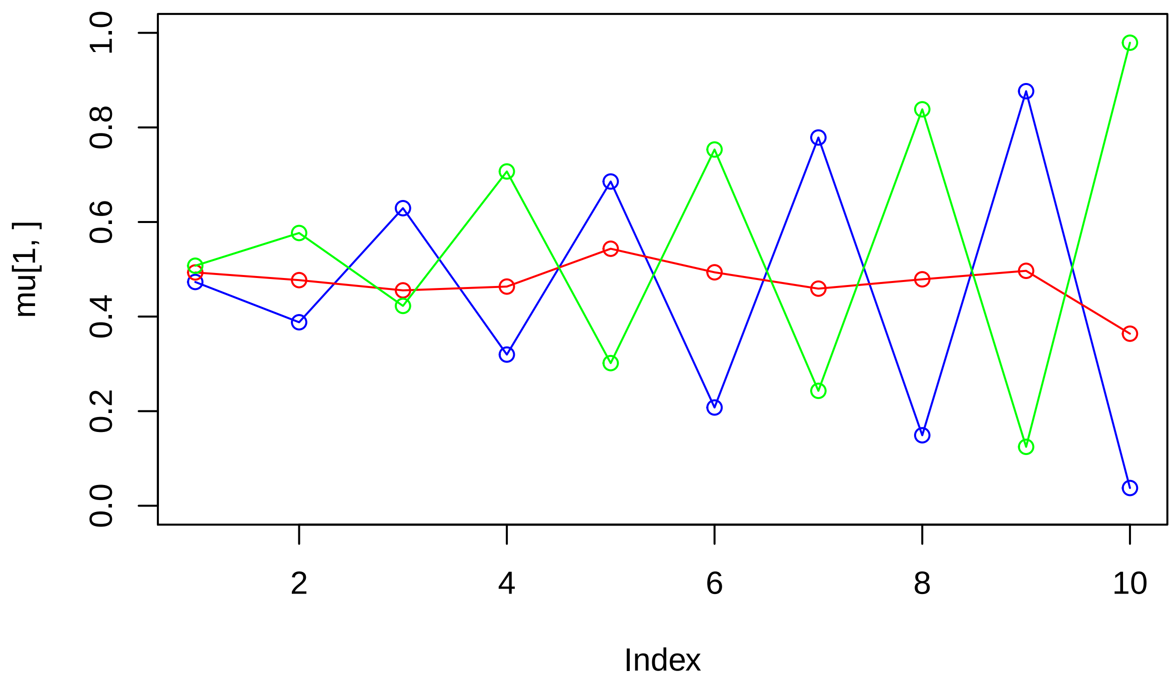

```
## iteration:  16 log likelihood:  -5339.255
```



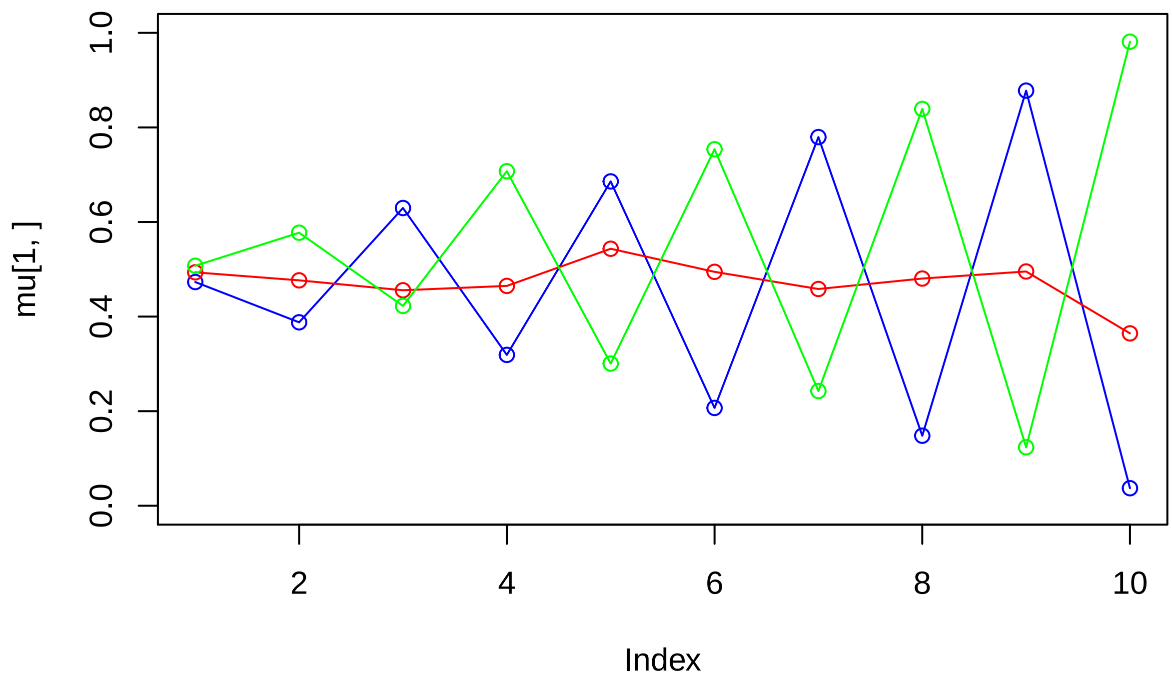

```
## iteration:  17 log likelihood:  -5337.222
```



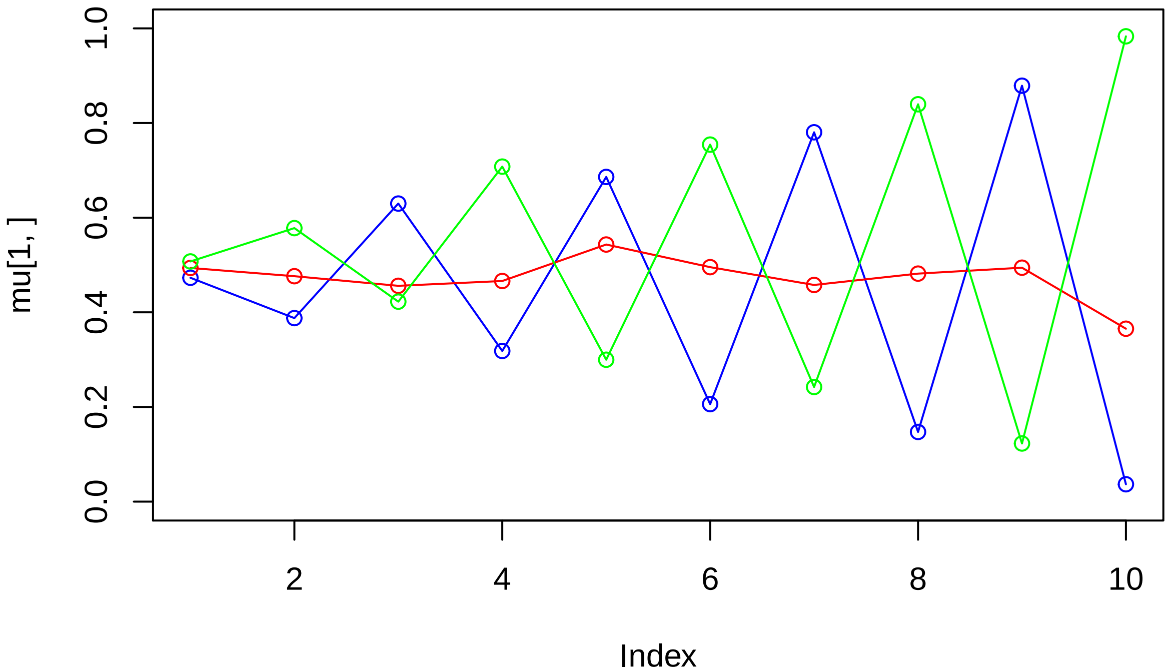

```
## iteration:  18 log likelihood:  -5335.372
```



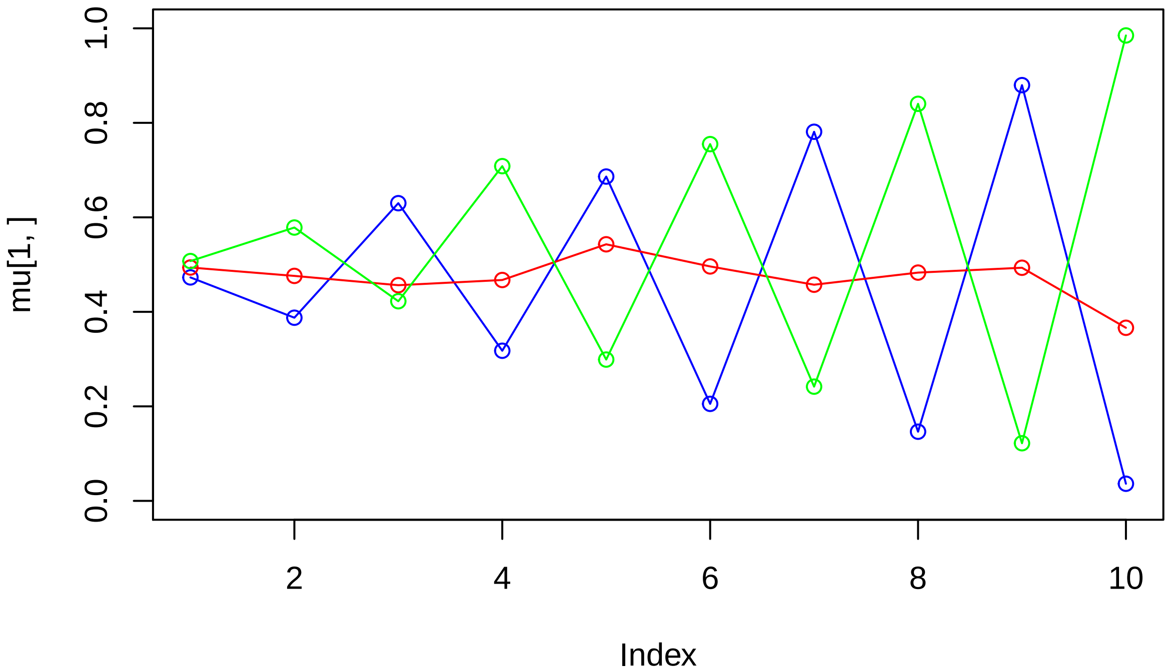

```
## iteration:  19 log likelihood:  -5333.675
```



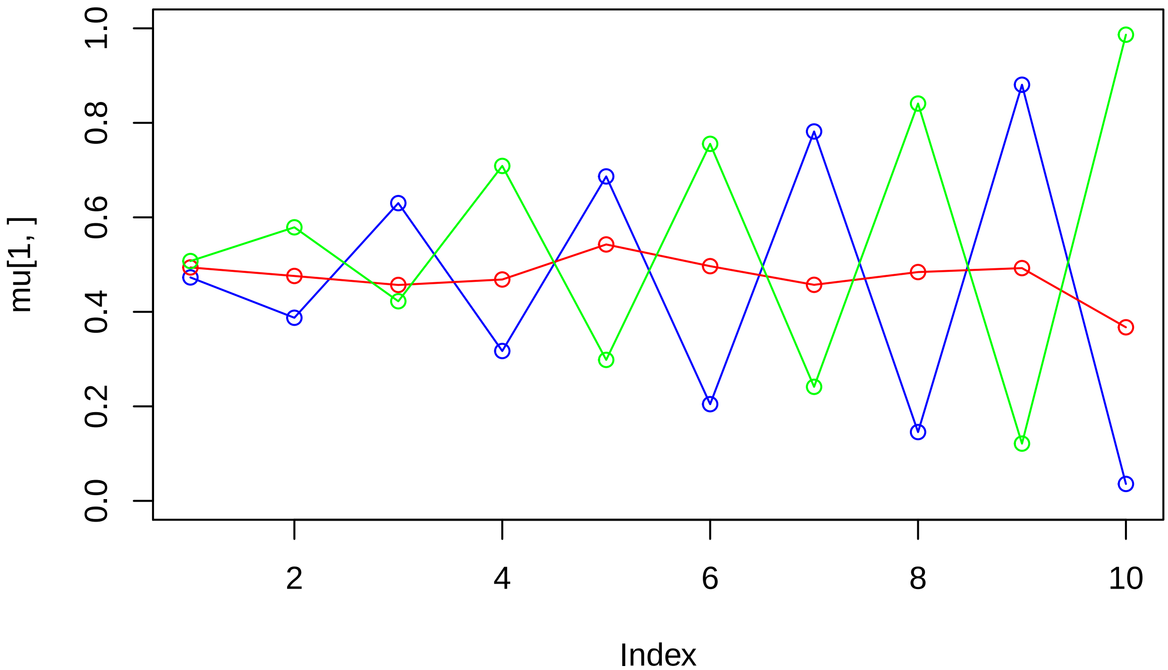

```
## iteration:  20 log likelihood:  -5332.111
```



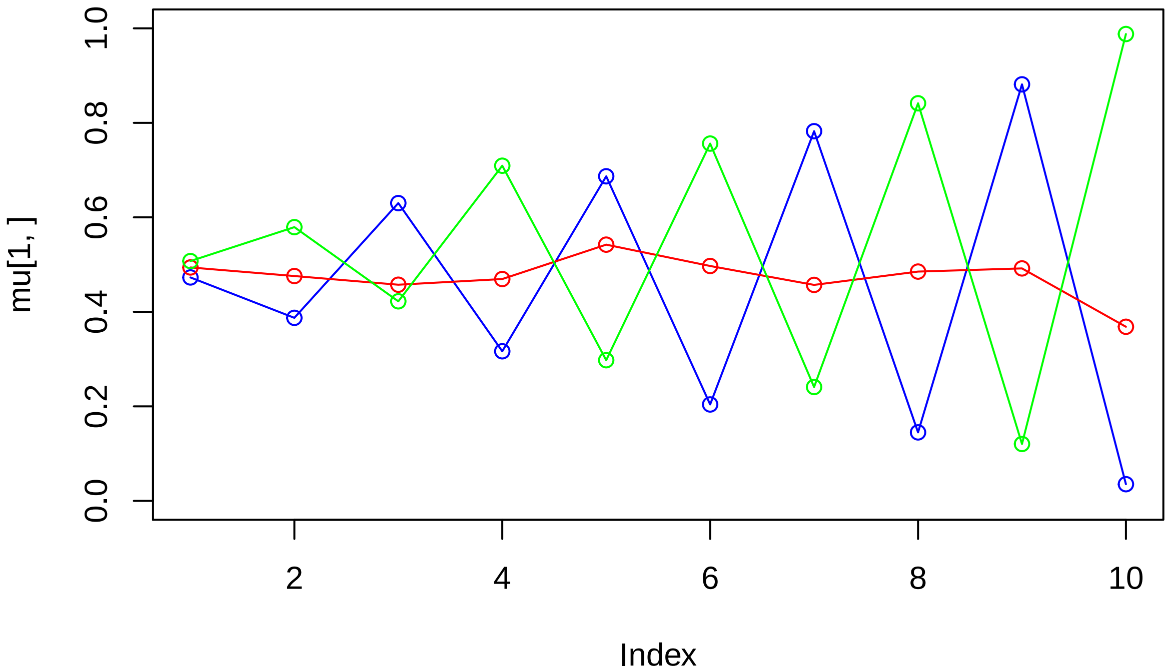

```
## iteration:  21 log likelihood:  -5330.664
```



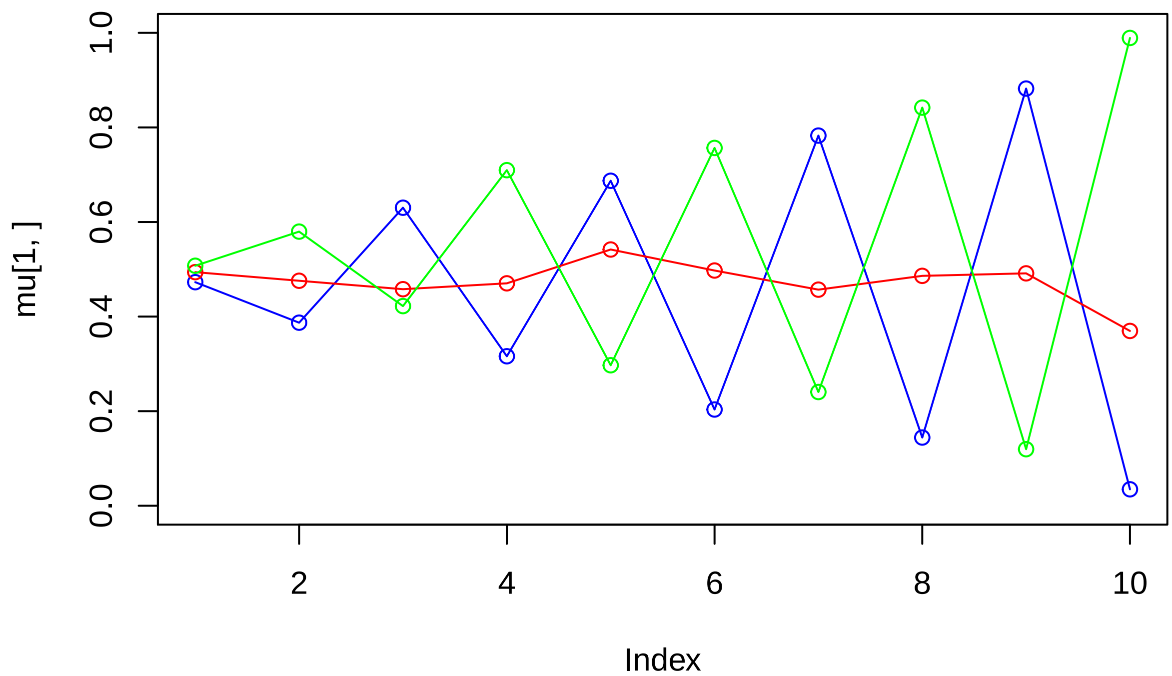

```
## iteration:  22 log likelihood:  -5329.325
```



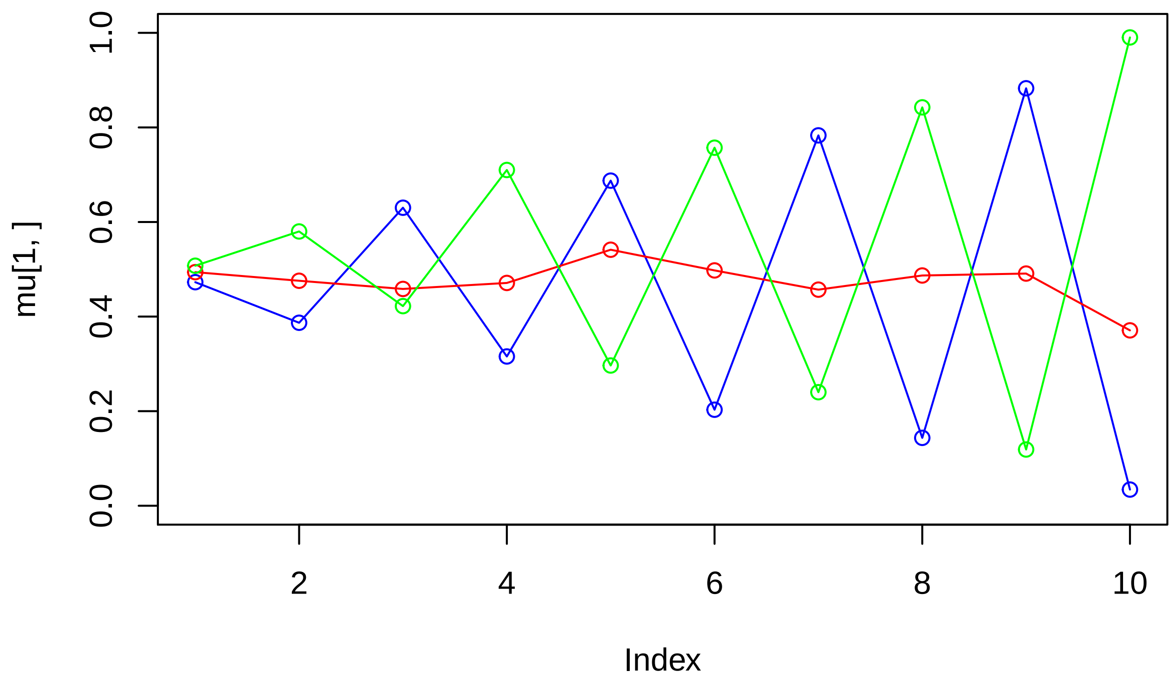

```
## iteration:  23 log likelihood:  -5328.085
```



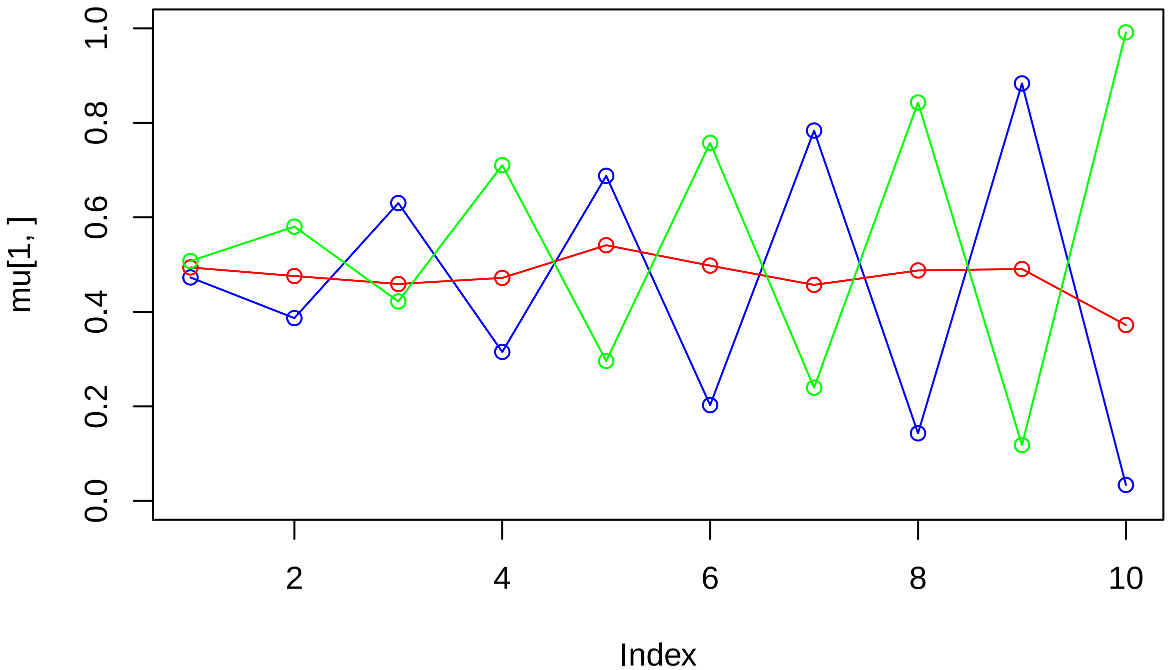

```
## iteration:  24 log likelihood:  -5326.935
```



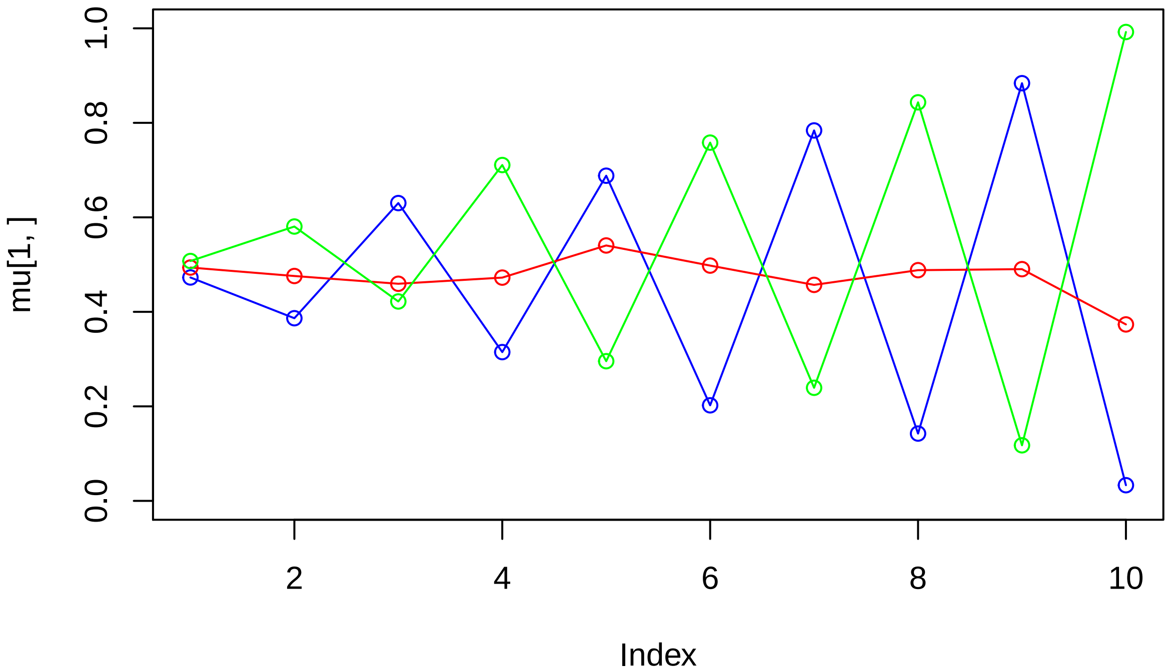

```
## iteration:  25 log likelihood:  -5325.871
```



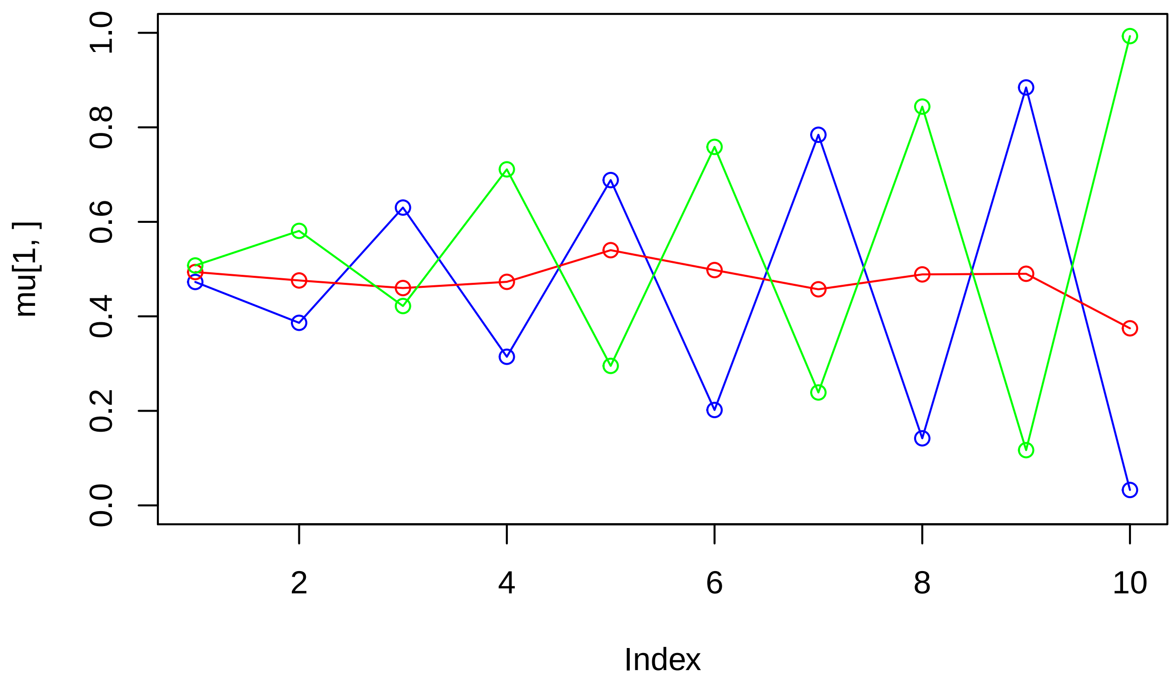

```
## iteration:  26 log likelihood:  -5324.884
```



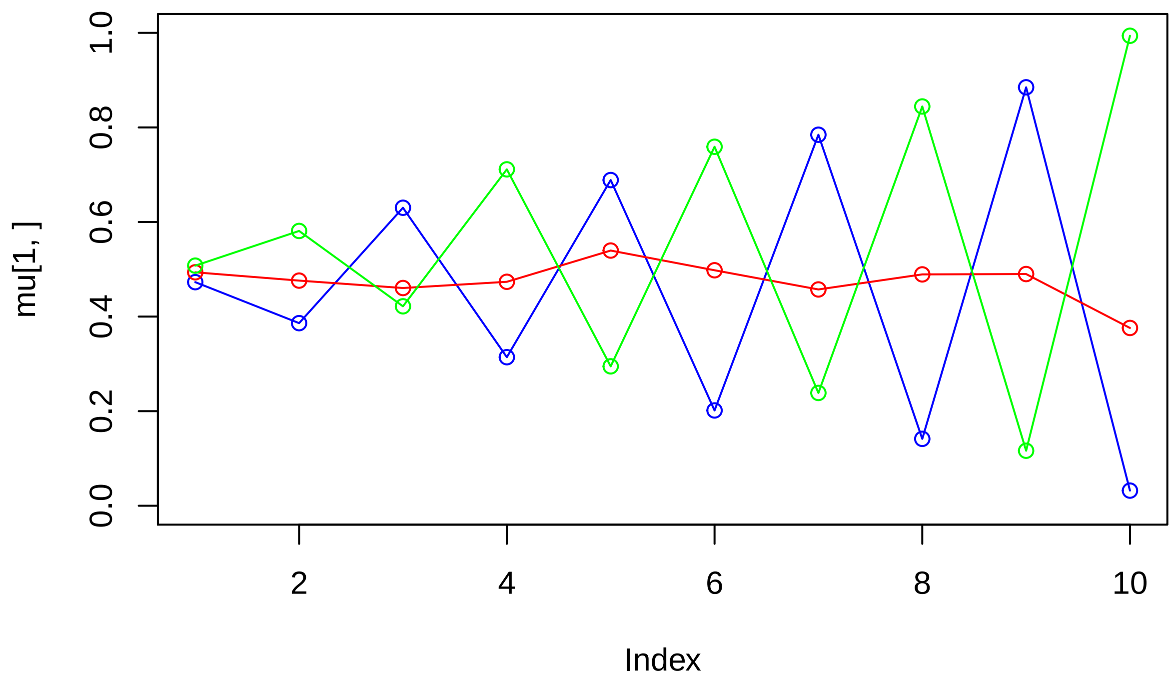

```
## iteration:  27 log likelihood:  -5323.971
```



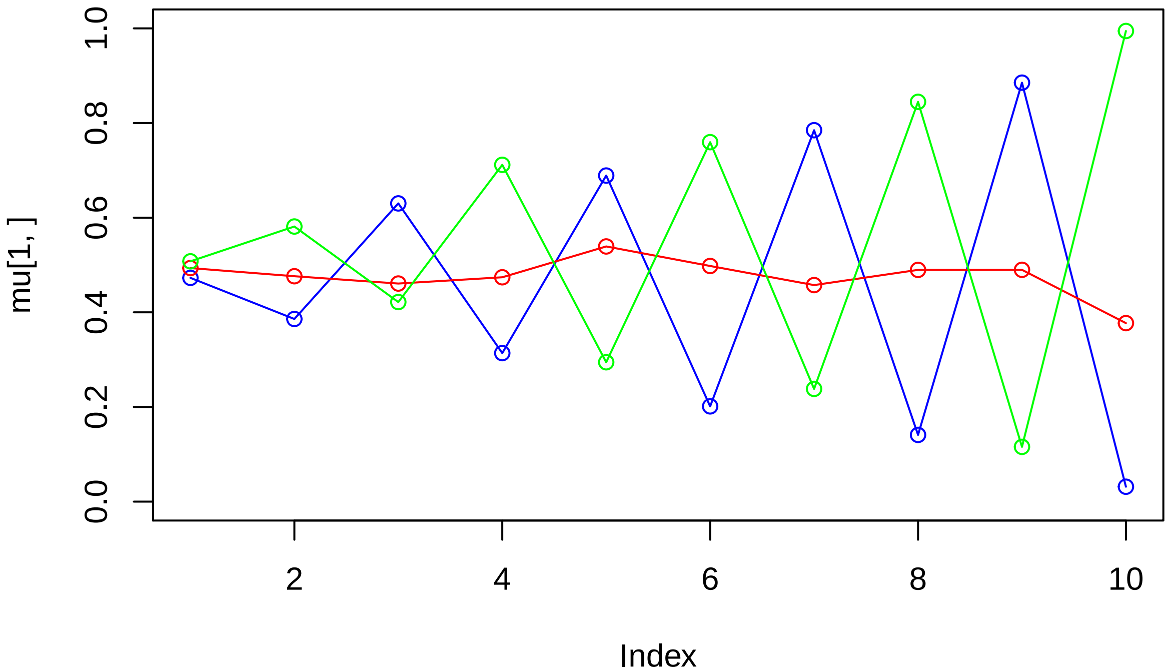

```
## iteration:  28 log likelihood:  -5323.124
```



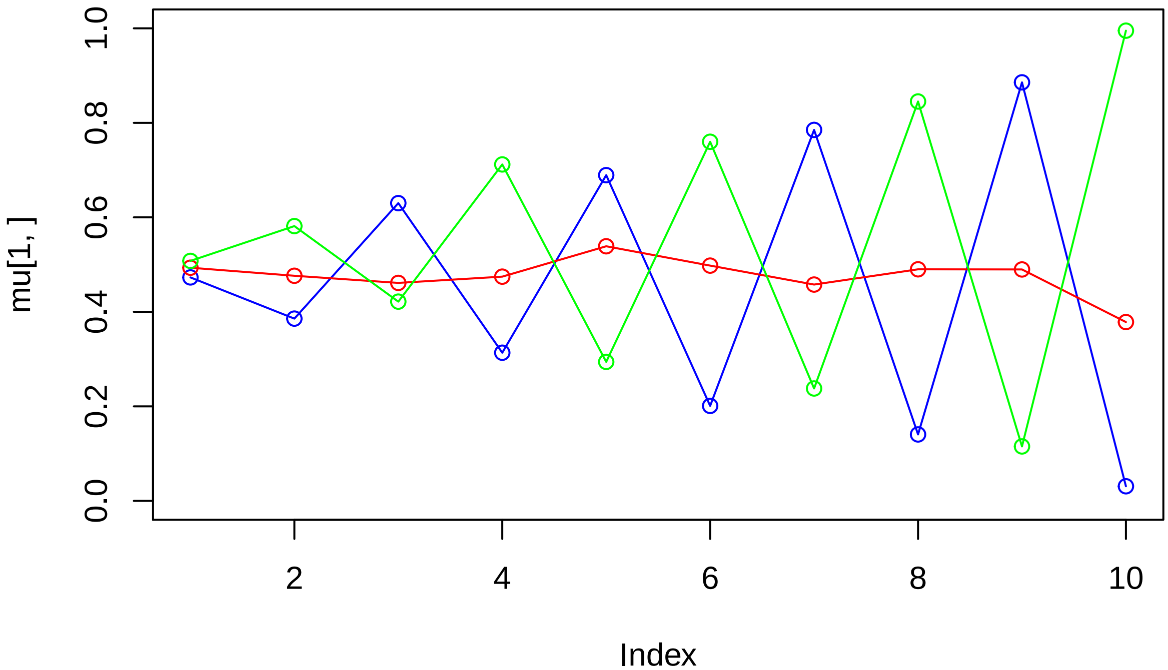

```
## iteration:  29 log likelihood:  -5322.34
```



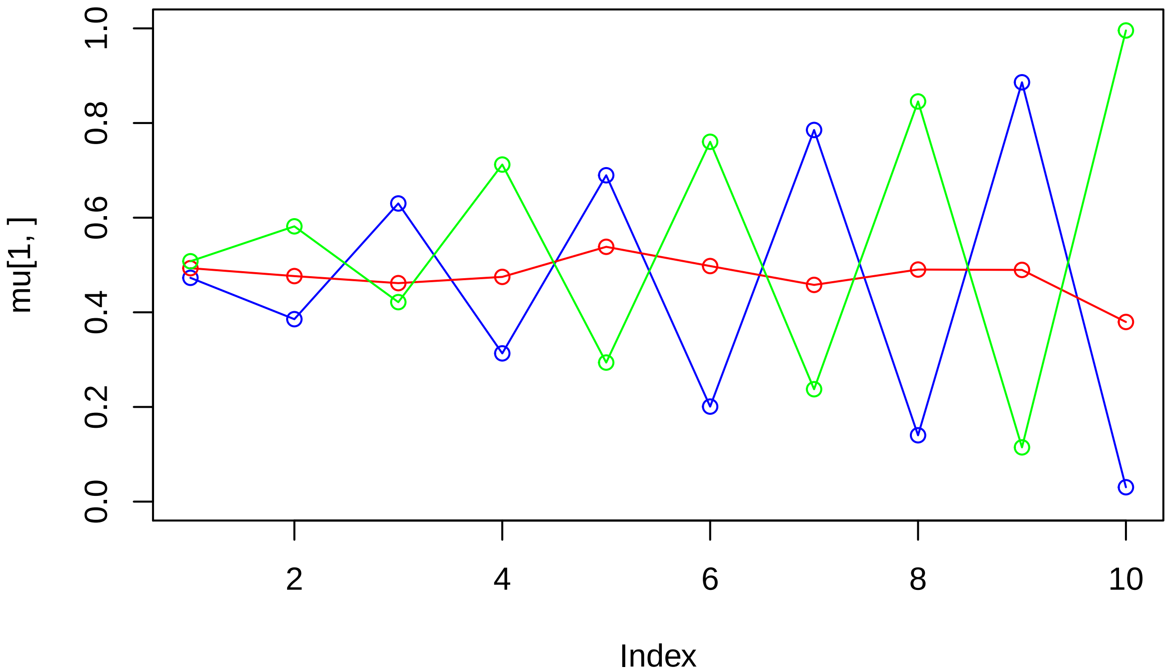

```
## iteration:  30 log likelihood:  -5321.613
```



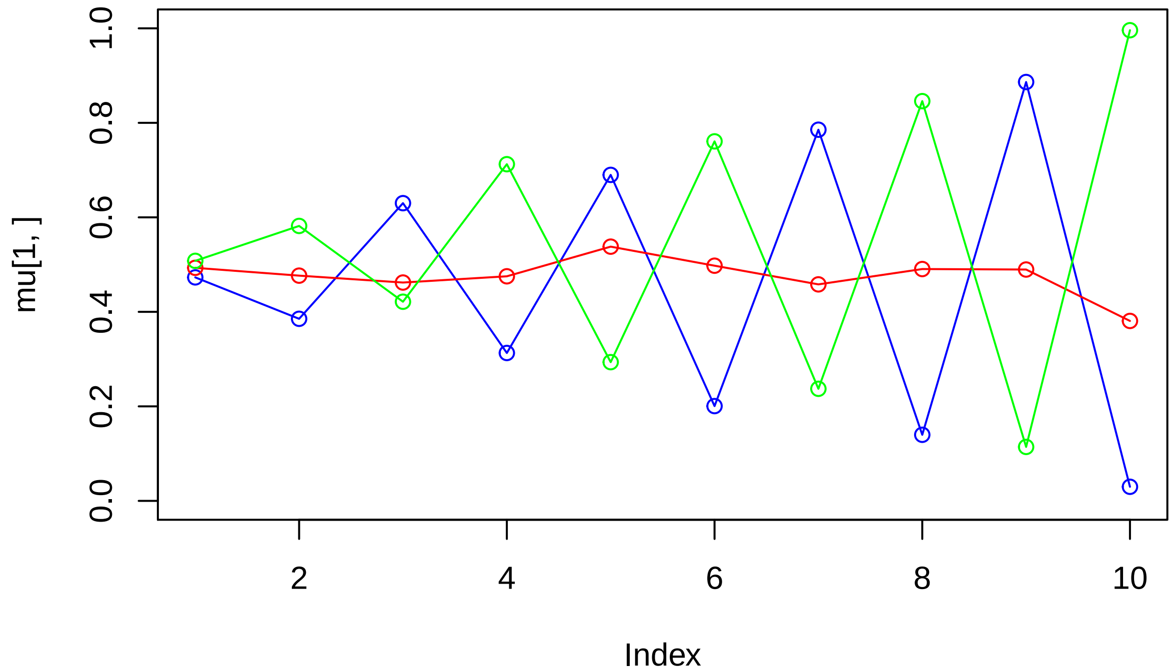

```
## iteration:  31 log likelihood:  -5320.939
```



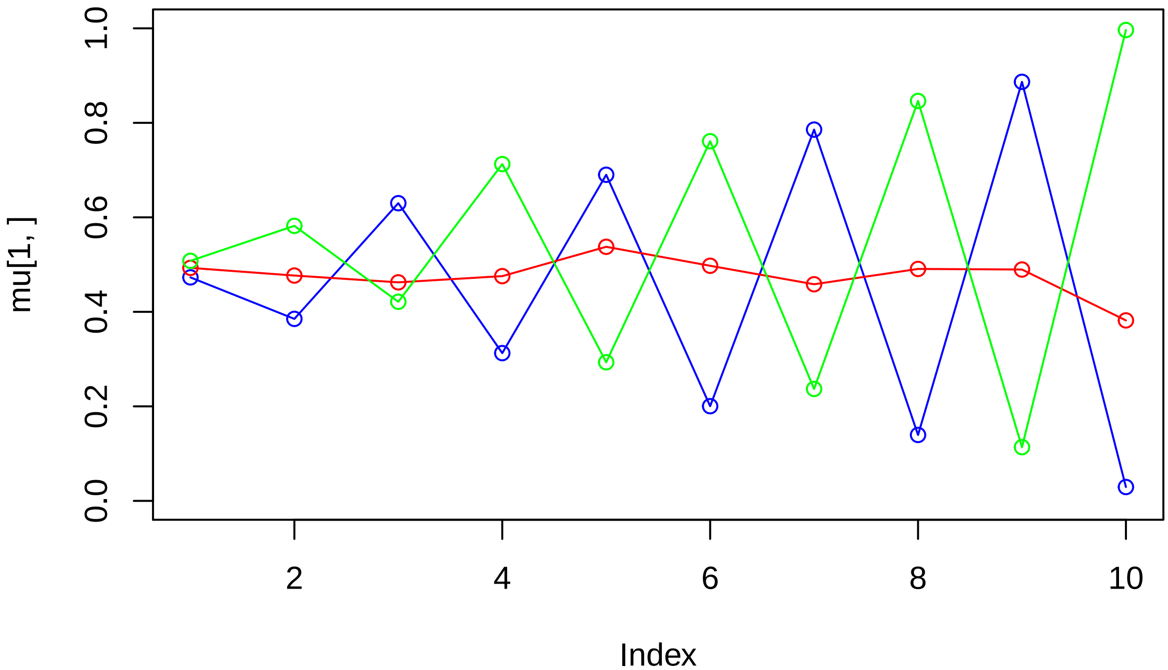

```
## iteration:  32 log likelihood:  -5320.314
```



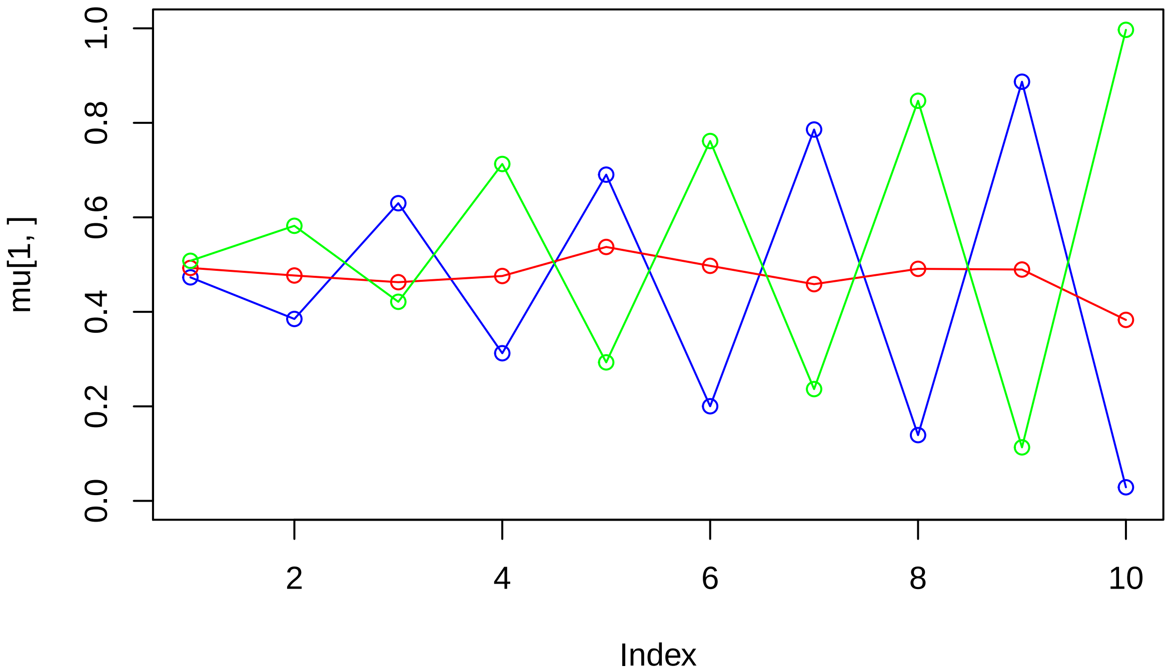

```
## iteration:  33 log likelihood:  -5319.734
```



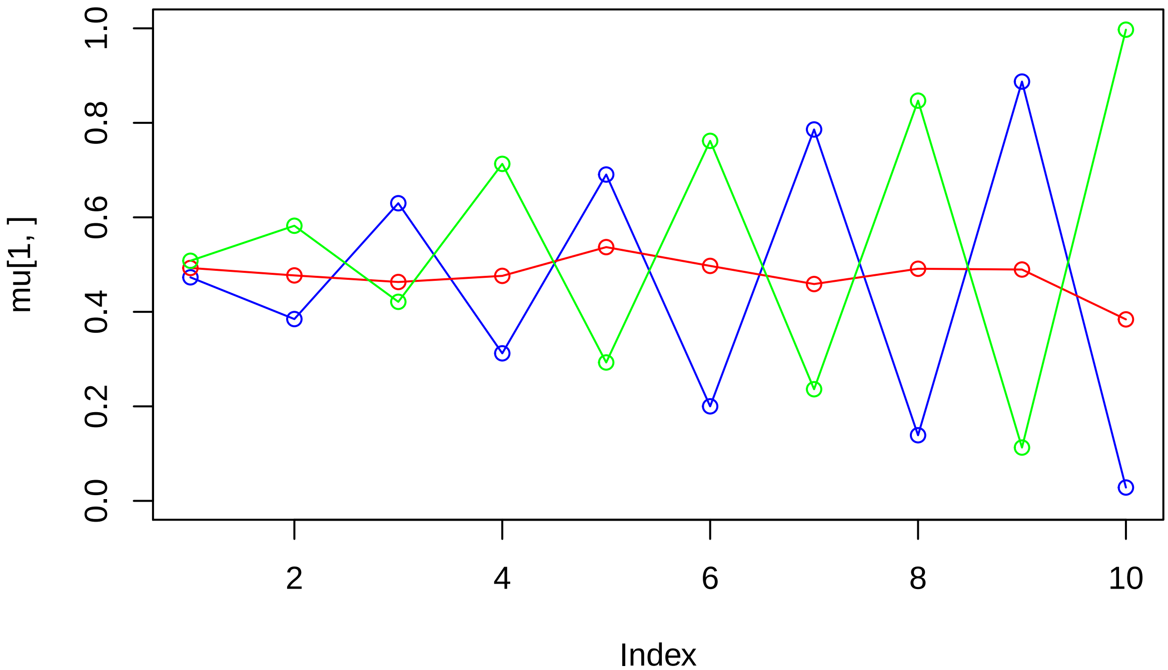

```
## iteration:  34 log likelihood:  -5319.195
```



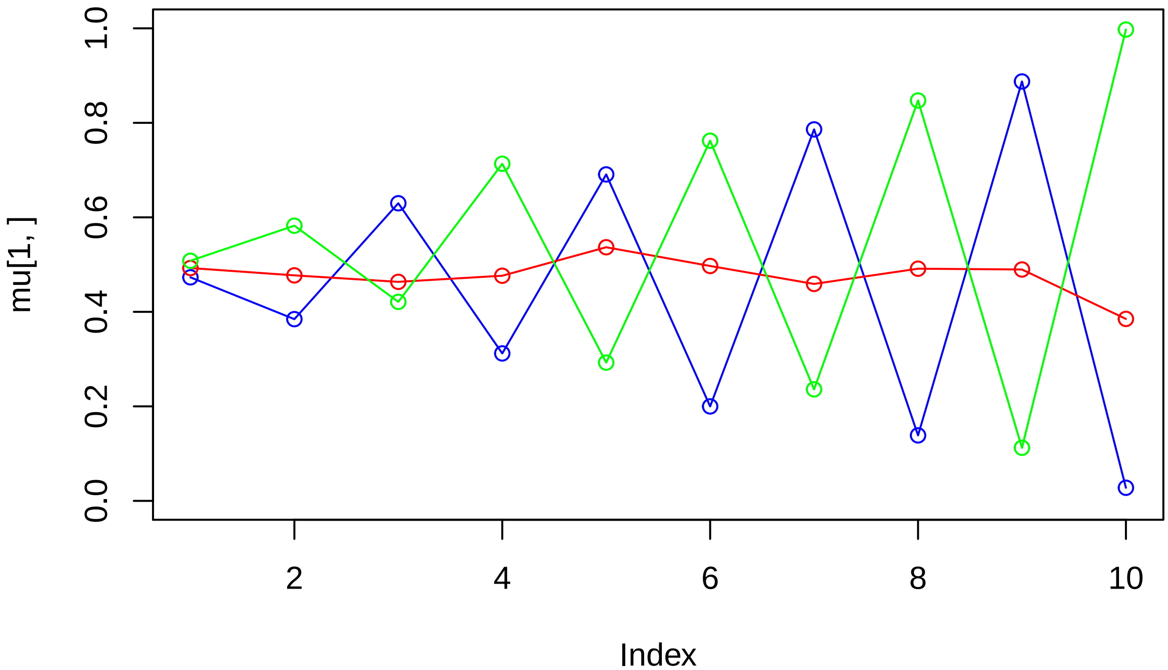

```
## iteration:  35 log likelihood:  -5318.694
```



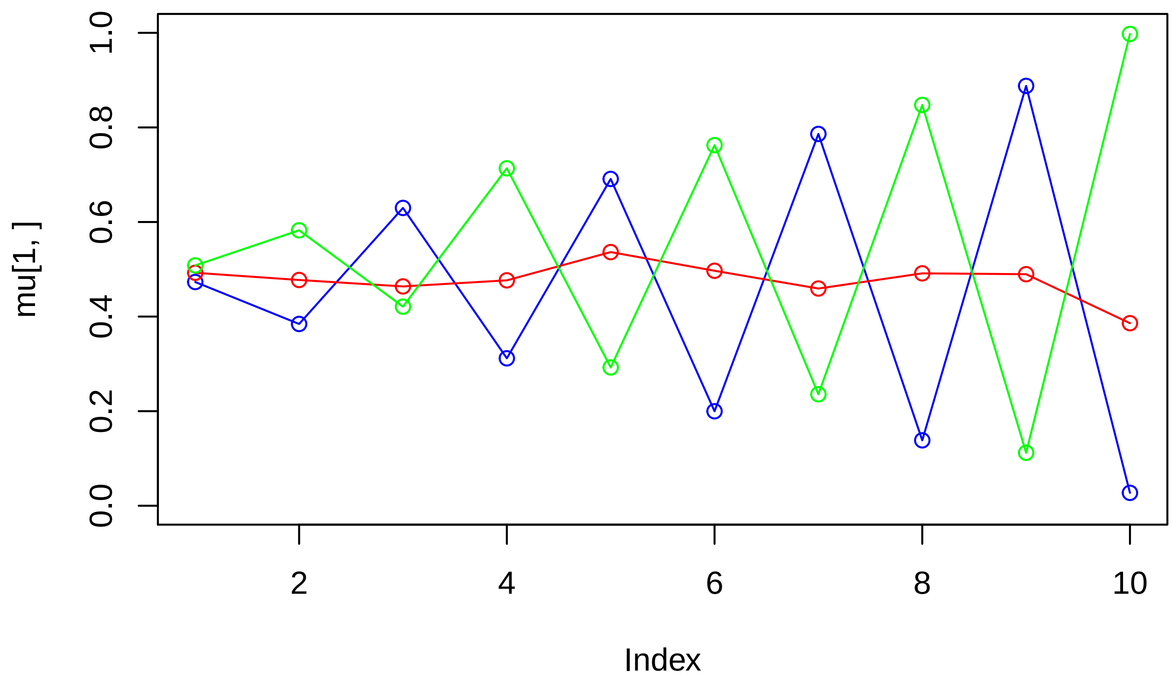

```
## iteration:  36 log likelihood:  -5318.228
```



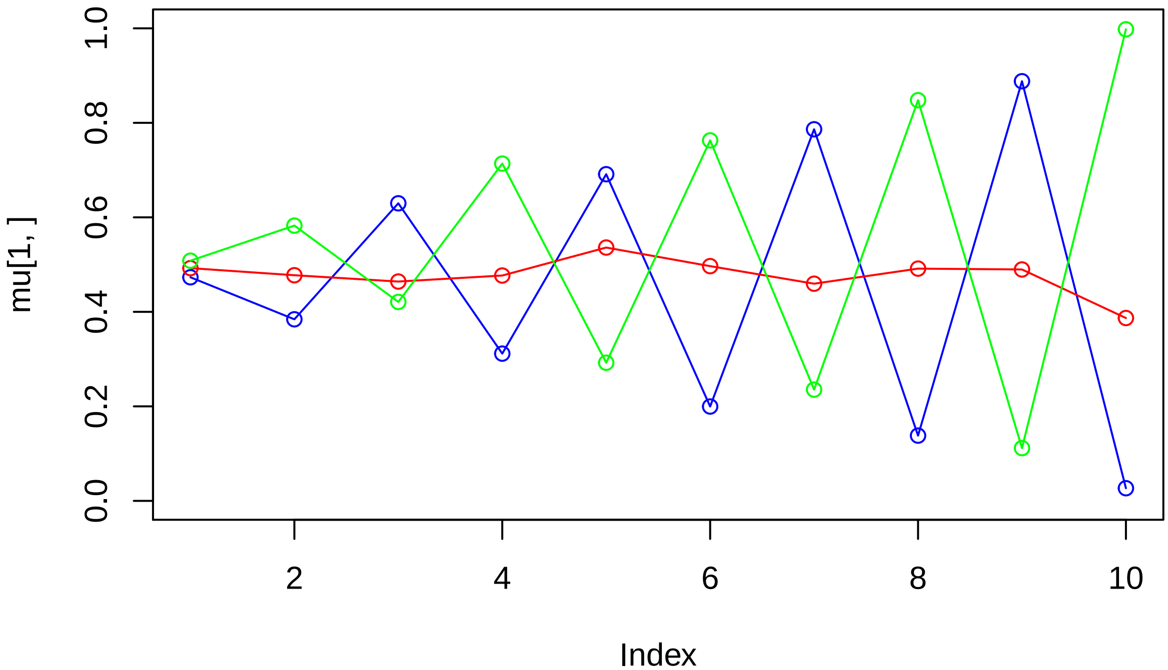

```
## iteration:  37 log likelihood:  -5317.794
```



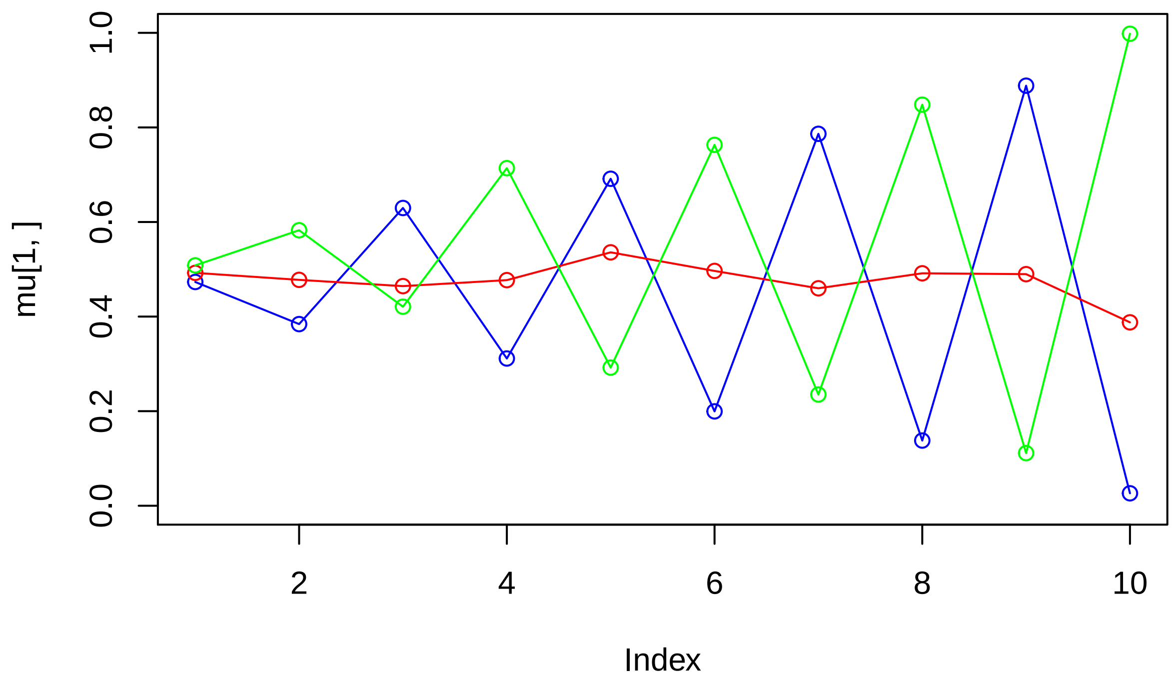

```
## iteration:  38 log likelihood:  -5317.389
```



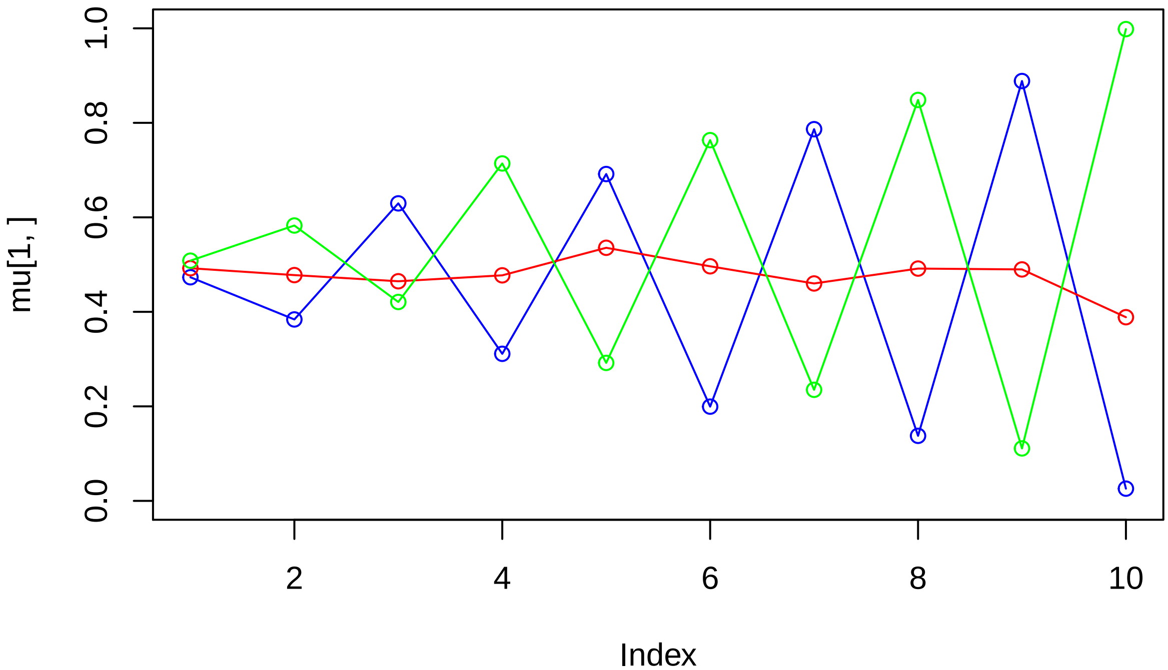

```
## iteration:  39 log likelihood:  -5317.012
```



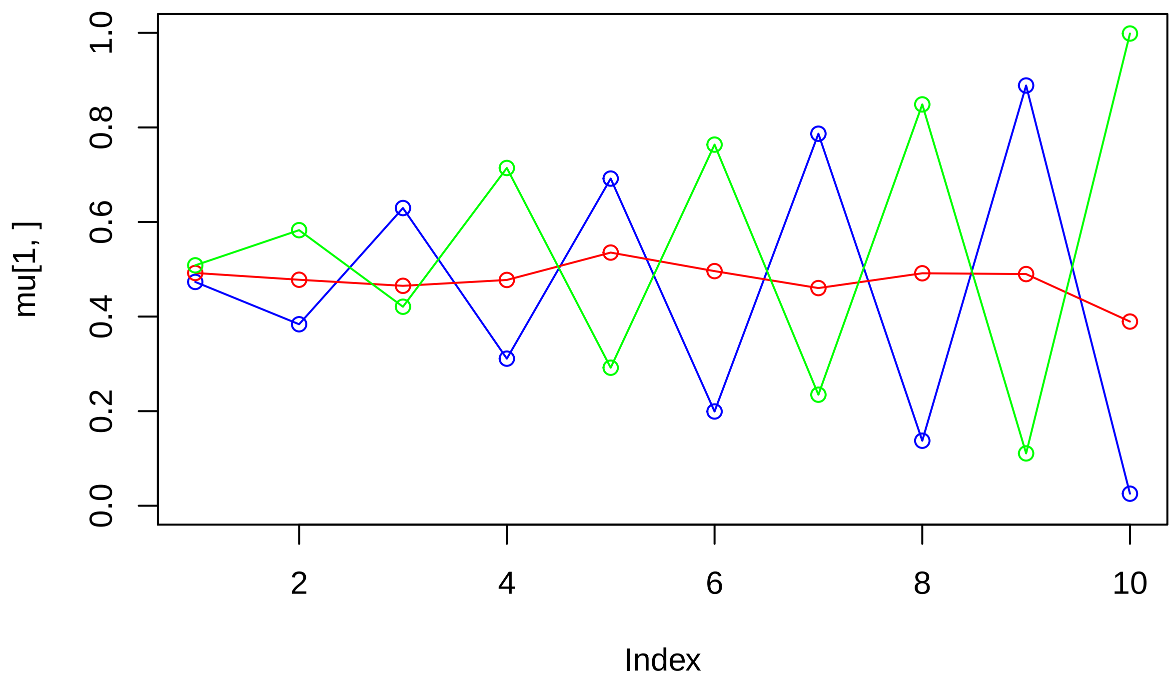

```
## iteration:  40 log likelihood:  -5316.66
```



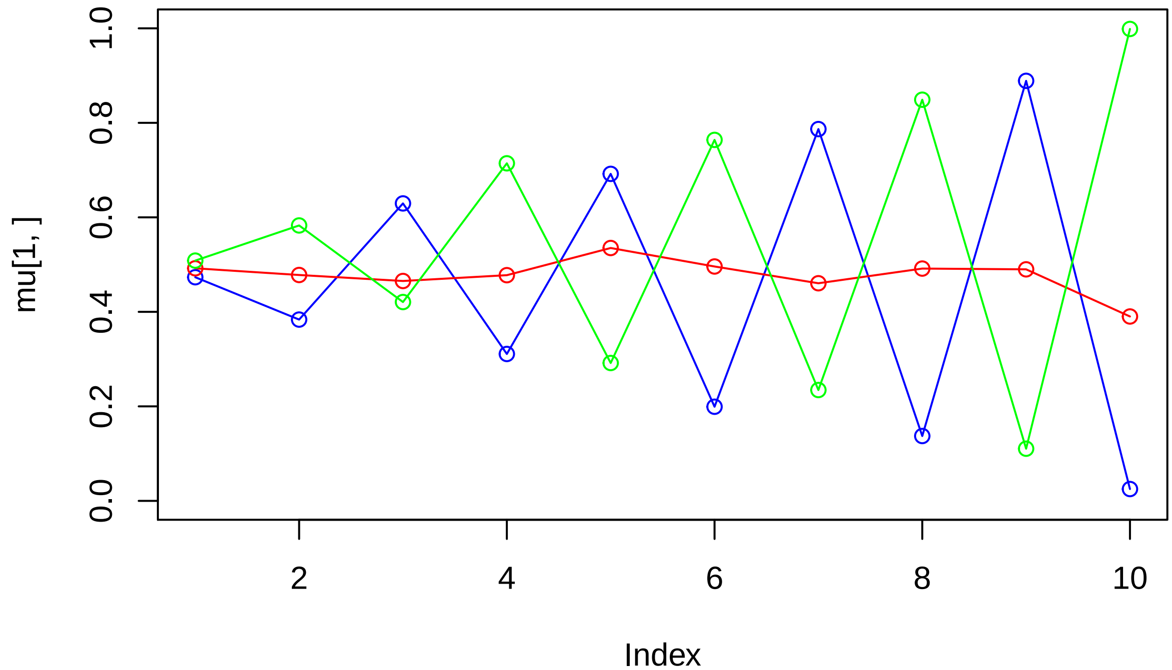

```
## iteration:  41 log likelihood:  -5316.33
```



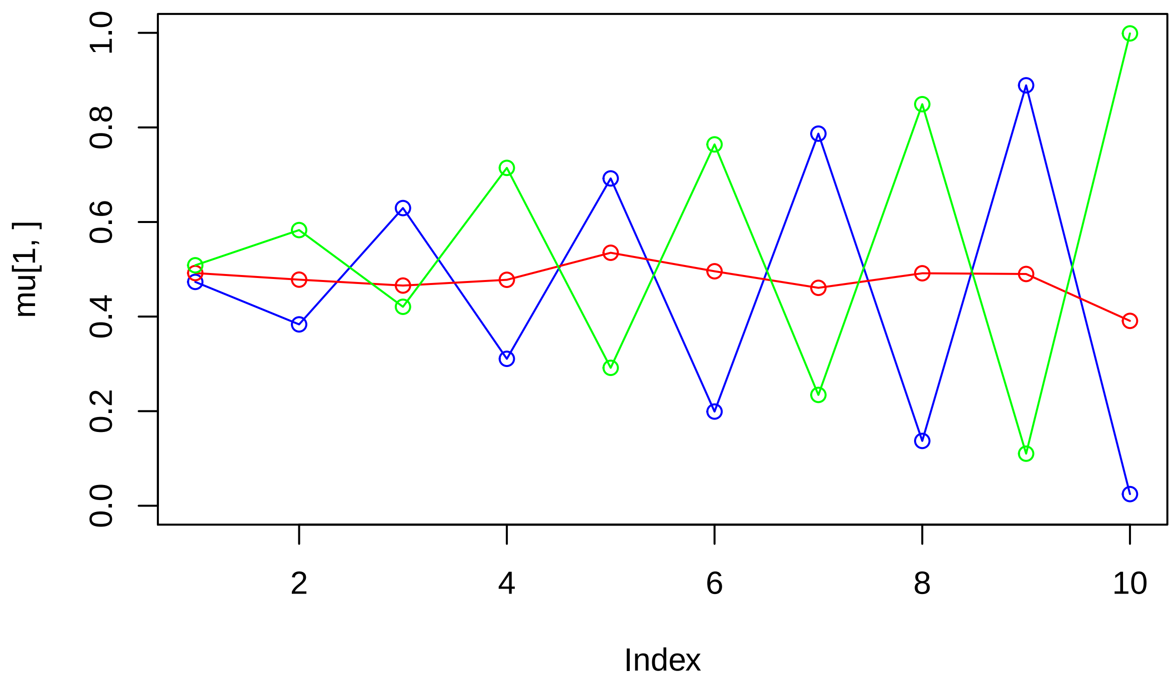

```
## iteration:  42 log likelihood:  -5316.021
```



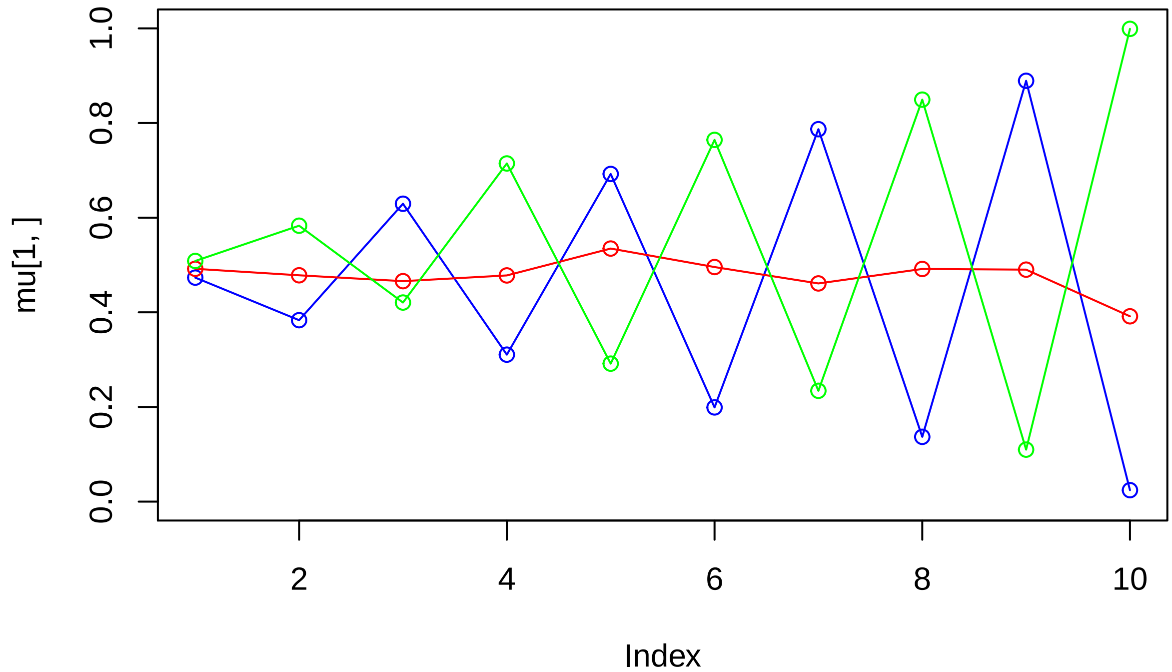

```
## iteration:  43 log likelihood:  -5315.732
```



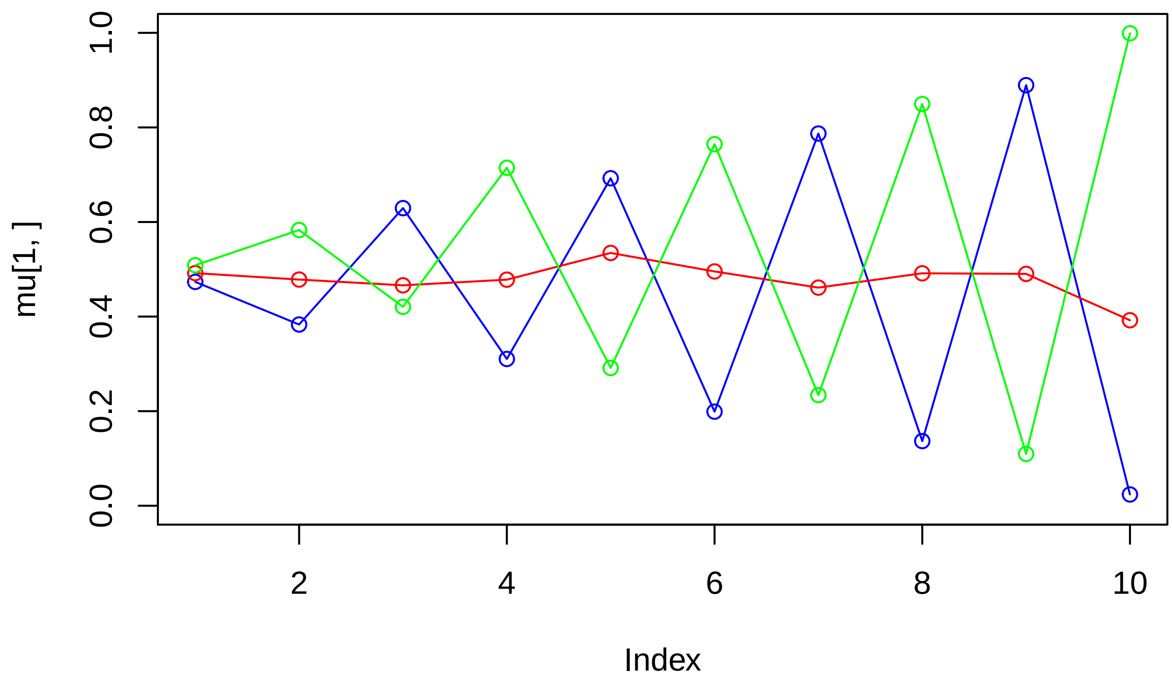

```
## iteration:  44 log likelihood:  -5315.461
```



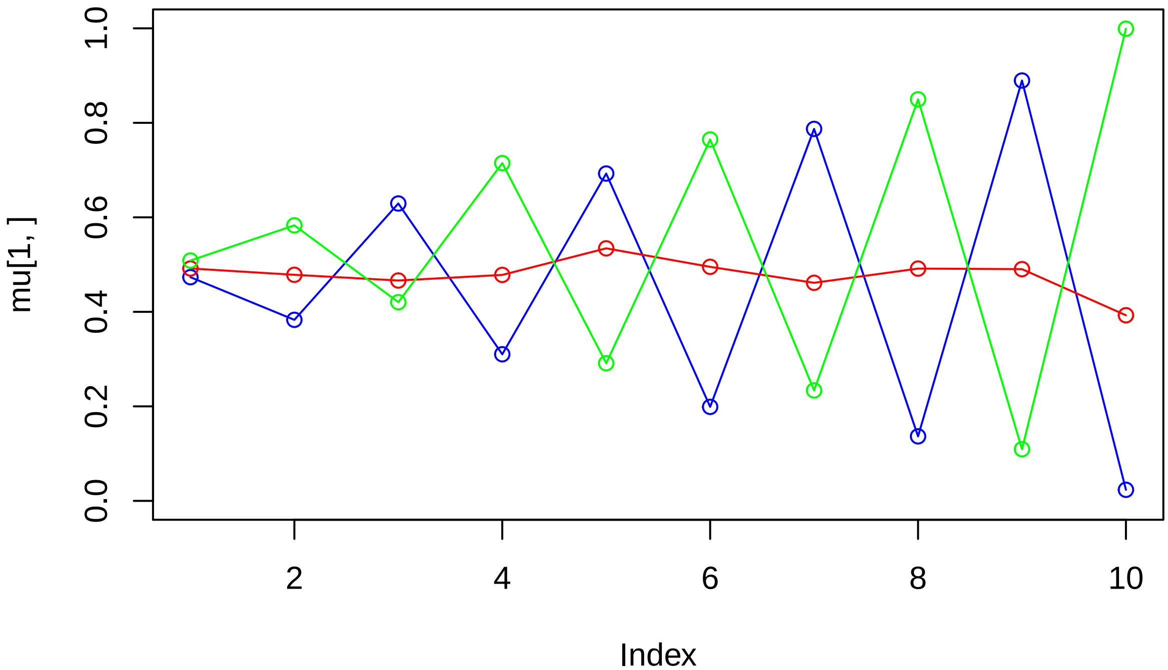

```
## iteration:  45 log likelihood:  -5315.205
```



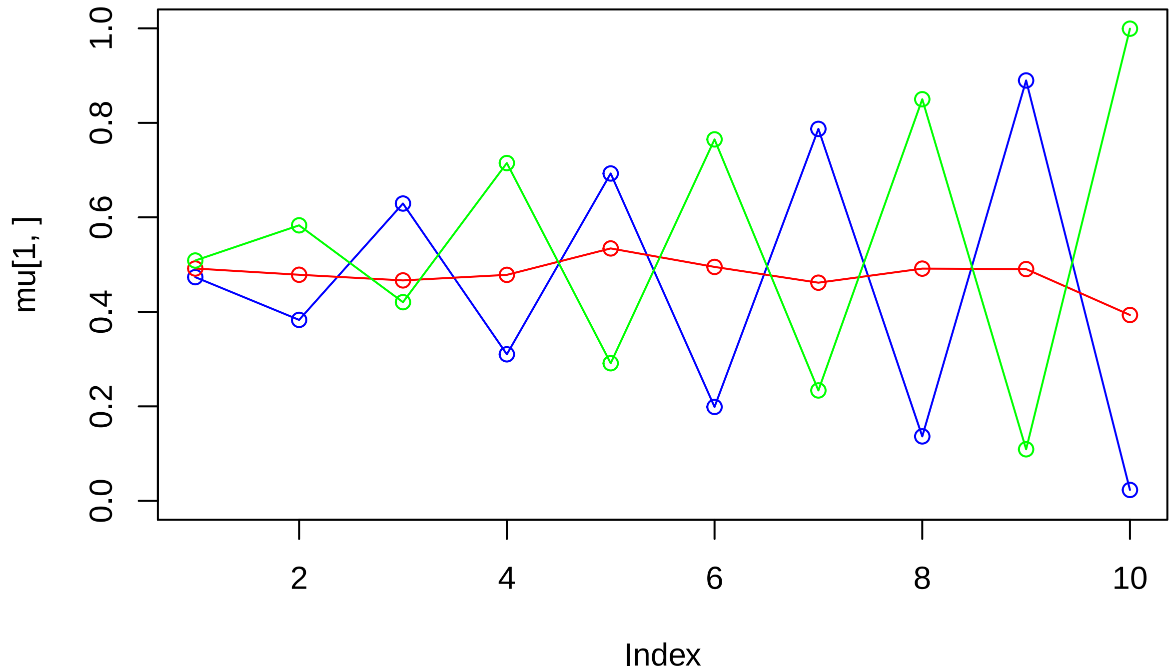

```
## iteration:  46 log likelihood:  -5314.965
```



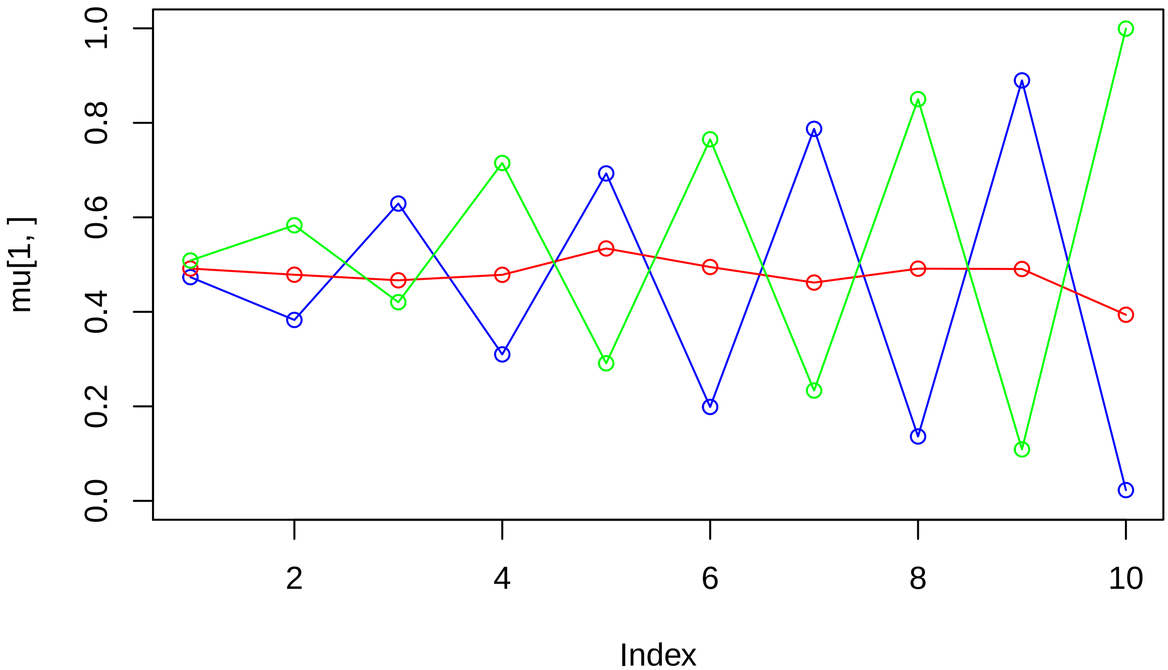

```
## iteration:  47 log likelihood:  -5314.739
```



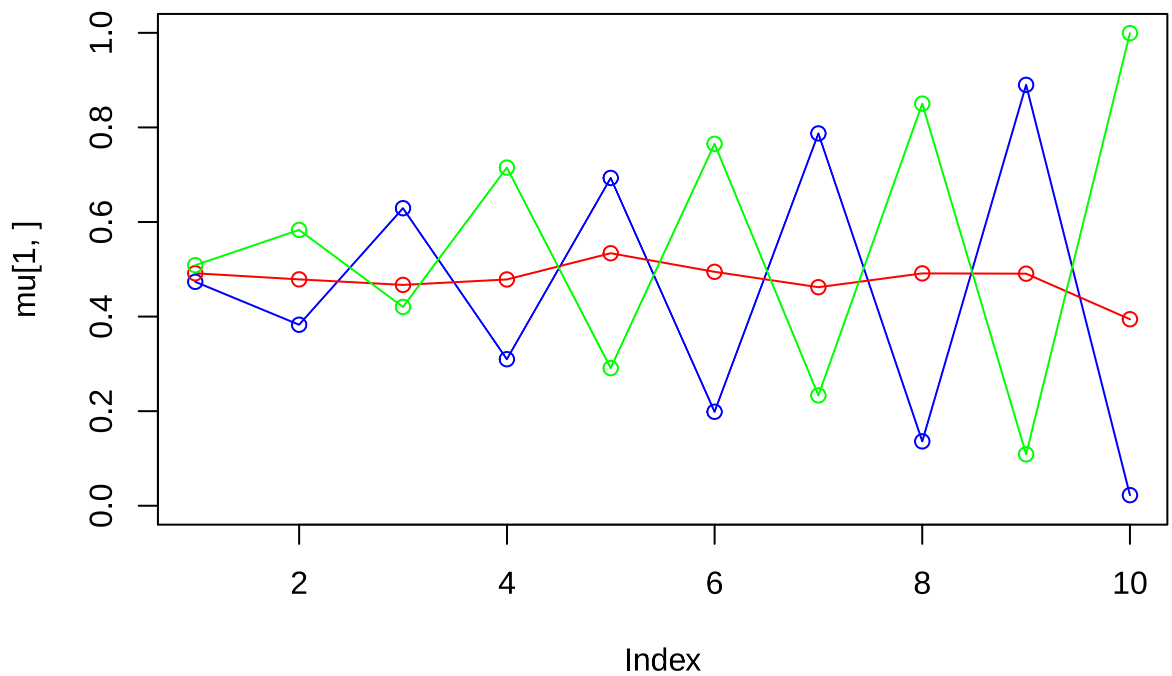

```
## iteration:  48 log likelihood:  -5314.526
```



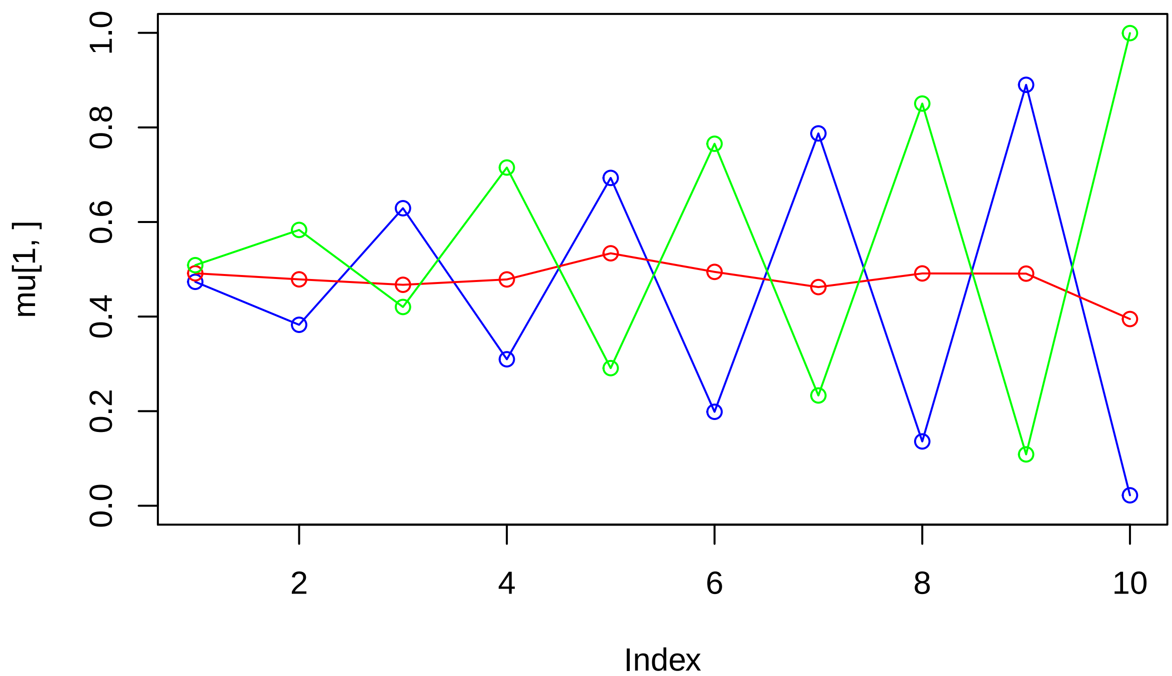

```
## iteration:  49 log likelihood:  -5314.324
```



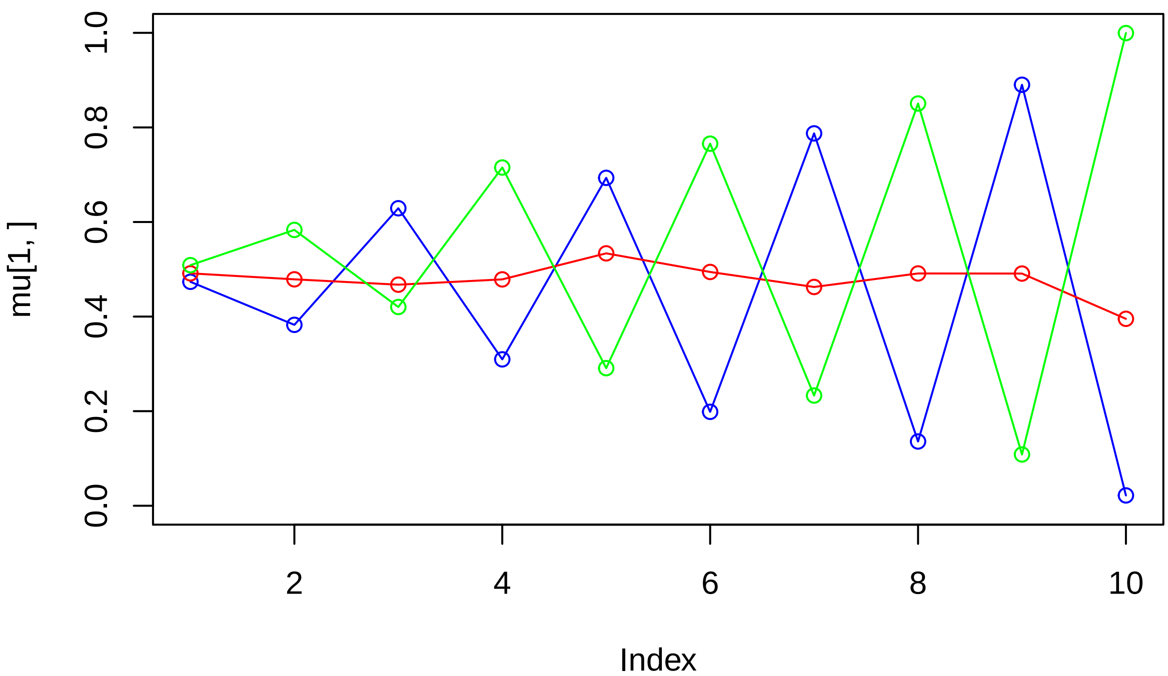

```
## iteration:  50 log likelihood:  -5314.133
```



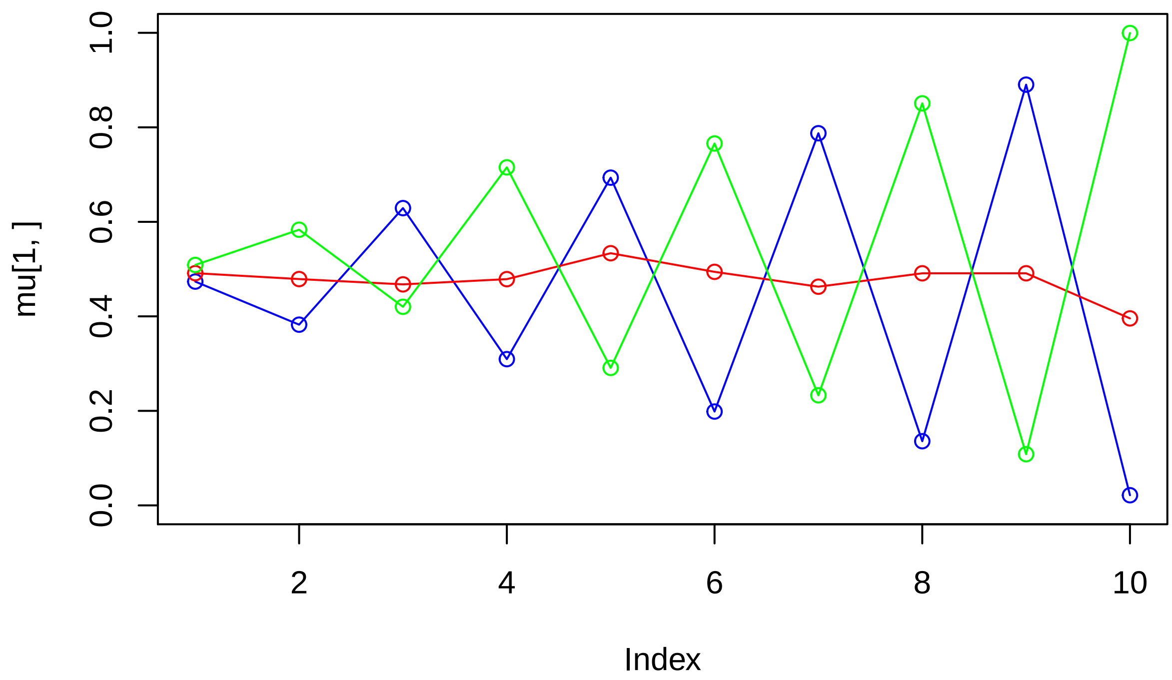

```
## iteration:  51 log likelihood:  -5313.952
```



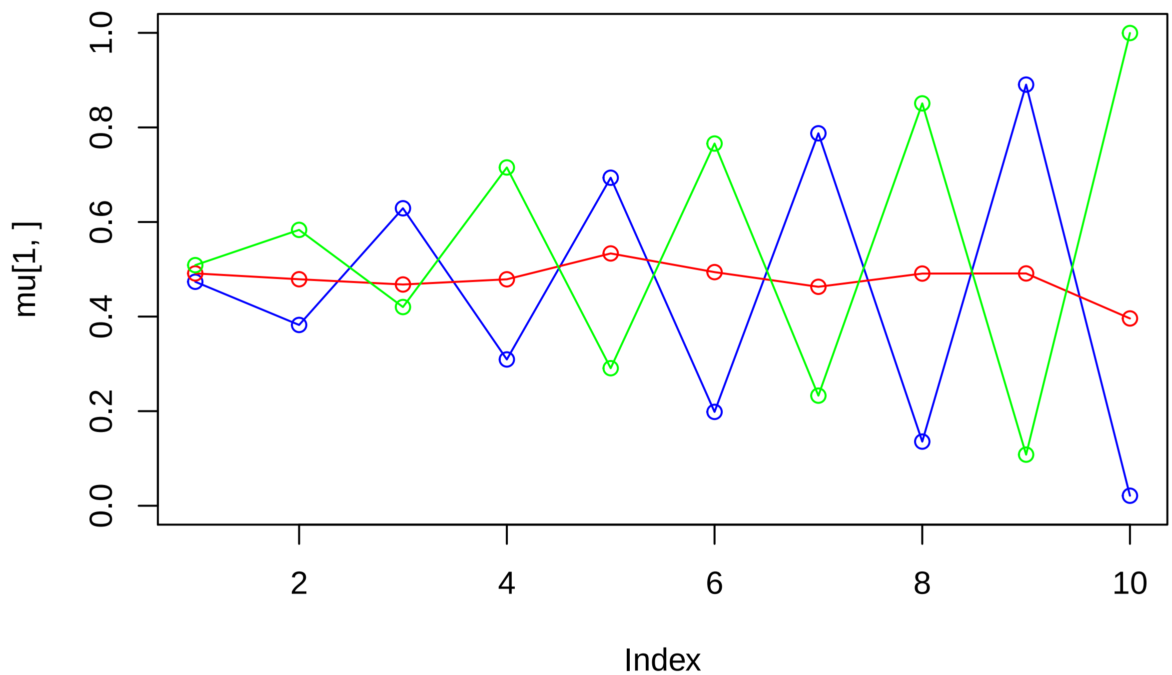

```
## iteration:  52 log likelihood:  -5313.78
```



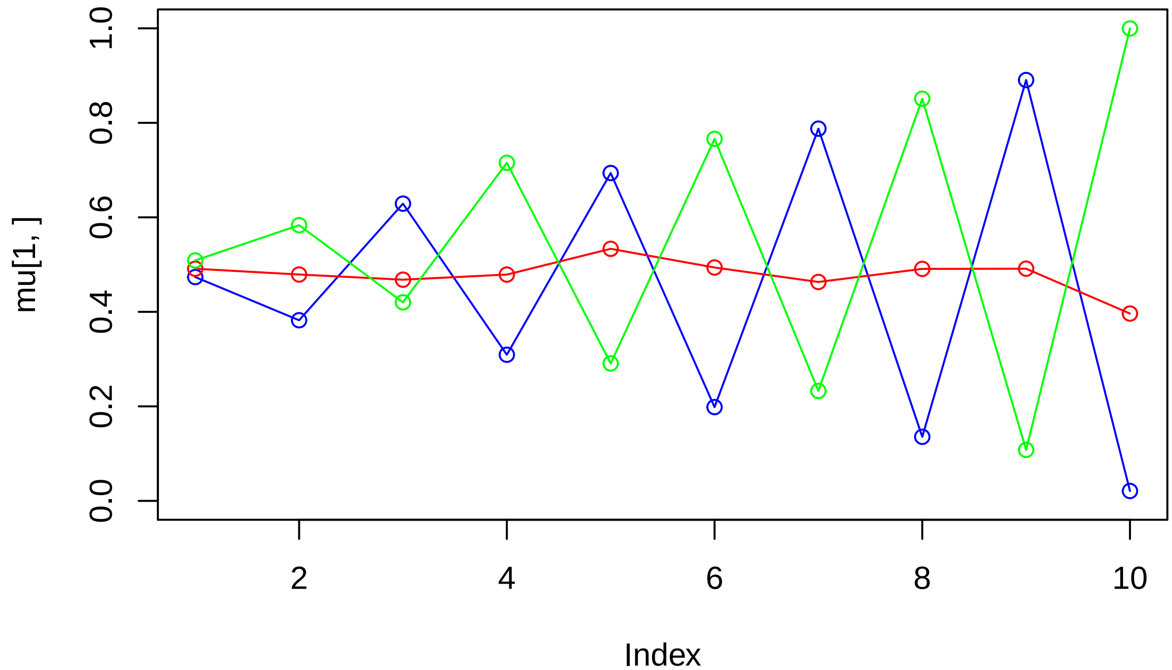

```
## iteration:  53 log likelihood:  -5313.617
```



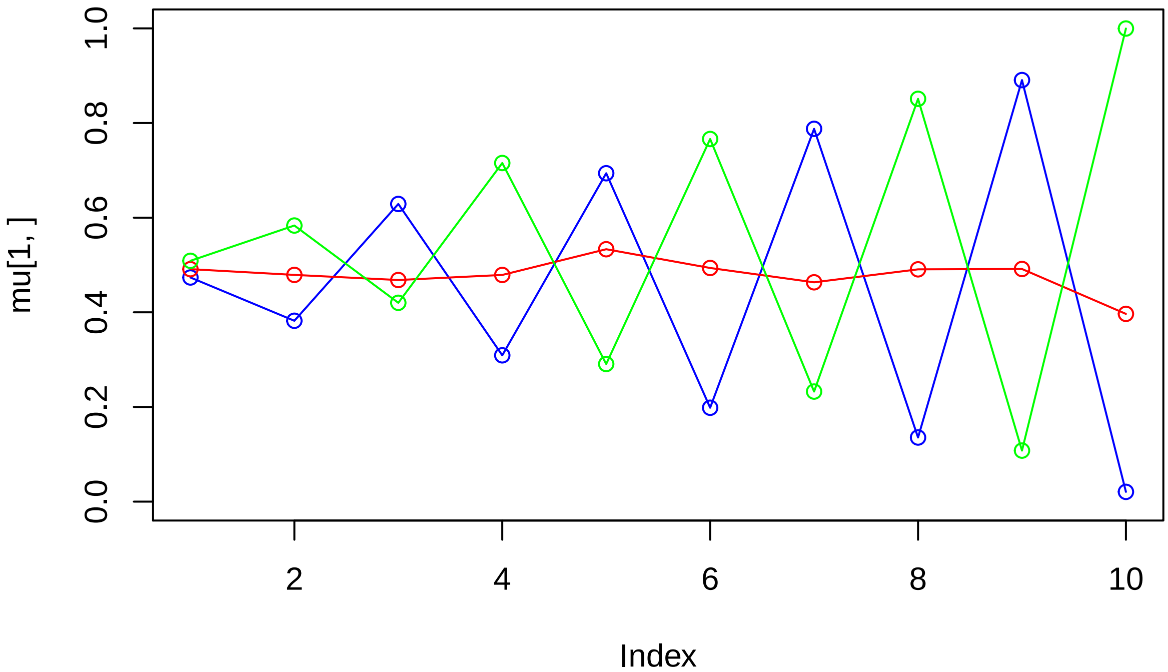

```
## iteration:  54 log likelihood:  -5313.461
```



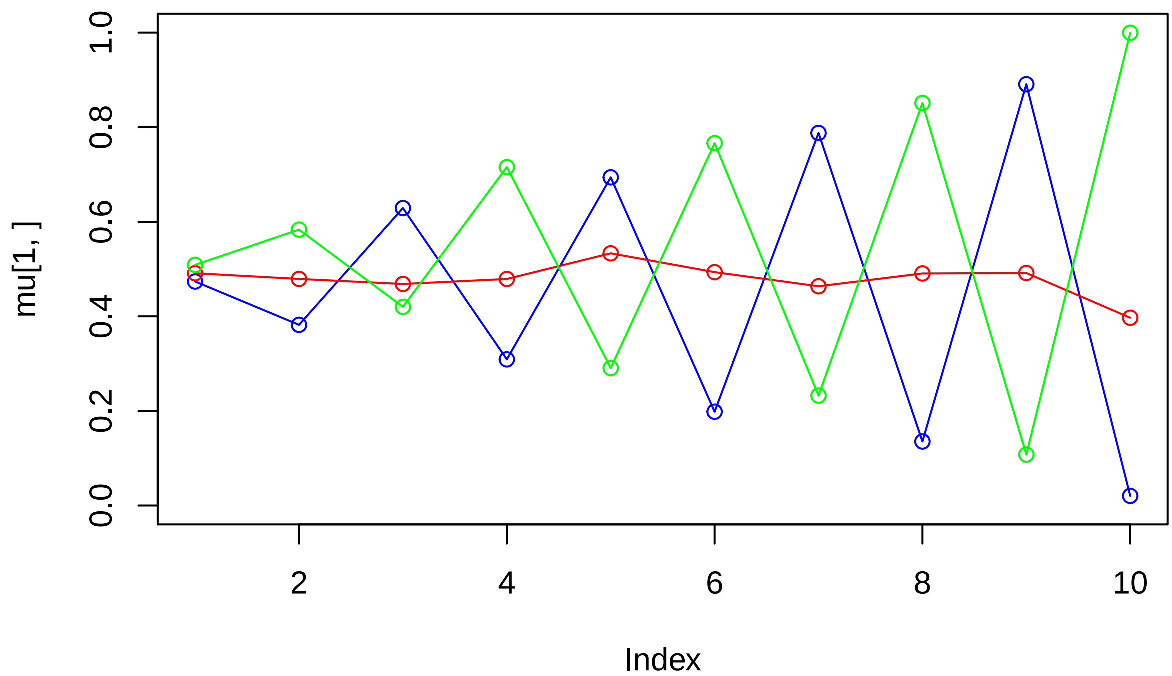

```
## iteration:  55 log likelihood:  -5313.313
```



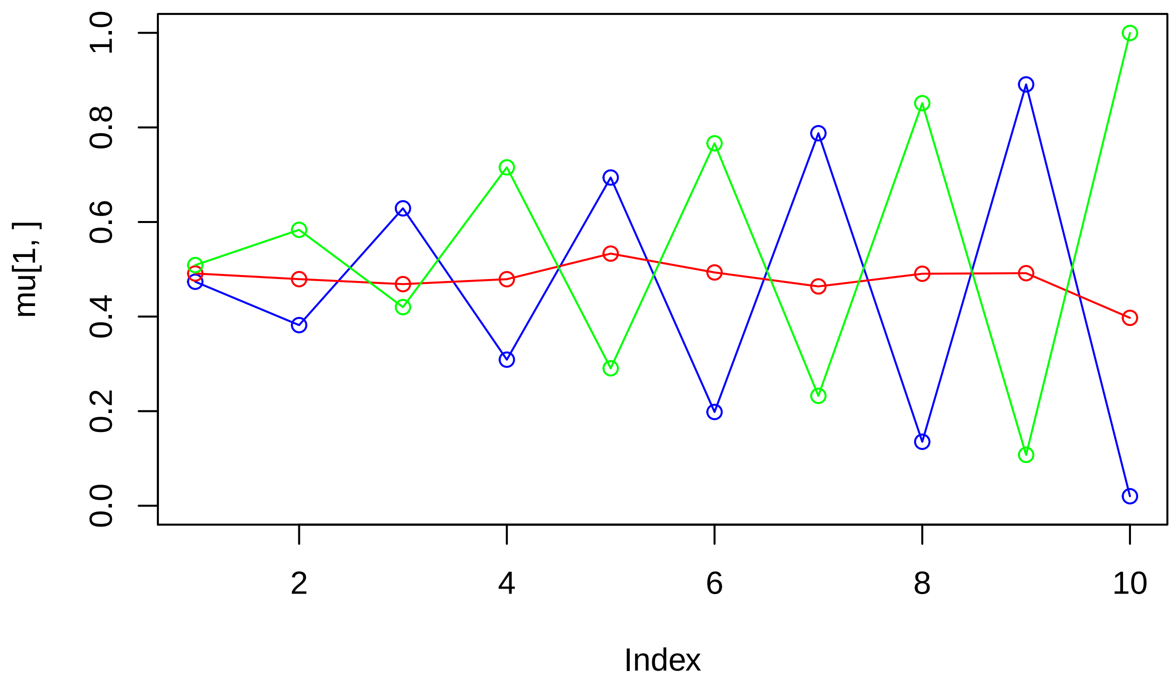

```
## iteration:  56 log likelihood:  -5313.172
```



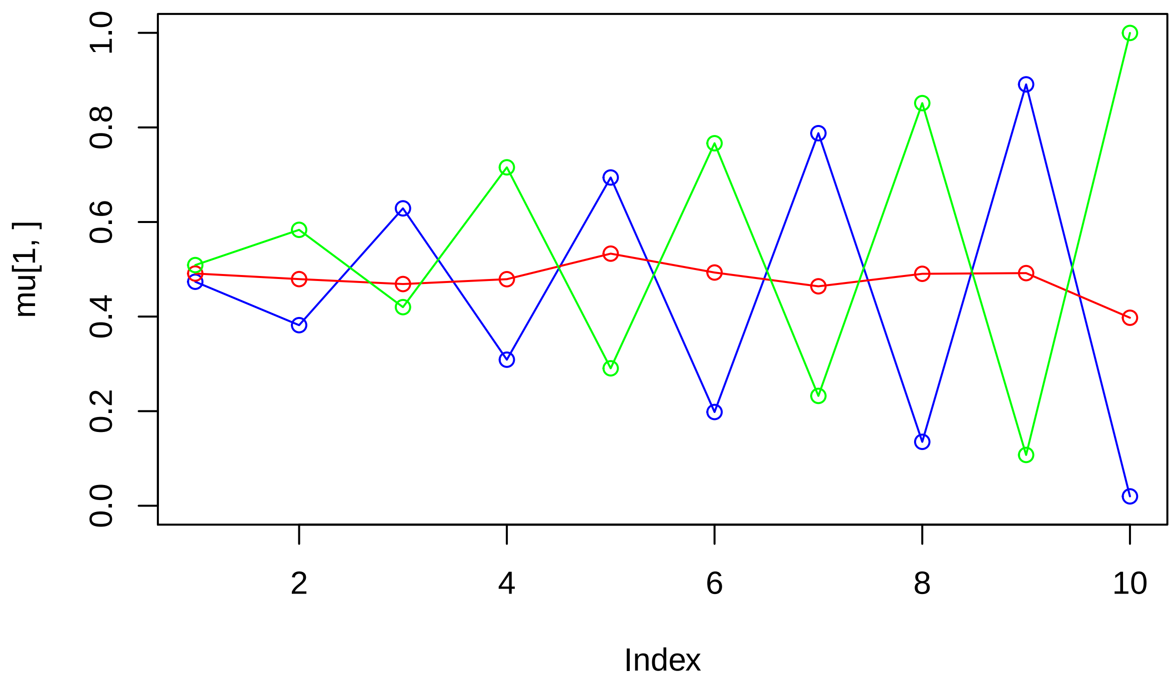

```
## iteration:  57 log likelihood:  -5313.036
```



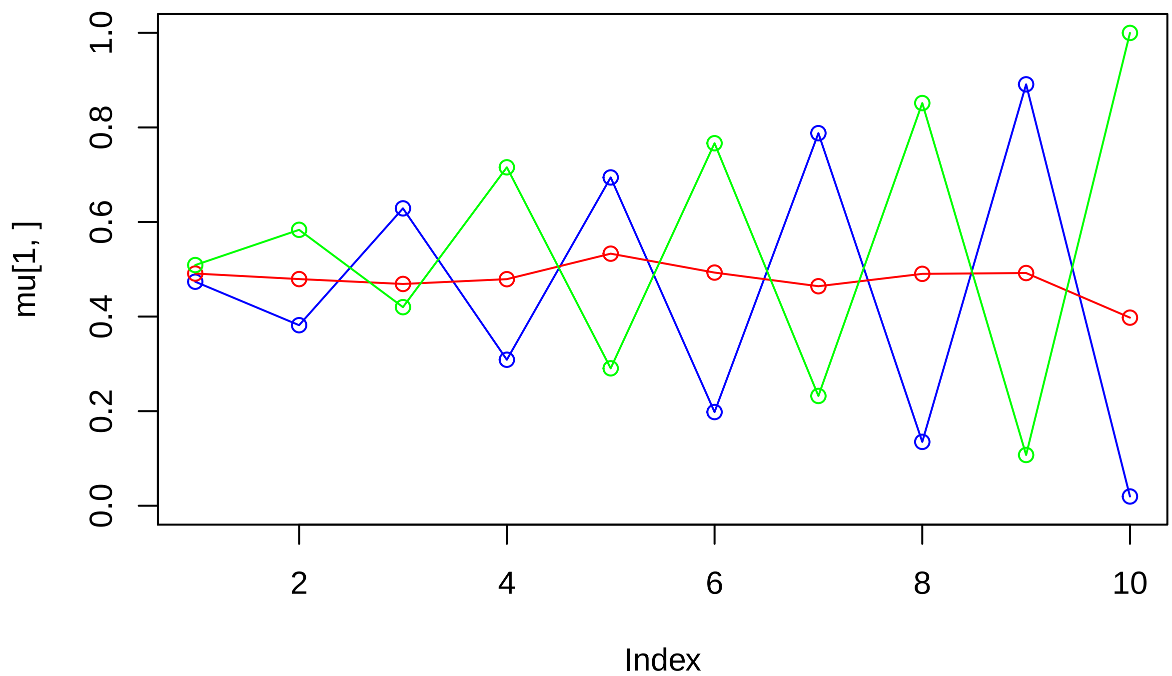

```
## iteration:  58 log likelihood:  -5312.907
```



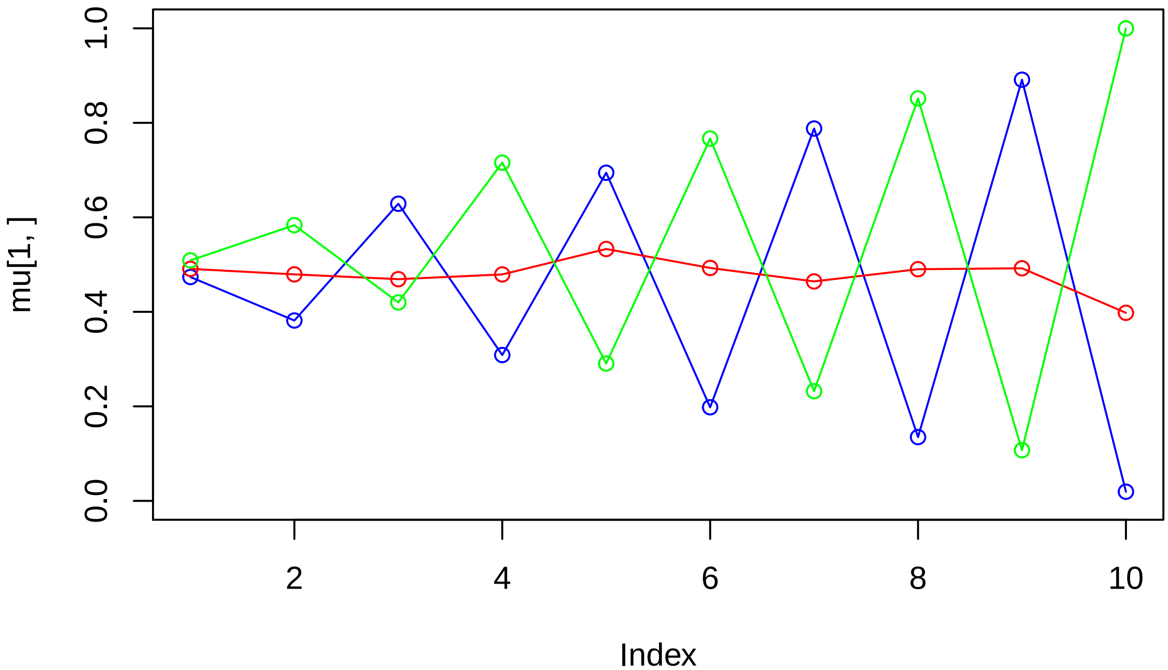

```
## iteration:  59 log likelihood:  -5312.782
```



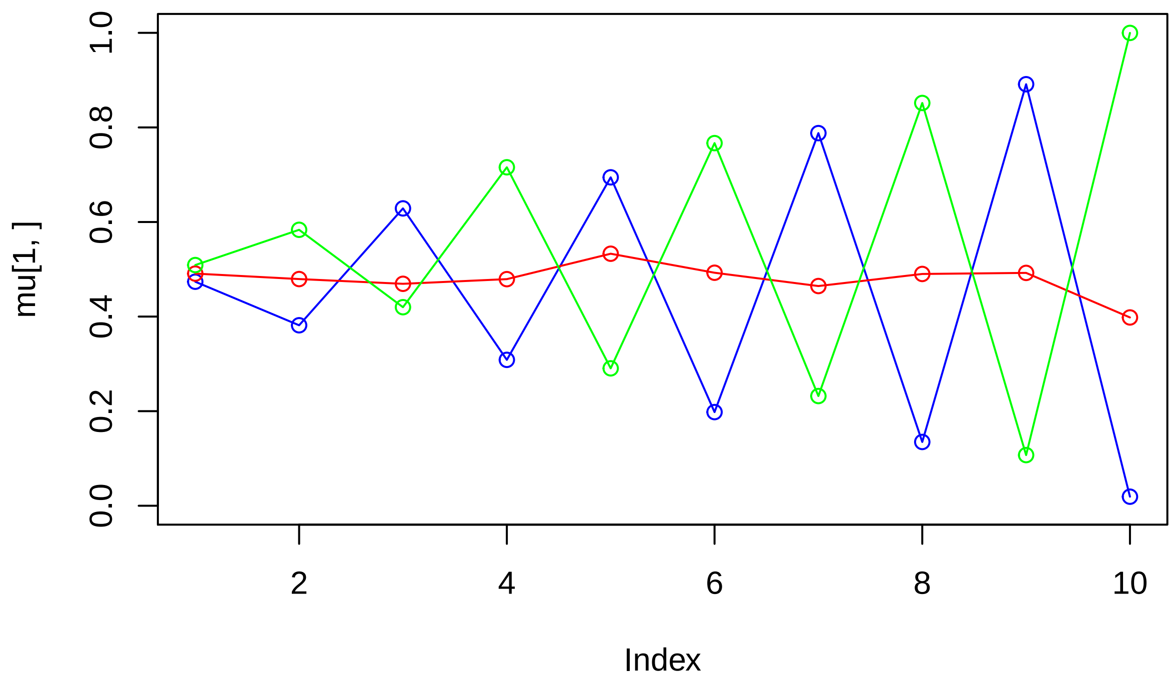

```
## iteration:  60 log likelihood:  -5312.663
```



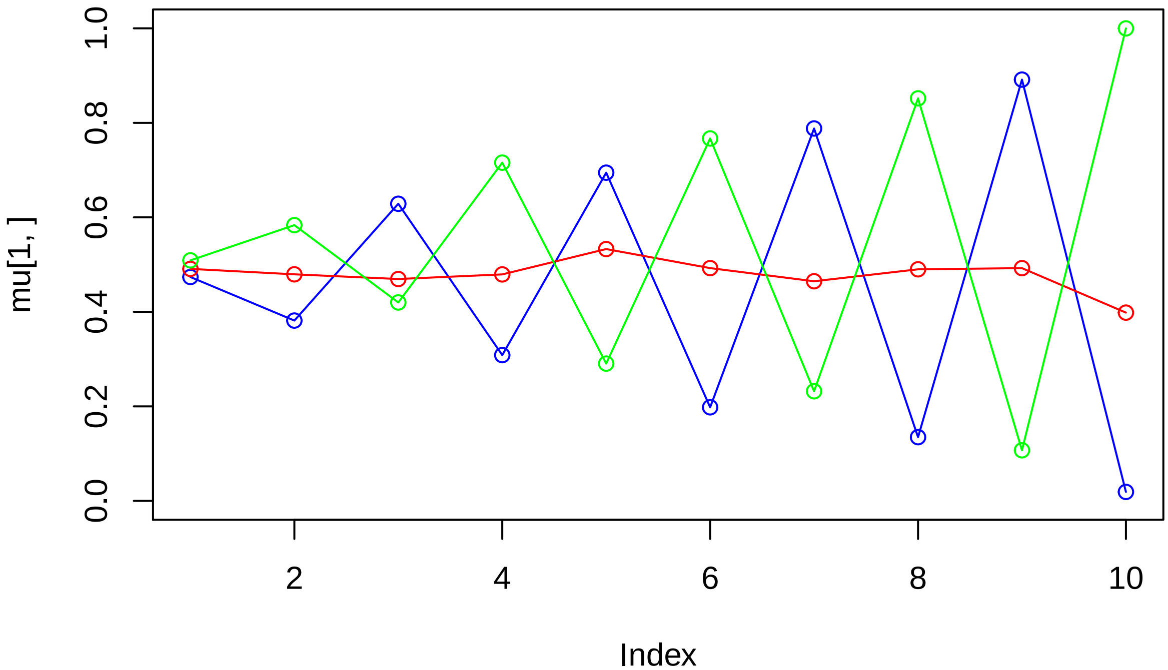

```
## iteration:  61 log likelihood:  -5312.548
```



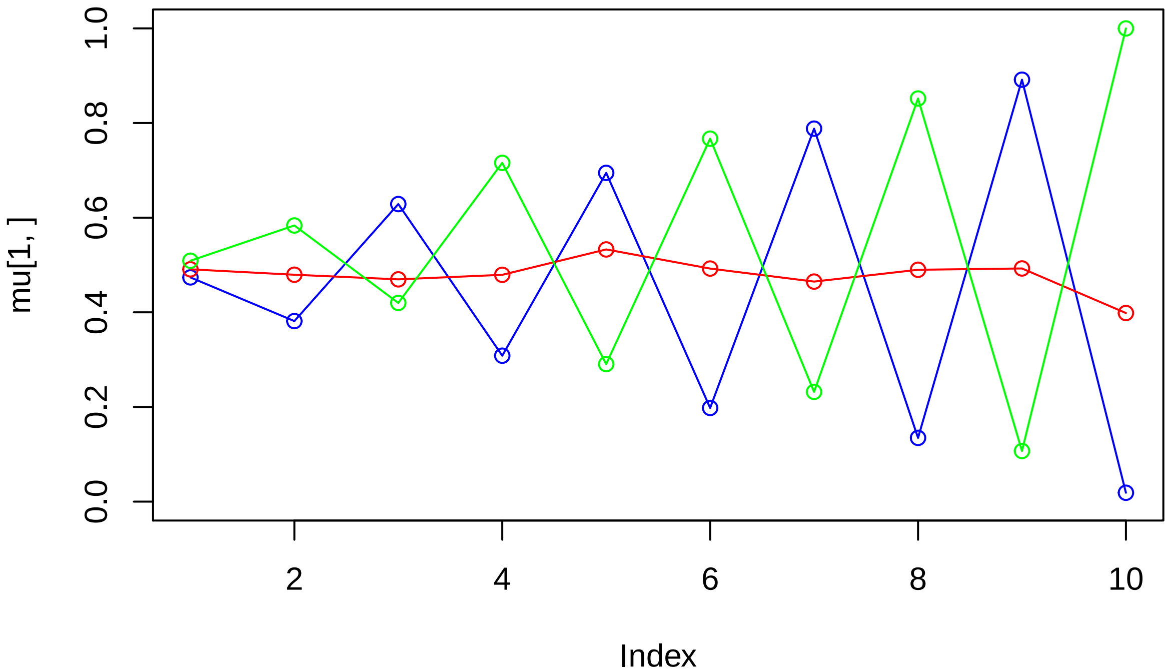

```
## iteration:  62 log likelihood:  -5312.437
```




```
## iteration:  63 log likelihood:  -5312.331
```

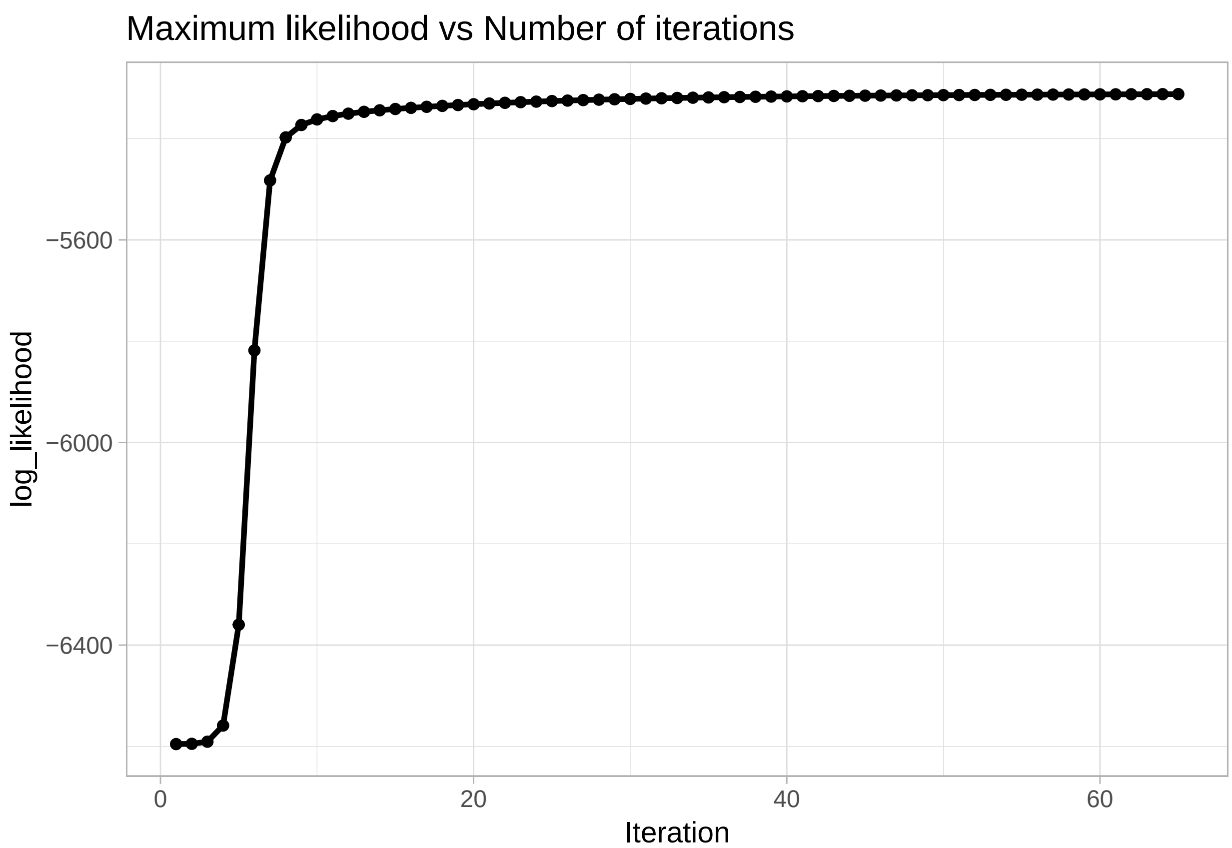



```
## iteration:  64 log likelihood:  -5312.228
```




```
## iteration:  65 log likelihood:  -5312.128
```

```
EM_3$plot
```



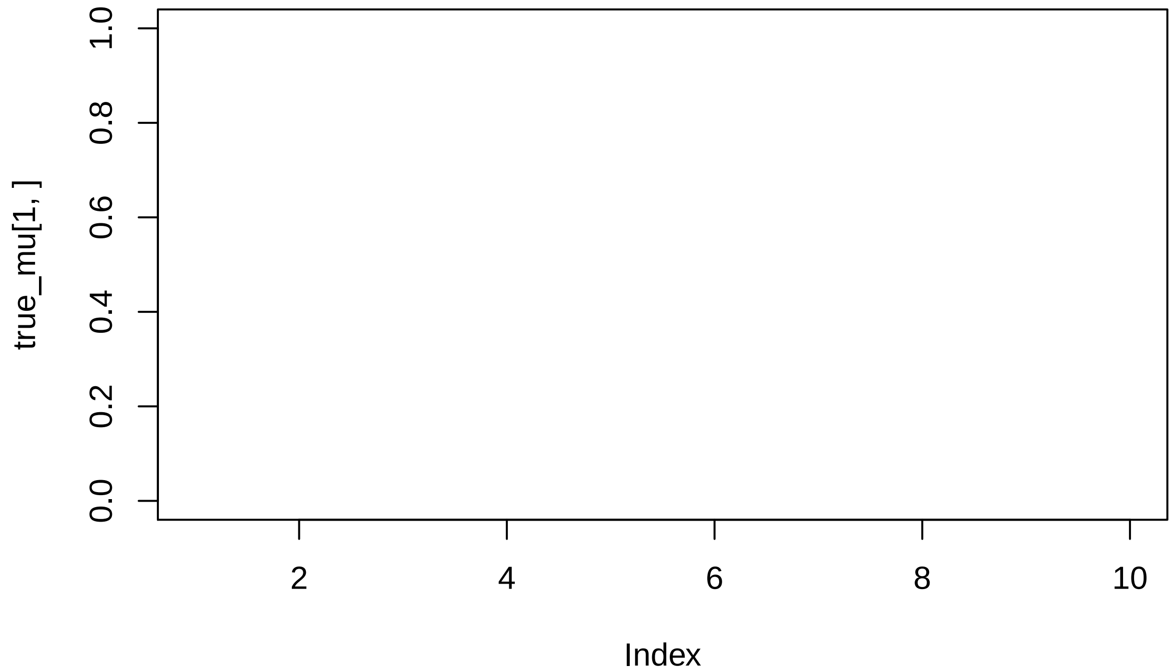

```
EM_3$pi_ML
```

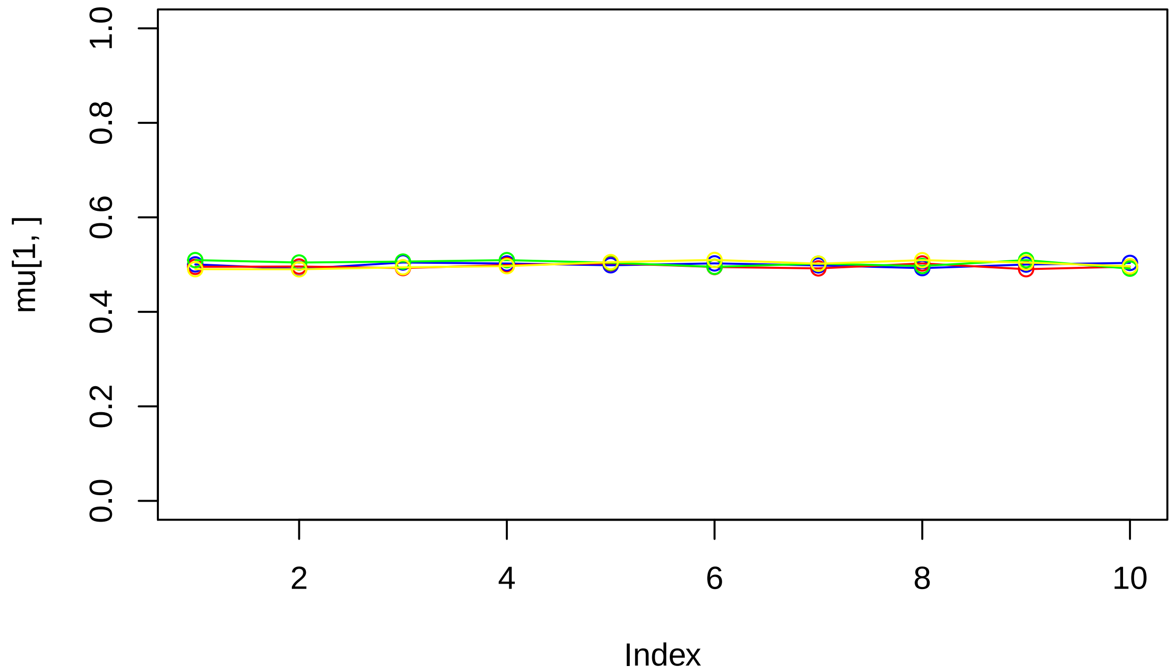
```
## [1] 0.3253079 0.3054648 0.3692273
```

```
EM_3$mu_ML
```

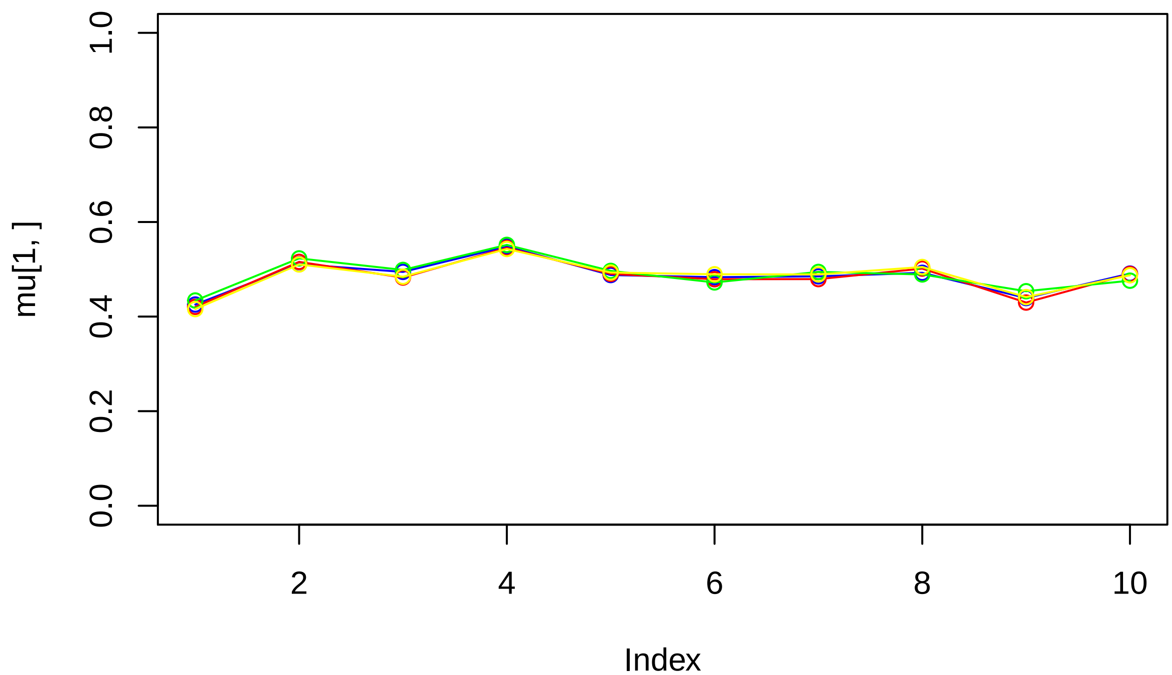
```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4737641 0.3814787 0.6287272 0.3083381 0.6946440 0.1979384 0.7880918
## [2,] 0.4908688 0.4794407 0.4696385 0.4792674 0.5328539 0.4924305 0.4649473
## [3,] 0.5090026 0.5835152 0.4198282 0.7158080 0.2904951 0.7669771 0.2318423
##           [,8]      [,9]      [,10]
## [1,] 0.1347064 0.8915450 0.01871995
## [2,] 0.4899193 0.4926427 0.39851709
## [3,] 0.8518488 0.1069488 0.99986452
```

```
EM_4 <- em_mat(4)
```

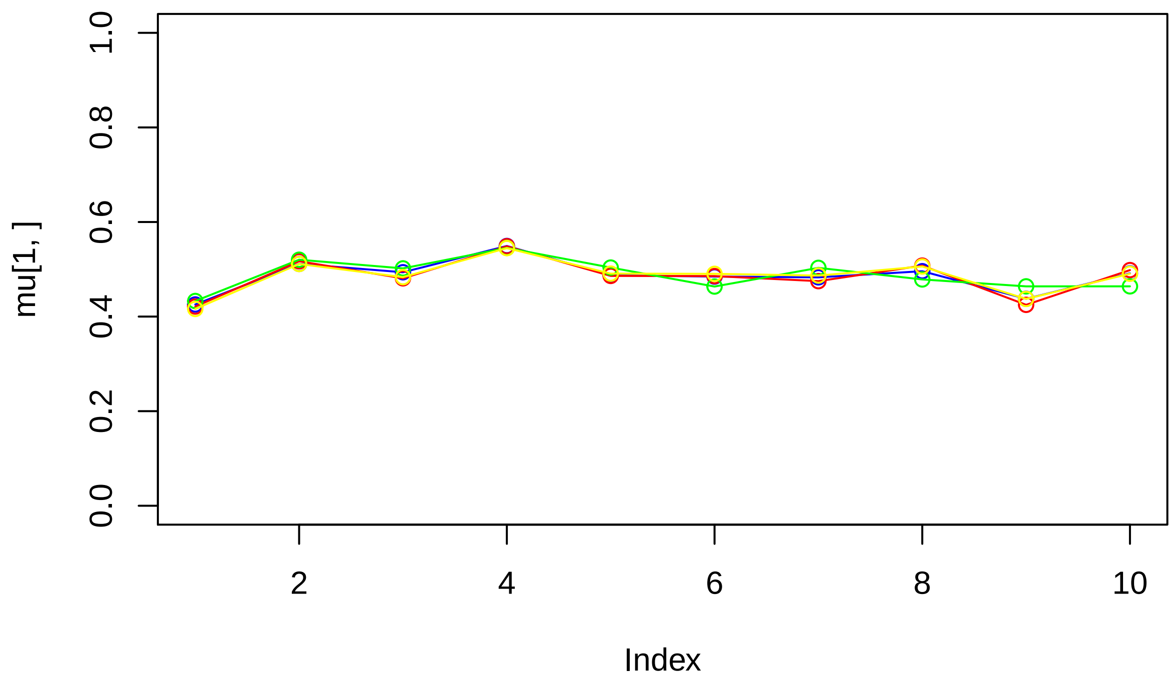

true_mu[1,]
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Index



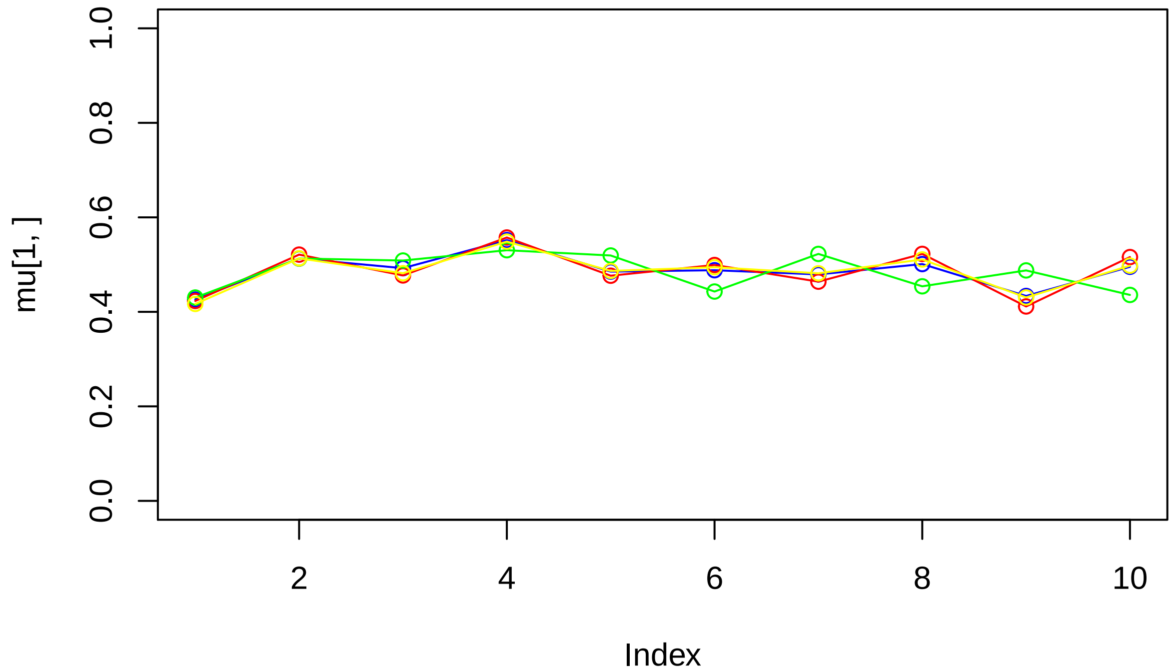

```
## iteration:  1 log likelihood:  -6655.473
```



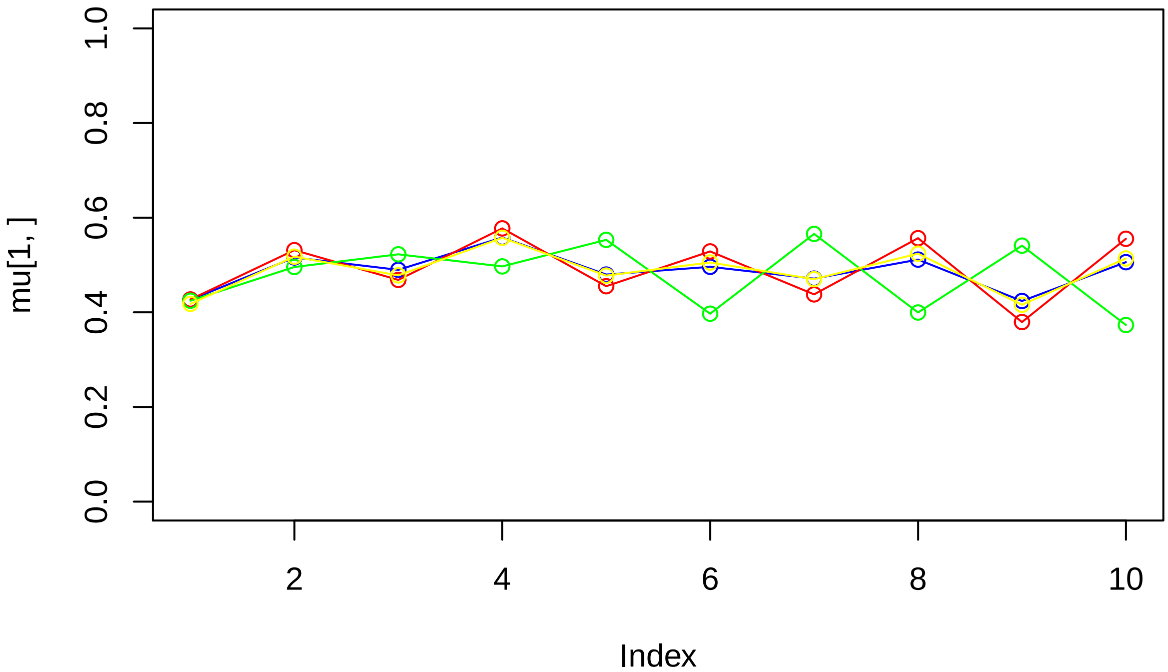

```
## iteration:  2 log likelihood:  -6654.384
```



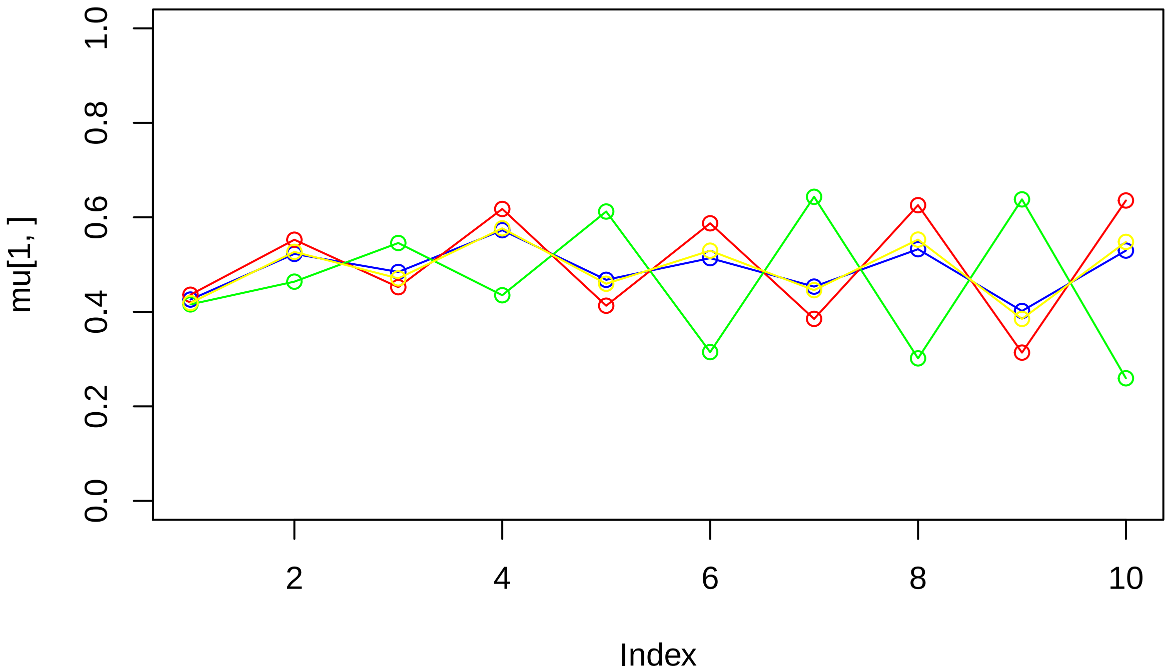

```
## iteration:  3 log likelihood:  -6648.24
```



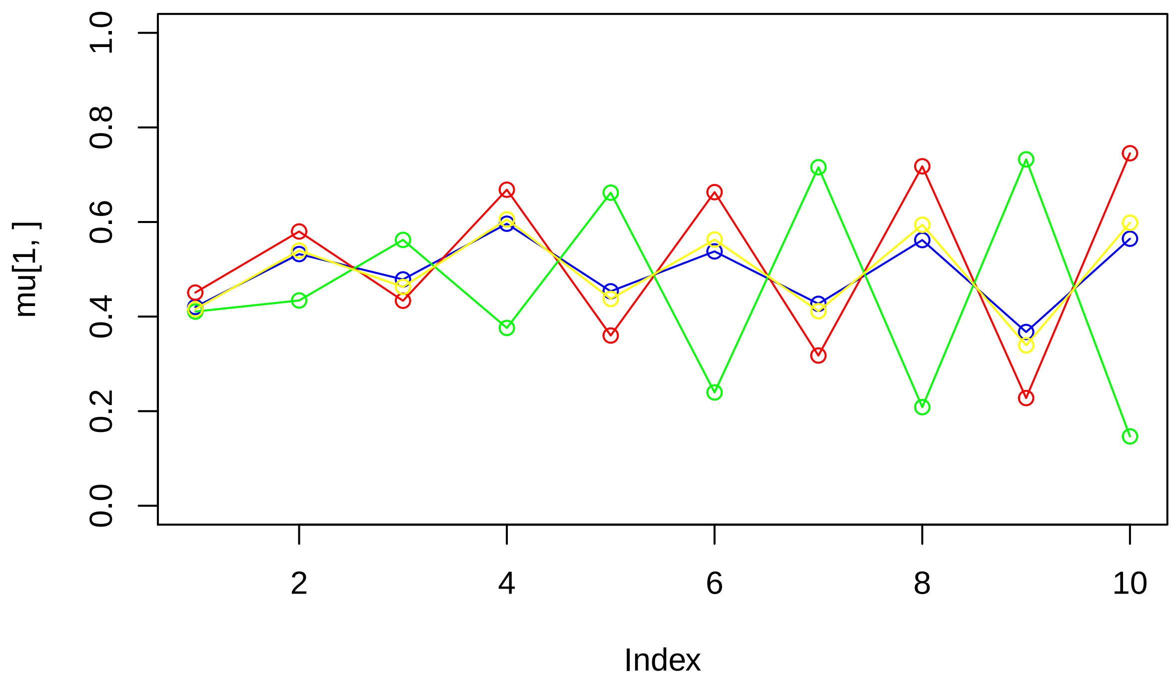

```
## iteration:  4 log likelihood:  -6615.956
```



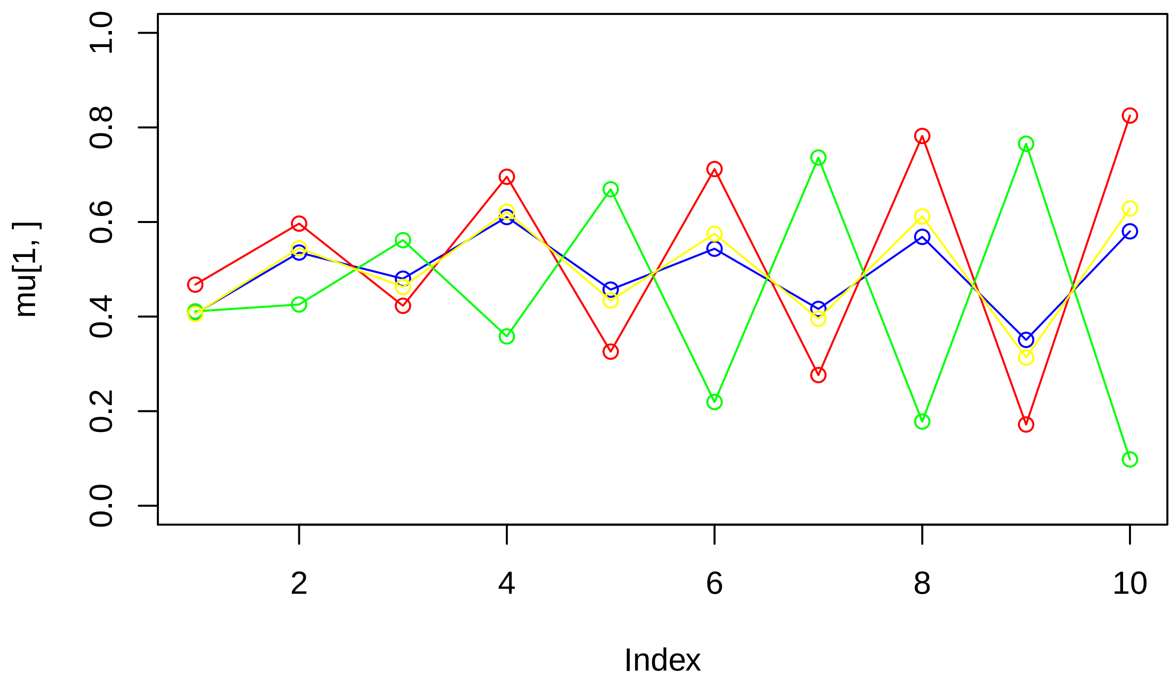

```
## iteration:  5 log likelihood:  -6476.236
```



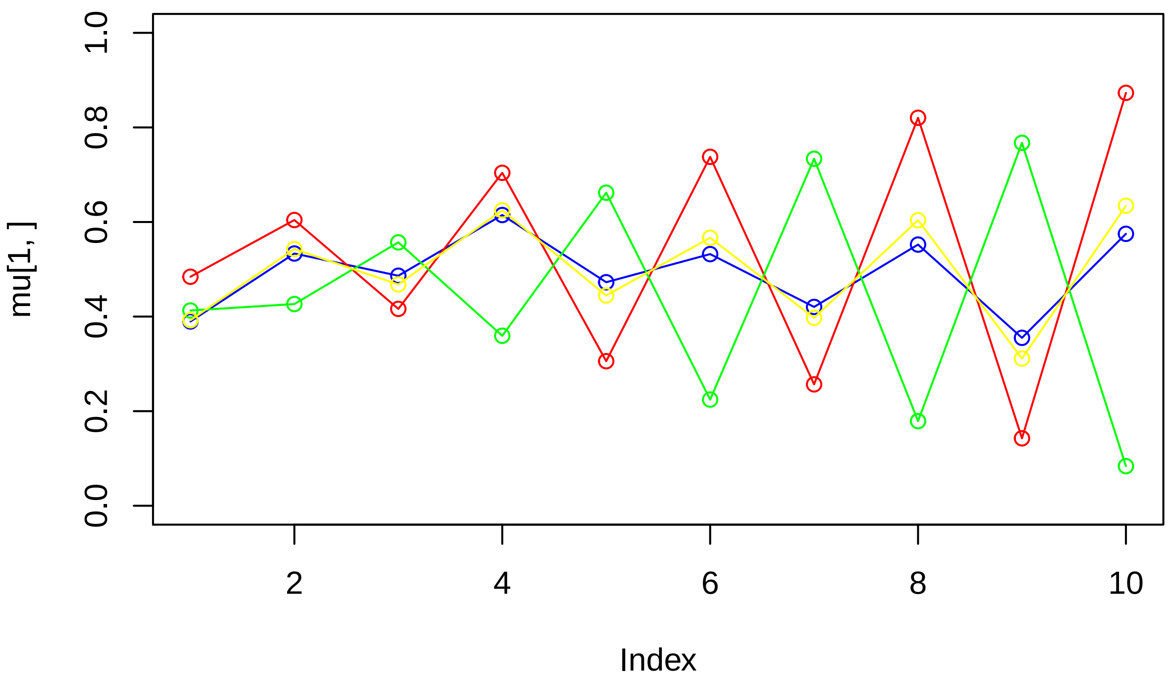

```
## iteration:  6 log likelihood:  -6163.491
```



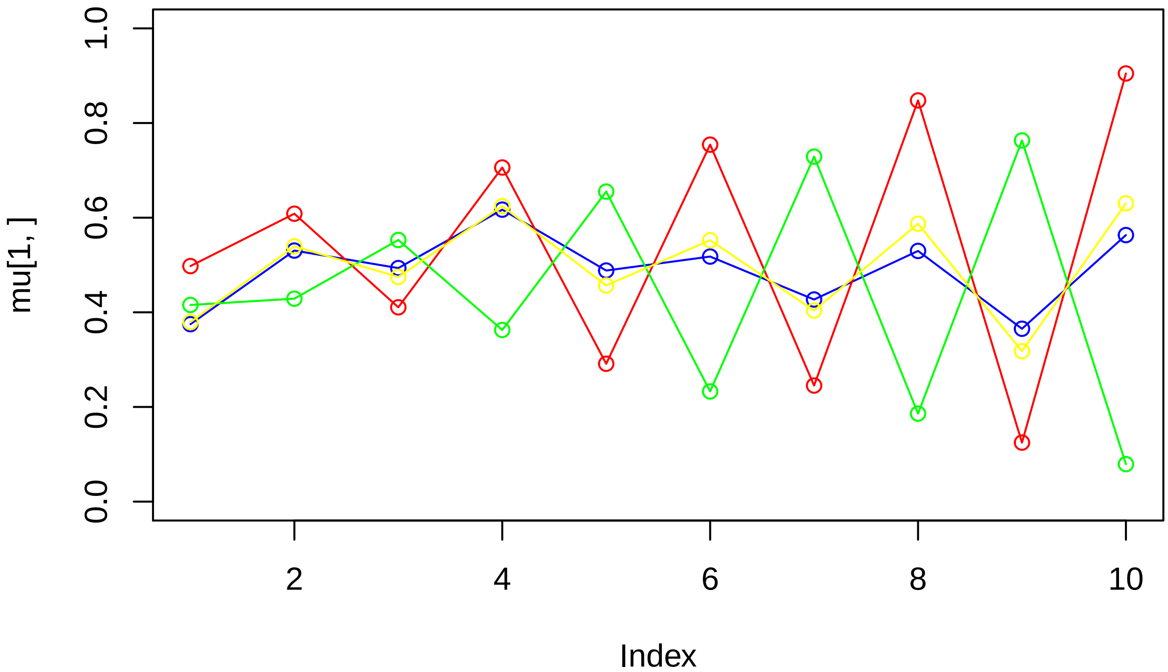

```
## iteration:  7 log likelihood:  -5923.696
```



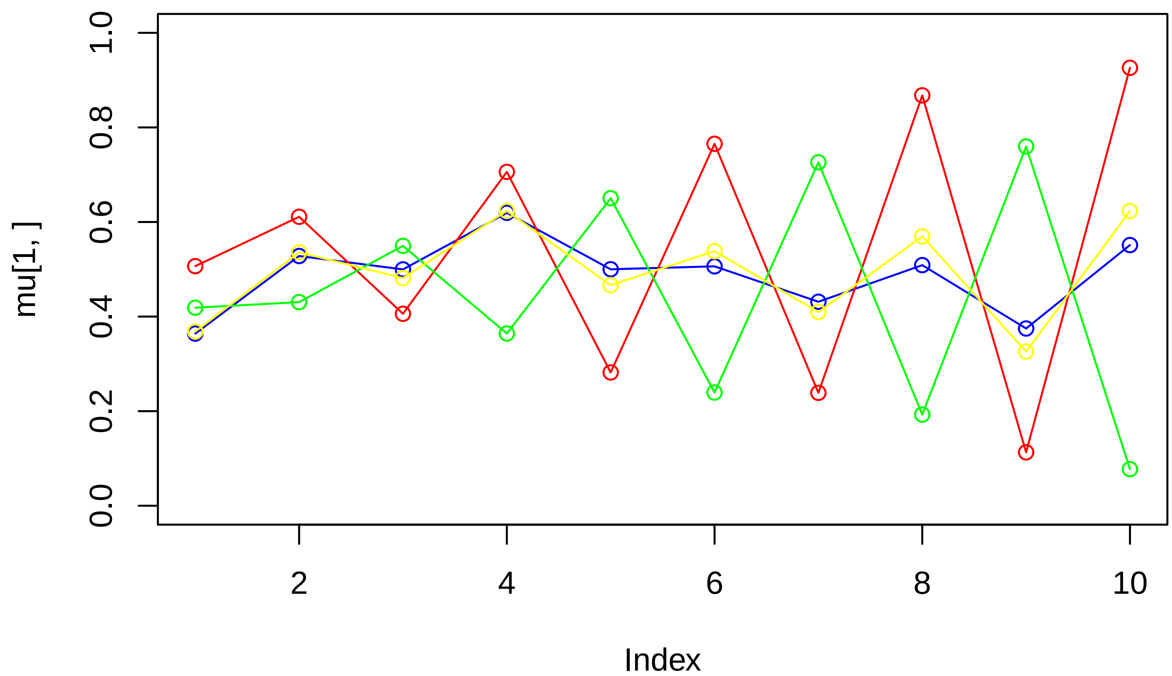

```
## iteration:  8 log likelihood:  -5823.047
```



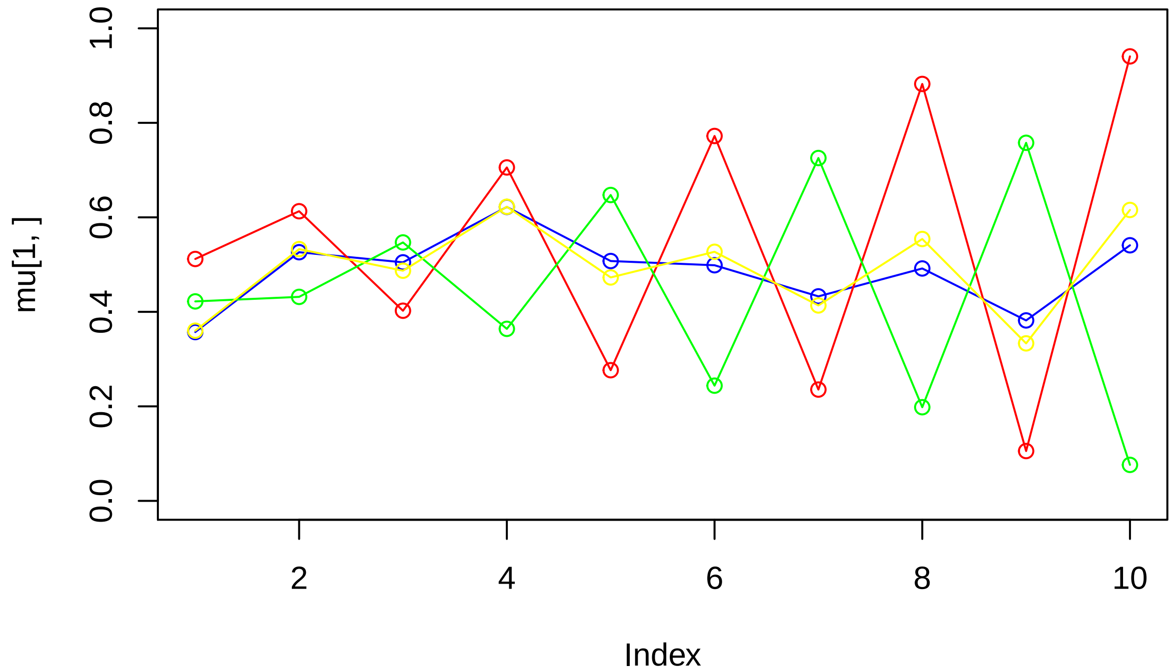

```
## iteration:  9 log likelihood:  -5773.59
```



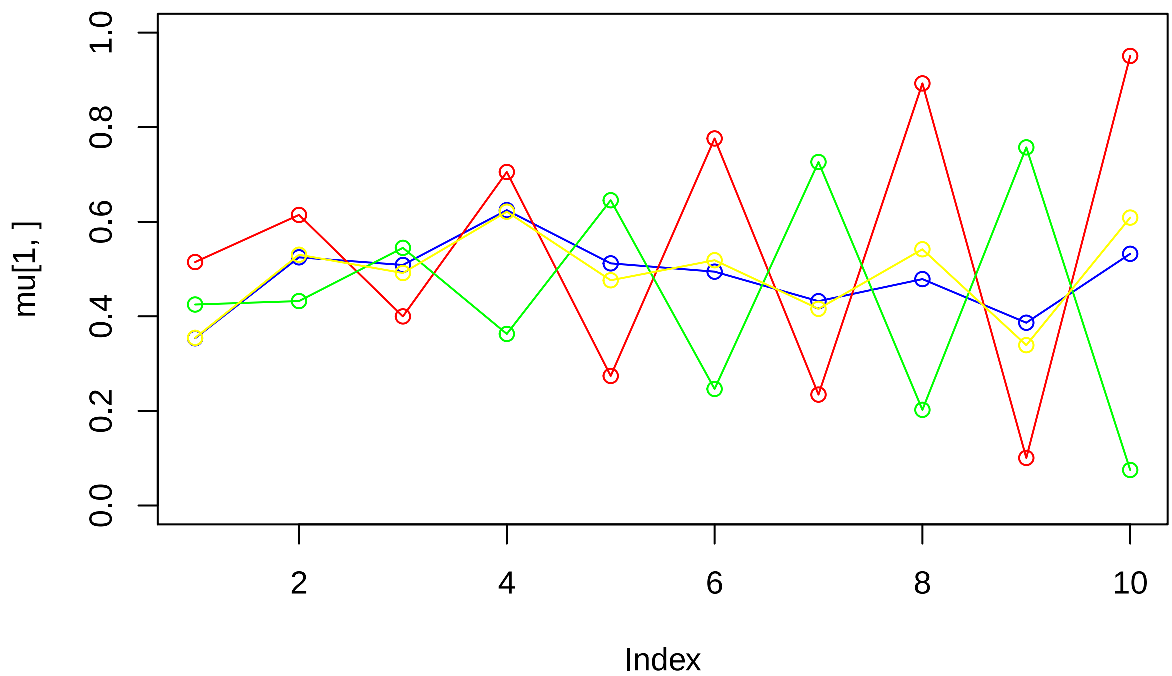

```
## iteration:  10 log likelihood:  -5743.739
```



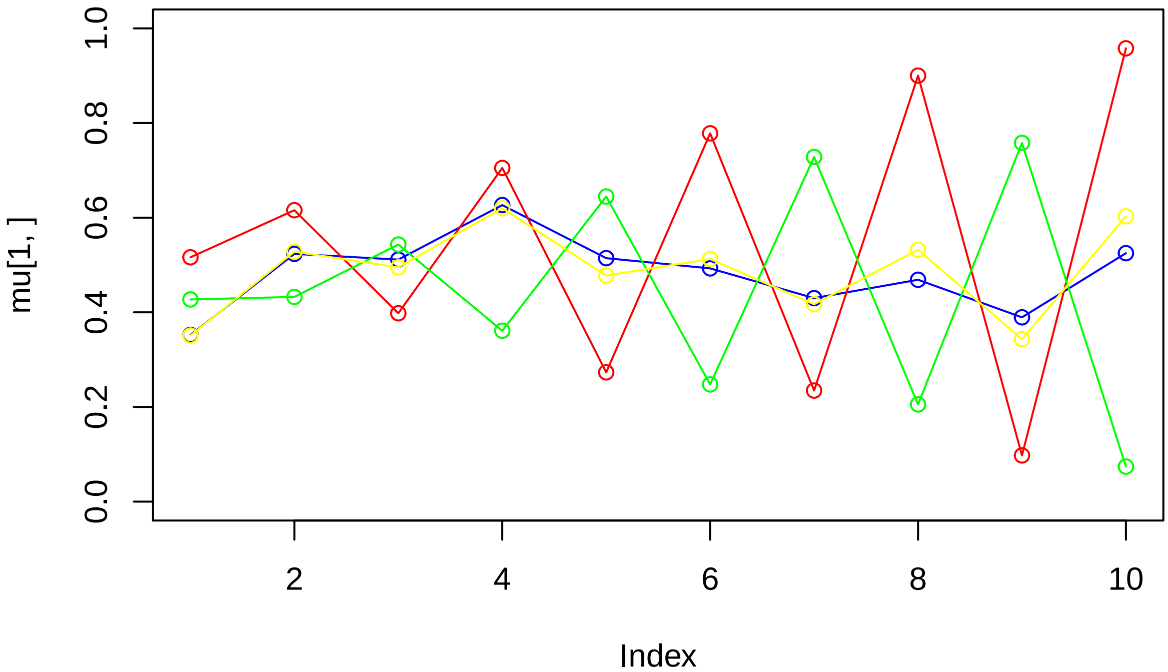

```
## iteration:  11 log likelihood:  -5725.322
```



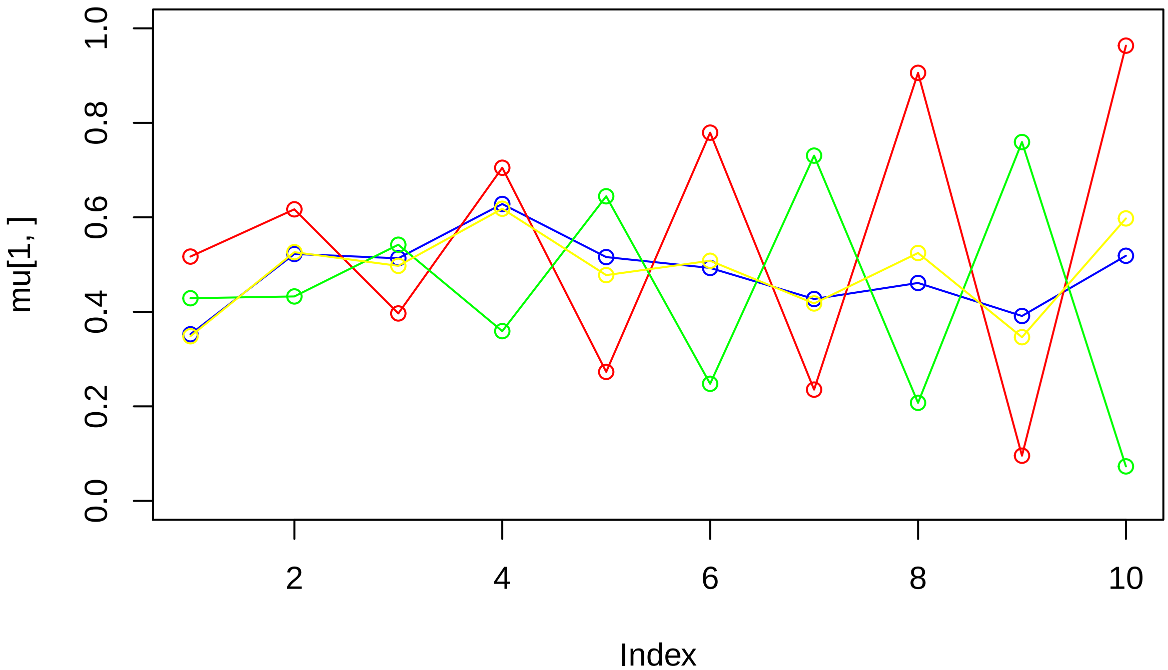

```
## iteration:  12 log likelihood:  -5714.141
```



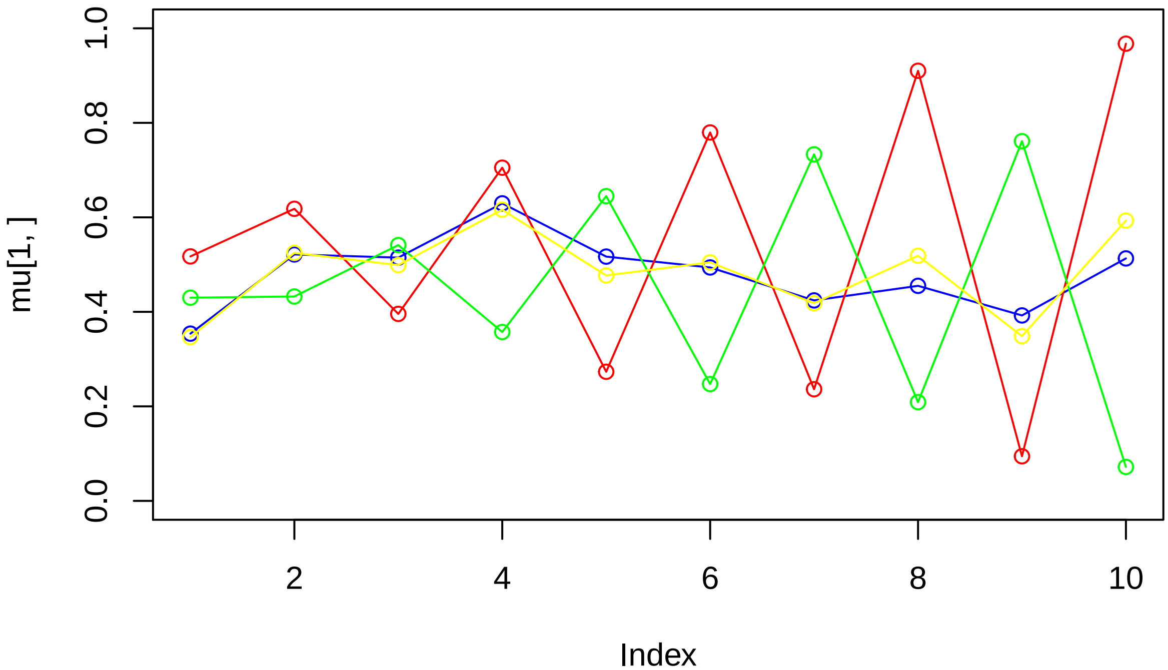

```
## iteration:  13 log likelihood:  -5707.236
```



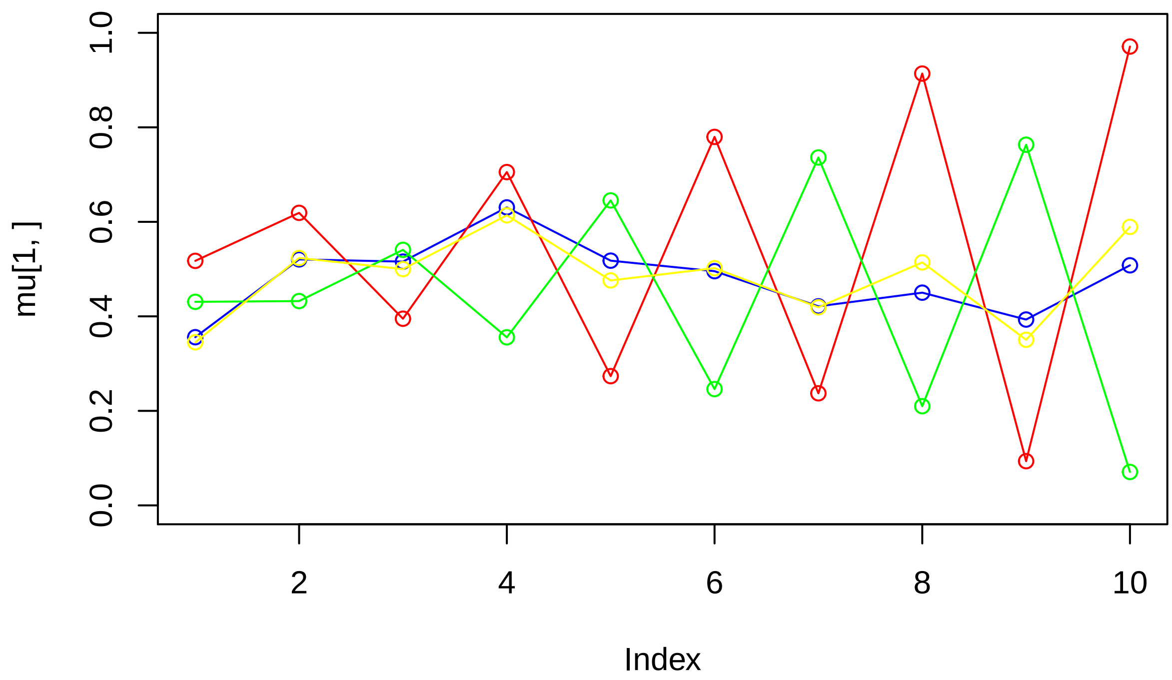

```
## iteration:  14 log likelihood:  -5702.736
```



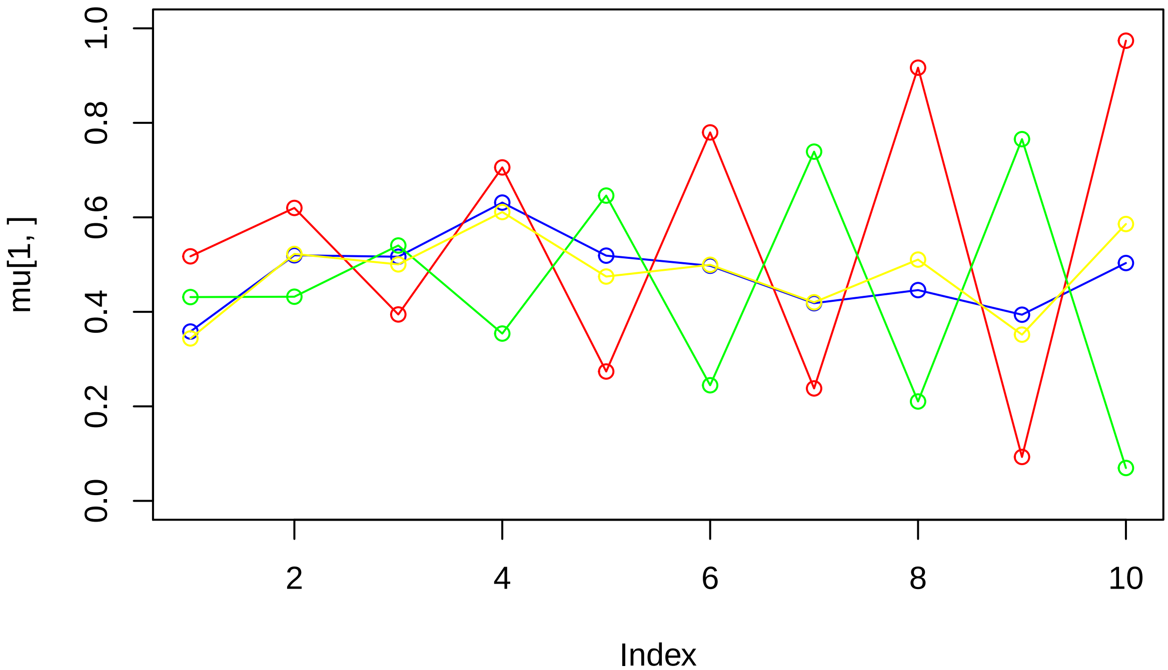

```
## iteration:  15 log likelihood:  -5699.586
```



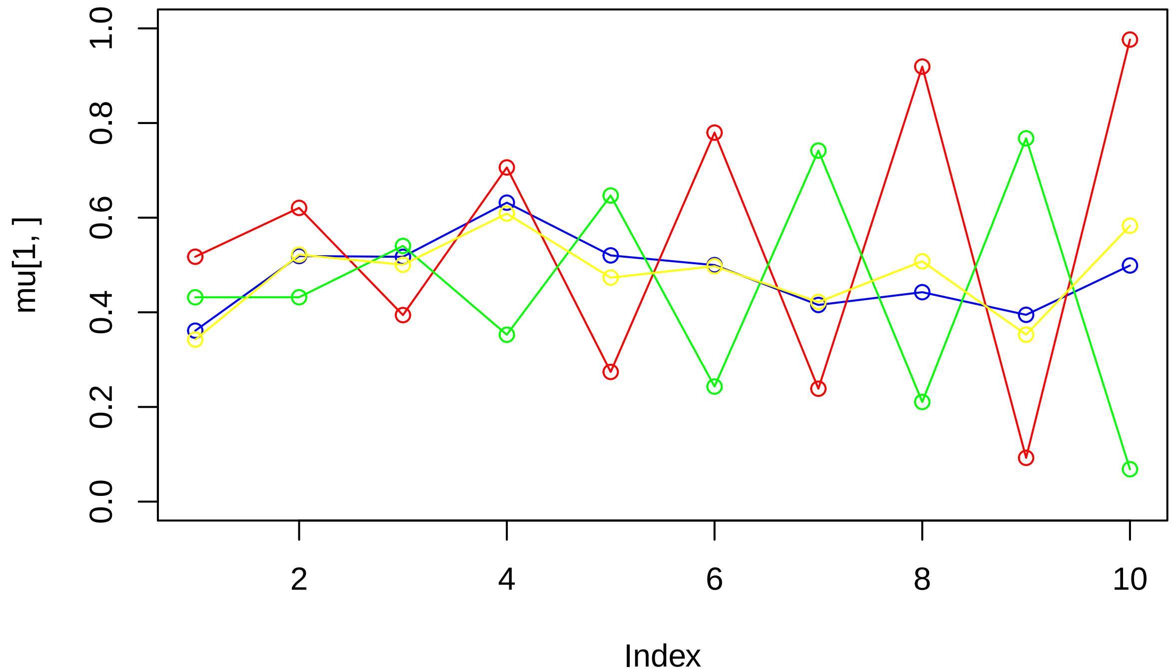

```
## iteration:  16 log likelihood:  -5697.217
```



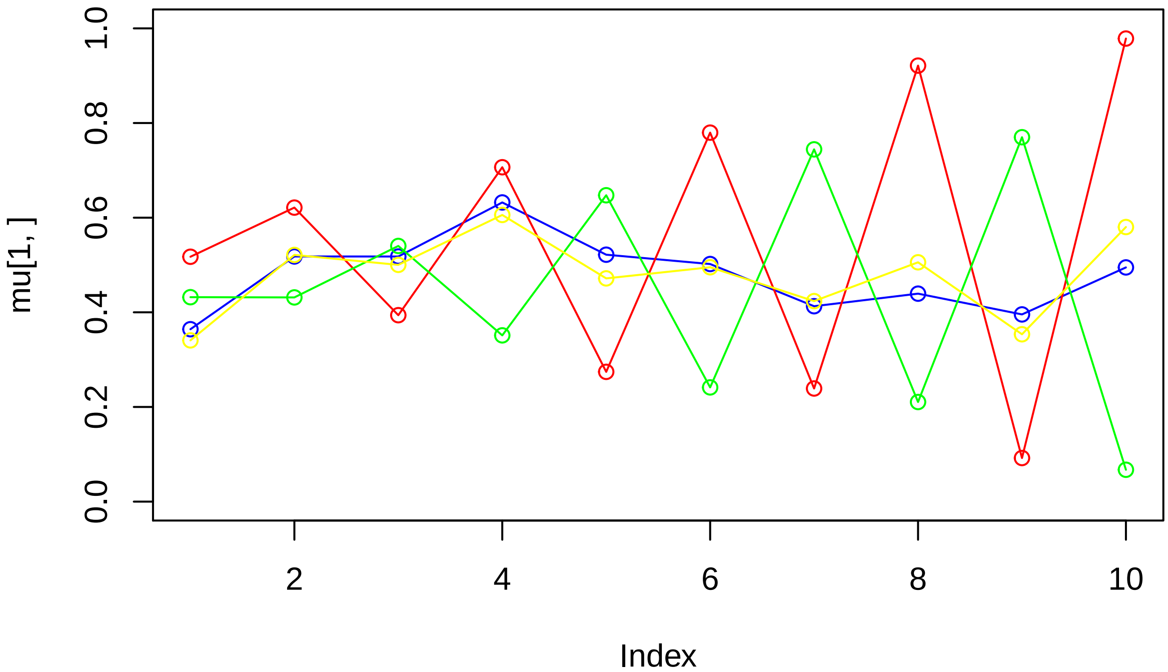

```
## iteration:  17 log likelihood:  -5695.317
```



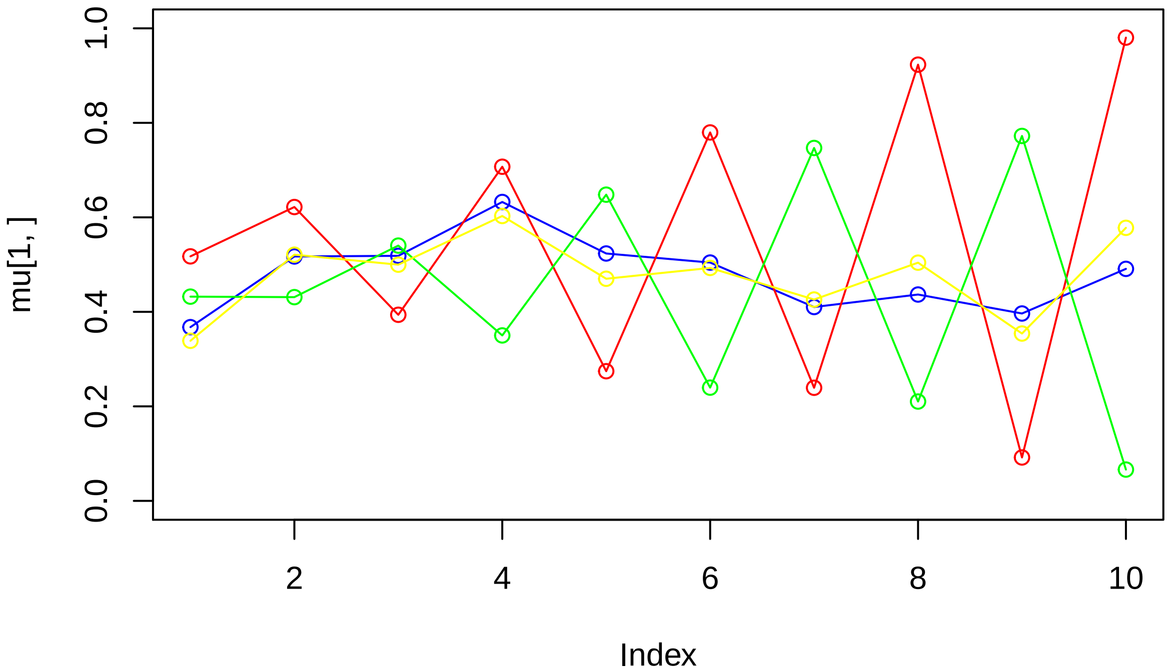

```
## iteration:  18 log likelihood:  -5693.707
```



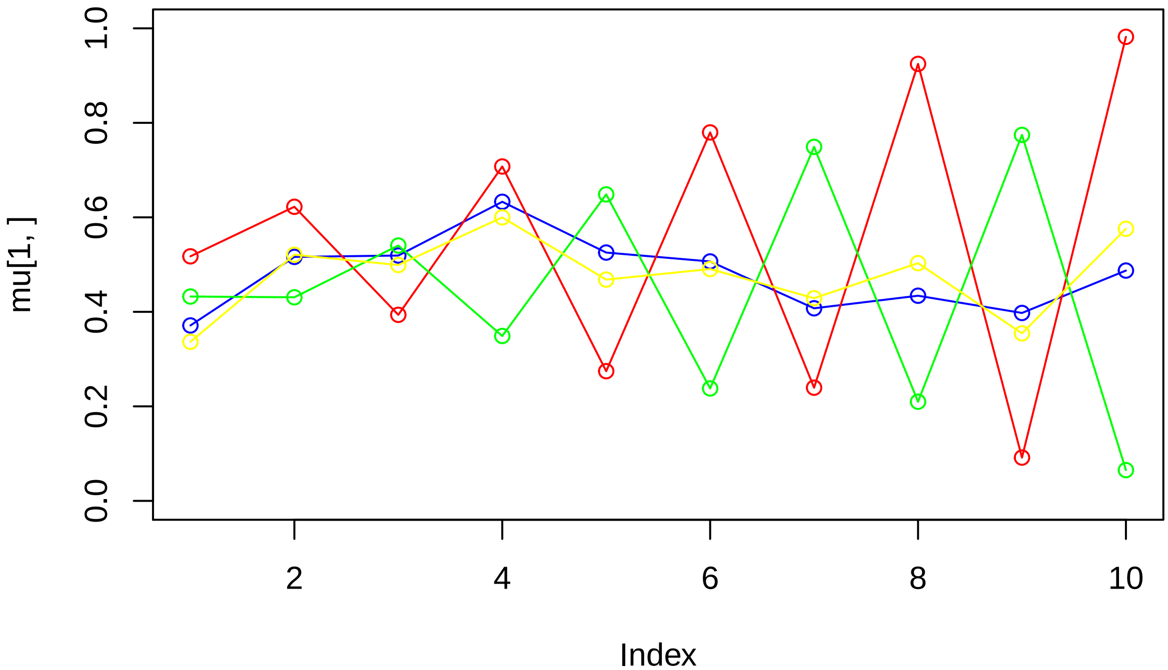

```
## iteration:  19 log likelihood:  -5692.276
```



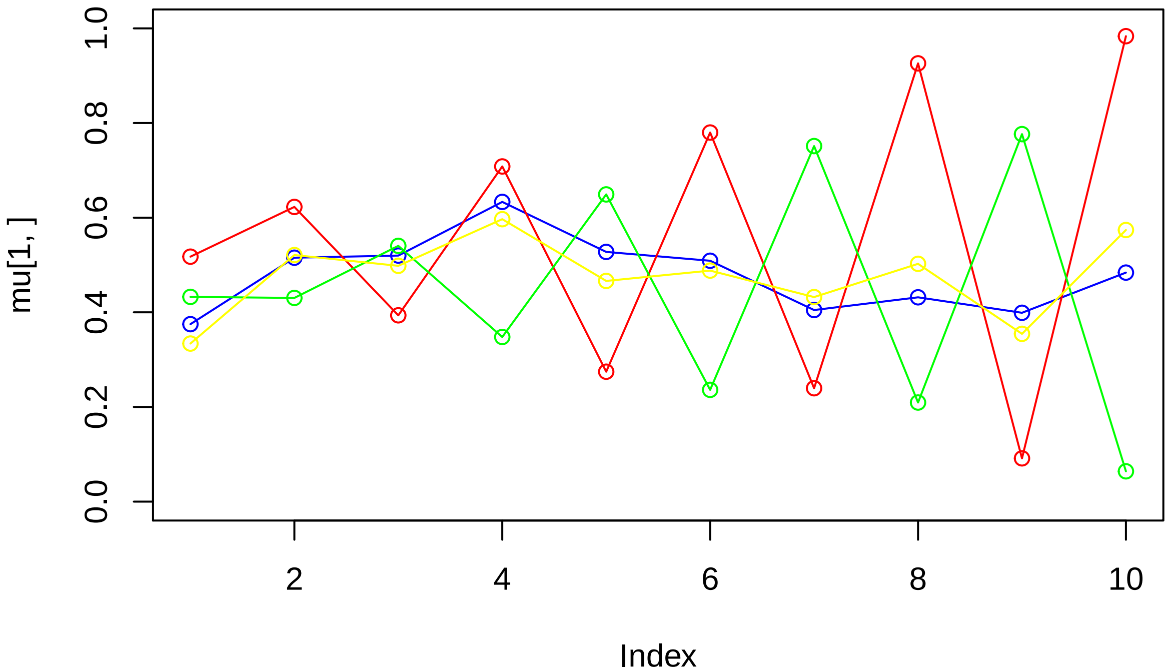

```
## iteration:  20 log likelihood:  -5690.95
```



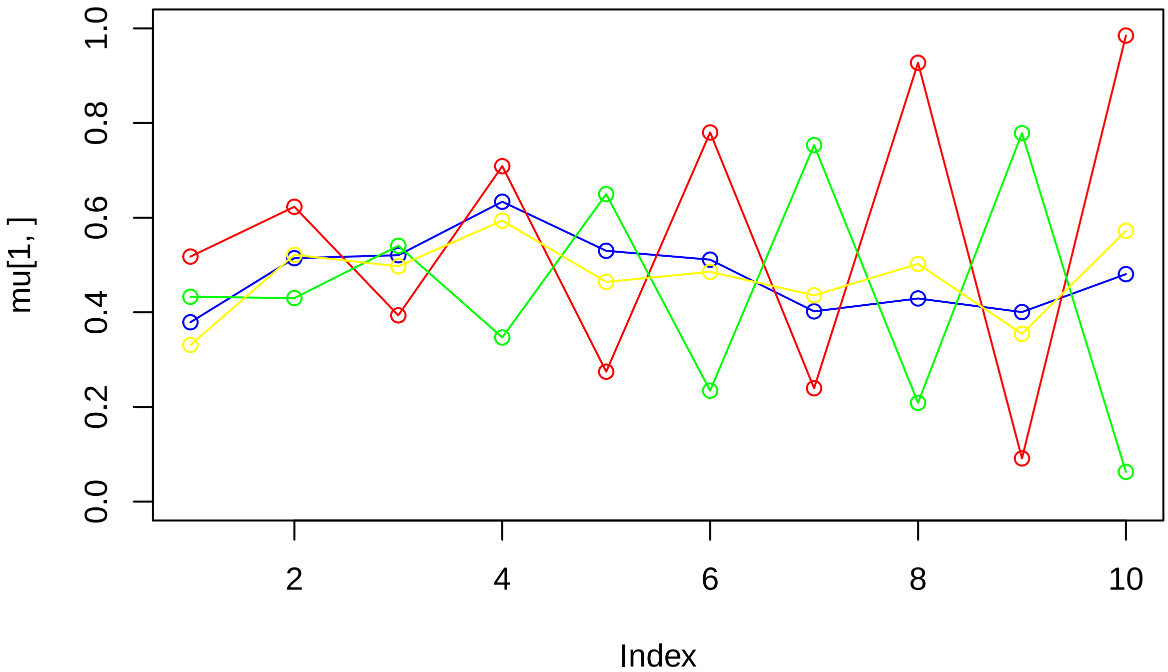

```
## iteration:  21 log likelihood:  -5689.674
```



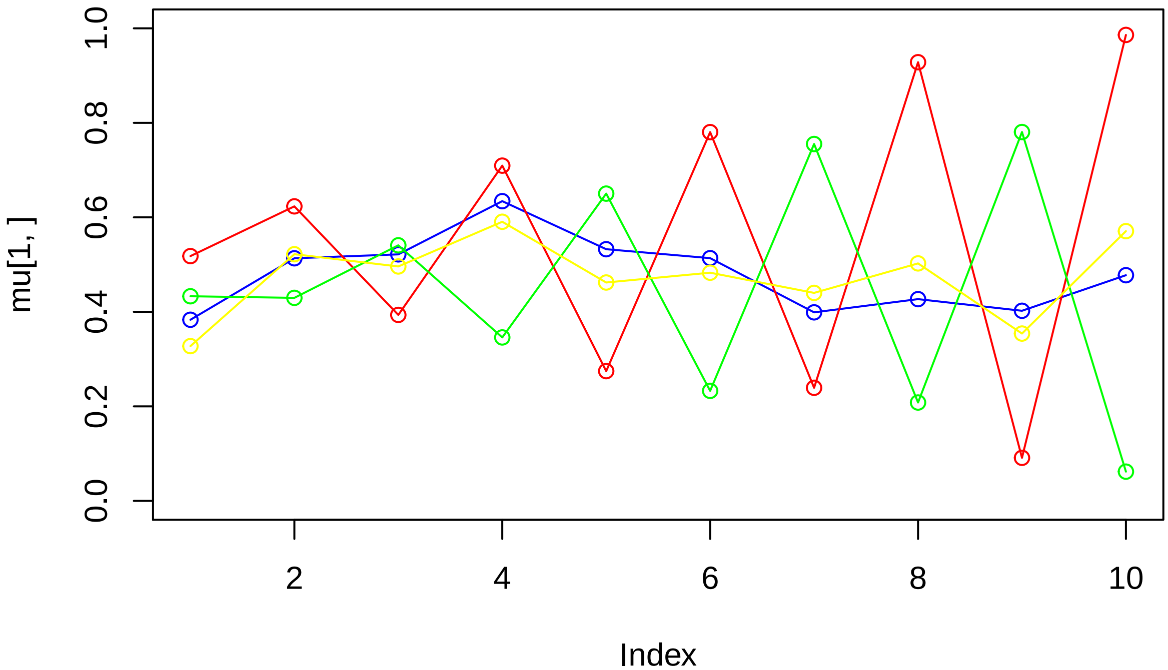

```
## iteration:  22 log likelihood:  -5688.403
```



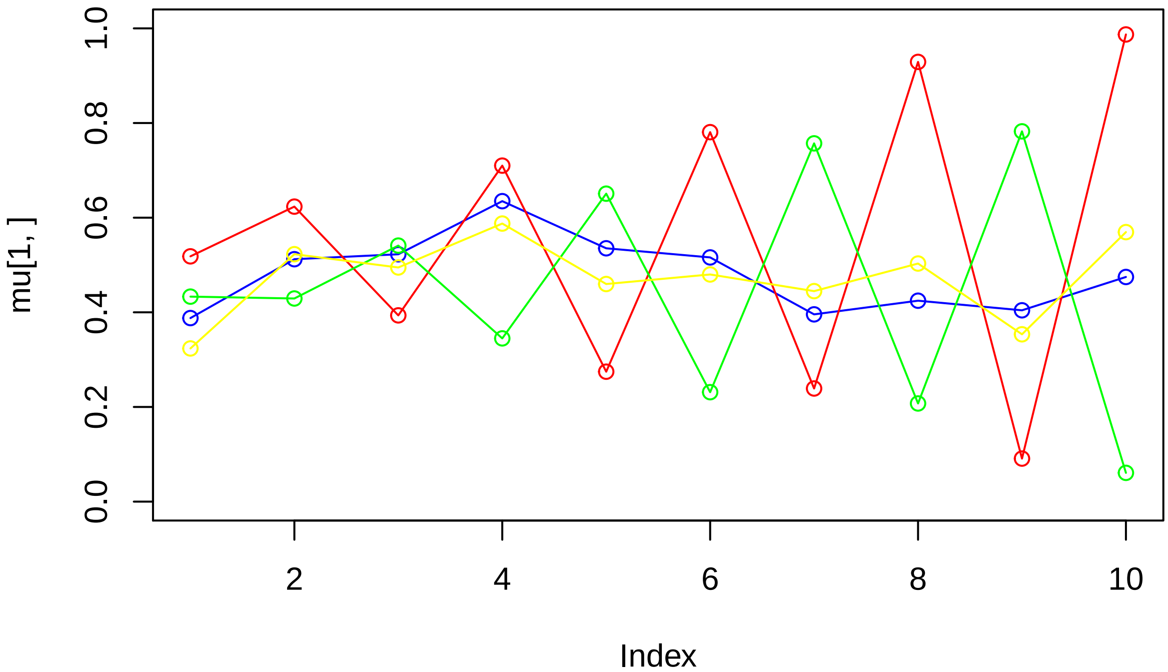

```
## iteration:  23 log likelihood:  -5687.099
```



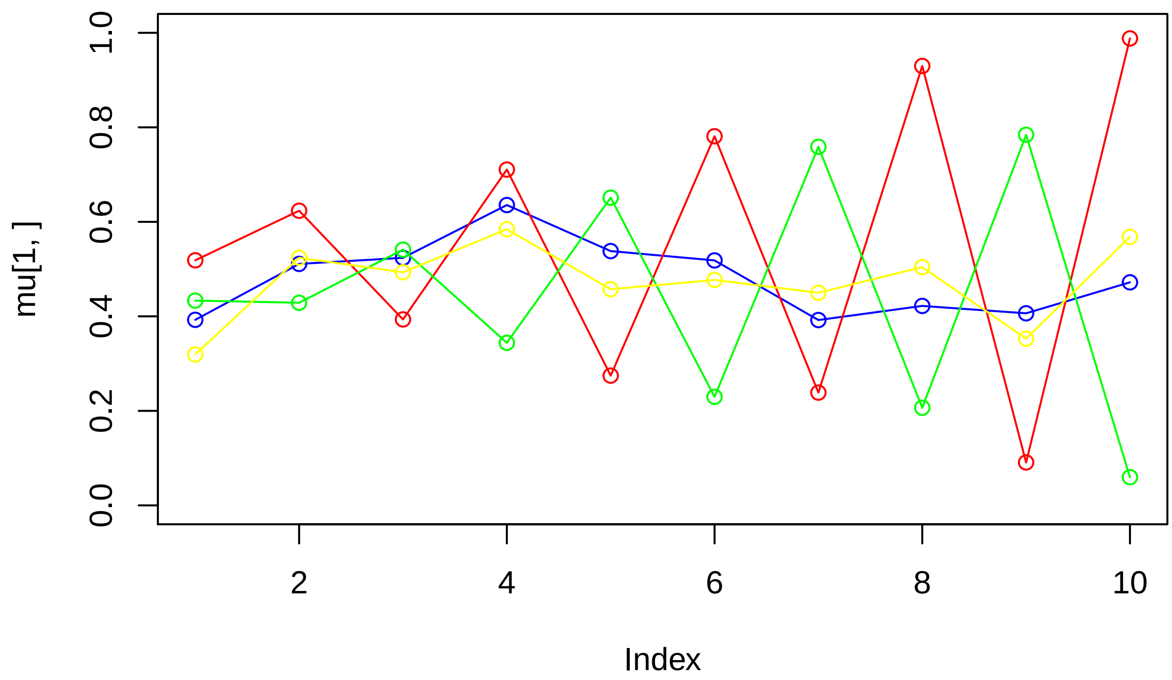

```
## iteration:  24 log likelihood:  -5685.724
```



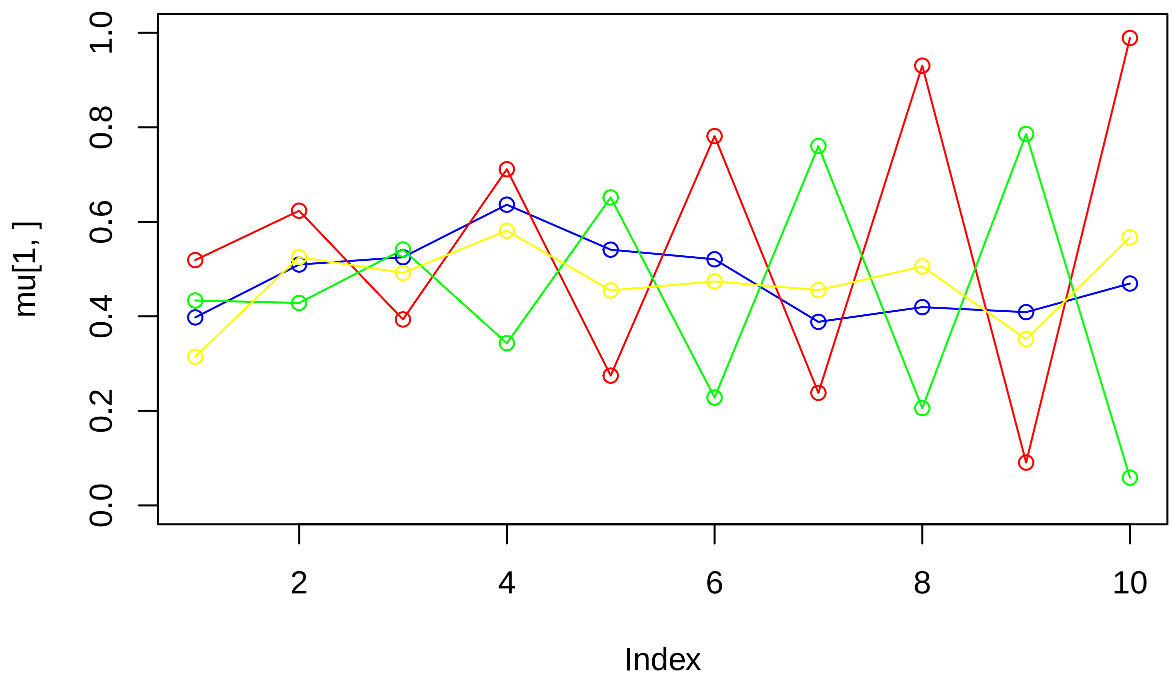

```
## iteration:  25 log likelihood:  -5684.246
```



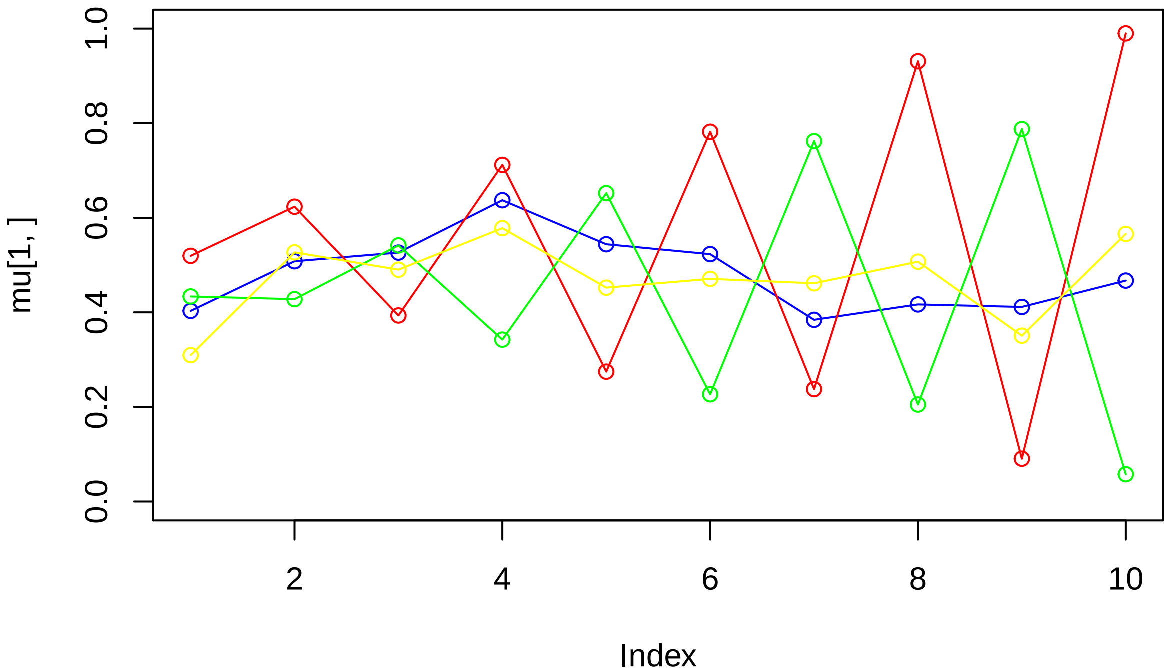

```
## iteration:  26 log likelihood:  -5682.631
```



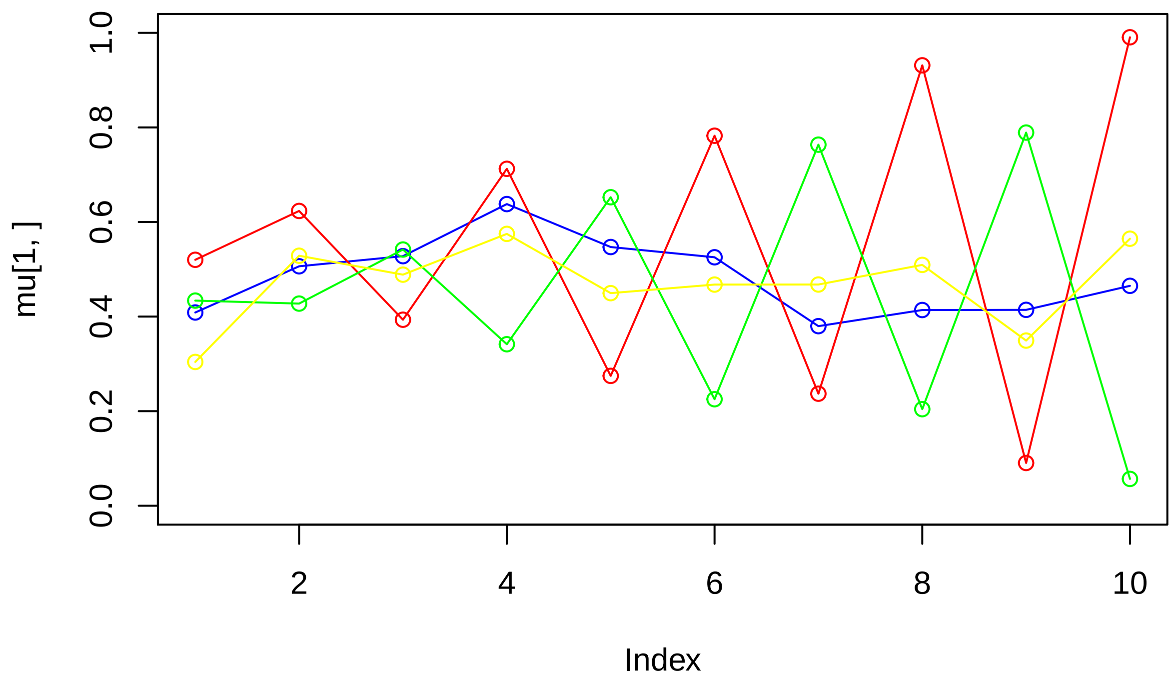

```
## iteration:  27 log likelihood:  -5680.847
```



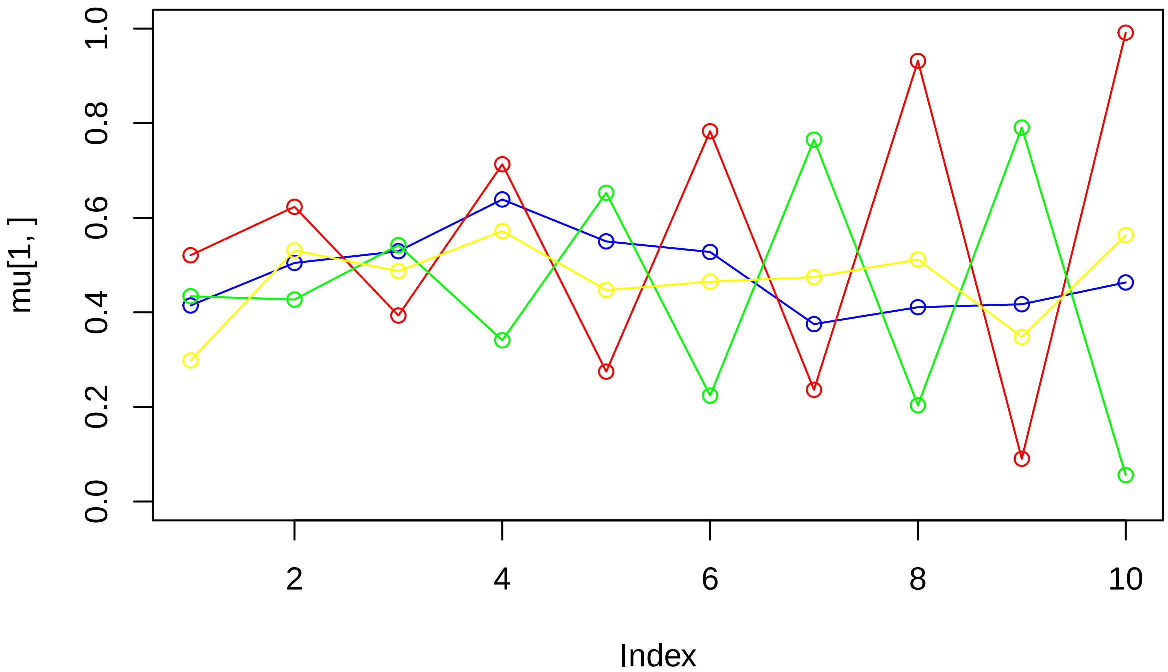

```
## iteration:  28 log likelihood:  -5678.867
```



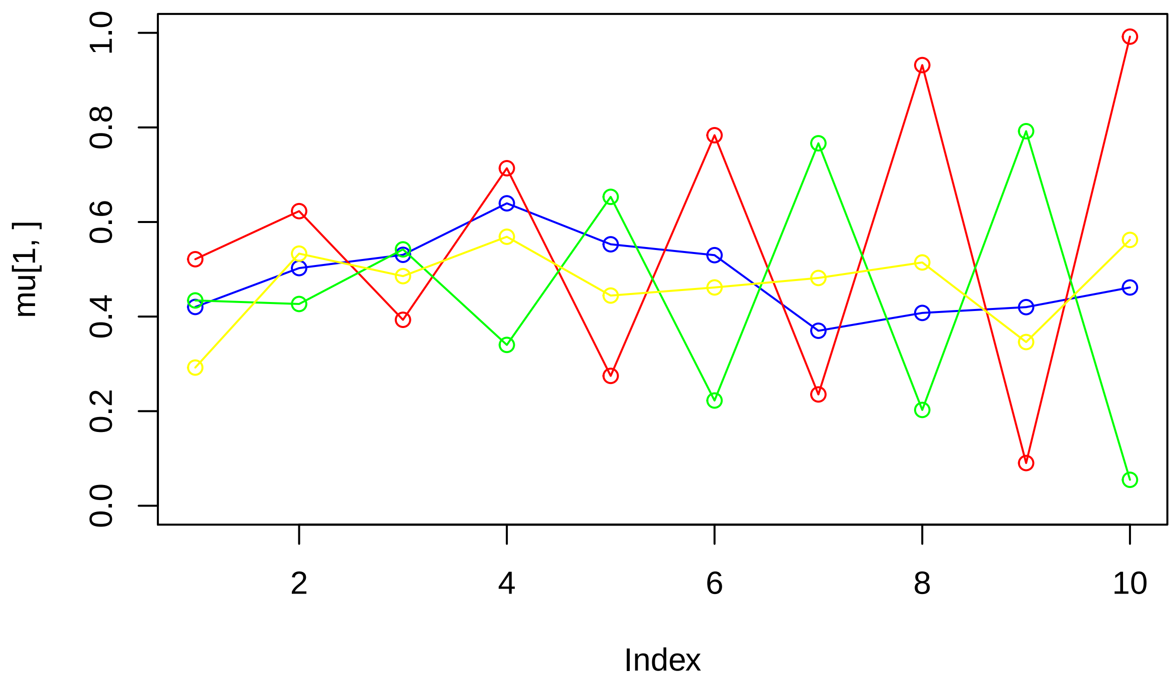

```
## iteration:  29 log likelihood:  -5676.669
```



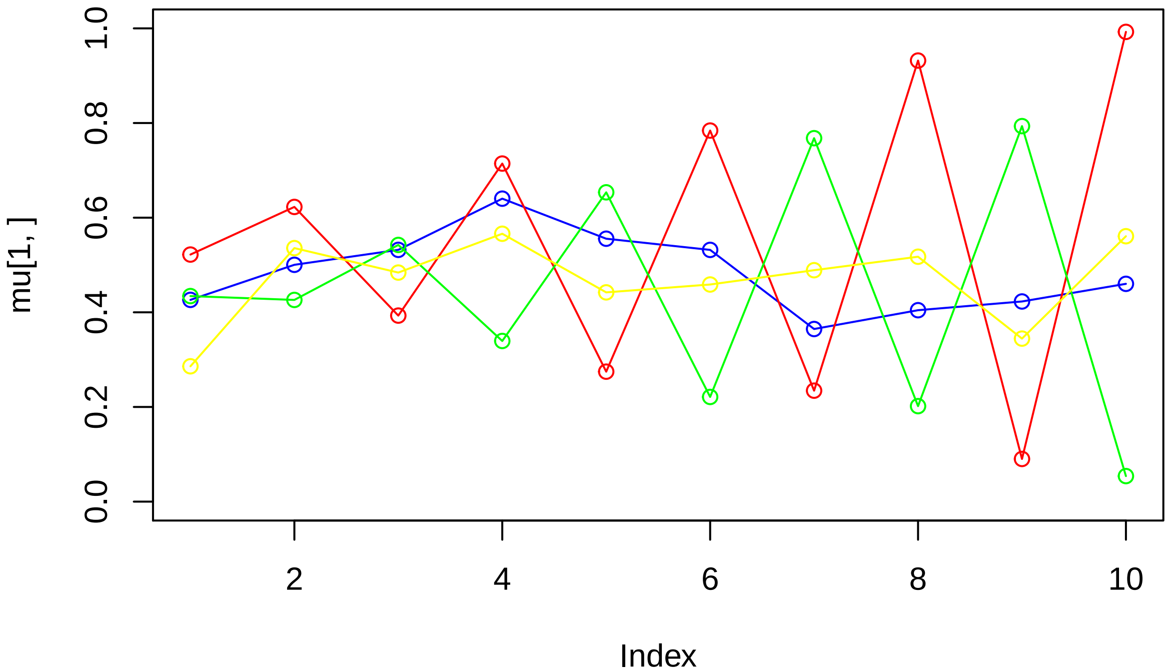

```
## iteration:  30 log likelihood:  -5674.239
```



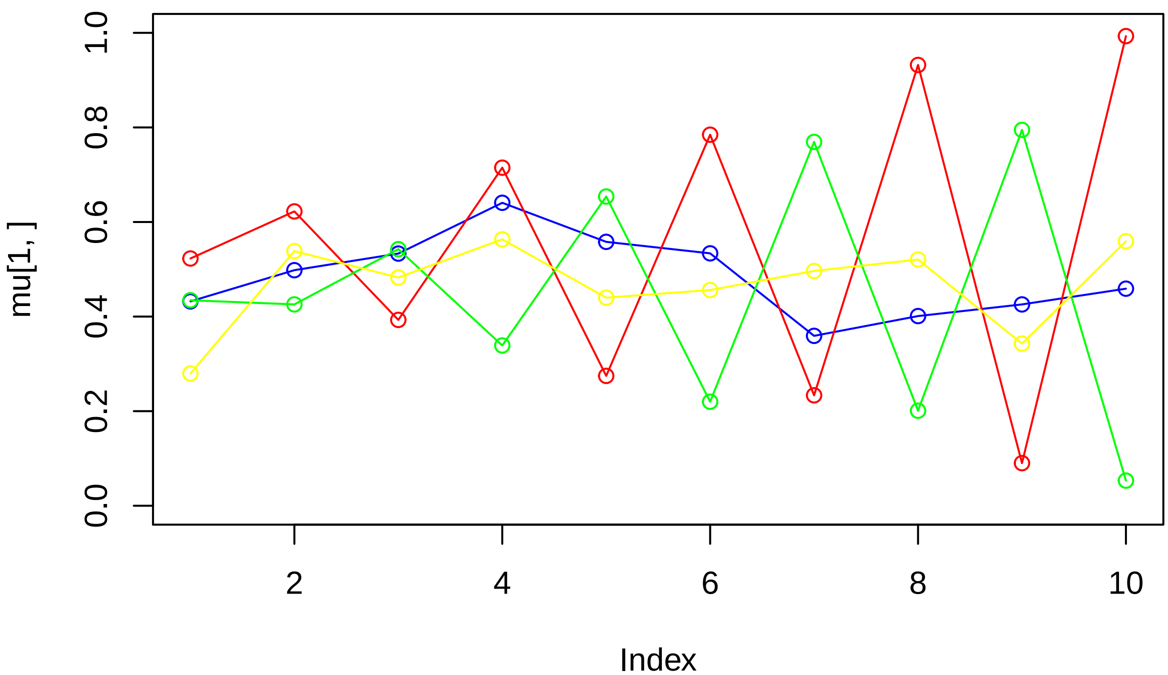

```
## iteration:  31 log likelihood:  -5671.576
```



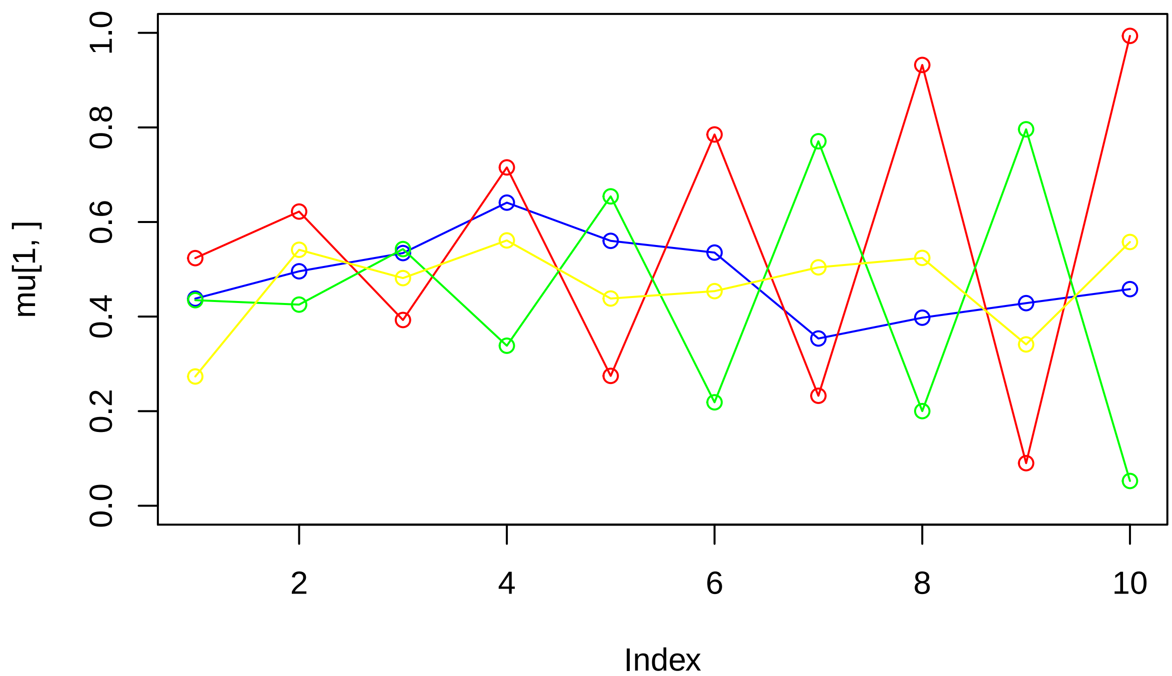

```
## iteration:  32 log likelihood:  -5668.691
```



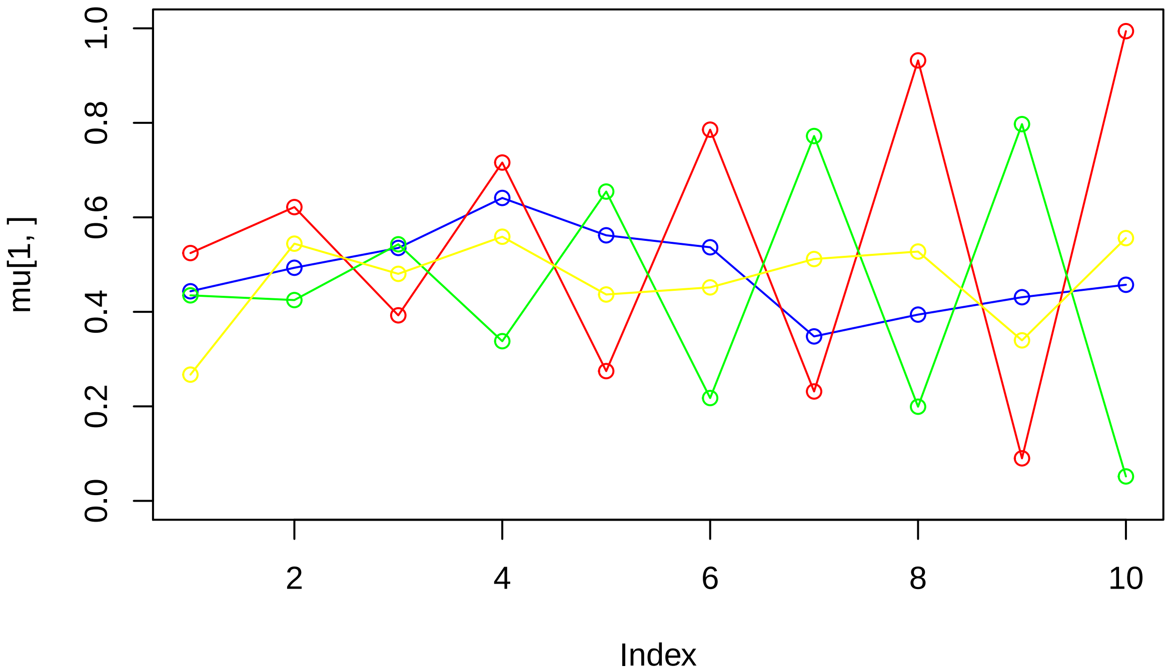

```
## iteration:  33 log likelihood:  -5665.614
```



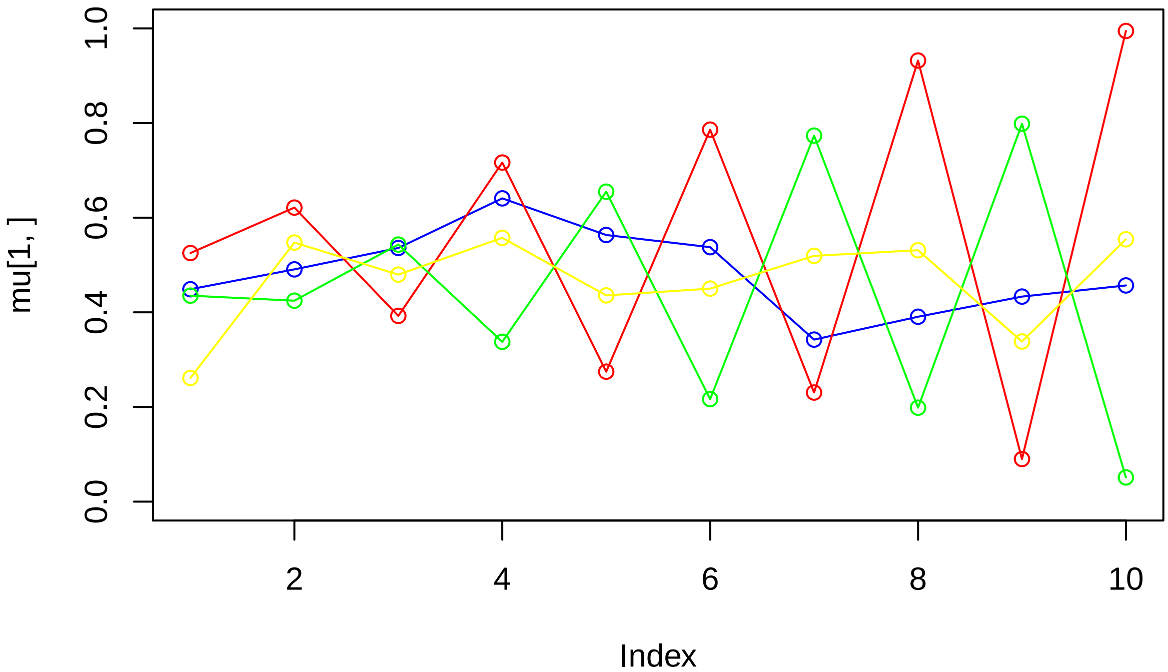

```
## iteration:  34 log likelihood:  -5662.386
```



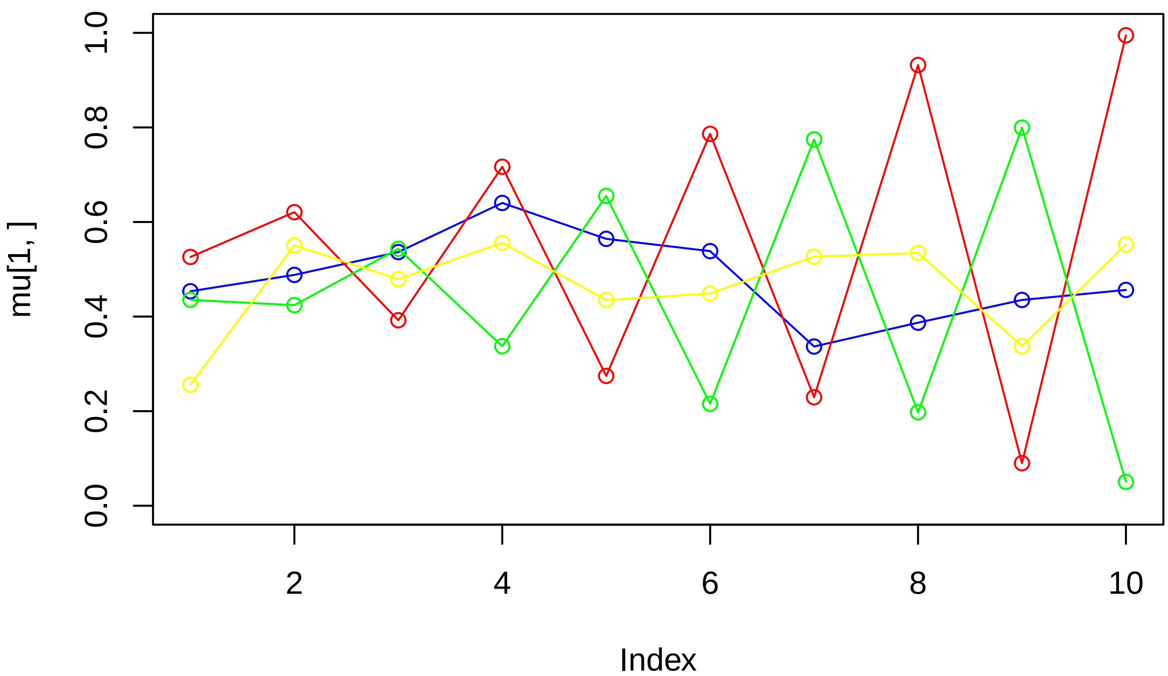

```
## iteration:  35 log likelihood:  -5659.059
```



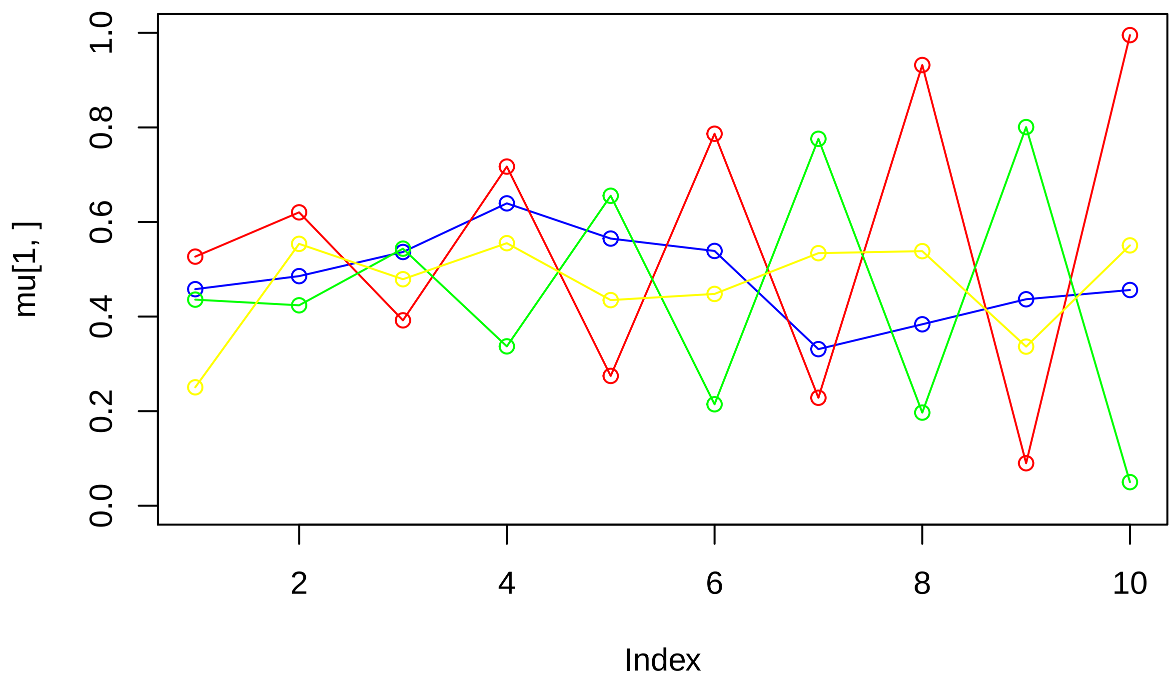

```
## iteration:  36 log likelihood:  -5655.693
```



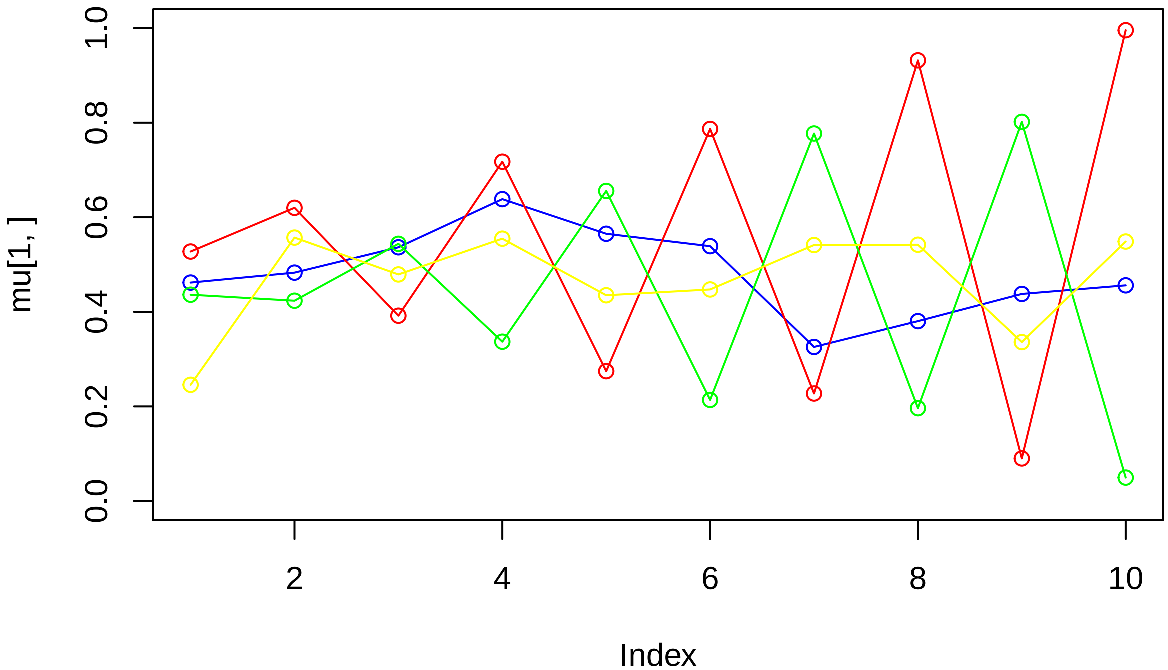

```
## iteration:  37 log likelihood:  -5652.346
```



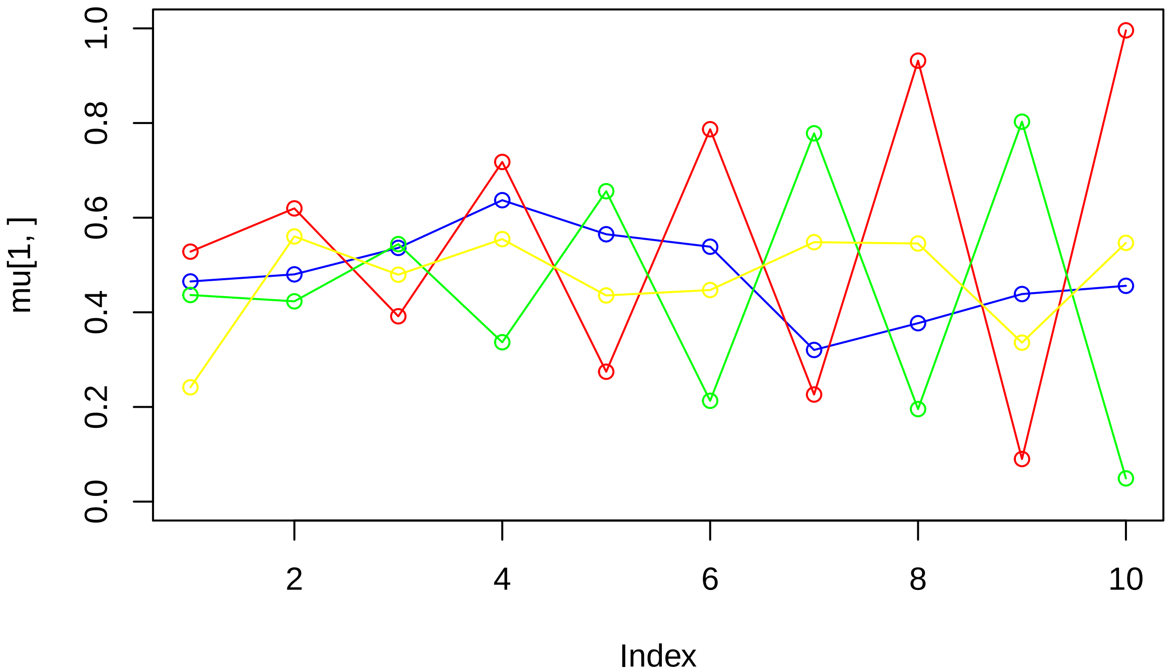

```
## iteration:  38 log likelihood:  -5649.071
```



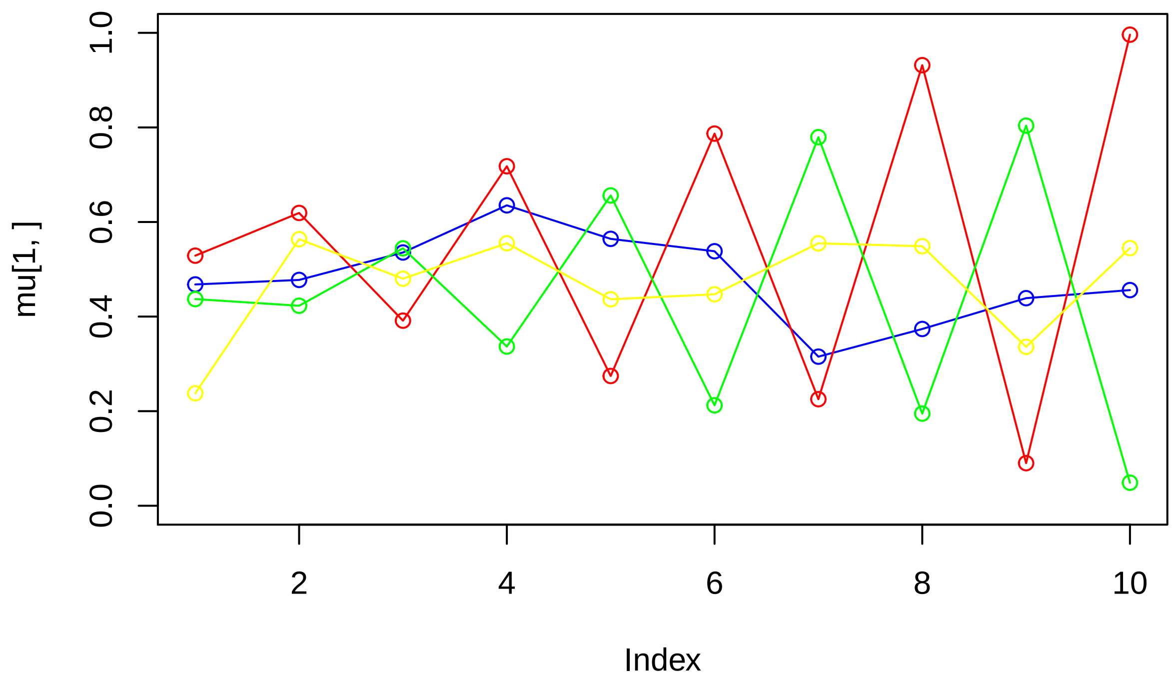

```
## iteration:  39 log likelihood:  -5645.908
```



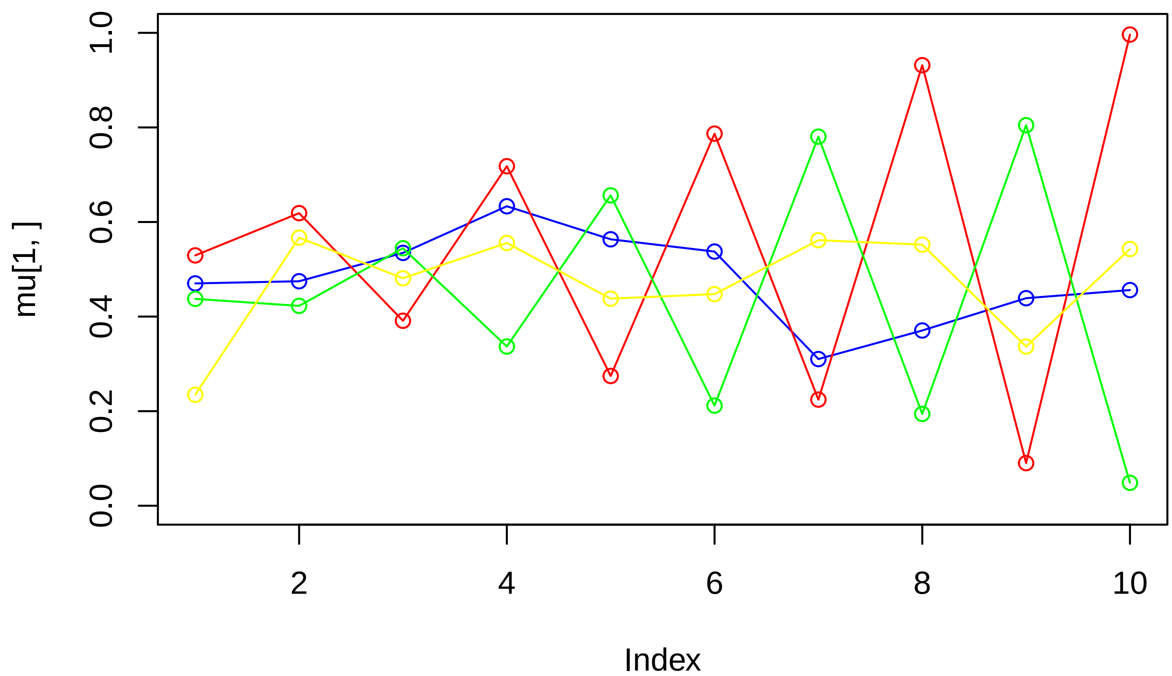

```
## iteration:  40 log likelihood:  -5642.885
```



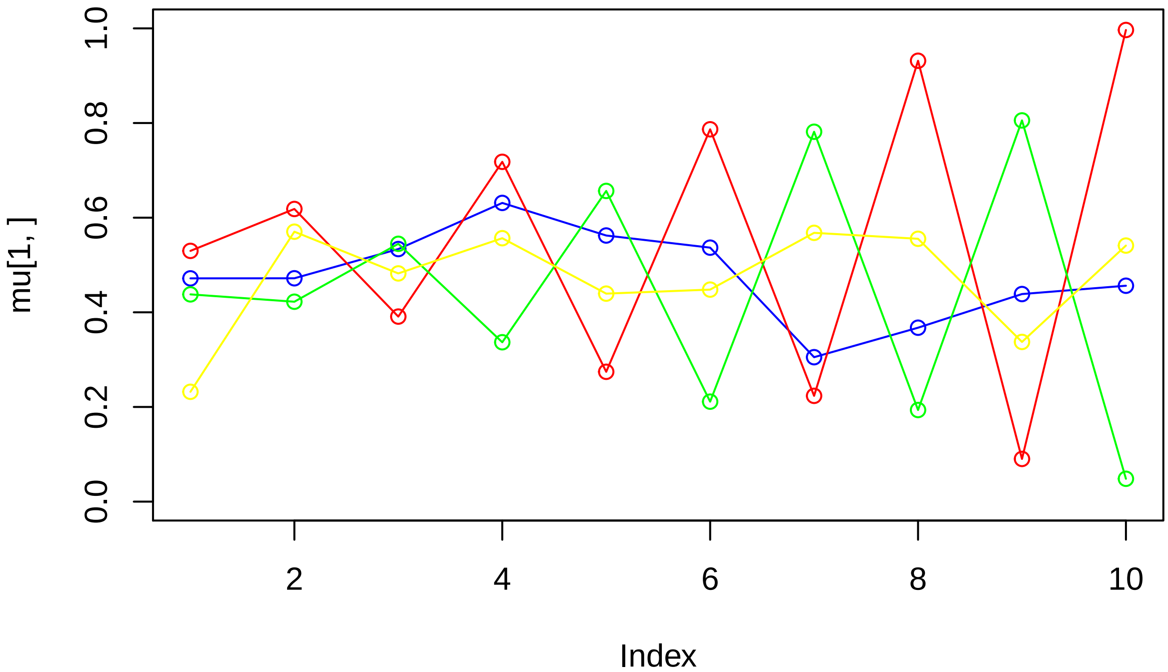

```
## iteration:  41 log likelihood:  -5640.016
```



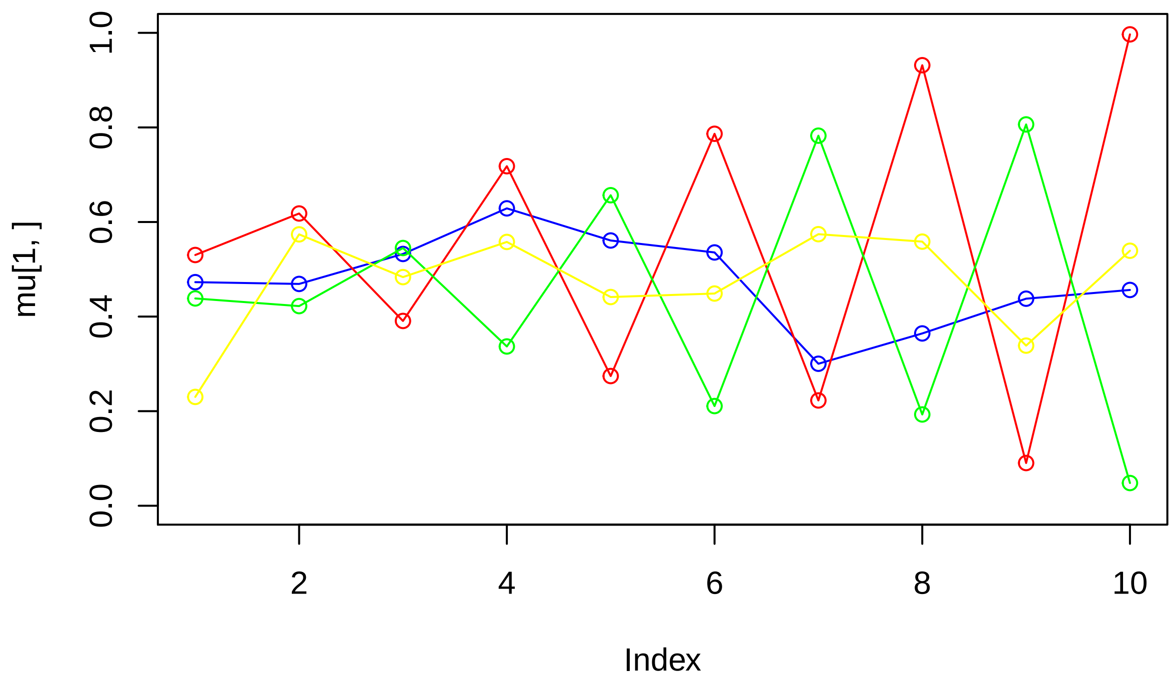

```
## iteration:  42 log likelihood:  -5637.303
```



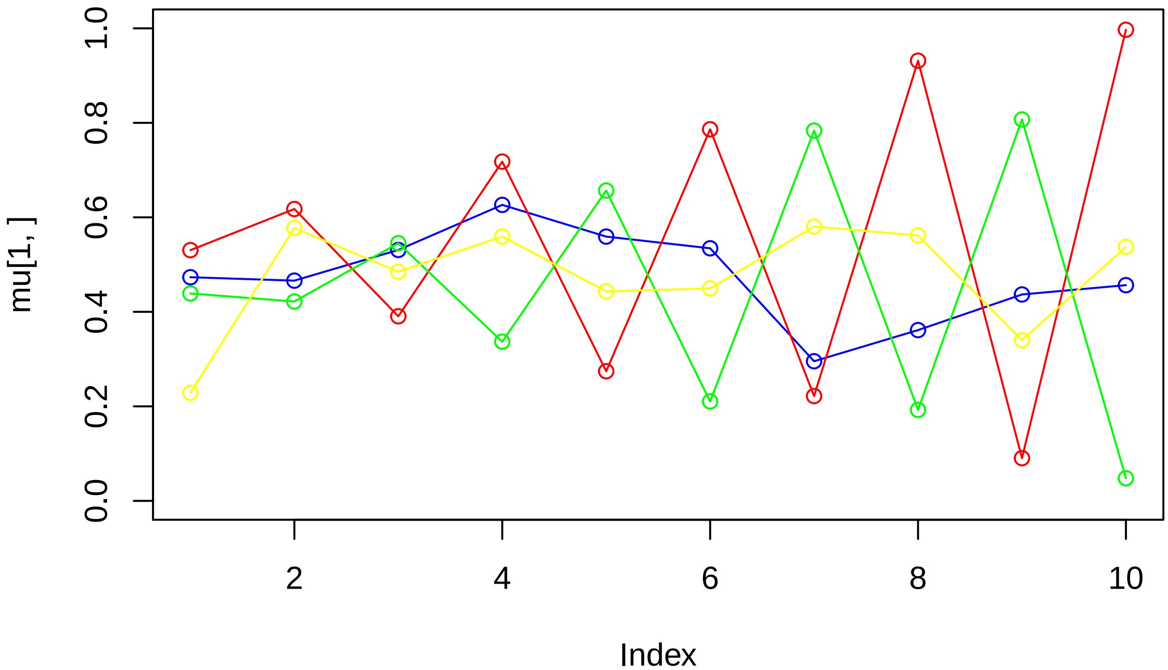

```
## iteration:  43 log likelihood:  -5634.736
```



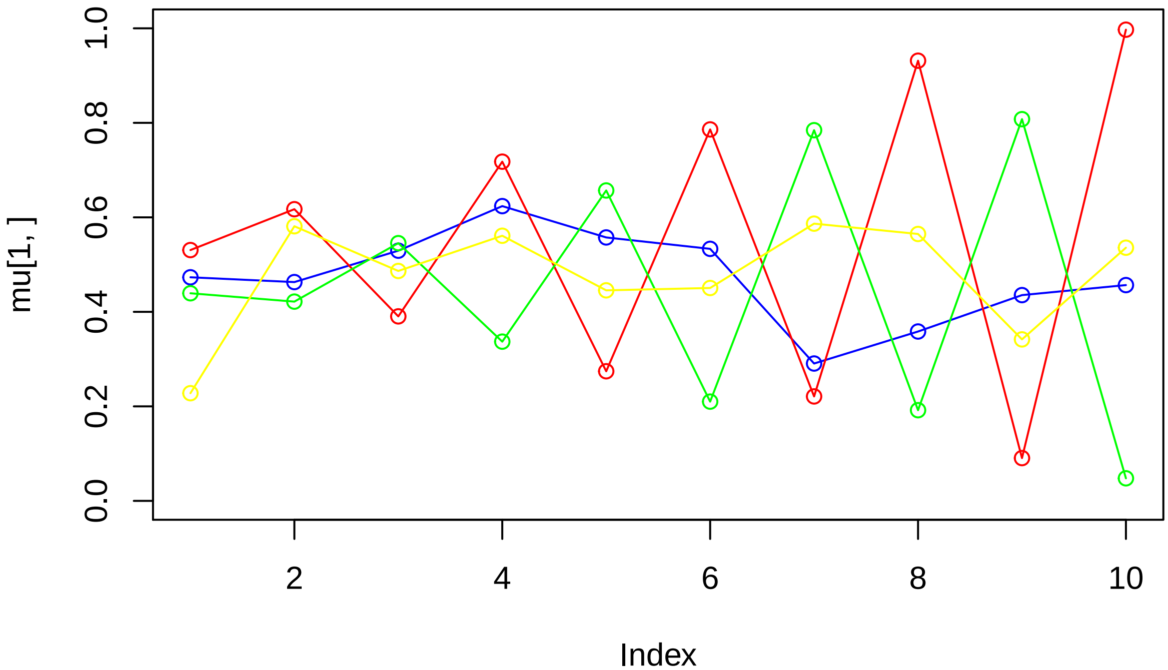

```
## iteration:  44 log likelihood:  -5632.298
```



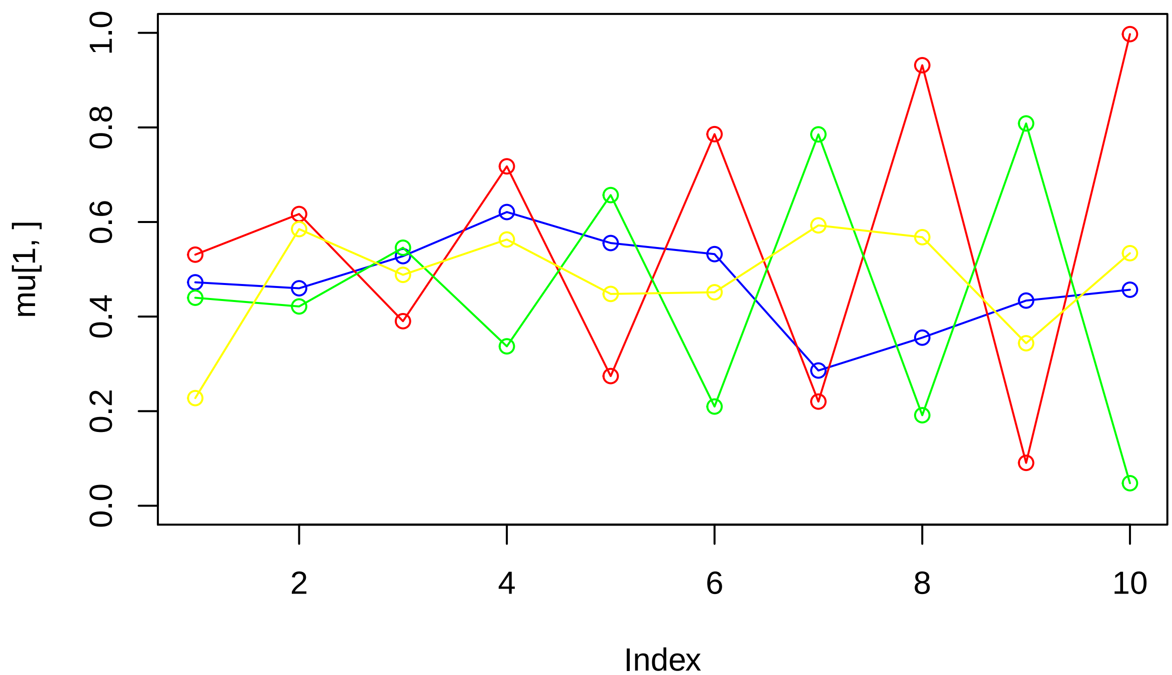

```
## iteration:  45 log likelihood:  -5629.969
```



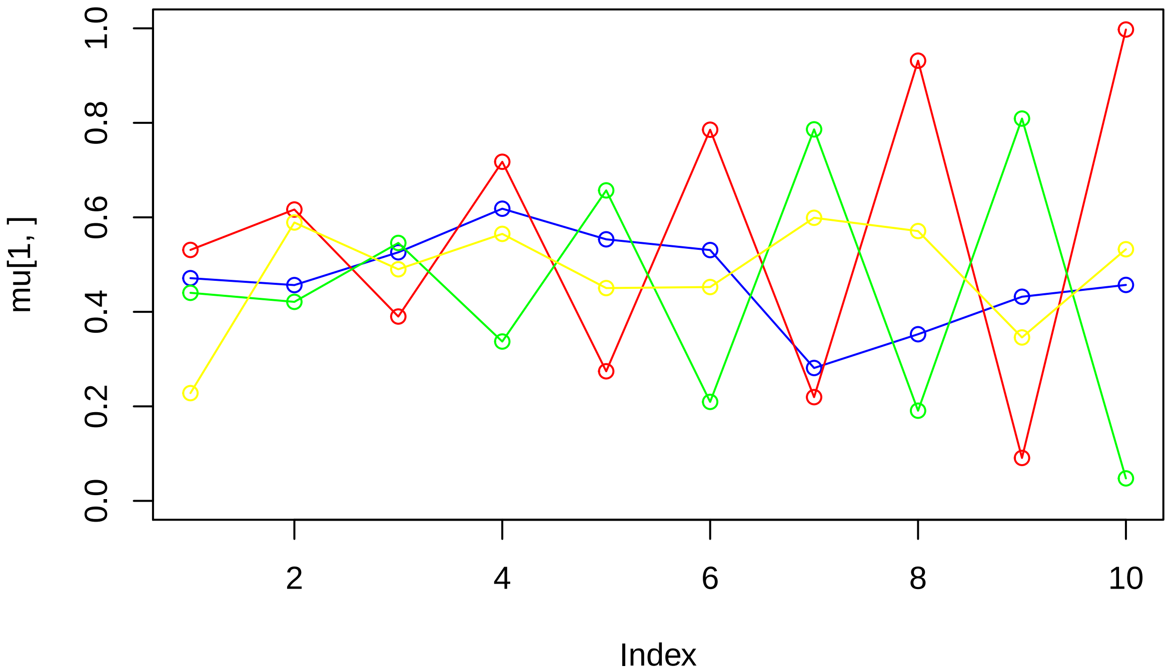

```
## iteration:  46 log likelihood:  -5627.724
```



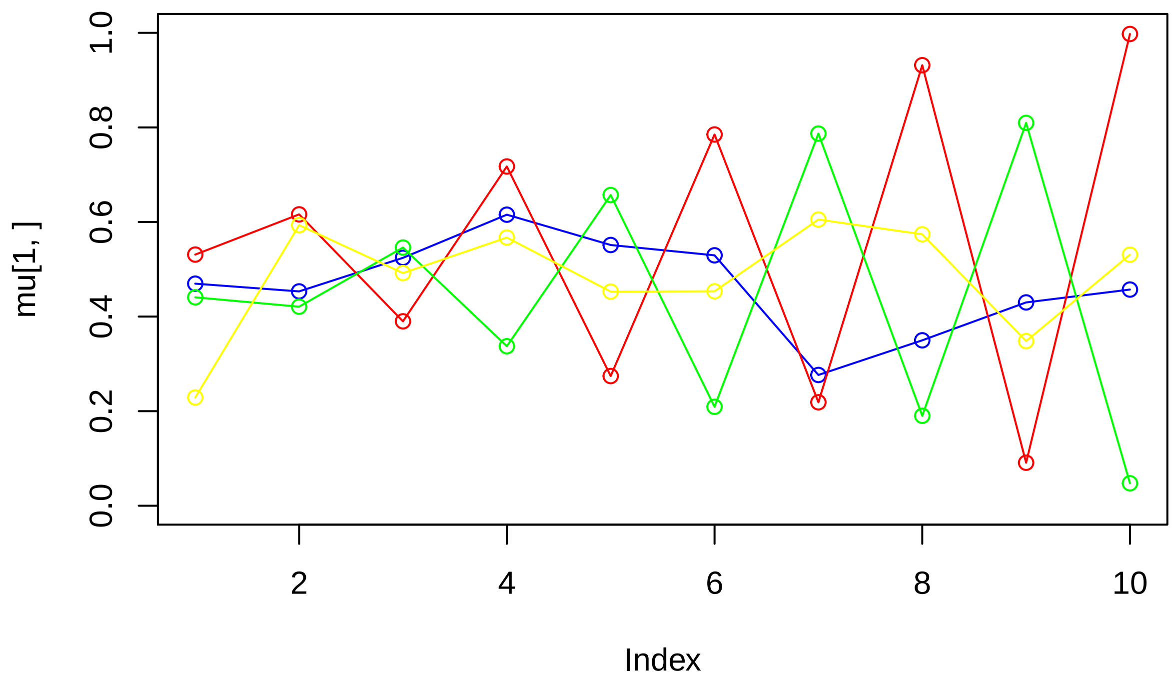

```
## iteration:  47 log likelihood:  -5625.538
```



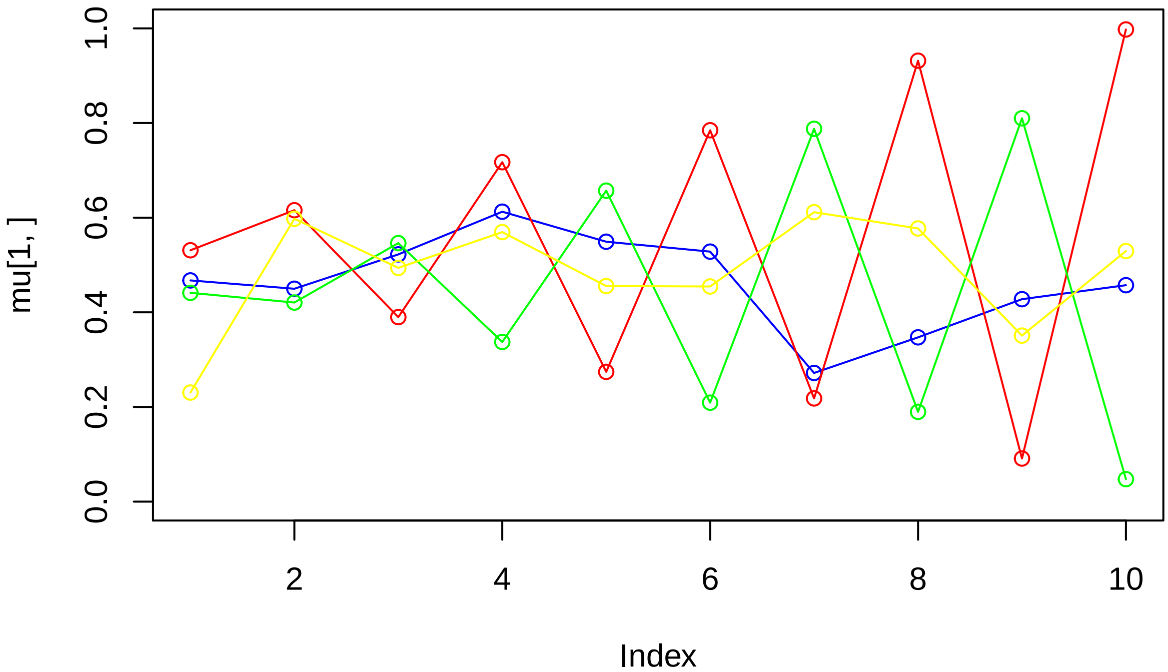

```
## iteration:  48 log likelihood:  -5623.385
```



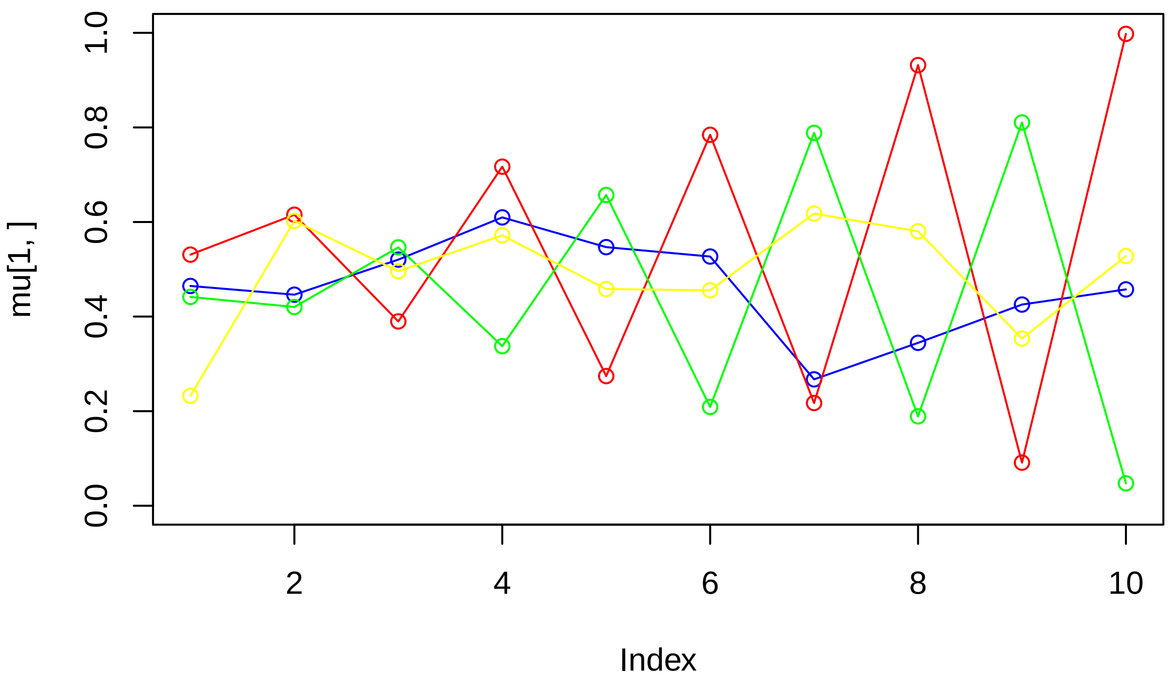

```
## iteration:  49 log likelihood:  -5621.241
```



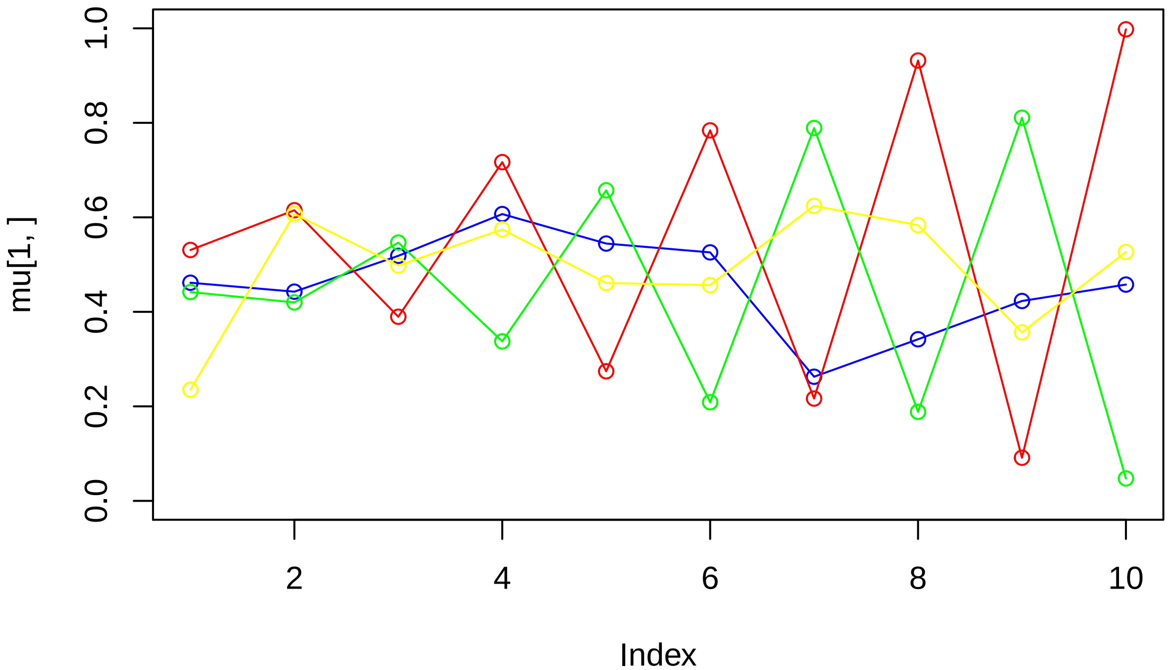

```
## iteration:  50 log likelihood:  -5619.084
```



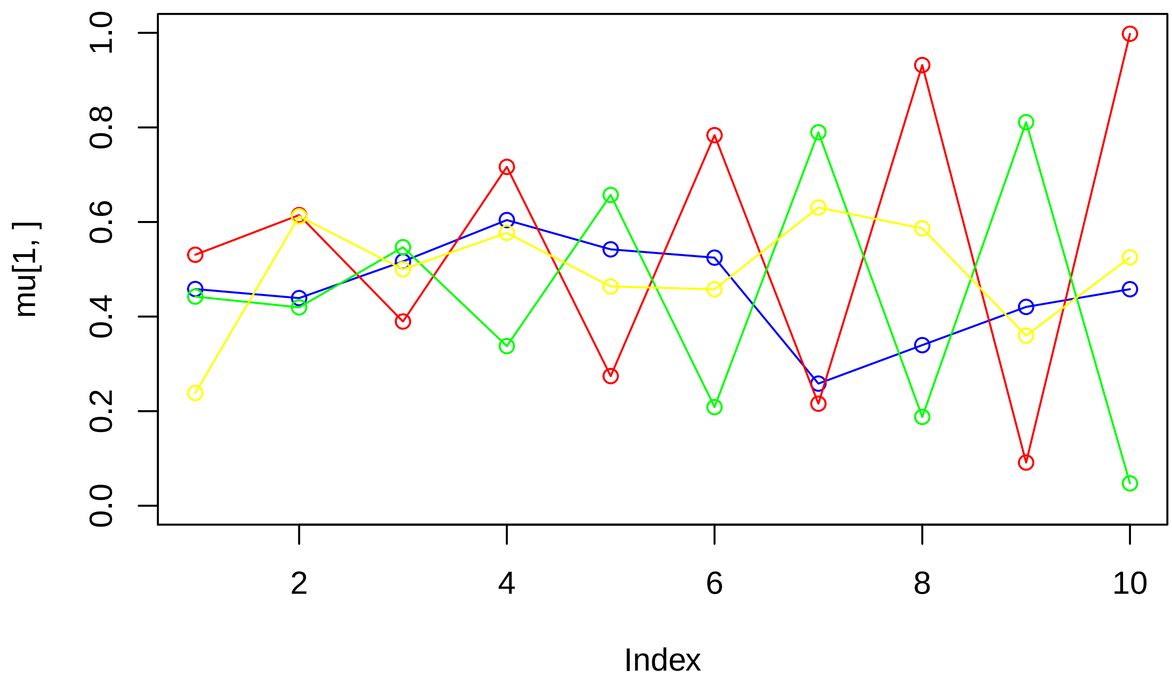

```
## iteration:  51 log likelihood:  -5616.894
```



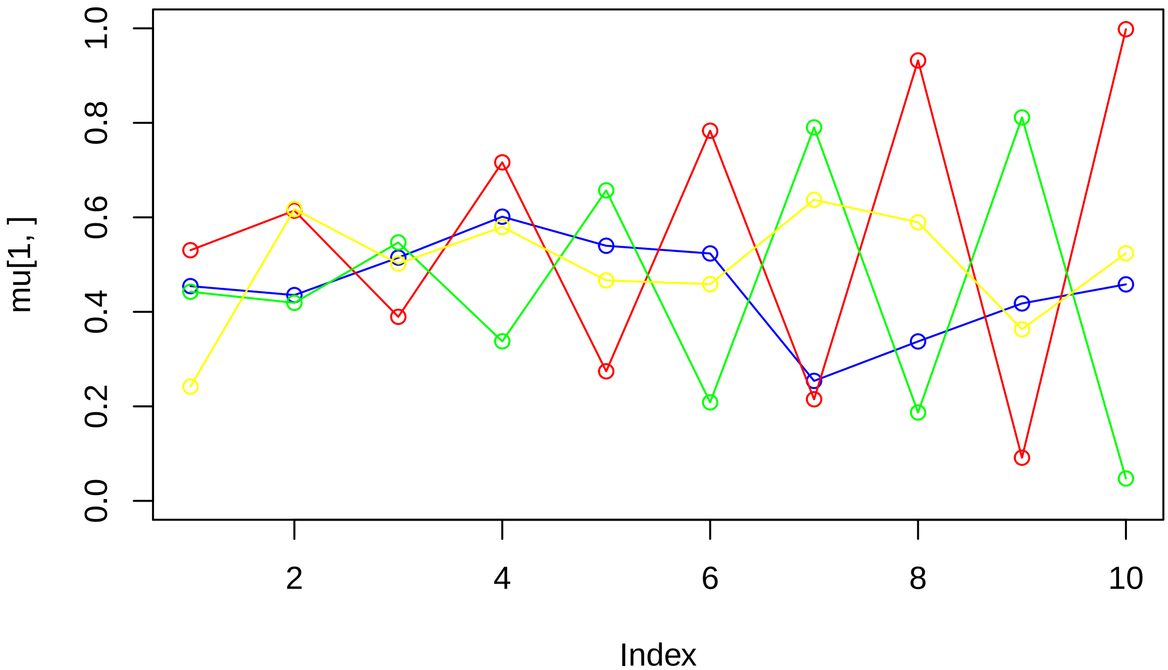

```
## iteration:  52 log likelihood:  -5614.658
```



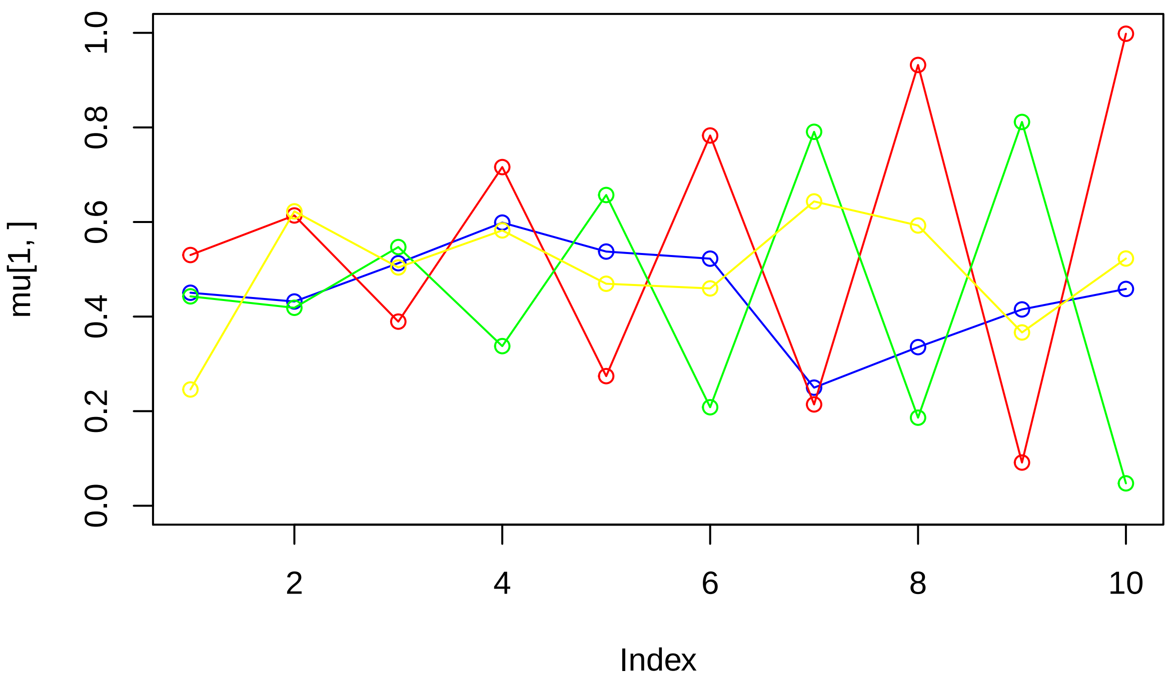

```
## iteration:  53 log likelihood:  -5612.364
```



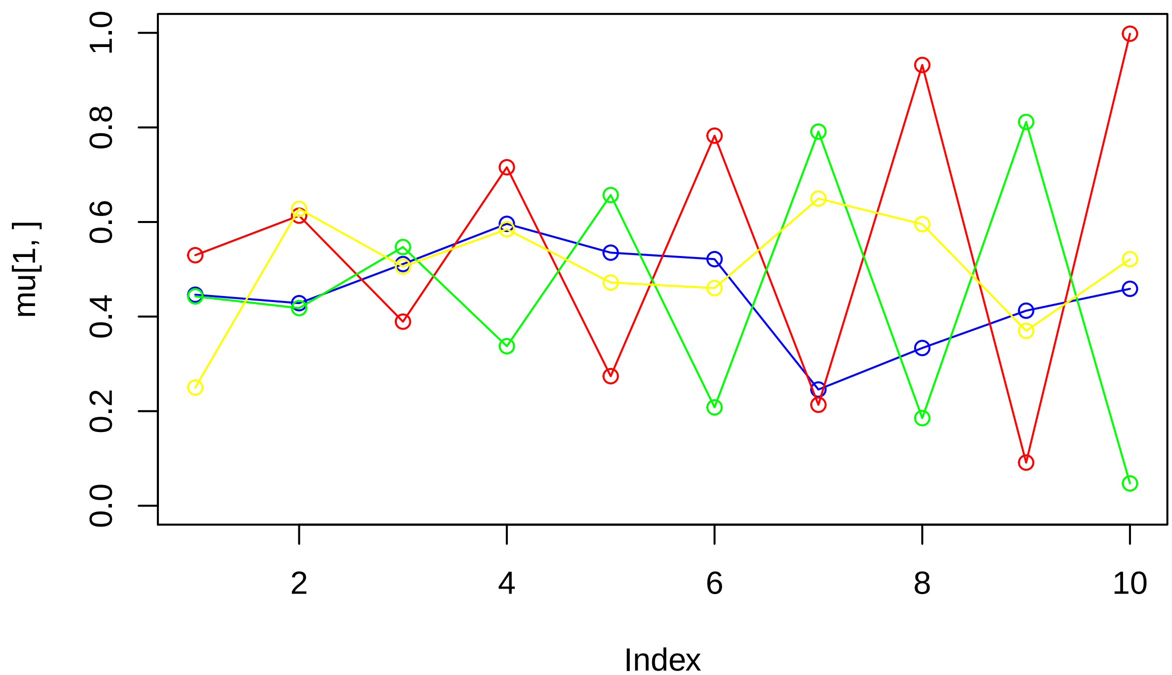

```
## iteration:  54 log likelihood:  -5610.009
```



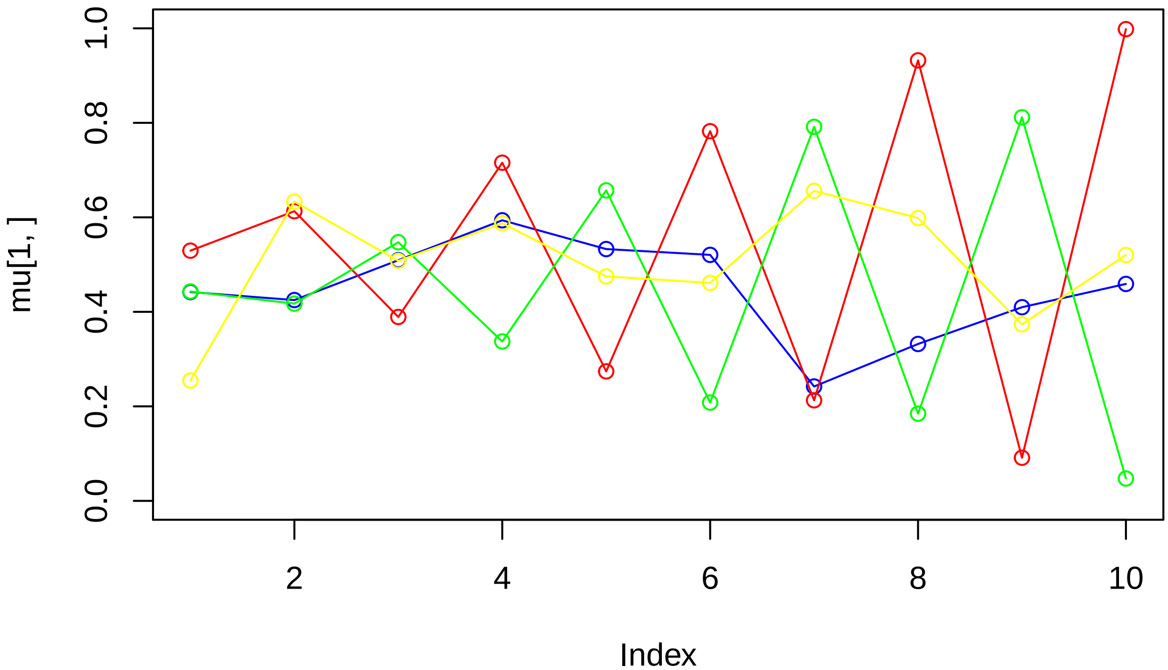

```
## iteration:  55 log likelihood:  -5607.596
```



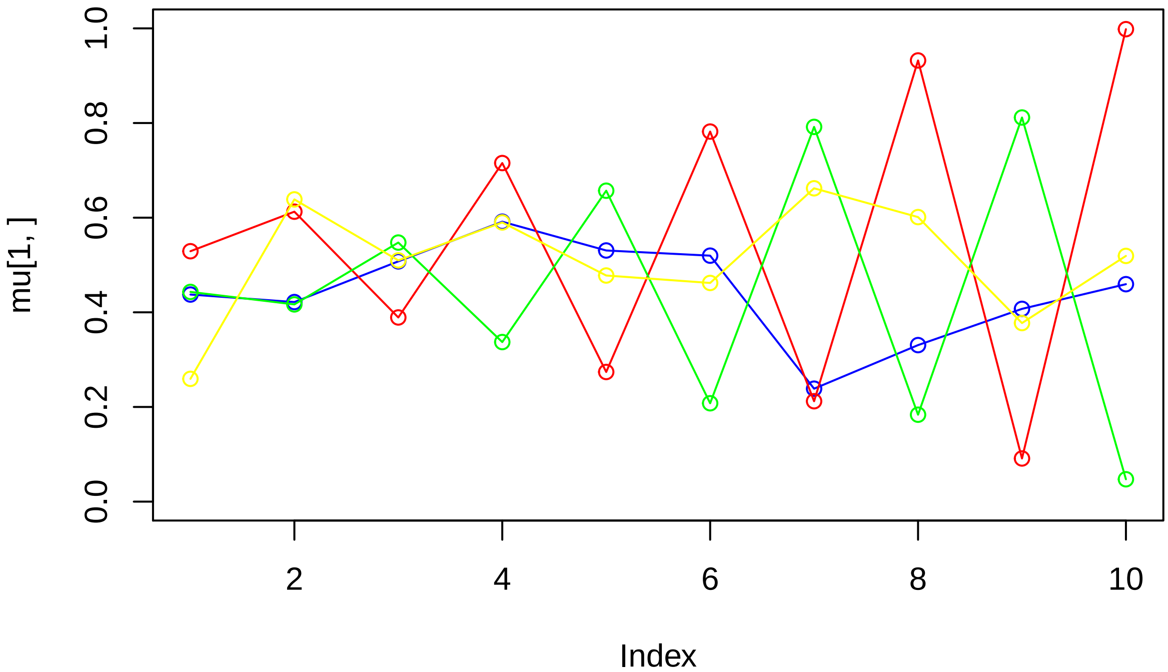

```
## iteration:  56 log likelihood:  -5605.138
```



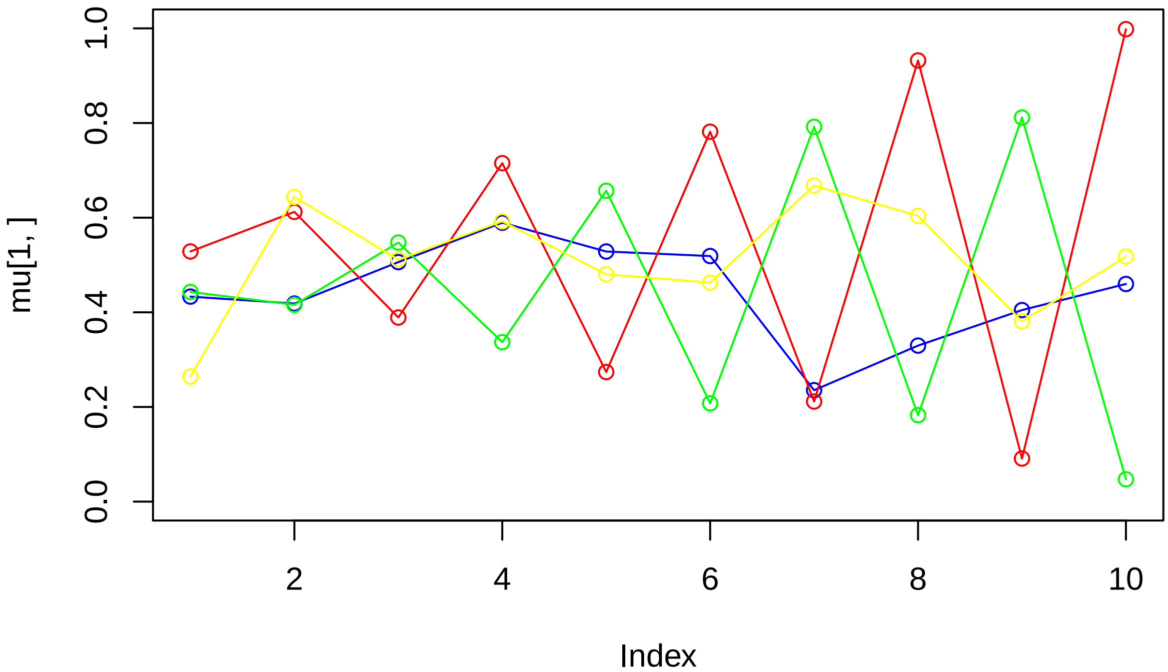

```
## iteration:  57 log likelihood:  -5602.654
```



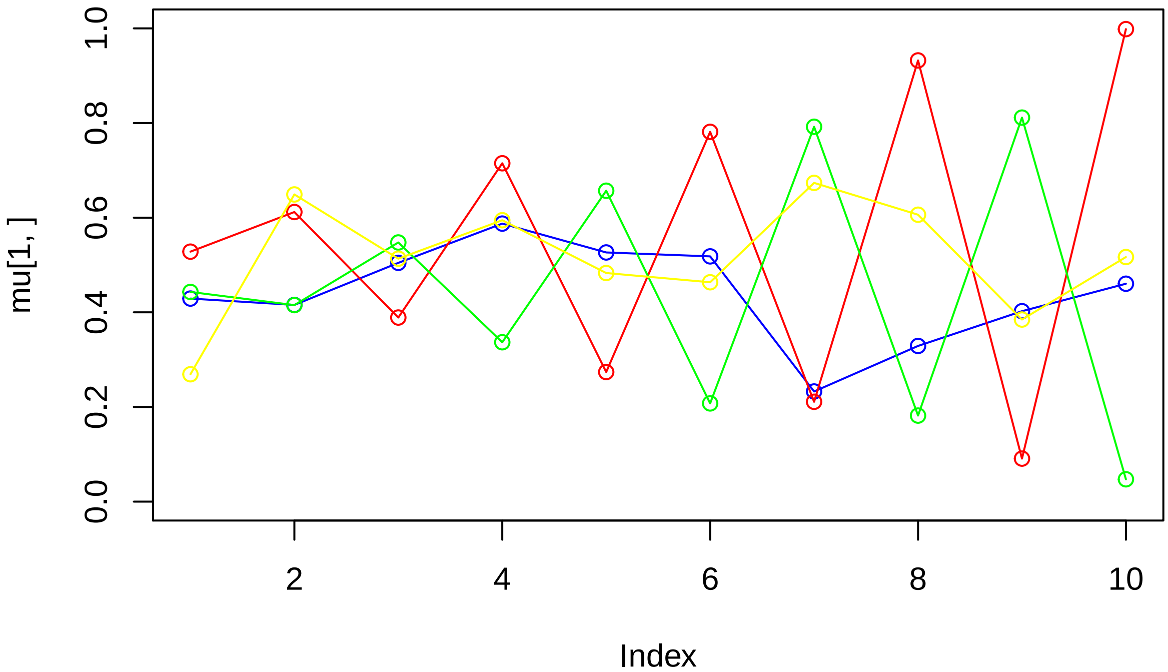

```
## iteration:  58 log likelihood:  -5600.168
```



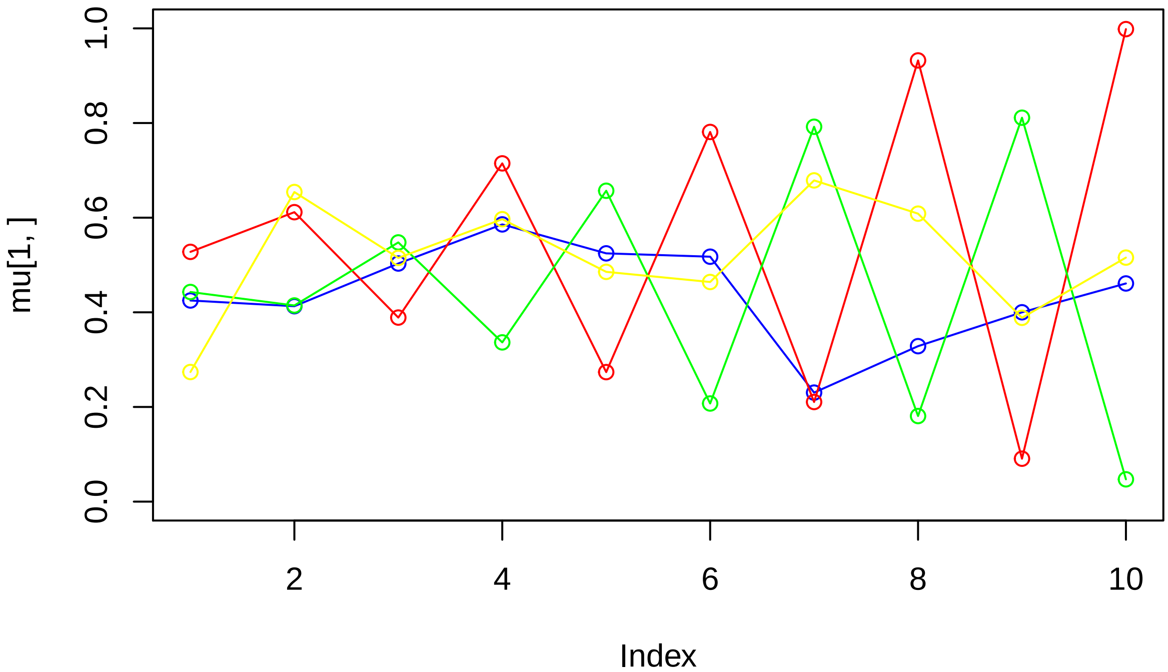

```
## iteration:  59 log likelihood:  -5597.712
```



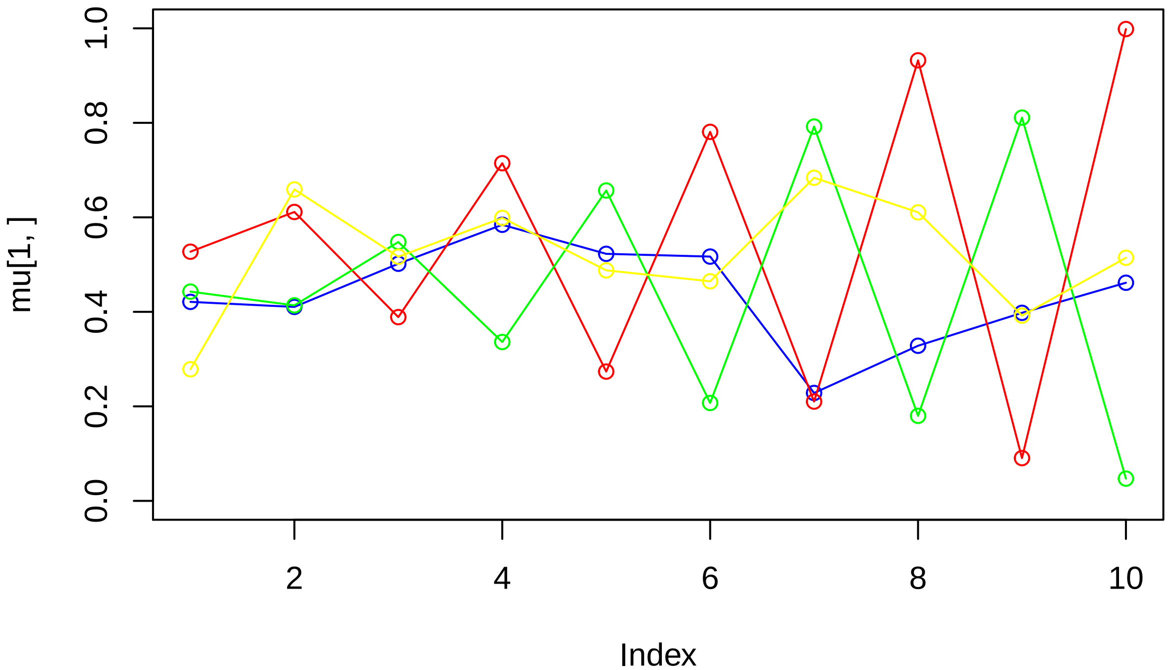

```
## iteration:  60 log likelihood:  -5595.319
```



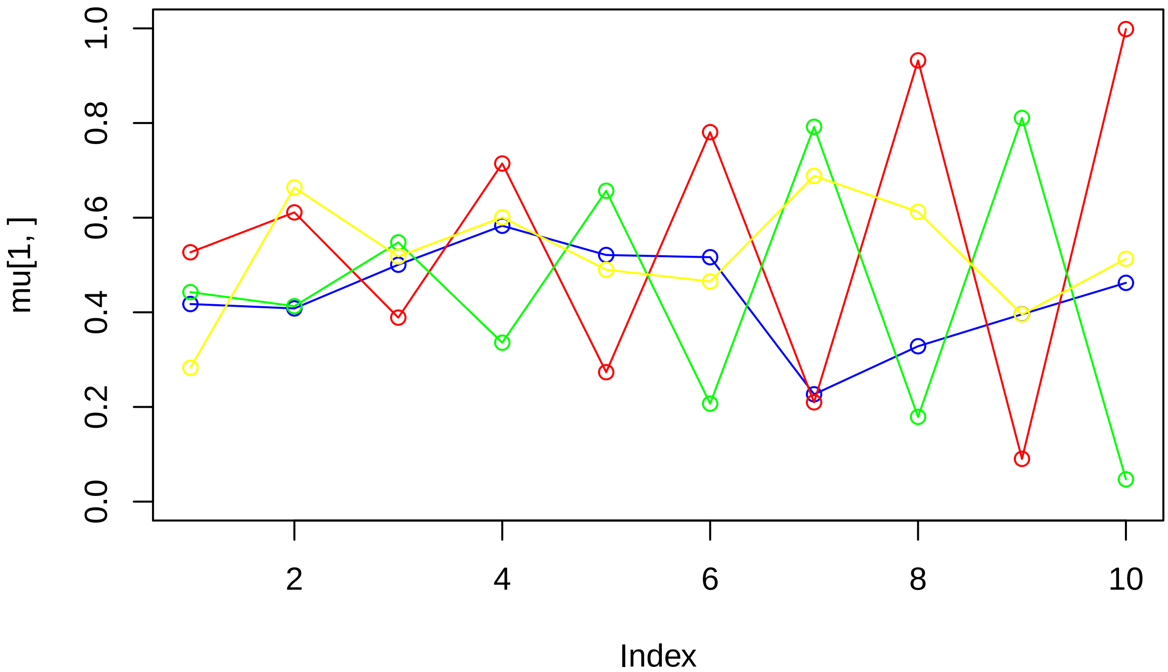

```
## iteration:  61 log likelihood:  -5593.02
```



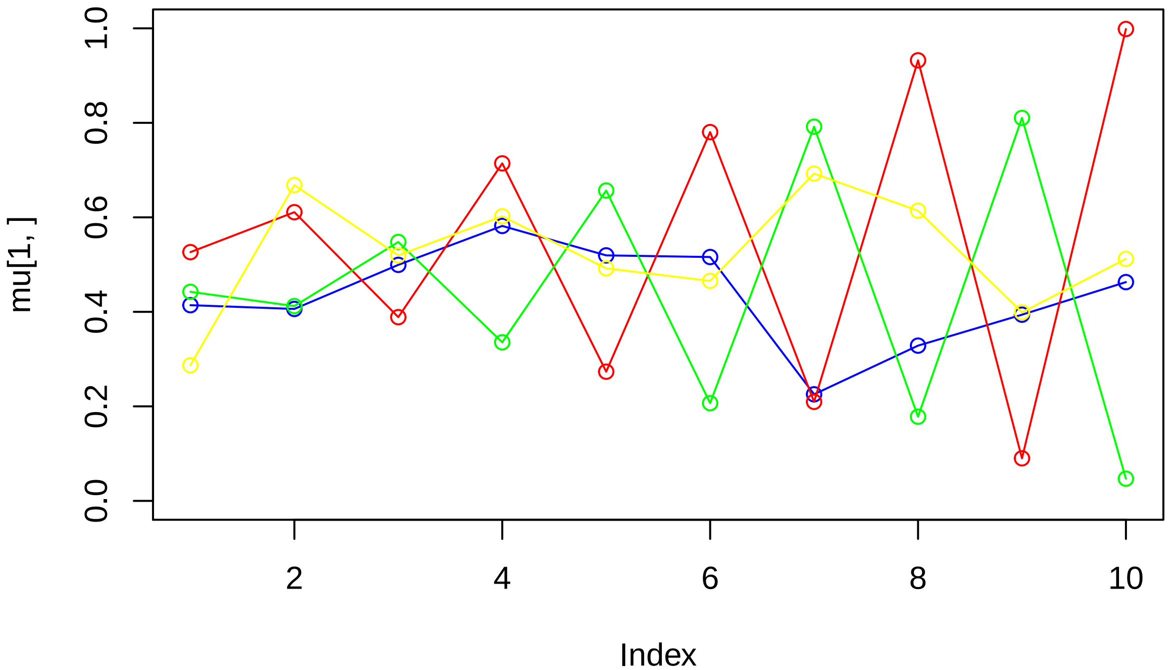

```
## iteration:  62 log likelihood:  -5590.845
```



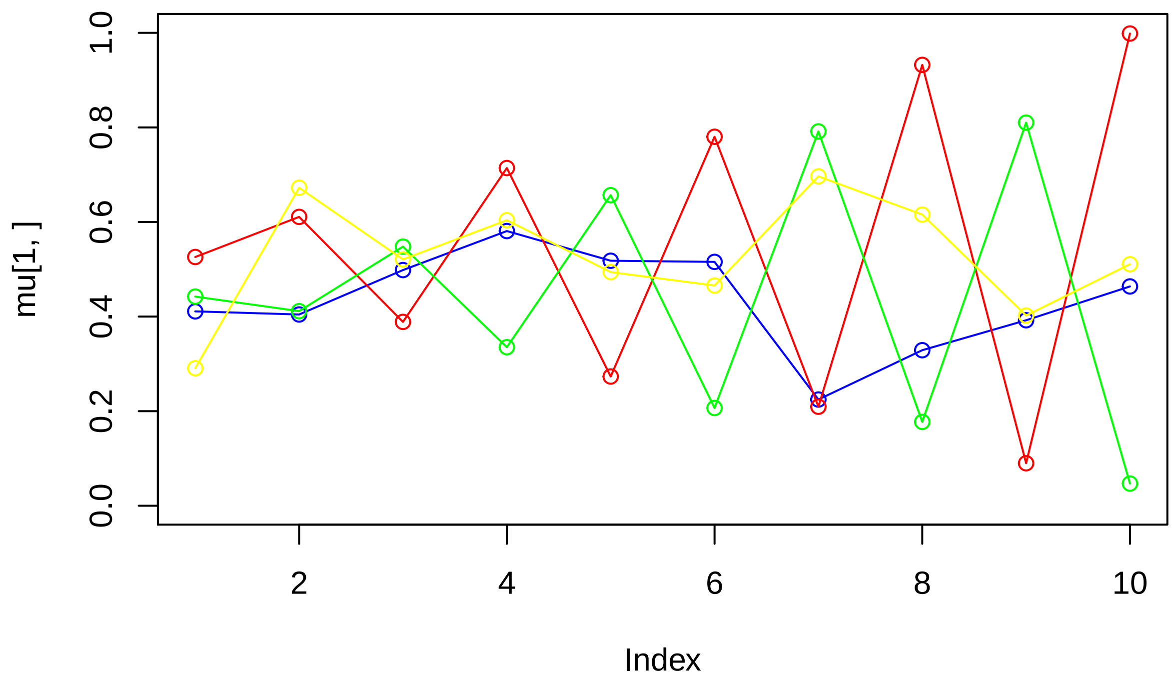

```
## iteration:  63 log likelihood:  -5588.818
```



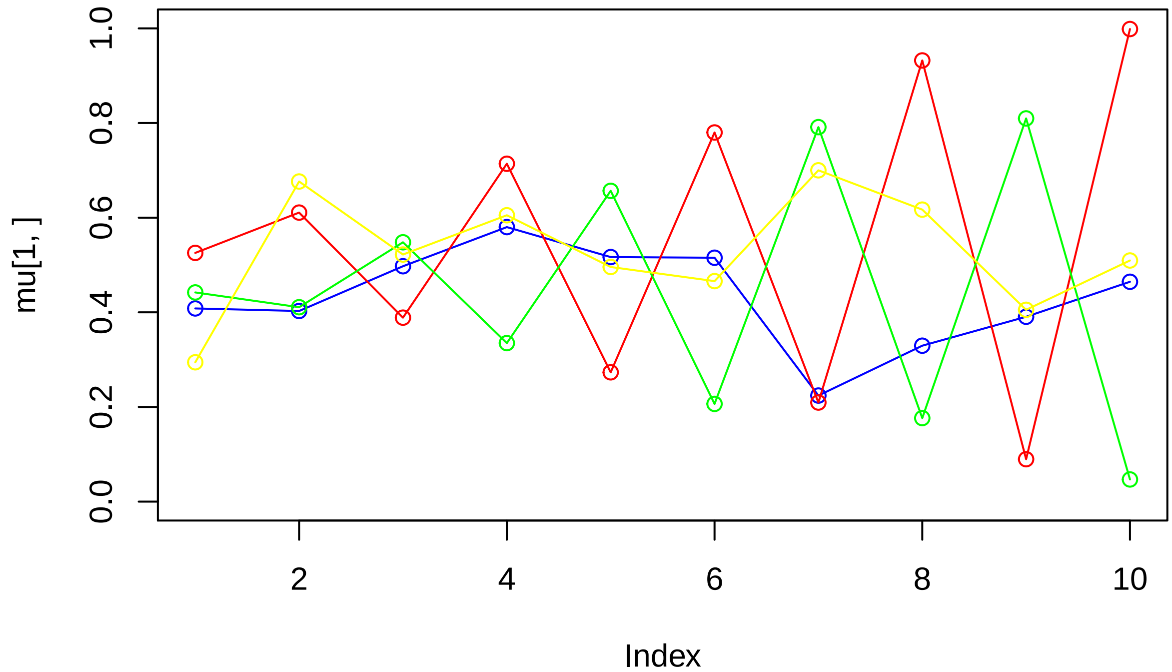

```
## iteration:  64 log likelihood:  -5586.954
```



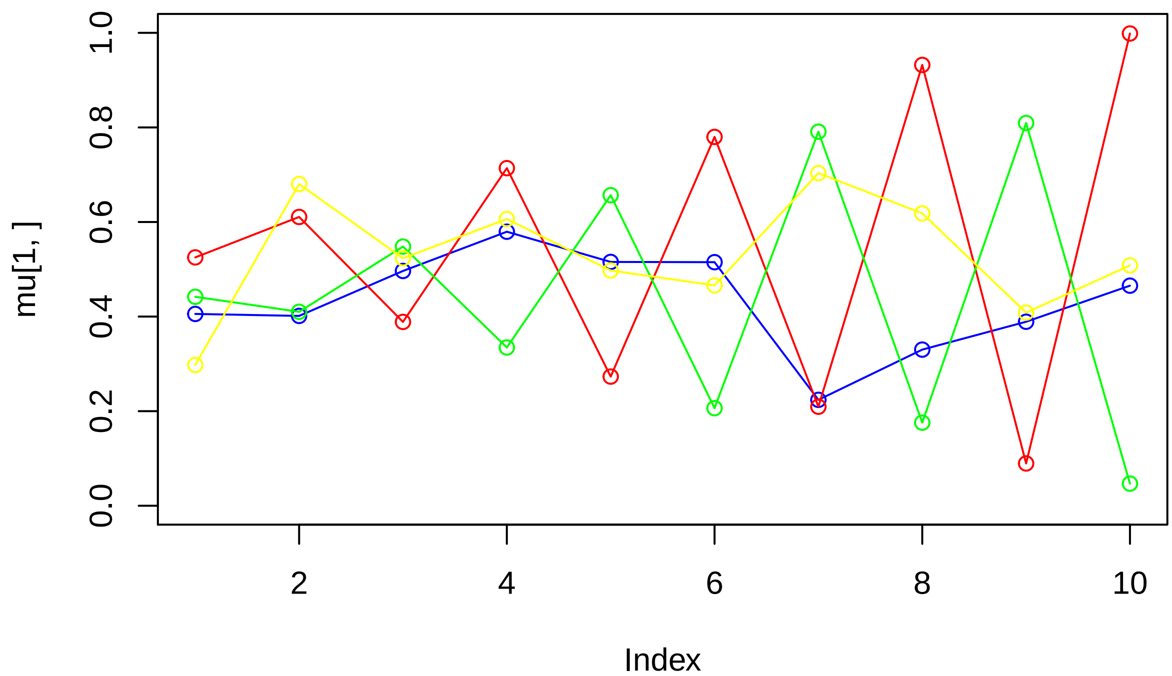

```
## iteration:  65 log likelihood:  -5585.263
```



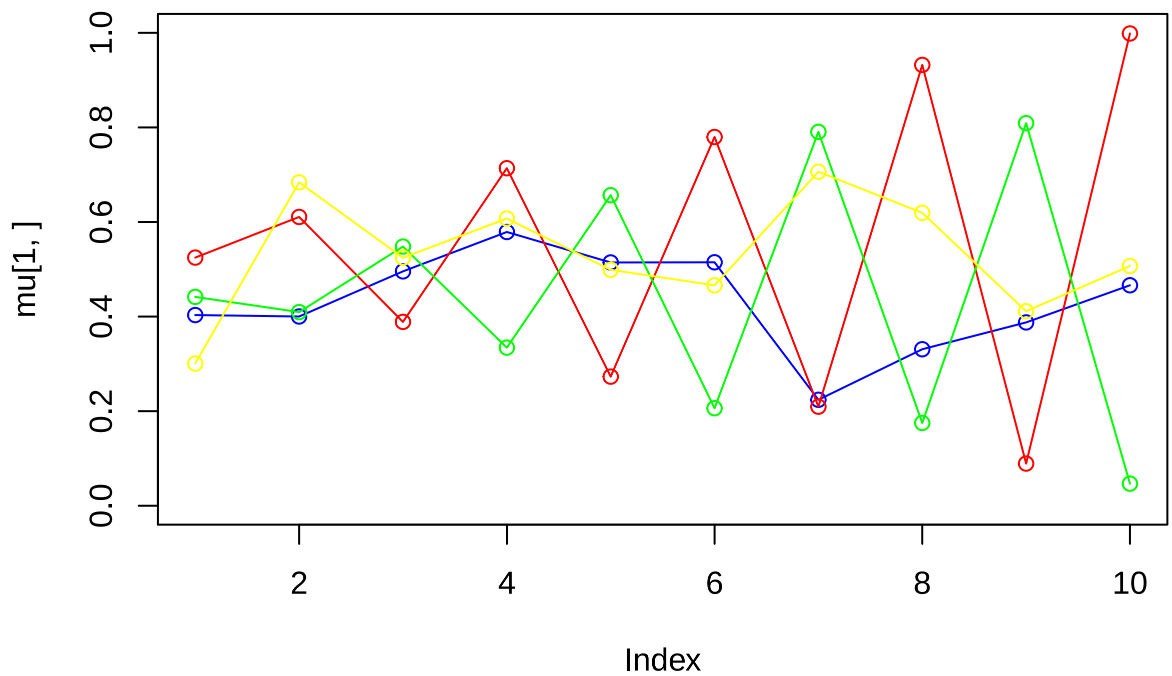

```
## iteration:  66 log likelihood:  -5583.748
```



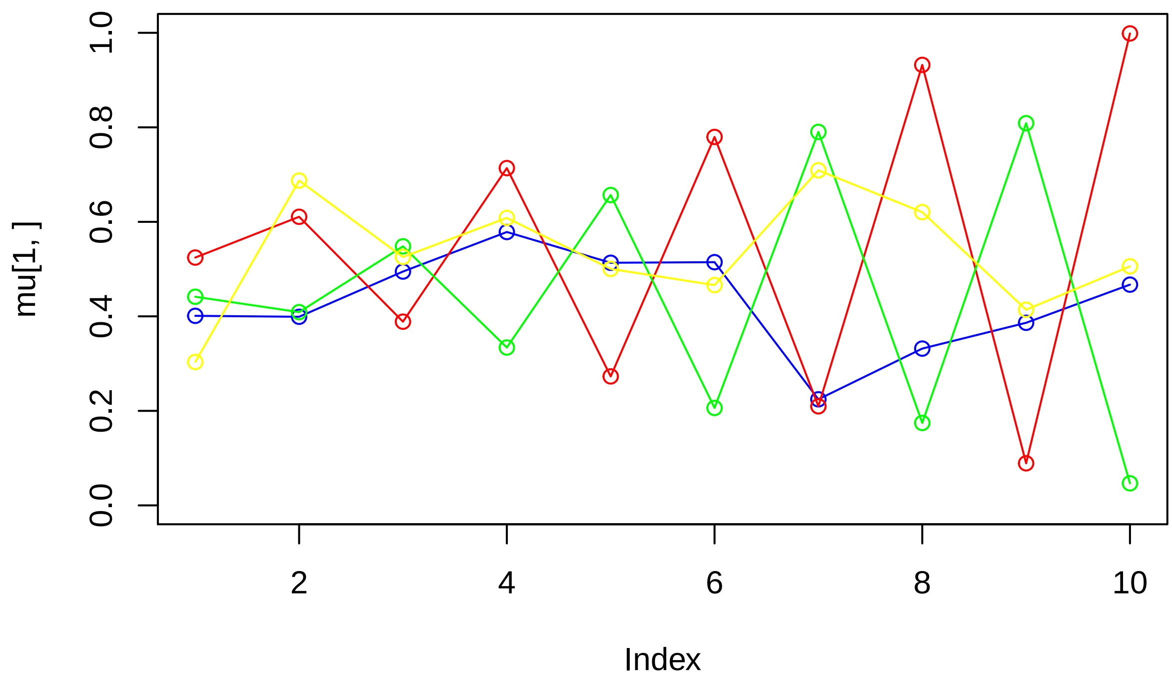

```
## iteration:  67 log likelihood:  -5582.404
```



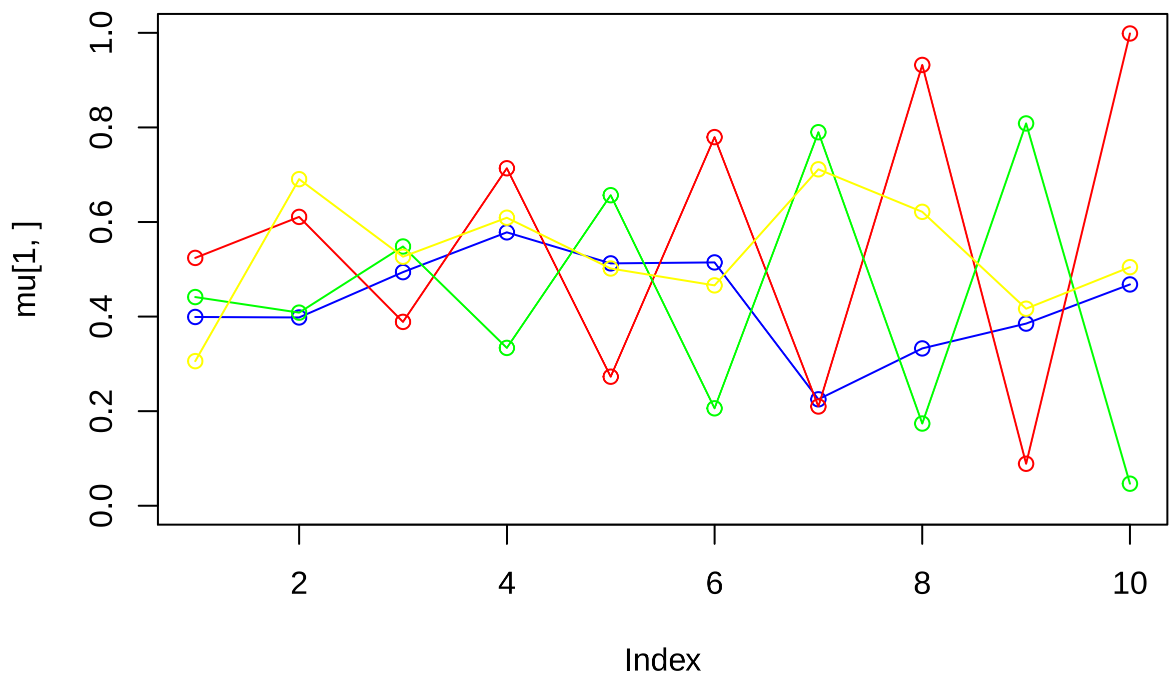

```
## iteration:  68 log likelihood:  -5581.223
```



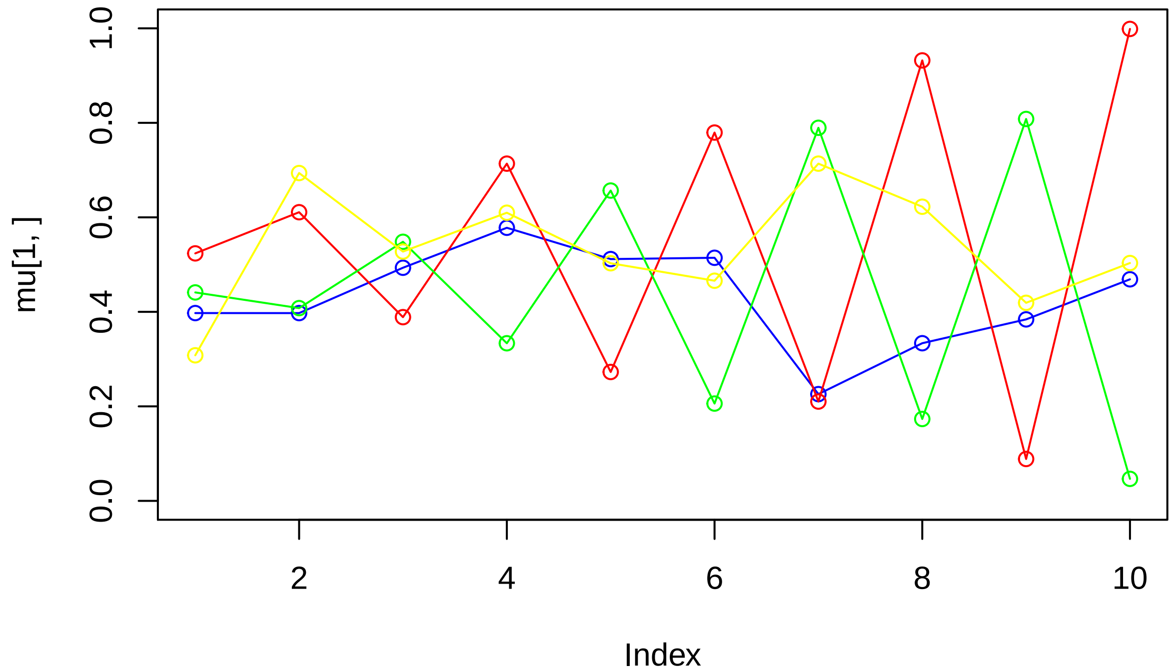

```
## iteration:  69 log likelihood:  -5580.191
```



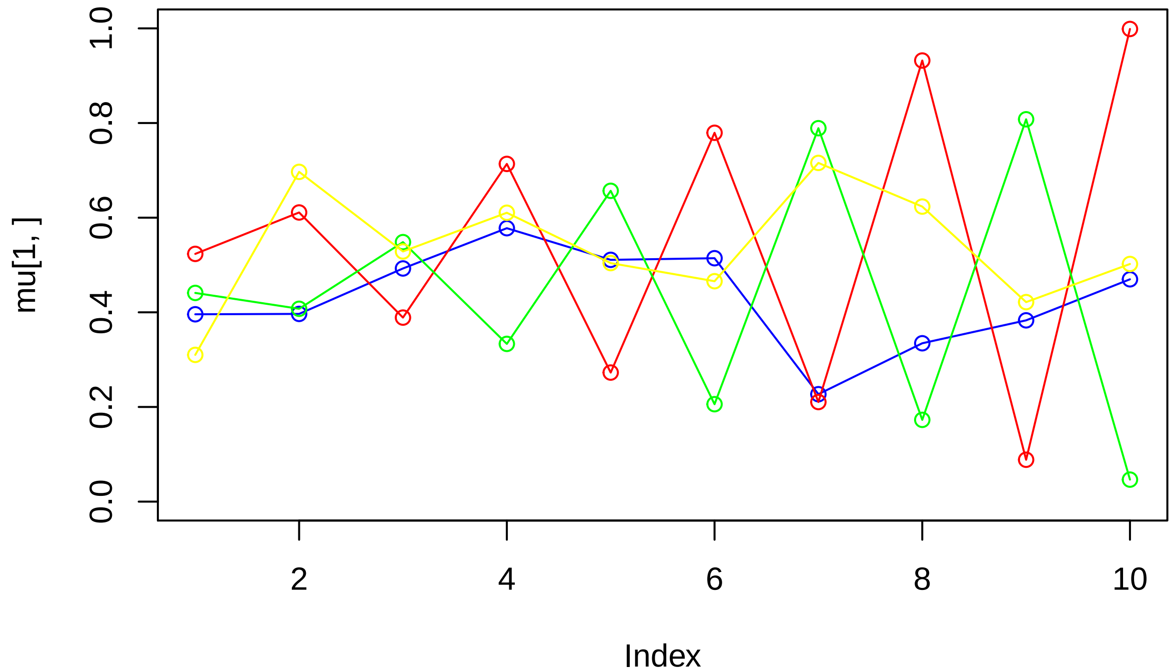

```
## iteration:  70 log likelihood:  -5579.295
```



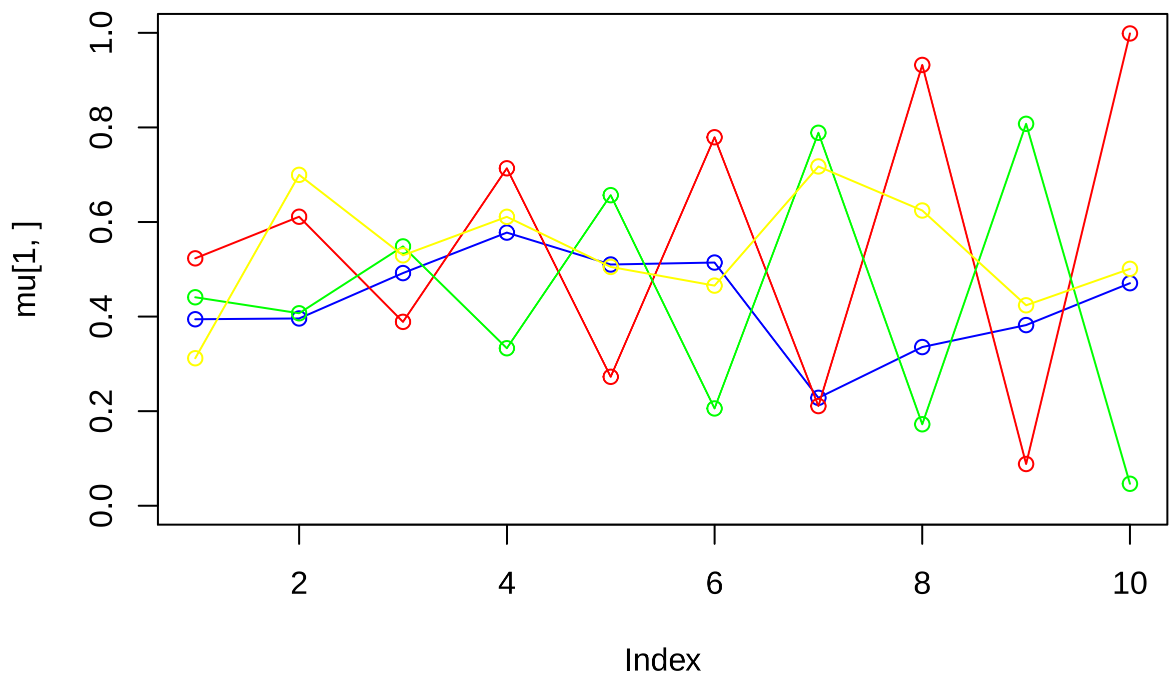

```
## iteration:  71 log likelihood:  -5578.519
```



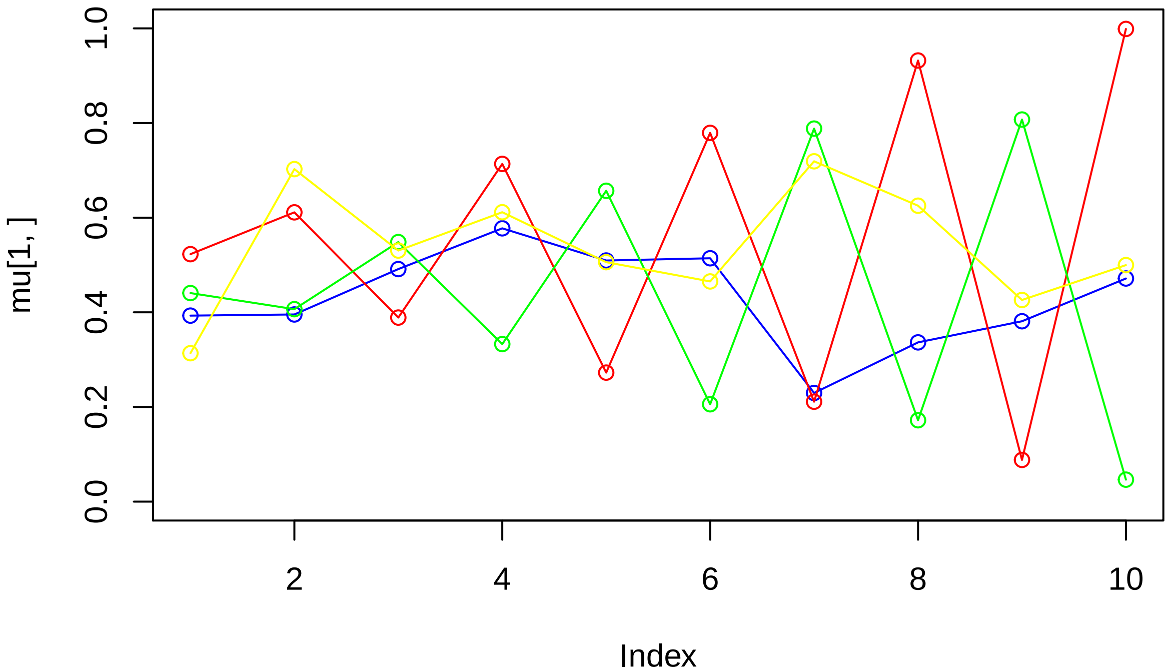

```
## iteration:  72 log likelihood:  -5577.847
```



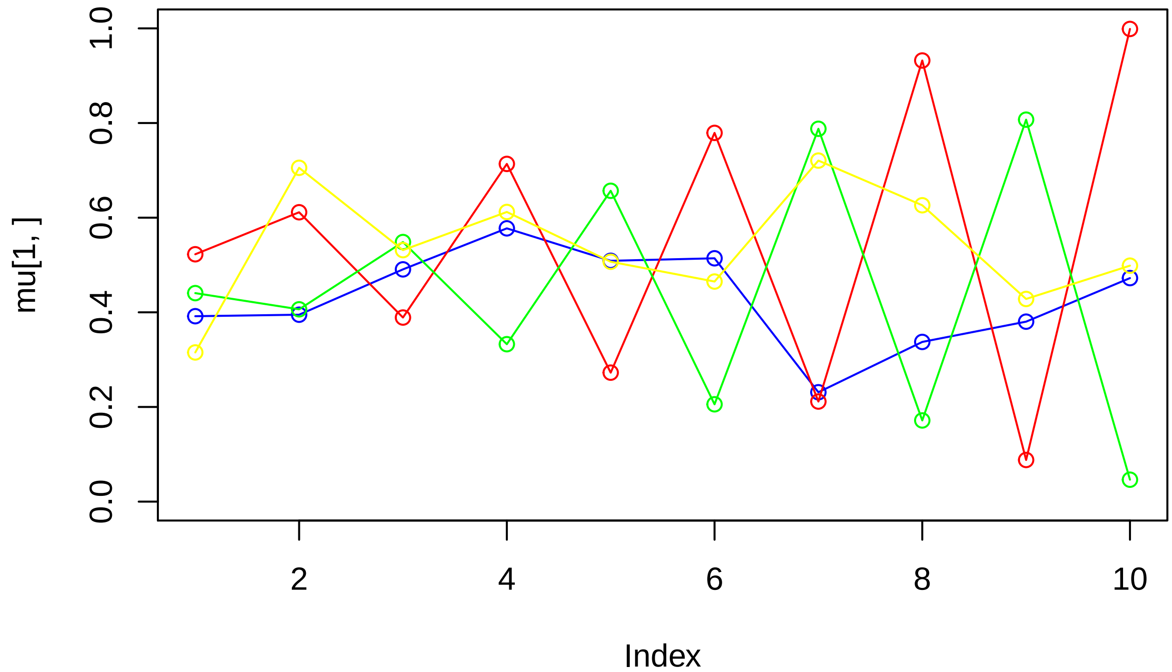

```
## iteration:  73 log likelihood:  -5577.263
```



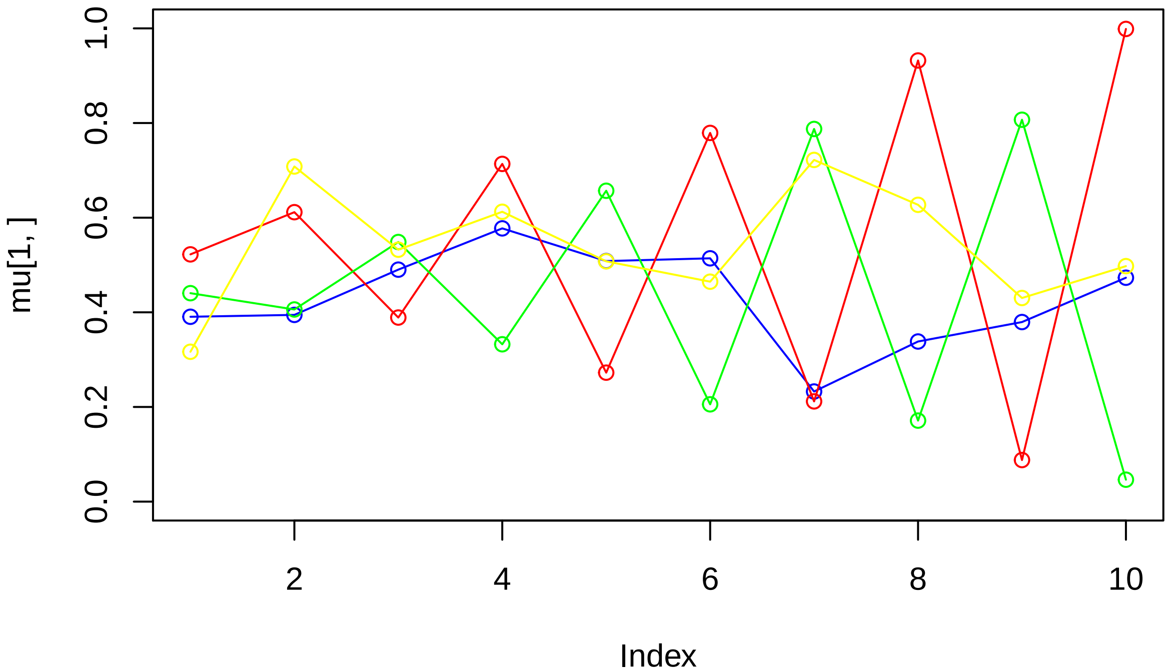

```
## iteration:  74 log likelihood:  -5576.755
```



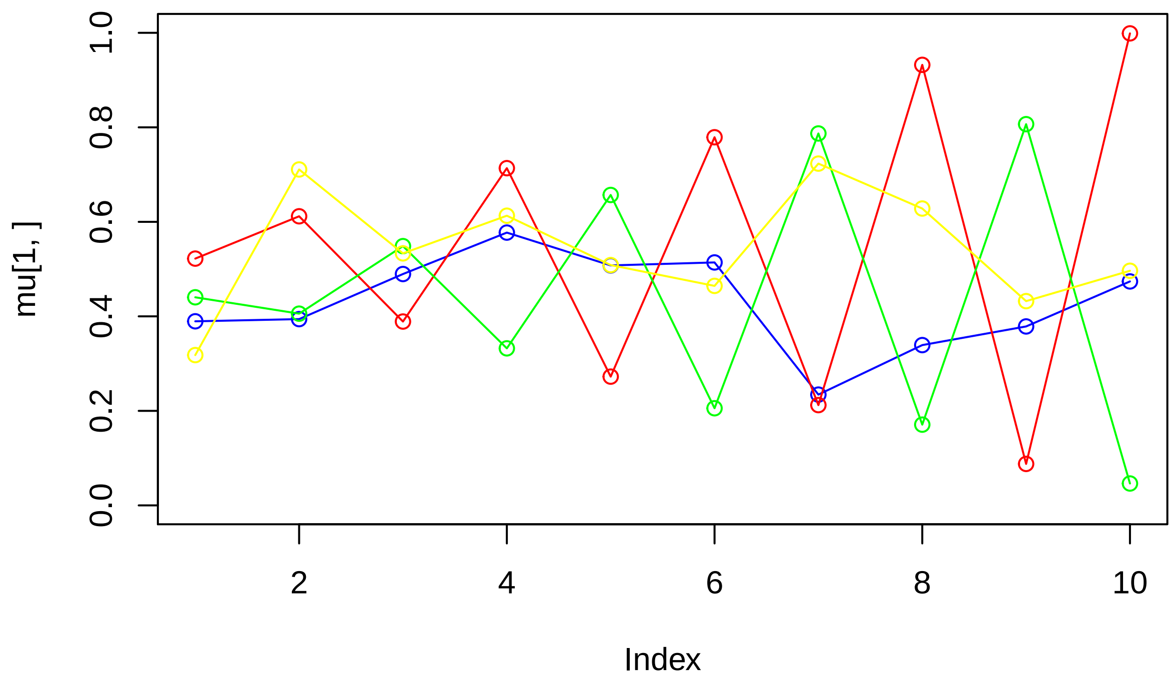

```
## iteration:  75 log likelihood:  -5576.308
```



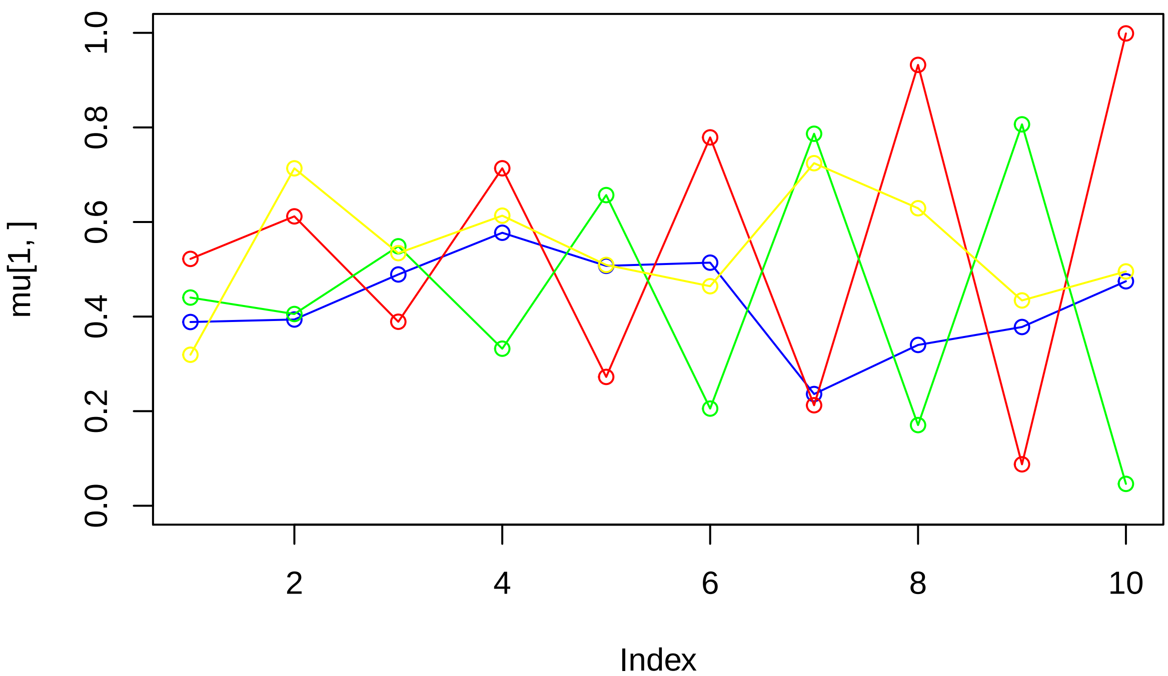

```
## iteration:  76 log likelihood:  -5575.911
```



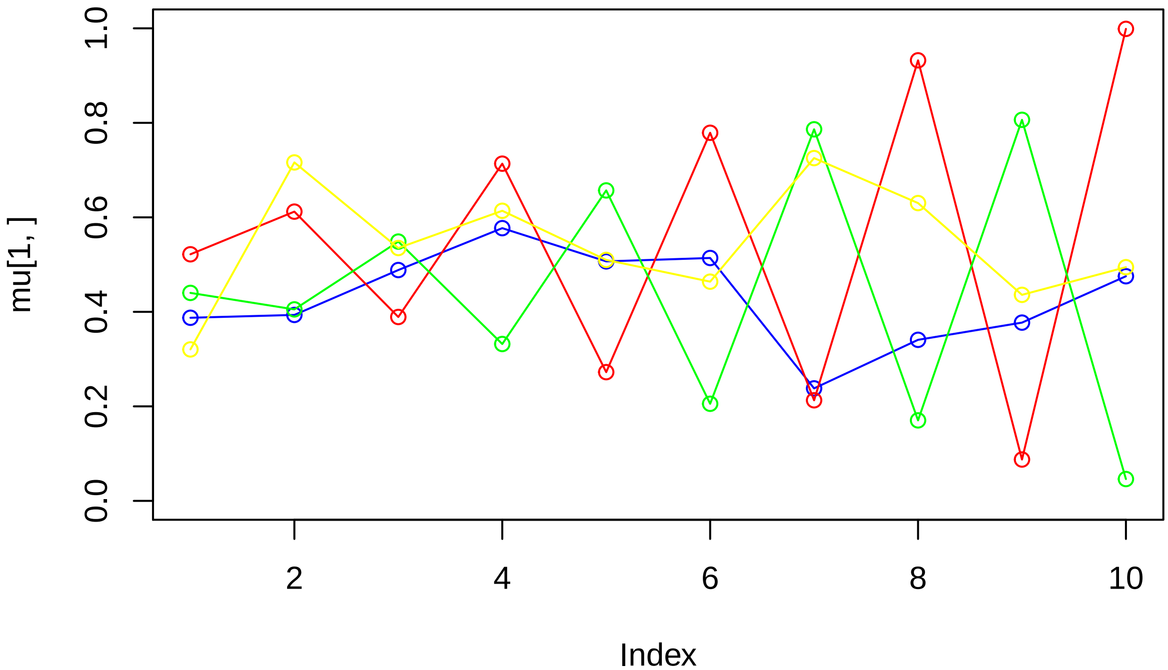

```
## iteration:  77 log likelihood:  -5575.555
```



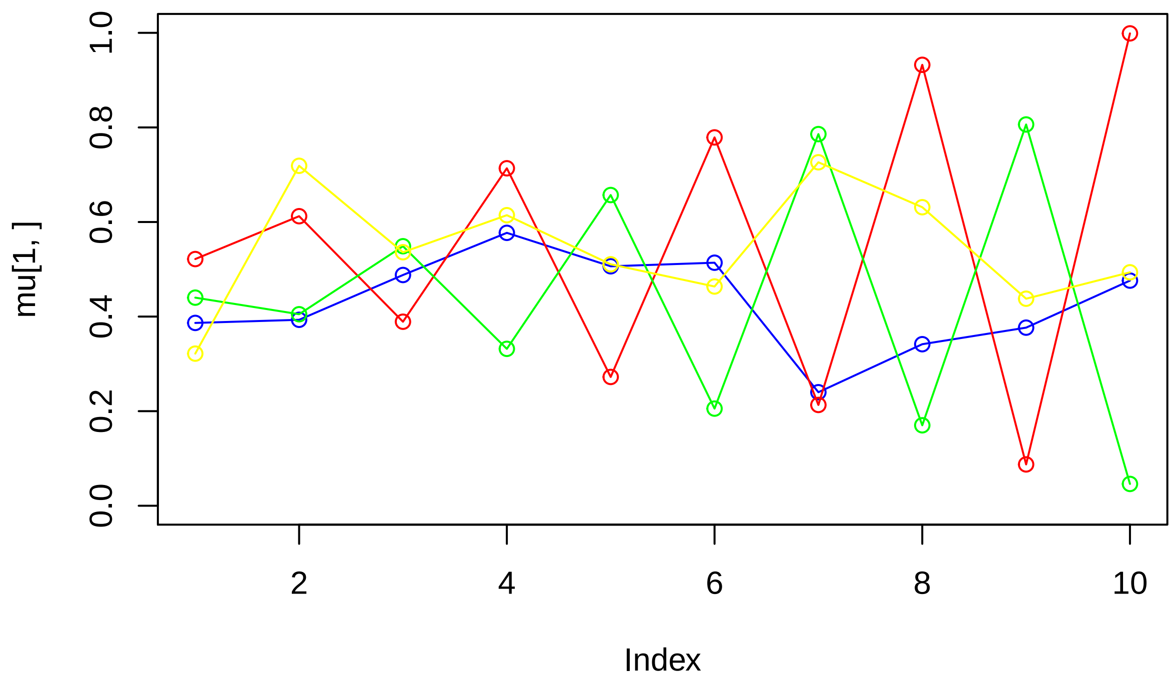

```
## iteration:  78 log likelihood:  -5575.229
```



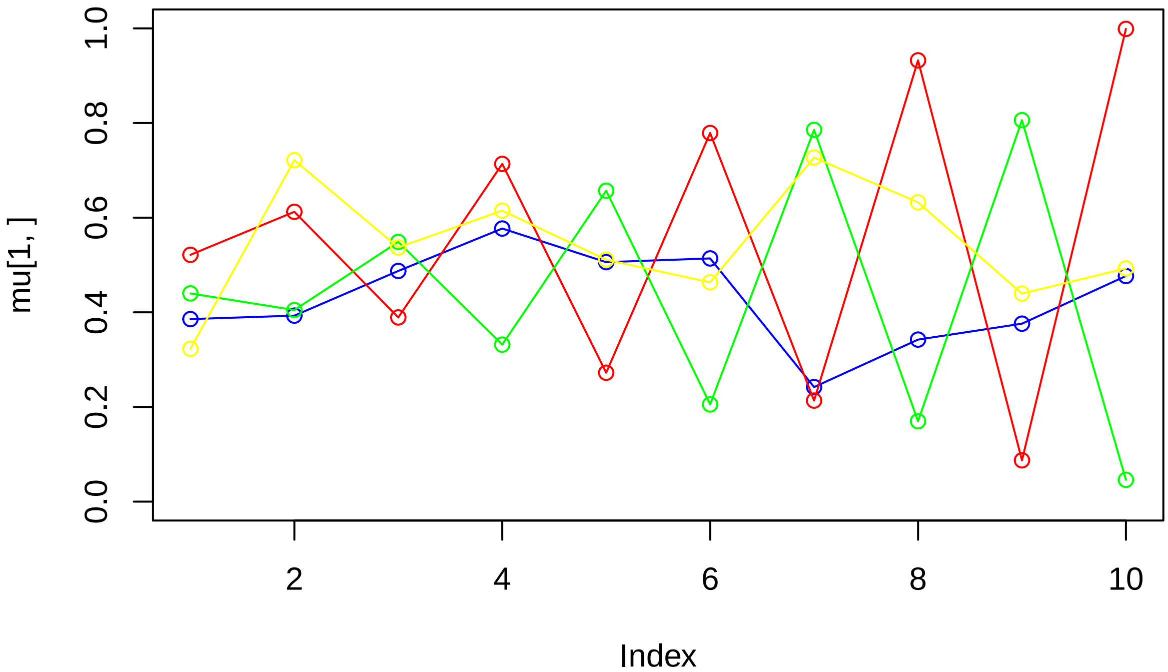

```
## iteration:  79 log likelihood:  -5574.926
```



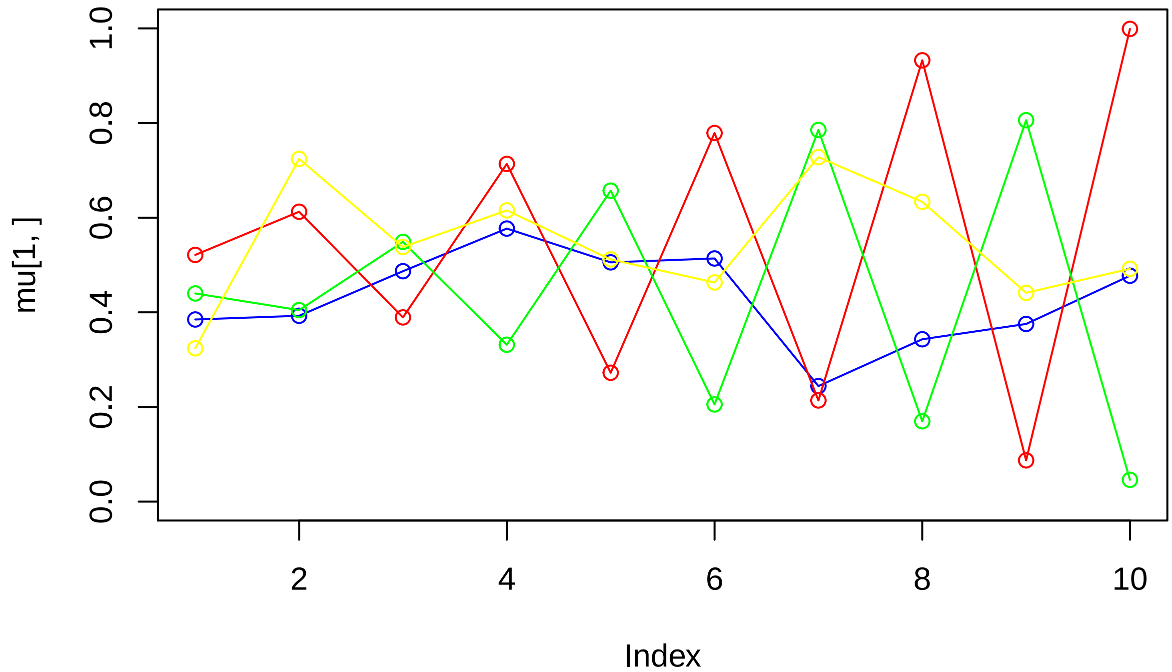

```
## iteration:  80 log likelihood:  -5574.641
```



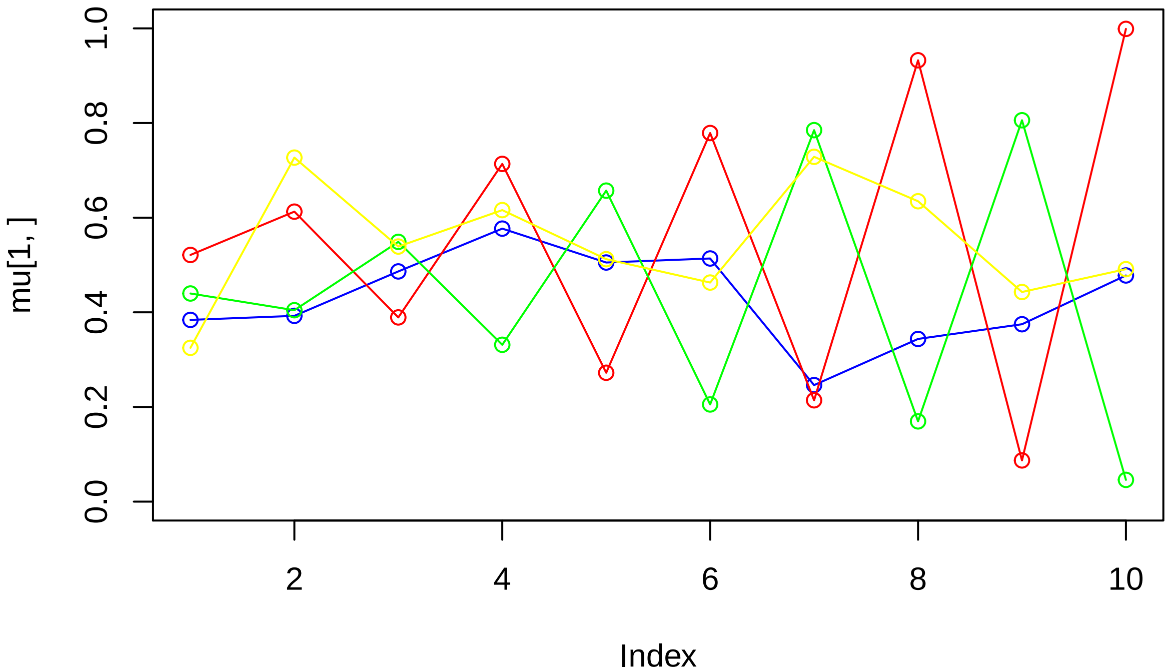

```
## iteration:  81 log likelihood:  -5574.366
```



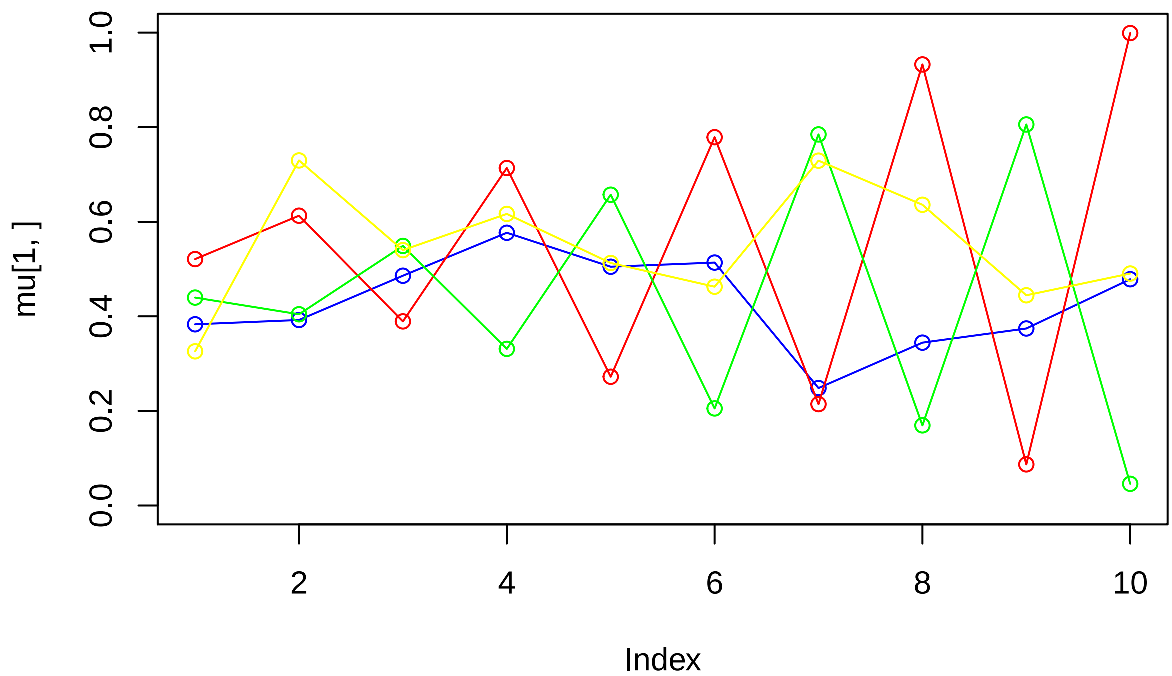

```
## iteration:  82 log likelihood:  -5574.097
```



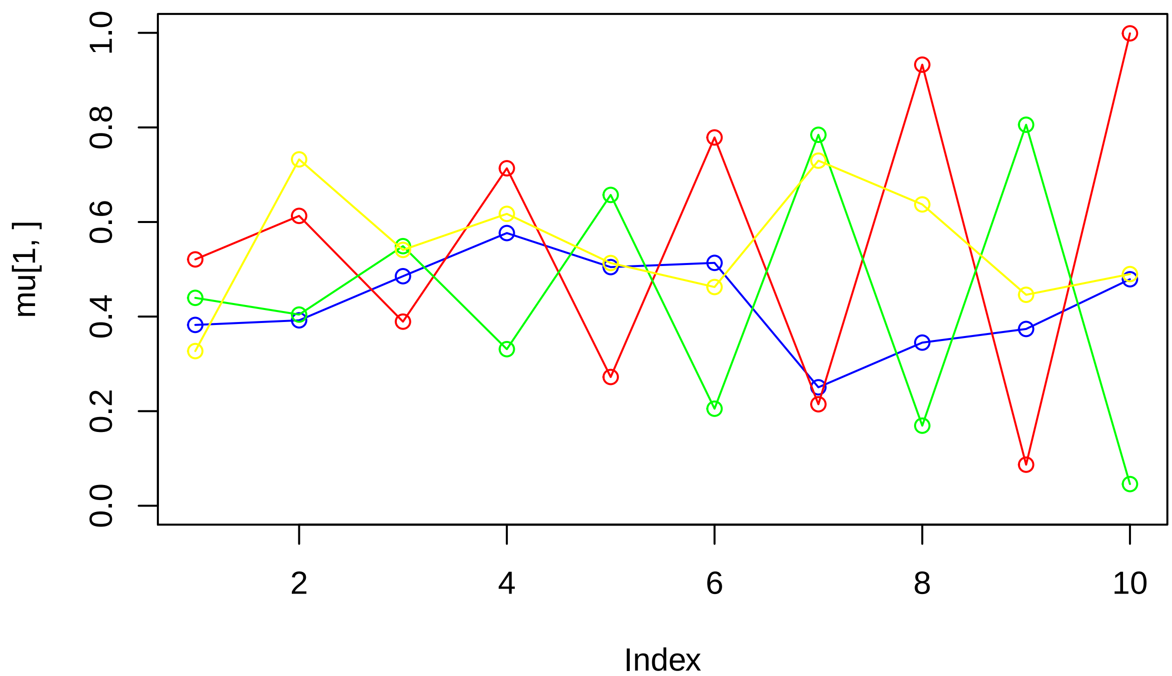

```
## iteration:  83 log likelihood:  -5573.831
```



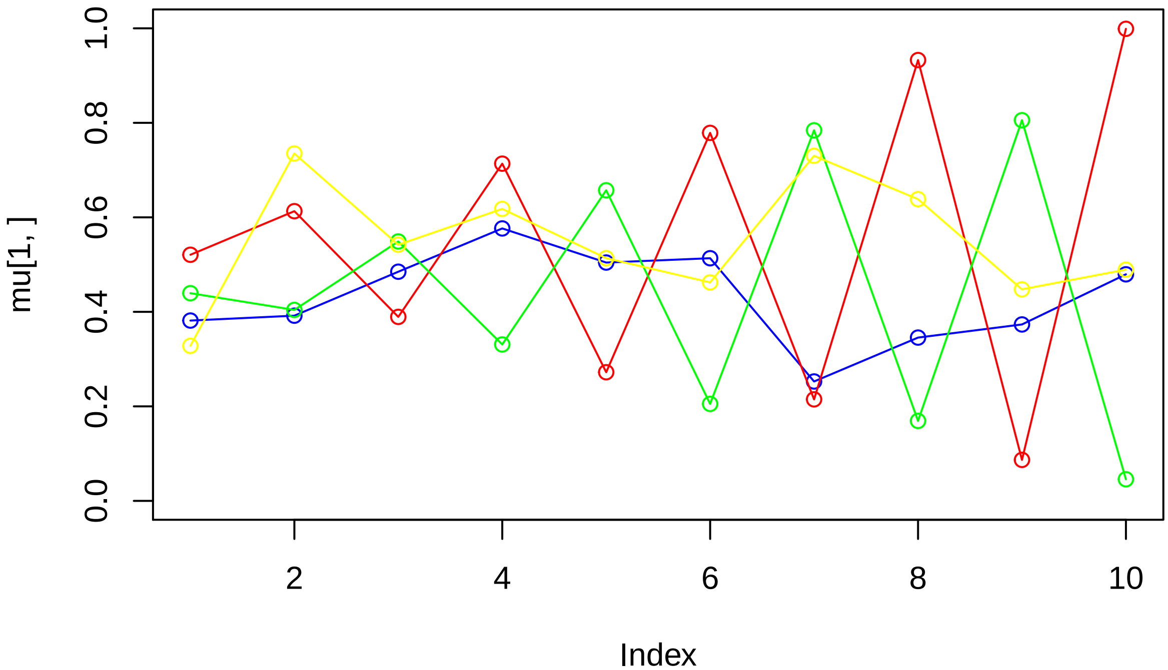

```
## iteration:  84 log likelihood:  -5573.563
```



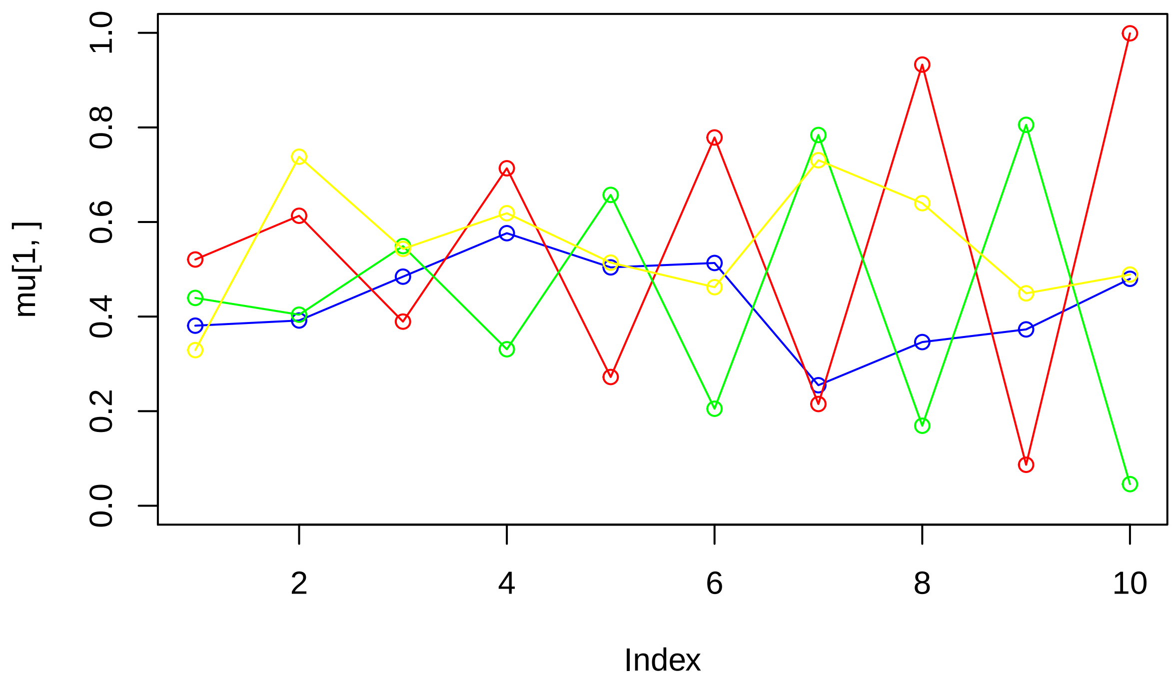

```
## iteration:  85 log likelihood:  -5573.291
```



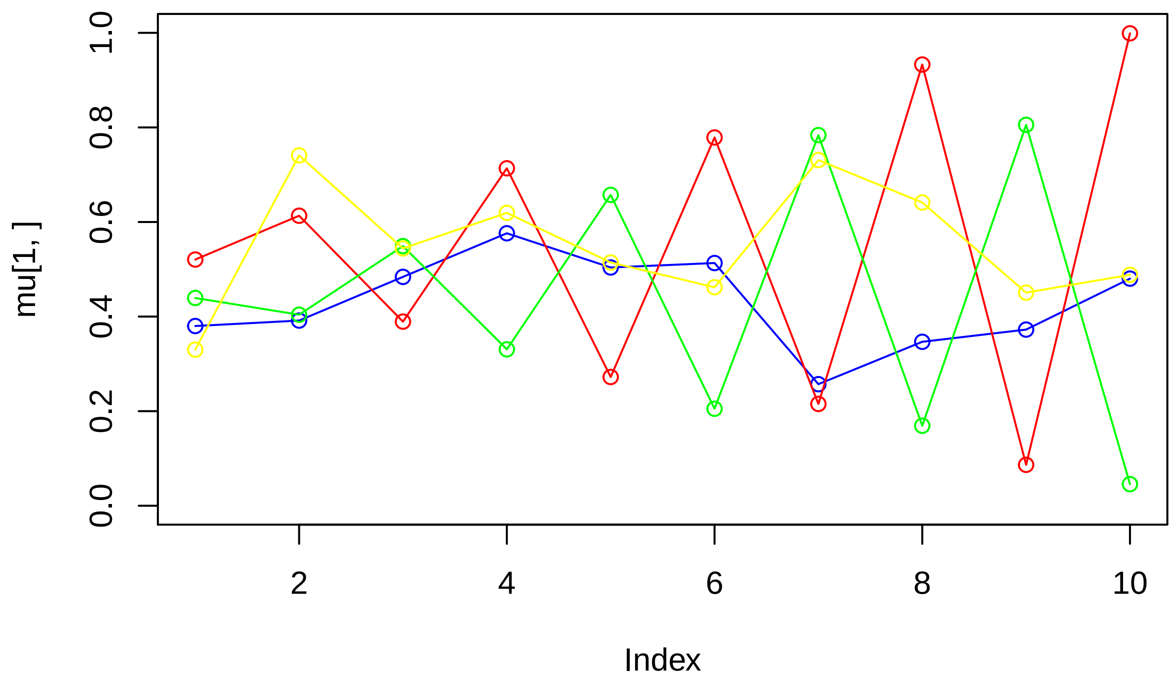

```
## iteration:  86 log likelihood:  -5573.014
```



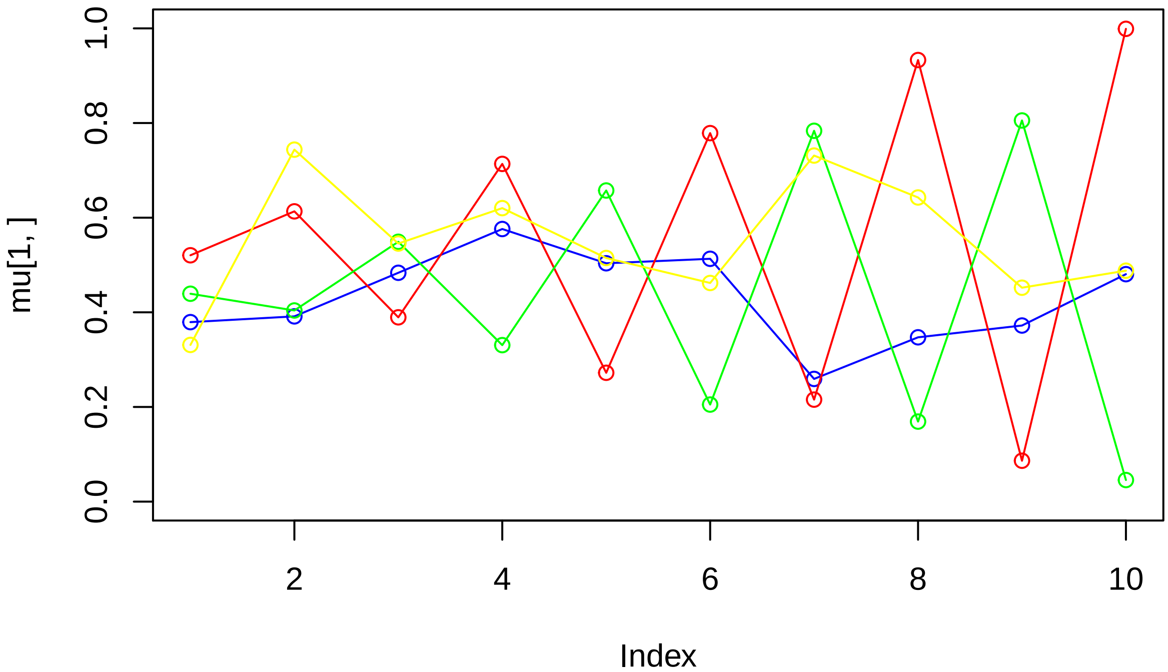

```
## iteration:  87 log likelihood:  -5572.727
```



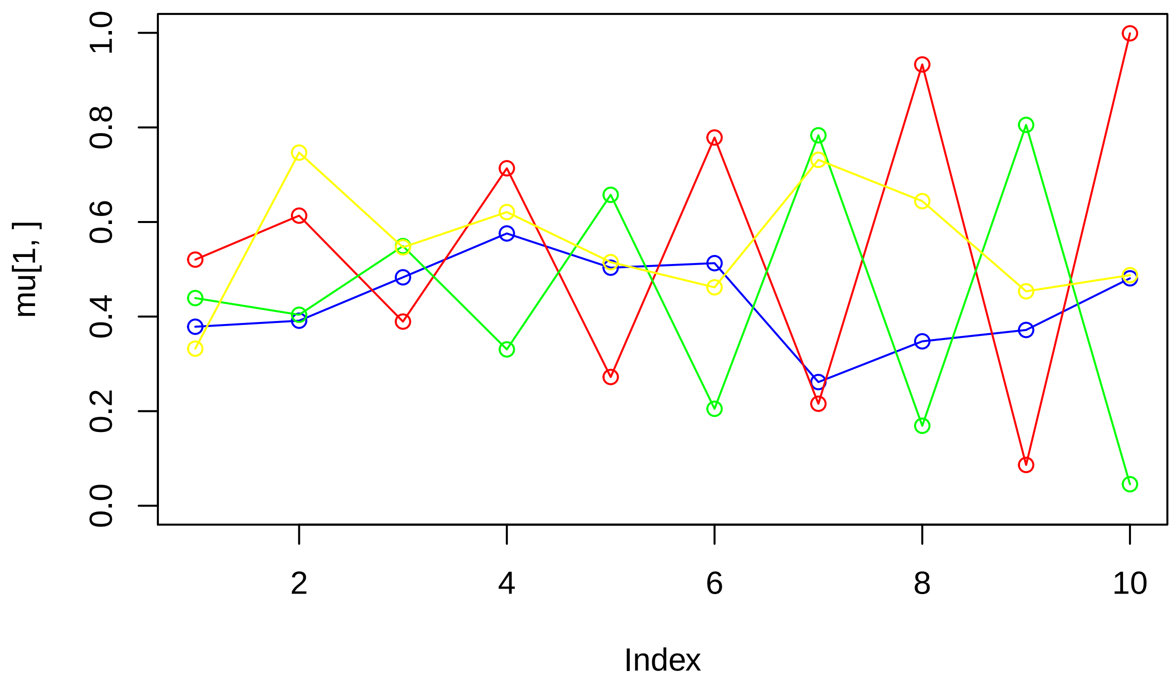

```
## iteration:  88 log likelihood:  -5572.431
```



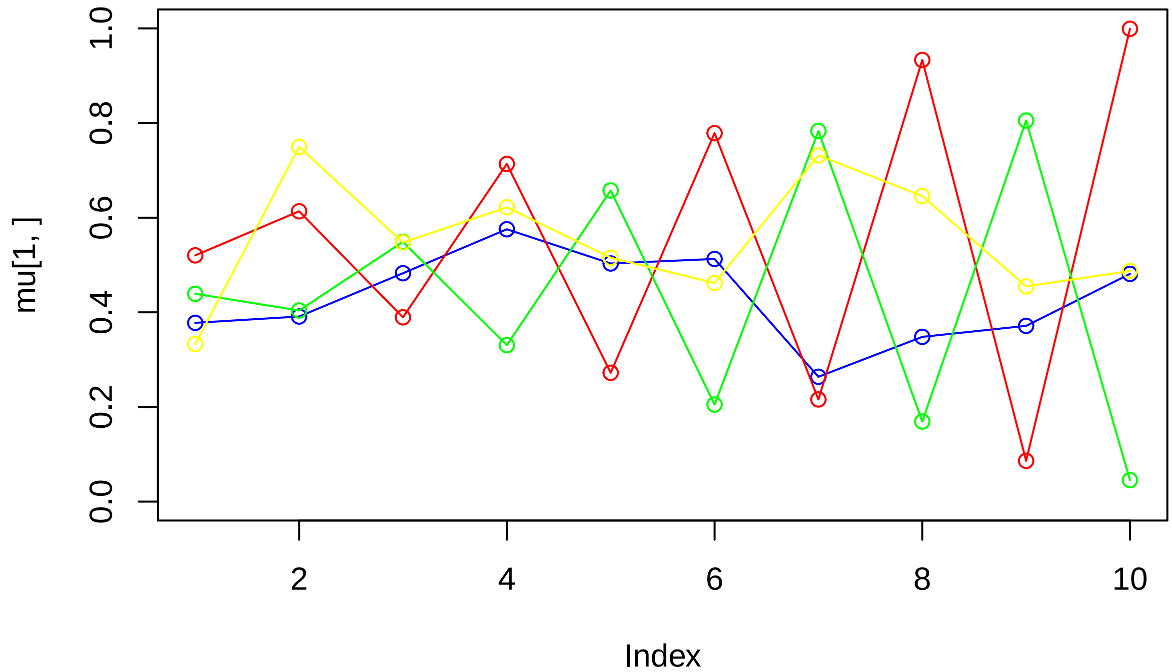

```
## iteration:  89 log likelihood:  -5572.124
```



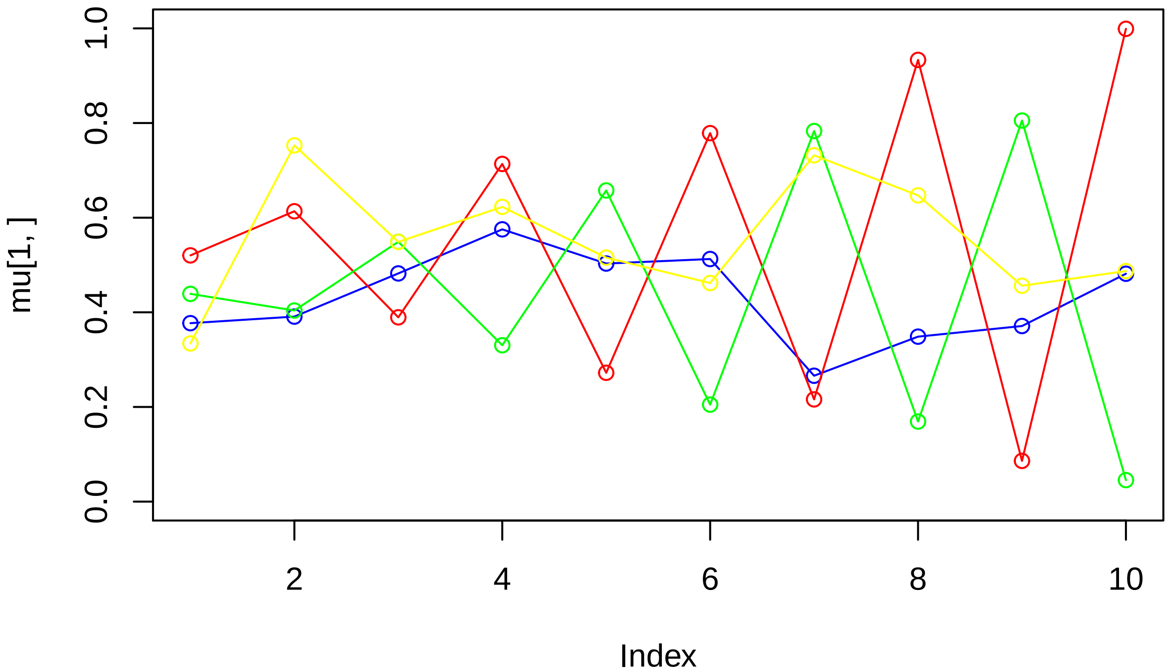

```
## iteration:  90 log likelihood:  -5571.804
```



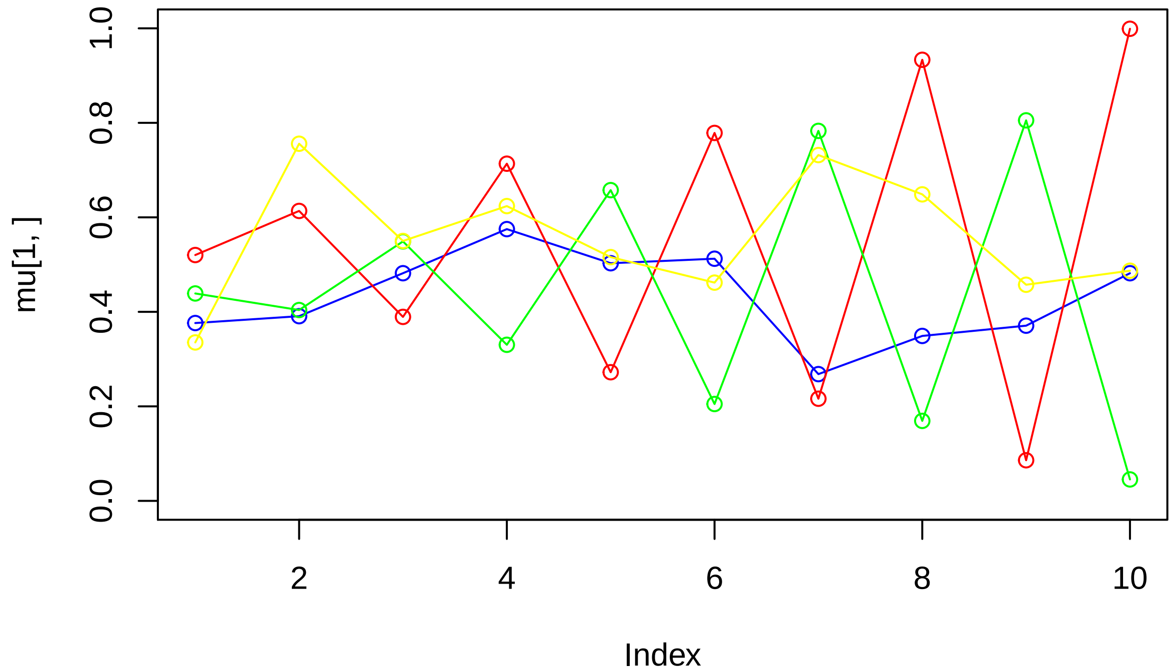

```
## iteration:  91 log likelihood:  -5571.471
```



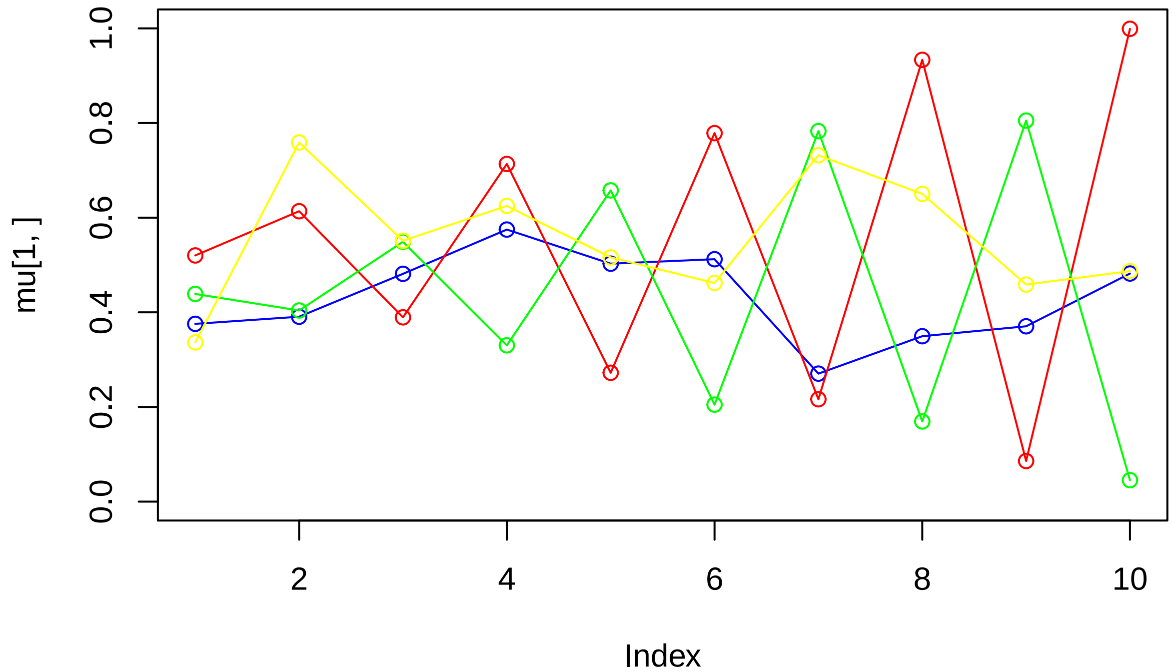

```
## iteration:  92 log likelihood:  -5571.123
```



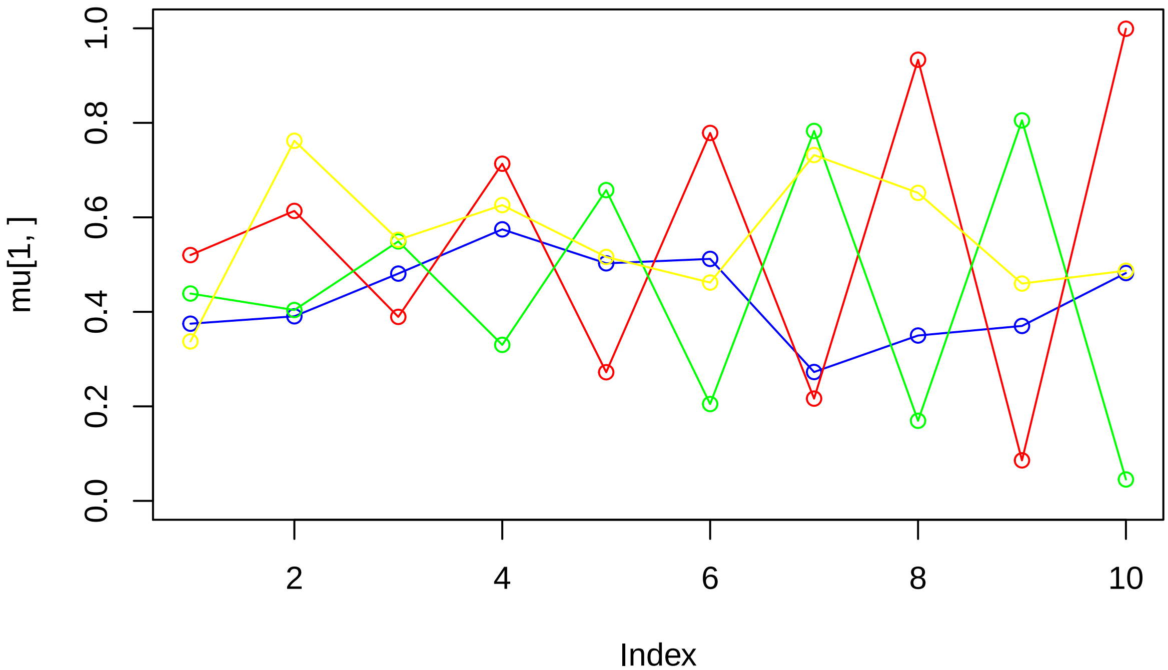

```
## iteration:  93 log likelihood:  -5570.762
```



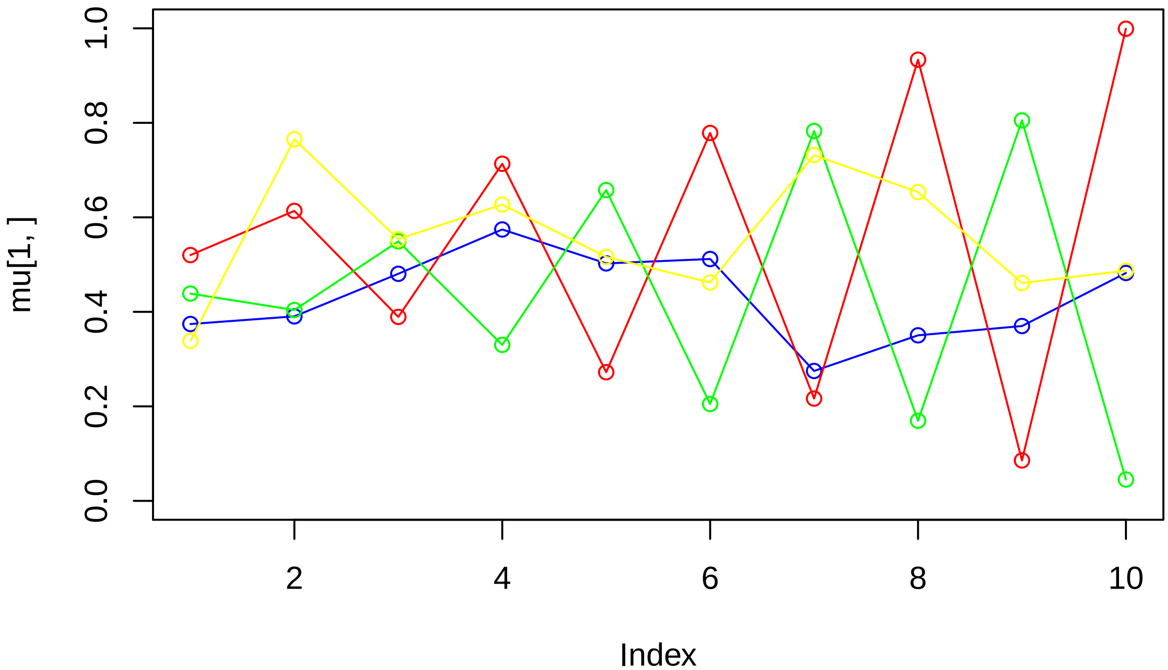

```
## iteration:  94 log likelihood:  -5570.385
```



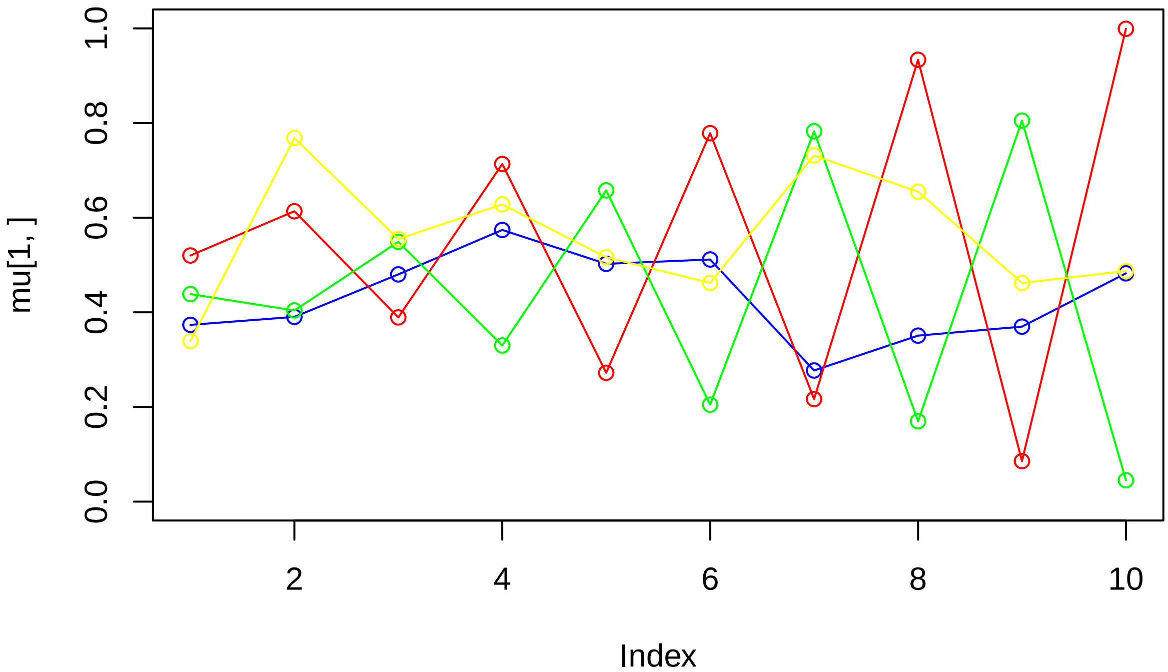

```
## iteration:  95 log likelihood:  -5569.993
```



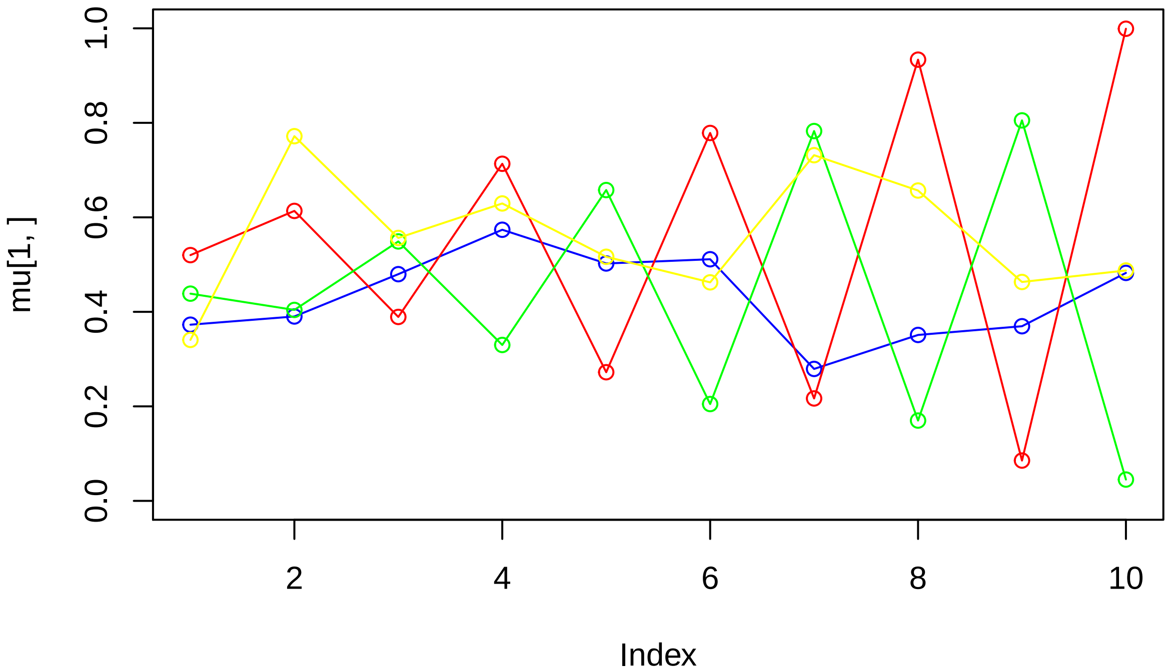

```
## iteration:  96 log likelihood:  -5569.585
```



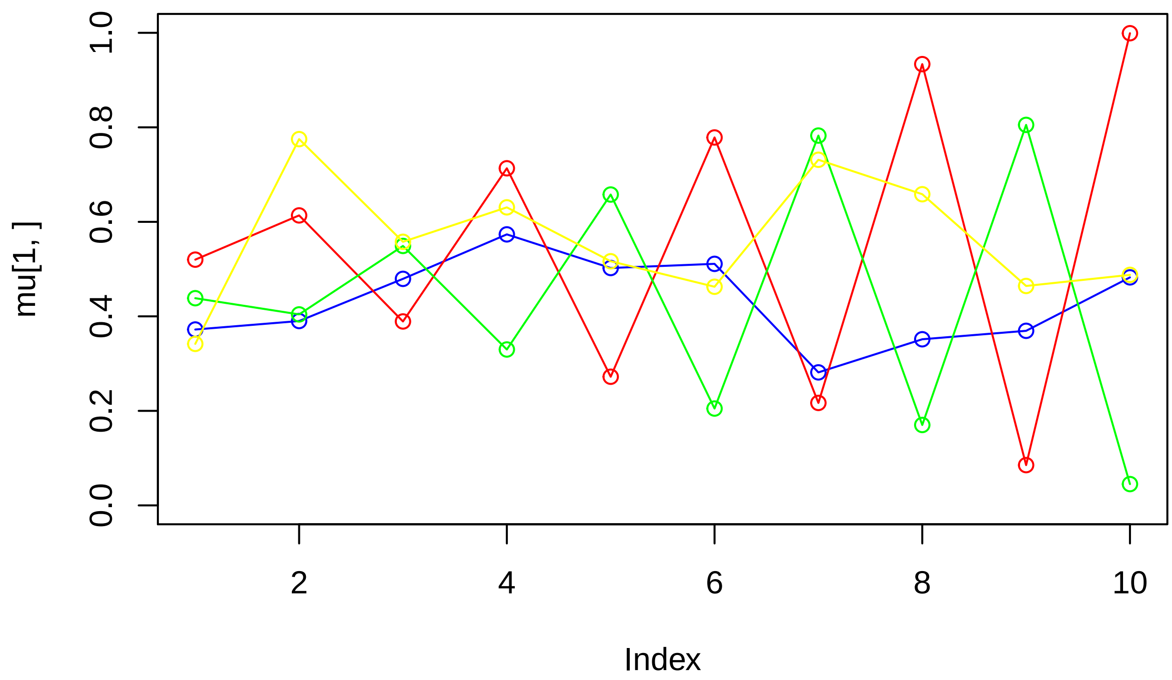

```
## iteration:  97 log likelihood:  -5569.161
```




```
## iteration:  98 log likelihood:  -5568.722
```

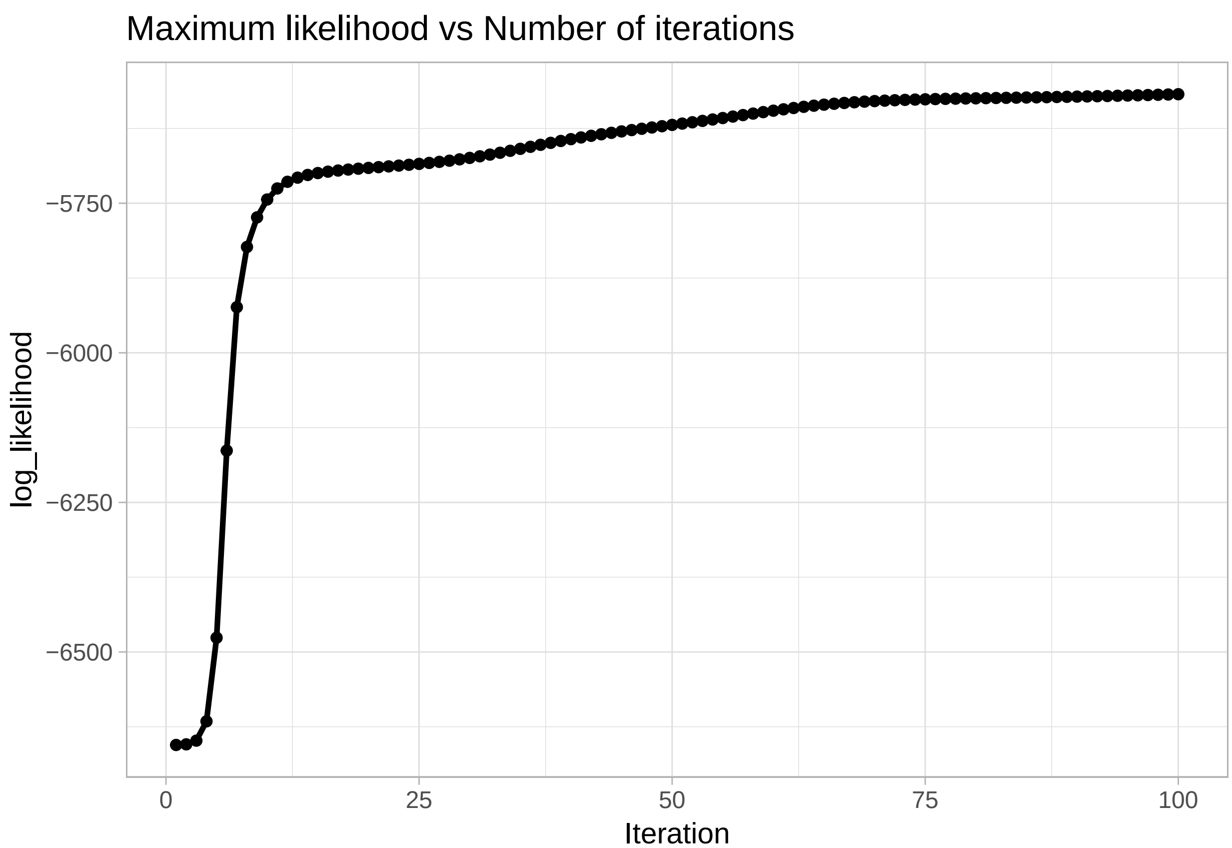



```
## iteration:  99 log likelihood:  -5568.267
```




```
## iteration:  100 log likelihood:  -5567.797
```

```
EM_4$plot
```




```
EM_4$pi_ML
```

```
## [1] 0.2880470 0.2533761 0.2933710 0.1652060
```

```
EM_4$mu_ML
```

```
##           [,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
```

Appendix

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

options(scipen = 999)
```

```r
spam_data <- read.csv(file = "spambase.data", header = FALSE)
colnames(spam_data)[58] <- "Spam"
spam_data$Spam <- factor(spam_data$Spam, levels = c(0,1), labels = c("0", "1"))
set.seed(12345)
n =  NROW(spam_data)
id = sample(1:n, floor(n*(2/3)))
train = spam_data[id,]
test = spam_data[-id,]

final_result <- NULL
for(i in seq(from = 10, to = 100, by = 10)){

ada_model <- mboost::blackboost(Spam~.,
                                 data = train,
                                 family = AdaExp(),
                               control=boost_control(mstop=i))

forest_model <- randomForest(Spam~., data = train, ntree = i)


prediction_function <- function(model, data){
  predicted <- predict(model, newdata = data, type = c("class"))
  predict_correct <- ifelse(data$Spam == predicted, 1, 0)
  score <- sum(predict_correct)/NROW(data)
  return(score)
}


train_ada_model_predict <- predict(ada_model, newdata = train, type = c("class"))
test_ada_model_predict <- predict(ada_model, newdata = test, type = c("class"))
train_forest_model_predict <- predict(forest_model, newdata = train, type = c("class"))
test_forest_model_predict <- predict(forest_model, newdata = test, type = c("class"))

test_predict_correct <- ifelse(test$Spam == test_forest_model_predict, 1, 0)
train_predict_correct <- ifelse(train$Spam == train_forest_model_predict, 1, 0)


train_ada_score <-  prediction_function(ada_model, train)
test_ada_score <-  prediction_function(ada_model, test)
train_forest_score <-  prediction_function(forest_model, train)
test_forest_score <-  prediction_function(forest_model, test)

iteration_result <- data.frame(number_of_trees = i,
                               accuracy = c(train_ada_score,
                                            test_ada_score,
                                            train_forest_score,
                                            test_forest_score),
                               type  = c("train", "test", "train", "test"),
                               model = c("ADA", "ADA",  "Forest", "Forest"))


final_result <- rbind(iteration_result, final_result)
}
```

```r
final_result$error_rate_percentage <- 100*(1 - final_result$accuracy)
ggplot(data = final_result, aes(x = number_of_trees,
                                y = error_rate_percentage,
                                group = type, color = type)) +
  geom_point() +
  geom_line() +
  ggtitle("Error Rate vs. increase in trees") + facet_grid(rows = vars(model))

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
```

```r
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)
em_mat <- function(k){
set.seed(1234567890)
# max number of EM iterations
max_it <- 100
# min change in log likelihood between two consecutive EM iterations
min_change <- 0.1
#------------- Producing Training data and Initialization ----------------#
```

```r
# number of training points
N <- 1000
# number of dimensions
D <- 10
# training data
x <- matrix(nrow=N, ncol=D)
# true mixing coefficients
true_pi <- vector(length = k)
true_pi <- rep(1/k, k)
# true conditional distributions
true_mu <- matrix(nrow = k, ncol = D)
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) {
l <- sample(1:k,1,prob=true_pi)
for(d in 1:D) {
x[n,d] <- rbinom(1,1,true_mu[l,d])
}
}
# fractional component assignments
z <- matrix(nrow = N, ncol = k)
# mixing coefficients
pi <- vector(length = k)
# conditional distributions
mu <- matrix(nrow = k, ncol = D)
# log likelihood of the EM iterations
llik <- vector(length = max_it)
# Random initialization of the paramters
pi <- runif(k,0.49,0.51)
pi <- pi / sum(pi)
for(i in 1:k) {
mu[i,] <- runif(D,0.49,0.51)
}
```

```r
#------------------------ Iteration stage -----------------------#
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
# Updating z matrix
p_Xn_MUn <- exp(x %*% log(t(mu)) + (1 - x) %*% log(1 - t(mu)))
numerator <- matrix(rep(pi,N), ncol = k, byrow = TRUE) * p_Xn_MUn
denominator <- rowSums(numerator)
Z_nk <- numerator/denominator
# Updating pi
pi <- colSums(Z_nk)/N
# Updating mu
mu <- (t(Z_nk) %*% x)/colSums(Z_nk)
#Log likelihood computation.
llik[it] <- sum(Z_nk * ((x %*% log(t(mu)) + (1 - x) %*% log(1 - t(mu))
) + matrix(rep(pi,N), ncol = k, byrow = TRUE)))
cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
flush.console()
# Stop if the log likelihood has not changed significantly
if(it >= 2){
if((llik[it] - llik[it-1]) < min_change){break()}
}
#M-step: ML parameter estimation from the data and fractional component assignments
# pi_ML
pi_ML <- pi
#mu_ML
mu_ML <- mu
}
#------------------------ output stage -----------------------#
df <- data.frame(Iteration = 1:length(llik[which(llik != 0.000)])
, log_likelihood = llik[which(llik != 0.000)])
plot <- ggplot(data = df) +
geom_point(mapping = aes(x = Iteration, y = log_likelihood),
color = 'black') +
geom_line(mapping = aes(x = Iteration, y = log_likelihood),
color = 'black', size = 1) +
ggtitle('Maximum likelihood vs Number of iterations') +
theme(plot.title = element_text(hjust = 0.5)) +
theme_light()
output <- list(pi_ML = pi_ML,
mu_ML = mu_ML,
```

```
plot = plot
)
output
}

EM_2 <- em_mat(2)
EM_2$plot
EM_2$pi_ML
EM_2$mu_ML
EM_3 <- em_mat(3)
EM_3$plot
EM_3$pi_ML
EM_3$mu_ML
EM_4 <- em_mat(4)
EM_4$plot
EM_4$pi_ML
EM_4$mu_ML
```