ST340 Assignment 2

Xintian Han 1909780, Runze Wang 1907544, Jingyuan Chen 2029628

26/02/2023

```
library(mvtnorm)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.0     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.1     v tibble    3.1.8
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

Question 1

(a)

$$f(\mu_{1:K}) = f(\mu_1, ..., \mu_K) = \sum_{i=1}^{n}\sum_{k=1}^{K}\gamma_{ik} logp(x_i|\mu_k)$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{K}\gamma_{ik} log \prod_{j=1}^{p}\mu_{kj}^{xij}(1-\mu_{kj})^{1-xij}$$

We differentiate the equation wrt $\mu_{1:K}$:

$$\frac{\partial}{\partial \mu_{1:K}} f(\mu_{1:K}) = \frac{\partial}{\partial \mu_{kj}} \sum_{i=1}^{n}\sum_{k=1}^{K}\gamma_{ik} log \prod_{j=1}^{p}\mu_{kj}^{xij}(1-\mu_{kj})^{1-xij} = \frac{\partial}{\partial \mu_{kj}} \sum_{i=1}^{n}\sum_{k=1}^{K}\sum_{j=1}^{p}\gamma_{ik} log \mu_{kj}^{xij}(1-\mu_{kj})^{1-xij}$$

$$= \frac{\partial}{\partial \mu_{kj}} \sum_{i=1}^{n}\gamma_{ik} log \mu_{kj}^{xij}(1-\mu_{kj})^{1-xij} = \sum_{i=1}^{n}\gamma_{ik}(\frac{x_{ij}}{\mu_{kj}} - \frac{1-x_{ij}}{1-\mu_{kj}}) = 0$$

$$\sum_{i=1}^{n}\gamma_{ik}x_{ij} - \sum_{i=1}^{n}\gamma_{ik}x_{ij}\mu_{kj} = \mu_{kj}\sum_{i=1}^{n}\gamma_{ik}(1-x_{ij})$$

$$\mu_{kj} = \frac{\sum_{i=1}^{n}\gamma_{ik}x_{ij}}{\sum_{i=1}^{n}\gamma_{ik}}$$

Therefore

$$\mu_{k} = \frac{\sum_{i=1}^{n}\gamma_{ik}x_{i}}{\sum_{i=1}^{n}\gamma_{ik}}$$

(bi)

In this part, we are going to use some part of the code from the lab4 and creating the mixture bernoulli distribution and the EM algorithm.

```
load("~/Downloads/20newsgroups.rdata")
## EM algorithm for a mixture of multivariate Bernoulli

# fix random number generator (seed)


## logsumexp(x) returns log(sum(exp(x))) but performs the computation in a more stable manner
logsumexp <- function(x) return(log(sum(exp(x - max(x)))) + max(x))


# we create the density of multivariate bernoulli distribution
bernoulli <- function(a) {
  b <- prod(a)
  return(b)
}

#we can test the bernoulli function is actually working
bernoulli(c(0.4,0.6))
```

```
## [1] 0.24
```

```
bernoulli(c(1,0))
```

```
## [1] 0
```

```r
# compute the log-likelihood
compute_ll <- function(xs,mus,lws,gammas) {
  ll <- 0
  n <- dim(xs)[1]
  K <- dim(mus)[1]
  v1 <- rep(1,dim(xs)[2])
  for (i in 1:n) {
    for (k in 1:K) {
      if (gammas[i,k] > 0) {
        ll <- ll +
  gammas[i,k]*(lws[k]+log(bernoulli((mus[k,])^(xs[i,])*(v1-mus[k,])^(v1-xs[i,])))-log(gammas[i,k]))
      }
    }
  }
  return(ll)
}

# compute the log-likelihood directly
compute_ll.direct <- function(xs,mus,lws) {
  ll <- 0
  v1 <- rep(1,dim(xs)[2])
  n <- dim(xs)[1]
  K <- dim(mus)[1]
  for (i in 1:n) {
    s <- 0
    for (k in 1:K) {
      s <- s + exp(lws[k])*(bernoulli((mus[k,])^(xs[i,])*((v1-mus[k,])^(v1-xs[i,]))))
    }
    ll <- ll + log(s)
  }
  return(ll)
}

em_mix_bernoulli <- function(xs,K,max.numit=Inf) {
  p <- dim(xs)[2]
  n <- dim(xs)[1]
  v1 <- rep(1,dim(xs)[2])
  # lws is log(ws)
  # we work with logs to keep the numbers stable
  # start off with ws all equal
  lws <- rep(log(1/K),K)

  # start off with a random selection of cluster means as a matrix from uniform distribution
  mus <- matrix(data = runif(n = K * p), nrow = K, ncol = p)

  # gammas will be set in the first iteration
  gammas <- matrix(0,n,K)

  converged <- FALSE
  numit <- 0
  ll <- -Inf
  print("iteration : log-likelihood")
  while(!converged && numit < max.numit) {
```

```
    numit <- numit + 1
    mus.old <- mus
    ll.old <- ll

    ## E step - calculate gammas
    for (i in 1:n) {
      # the elements of lprs are log(w_k * p_k(x)) for each k in {1,...K}
      lprs <- rep(0,K)
      for (k in 1:K) {
        lprs[k] <- lws[k] + log(bernoulli((mus[k,])^(xs[i,])*((v1-mus[k,])^(v1-xs[i,]))))
      }
      # gammas[i,k] = w_k * p_k(x) / sum_j {w_j * p_j(x)}
      gammas[i,] <- exp(lprs - logsumexp(lprs))
    }

    ll <- compute_ll(xs,mus,lws,gammas)
    # we could also compute the log-likelihood directly below
    # ll <- compute_ll.direct(xs,mus,Sigmas,lws)

    # M step - update ws, mus
    Ns <- rep(0,K)
    for (k in 1:K) {
      Ns[k] <- sum(gammas[,k])
      lws[k] <- log(Ns[k])-log(n)

      mus[k,] <- rep(0,p)

      for (i in 1:n) {
        mus[k,] <- mus[k,]+gammas[i,k]/Ns[k]*xs[i,]

        # potential non positive definite matrix and non simmetric
        # just a triangle is needed to computed, the rest is the mirror of it
      }

    }
    print(paste(numit,": ",ll))

    mus[which(mus > 1,arr.ind=TRUE)] <- 1 - 1e-15

    # we stop once the increase in the log-likelihood is "small enough"

    if (abs(ll-ll.old) < 1e-5) converged <- TRUE
  }



  return(list(lws=lws,mus=mus,gammas=gammas,ll=ll))

}
```

Then we run the algorithm for K=4, could get the following. We finally observed that , after a large number of iterations , the log-likelihood converged.

```
out10<- em_mix_bernoulli(documents,4)
```

```
## [1] "iteration : log-likelihood"
## [1] "1 :  -1632751.21607543"
## [1] "2 :  -253149.26830469"
## [1] "3 :  -247914.610117651"
## [1] "4 :  -244186.926817361"
## [1] "5 :  -242235.334407512"
## [1] "6 :  -241084.57052239"
## [1] "7 :  -240129.223837677"
## [1] "8 :  -239394.093420295"
## [1] "9 :  -238820.977956918"
## [1] "10 :  -238381.552127271"
## [1] "11 :  -238054.950165742"
## [1] "12 :  -237785.995331524"
## [1] "13 :  -237595.616300629"
## [1] "14 :  -237457.885433961"
## [1] "15 :  -237351.64368731"
## [1] "16 :  -237269.575571303"
## [1] "17 :  -237201.623493264"
## [1] "18 :  -237145.266444196"
## [1] "19 :  -237059.833268799"
## [1] "20 :  -236994.221667642"
## [1] "21 :  -236947.810964349"
## [1] "22 :  -236912.048823549"
## [1] "23 :  -236881.872085226"
## [1] "24 :  -236856.437125067"
## [1] "25 :  -236836.147457716"
## [1] "26 :  -236820.312640813"
## [1] "27 :  -236807.585323002"
## [1] "28 :  -236796.42489816"
## [1] "29 :  -236783.338619986"
## [1] "30 :  -236762.74747253"
## [1] "31 :  -236751.879446035"
## [1] "32 :  -236744.765455592"
## [1] "33 :  -236739.690017192"
## [1] "34 :  -236735.761754824"
## [1] "35 :  -236732.591524037"
## [1] "36 :  -236729.646410278"
## [1] "37 :  -236726.138311599"
## [1] "38 :  -236723.28462876"
## [1] "39 :  -236721.333366475"
## [1] "40 :  -236719.781487455"
## [1] "41 :  -236718.393065824"
## [1] "42 :  -236717.048969761"
## [1] "43 :  -236715.697204087"
## [1] "44 :  -236714.335757272"
## [1] "45 :  -236712.946492059"
## [1] "46 :  -236711.431596592"
## [1] "47 :  -236709.592522751"
## [1] "48 :  -236707.205140787"
## [1] "49 :  -236704.360081273"
## [1] "50 :  -236701.263596302"
```

```
## [1] "51 :  -236697.609695573"
## [1] "52 :  -236692.869165489"
## [1] "53 :  -236686.493067627"
## [1] "54 :  -236677.998042367"
## [1] "55 :  -236667.36264209"
## [1] "56 :  -236655.513749808"
## [1] "57 :  -236644.043283006"
## [1] "58 :  -236634.176369978"
## [1] "59 :  -236626.237901181"
## [1] "60 :  -236619.948820683"
## [1] "61 :  -236614.882711116"
## [1] "62 :  -236610.695737459"
## [1] "63 :  -236607.16302892"
## [1] "64 :  -236604.144234779"
## [1] "65 :  -236601.546520642"
## [1] "66 :  -236599.301878013"
## [1] "67 :  -236597.355998165"
## [1] "68 :  -236595.663375943"
## [1] "69 :  -236594.184954606"
## [1] "70 :  -236592.885588804"
## [1] "71 :  -236591.725720748"
## [1] "72 :  -236590.623510895"
## [1] "73 :  -236589.301485389"
## [1] "74 :  -236587.050254046"
## [1] "75 :  -236583.68317272"
## [1] "76 :  -236580.572123574"
## [1] "77 :  -236578.453619328"
## [1] "78 :  -236577.097886338"
## [1] "79 :  -236576.165389454"
## [1] "80 :  -236575.448805727"
## [1] "81 :  -236574.843685523"
## [1] "82 :  -236574.296643453"
## [1] "83 :  -236573.77418849"
## [1] "84 :  -236573.251545757"
## [1] "85 :  -236572.719945606"
## [1] "86 :  -236572.198015828"
## [1] "87 :  -236571.715219003"
## [1] "88 :  -236571.28137313"
## [1] "89 :  -236570.885588083"
## [1] "90 :  -236570.511722739"
## [1] "91 :  -236570.146295346"
## [1] "92 :  -236569.778990197"
## [1] "93 :  -236569.401127297"
## [1] "94 :  -236569.004198583"
## [1] "95 :  -236568.578716927"
## [1] "96 :  -236568.113171325"
## [1] "97 :  -236567.592840845"
## [1] "98 :  -236566.998217287"
## [1] "99 :  -236566.30273178"
## [1] "100 :  -236565.469349492"
## [1] "101 :  -236564.445360905"
## [1] "102 :  -236563.154311121"
## [1] "103 :  -236561.483394917"
## [1] "104 :  -236559.263655196"
```

```
## [1] "105 :  -236556.238565864"
## [1] "106 :  -236552.01315175"
## [1] "107 :  -236545.97115481"
## [1] "108 :  -236537.16103434"
## [1] "109 :  -236524.274448878"
## [1] "110 :  -236506.11617643"
## [1] "111 :  -236482.805557732"
## [1] "112 :  -236458.147975157"
## [1] "113 :  -236439.340774507"
## [1] "114 :  -236427.738232171"
## [1] "115 :  -236420.440505106"
## [1] "116 :  -236415.465209263"
## [1] "117 :  -236411.835528091"
## [1] "118 :  -236409.05576271"
## [1] "119 :  -236406.847011947"
## [1] "120 :  -236405.03554832"
## [1] "121 :  -236403.504744217"
## [1] "122 :  -236402.172252289"
## [1] "123 :  -236400.977809475"
## [1] "124 :  -236399.875981288"
## [1] "125 :  -236398.831454492"
## [1] "126 :  -236397.815753081"
## [1] "127 :  -236396.804783326"
## [1] "128 :  -236395.776852328"
## [1] "129 :  -236394.710917005"
## [1] "130 :  -236393.584866354"
## [1] "131 :  -236392.373644343"
## [1] "132 :  -236391.046983523"
## [1] "133 :  -236389.56642501"
## [1] "134 :  -236387.881115961"
## [1] "135 :  -236385.921539032"
## [1] "136 :  -236383.58971687"
## [1] "137 :  -236380.743287568"
## [1] "138 :  -236377.168583285"
## [1] "139 :  -236372.5331138"
## [1] "140 :  -236366.297336687"
## [1] "141 :  -236357.540638948"
## [1] "142 :  -236344.594144737"
## [1] "143 :  -236324.223536626"
## [1] "144 :  -236289.934221715"
## [1] "145 :  -236230.861831915"
## [1] "146 :  -236144.932757118"
## [1] "147 :  -236066.409773615"
## [1] "148 :  -236024.451586129"
## [1] "149 :  -236005.276715451"
## [1] "150 :  -235994.543285705"
## [1] "151 :  -235987.206104593"
## [1] "152 :  -235981.636650852"
## [1] "153 :  -235977.183868063"
## [1] "154 :  -235973.519787569"
## [1] "155 :  -235970.46269507"
## [1] "156 :  -235967.904780967"
## [1] "157 :  -235965.766364489"
## [1] "158 :  -235963.976315419"
```

```
## [1] "159 :  -235962.471286402"
## [1] "160 :  -235961.198540455"
## [1] "161 :  -235960.115989521"
## [1] "162 :  -235959.189752217"
## [1] "163 :  -235958.39113451"
## [1] "164 :  -235957.694365074"
## [1] "165 :  -235957.075222439"
## [1] "166 :  -235956.509933808"
## [1] "167 :  -235955.973649533"
## [1] "168 :  -235955.43802401"
## [1] "169 :  -235954.868009171"
## [1] "170 :  -235954.220370356"
## [1] "171 :  -235953.453464106"
## [1] "172 :  -235952.561880397"
## [1] "173 :  -235951.609033498"
## [1] "174 :  -235950.680517119"
## [1] "175 :  -235949.808826181"
## [1] "176 :  -235948.979877735"
## [1] "177 :  -235948.168510184"
## [1] "178 :  -235947.355155324"
## [1] "179 :  -235946.530701042"
## [1] "180 :  -235945.692346113"
## [1] "181 :  -235944.836808722"
## [1] "182 :  -235943.960889563"
## [1] "183 :  -235943.065149807"
## [1] "184 :  -235942.154284838"
## [1] "185 :  -235941.240261208"
## [1] "186 :  -235940.352220137"
## [1] "187 :  -235939.533985224"
## [1] "188 :  -235938.817292282"
## [1] "189 :  -235938.206336902"
## [1] "190 :  -235937.689768922"
## [1] "191 :  -235937.253675887"
## [1] "192 :  -235936.885584768"
## [1] "193 :  -235936.574806228"
## [1] "194 :  -235936.312237031"
## [1] "195 :  -235936.090166436"
## [1] "196 :  -235935.902099453"
## [1] "197 :  -235935.742588482"
## [1] "198 :  -235935.607076498"
## [1] "199 :  -235935.491756526"
## [1] "200 :  -235935.393449117"
## [1] "201 :  -235935.309497859"
## [1] "202 :  -235935.237681041"
## [1] "203 :  -235935.176137635"
## [1] "204 :  -235935.123304815"
## [1] "205 :  -235935.077864272"
## [1] "206 :  -235935.038692754"
## [1] "207 :  -235935.004809206"
## [1] "208 :  -235934.975301137"
## [1] "209 :  -235934.949189072"
## [1] "210 :  -235934.925127799"
## [1] "211 :  -235934.900699761"
## [1] "212 :  -235934.870766015"
```

```
## [1] "213 :  -235934.824065994"
## [1] "214 :  -235934.738869849"
## [1] "215 :  -235934.588568633"
## [1] "216 :  -235934.379925583"
## [1] "217 :  -235934.183184685"
## [1] "218 :  -235934.055943688"
## [1] "219 :  -235933.98681229"
## [1] "220 :  -235933.945790897"
## [1] "221 :  -235933.916340721"
## [1] "222 :  -235933.892305655"
## [1] "223 :  -235933.871590166"
## [1] "224 :  -235933.853384479"
## [1] "225 :  -235933.837273048"
## [1] "226 :  -235933.822975353"
## [1] "227 :  -235933.810270231"
## [1] "228 :  -235933.798971398"
## [1] "229 :  -235933.788917842"
## [1] "230 :  -235933.779968779"
## [1] "231 :  -235933.772000424"
## [1] "232 :  -235933.764903572"
## [1] "233 :  -235933.758581639"
## [1] "234 :  -235933.752949069"
## [1] "235 :  -235933.747929989"
## [1] "236 :  -235933.743456993"
## [1] "237 :  -235933.73947023"
## [1] "238 :  -235933.735916439"
## [1] "239 :  -235933.732748284"
## [1] "240 :  -235933.729923623"
## [1] "241 :  -235933.727404923"
## [1] "242 :  -235933.725158814"
## [1] "243 :  -235933.723155542"
## [1] "244 :  -235933.721368634"
## [1] "245 :  -235933.71977451"
## [1] "246 :  -235933.718352164"
## [1] "247 :  -235933.71708289"
## [1] "248 :  -235933.715950045"
## [1] "249 :  -235933.714938776"
## [1] "250 :  -235933.714035881"
## [1] "251 :  -235933.713229592"
## [1] "252 :  -235933.712509428"
## [1] "253 :  -235933.711866061"
## [1] "254 :  -235933.711291162"
## [1] "255 :  -235933.710777347"
## [1] "256 :  -235933.710318011"
## [1] "257 :  -235933.709907291"
## [1] "258 :  -235933.709539939"
## [1] "259 :  -235933.709211294"
## [1] "260 :  -235933.708917201"
## [1] "261 :  -235933.708653971"
## [1] "262 :  -235933.708418296"
## [1] "263 :  -235933.708207234"
## [1] "264 :  -235933.708018167"
## [1] "265 :  -235933.707848752"
## [1] "266 :  -235933.707696908"
```

```
## [1] "267 :  -235933.707560773"
## [1] "268 :  -235933.707438695"
## [1] "269 :  -235933.70732918"
## [1] "270 :  -235933.707230927"
## [1] "271 :  -235933.707142738"
## [1] "272 :  -235933.707063562"
## [1] "273 :  -235933.706992455"
## [1] "274 :  -235933.706928592"
## [1] "275 :  -235933.706871205"
## [1] "276 :  -235933.706819644"
## [1] "277 :  -235933.706773287"
## [1] "278 :  -235933.706731604"
## [1] "279 :  -235933.706694097"
## [1] "280 :  -235933.706660376"
## [1] "281 :  -235933.706630021"
## [1] "282 :  -235933.706602702"
## [1] "283 :  -235933.706578106"
## [1] "284 :  -235933.706555955"
## [1] "285 :  -235933.706536006"
## [1] "286 :  -235933.706518035"
## [1] "287 :  -235933.706501834"
## [1] "288 :  -235933.706487244"
## [1] "289 :  -235933.706474082"
## [1] "290 :  -235933.706462218"
## [1] "291 :  -235933.706451518"
## [1] "292 :  -235933.706441857"
```

b(ii)

We have K = 4,four clusters in the mixture bernoulli model. We know that the value in the newsgroups data set is the true value and we use γ_k to estimate the μ_k. In this case using part(a), we could find the maximum value of γ for each n where n=16242. Then we could perform a table for the true data set and the estimaated clusters. After that, we turned the table into a matrix for convenience in further calculations. We need to find the matrix element that has the maximum value both in each row and column, that wiil be the true value=estimated value for four clusters. Finally, we could get the total accuracy.

```
n<-16242
true_label <- as.vector(newsgroups)
# we could get the maximum value of gamma
cluster <- apply(out10$gammas, 1, which.max)

matrix.f<- function(true_label,cluster,k) {
   n<-16242
  table <- table(true_label,cluster)
  matrix4 <- table
  accuracy<-0
   m_element <-numeric(k)
repeat {
  max_r <- apply(matrix4, 1, max)
  max_c <- apply(matrix4, 2, max)
  max_v <- max(matrix4)
  if (max_v == 0) {
    break
  }
```

```
  max_n <- which(matrix4 == max_v, arr.ind = TRUE)
  i <- max_n[,1]
  j <- max_n[,2]
  if (max_r[i] == max_v && max_c[j] == max_v) {
    m_element[i] <- max_v
    accuracy <- accuracy + max_v
    matrix4[i,] <- rep(0, k)
    matrix4[,j] <- rep(0, k)
  } else {
    matrix4[i,j] <- 0
  }
}

total_accuracy <- paste(accuracy/n)
output<- list(table = table, m_element = m_element, total_accuracy = total_accuracy)
return(output)}
```

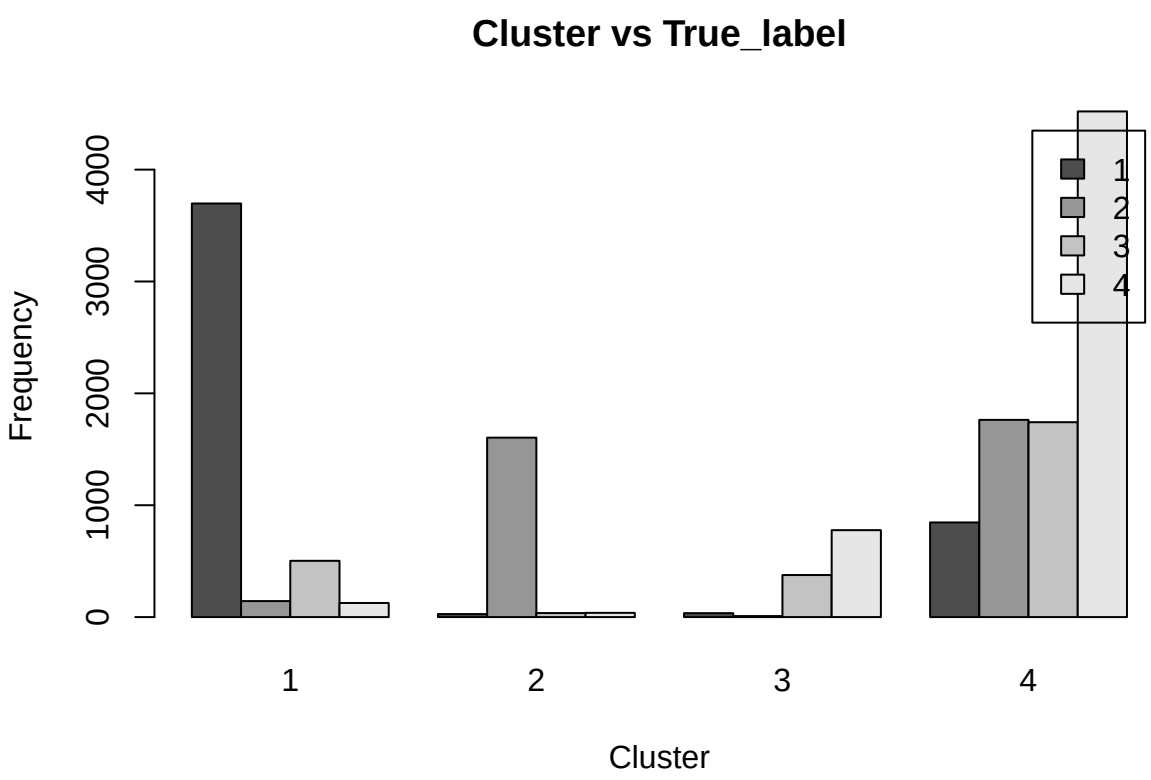
```
matrix.f(true_label,cluster,4)
```

```
## $table
##           cluster
## true_label    1    2    3    4
##          1 3697   27   35  846
##          2  143 1604    9 1763
##          3  503   36  376 1742
##          4  126   38  777 4520
##
## $m_element
## [1] 3697 1604  376 4520
##
## $total_accuracy
## [1] "0.62781677133358"
```

The total accuracy is 0.63.

Also using the code below we actually find there is misallocation of the clusters with estimated value using the bar chart to iluustrate the frequency.

```
as <- matrix.f(true_label, cluster, k = 4)
table <- as$table

# we could plot the frequency graph
barplot(table, beside = TRUE,col = grey.colors(ncol(table)),
        main = "Cluster vs True_label", xlab = "Cluster",
        ylab = "Frequency", legend.text = rownames(table))
```



This graph shows the data performed for the estimated values and true values. From the graphs we can there is misallocation of data in clusters and true labels for all clusters, so they did not perform so well in this algorithm. As a result, the em algorithm did't perform well in this data set, the reasons might be the optimal clusters do not match the four topic labels or clusters might not be reliable reproducible. To improve the accuracy,we could reallocated the values in order to get a more precise answer through permutation of tesing and assigning the true value to match the estimated value for four clusters.Otherwise, we could also find the words frequency in each cluster to check whether they correspond to the four board topics.

b(iii) The sensitivity of the EM algorithm with respect to the initial parameters can be examined by comparing the results obtained from different initial values. To reduce the sensitivity, it is common to run the EM algorithm multiple times with different initial values and choose the estimate that maximizes the likelihood or has the highest total accuracy value. We run the code for 3 times and obtained 3 different values for total accuracy for each time.

```
#we repeat the process for 3 times
accuracy_v <- replicate(3, {
  out10 <- em_mix_bernoulli(documents, 4)
  cluster <- apply(out10$gammas, 1, which.max)
  matrix.f(true_label, cluster, 4)$total_accuracy
})
```

```
## [1] "iteration : log-likelihood"
## [1] "1 :  -1580976.56341018"
## [1] "2 :  -252577.825859701"
## [1] "3 :  -246080.633553911"
## [1] "4 :  -241108.758199989"
```

```
## [1] "5 :  -239074.508261364"
## [1] "6 :  -238200.213576361"
## [1] "7 :  -237652.194479332"
## [1] "8 :  -237317.359784143"
## [1] "9 :  -237092.89585333"
## [1] "10 :  -236948.28187849"
## [1] "11 :  -236868.948521374"
## [1] "12 :  -236791.129119805"
## [1] "13 :  -236692.820519277"
## [1] "14 :  -236569.54820166"
## [1] "15 :  -236450.894948616"
## [1] "16 :  -236387.661177241"
## [1] "17 :  -236357.877158877"
## [1] "18 :  -236336.051268921"
## [1] "19 :  -236316.706272448"
## [1] "20 :  -236298.797167762"
## [1] "21 :  -236282.10963734"
## [1] "22 :  -236266.582912955"
## [1] "23 :  -236252.175473129"
## [1] "24 :  -236238.823112772"
## [1] "25 :  -236226.419521699"
## [1] "26 :  -236214.844654842"
## [1] "27 :  -236204.082534984"
## [1] "28 :  -236194.246385107"
## [1] "29 :  -236185.386955218"
## [1] "30 :  -236177.441285627"
## [1] "31 :  -236170.273414245"
## [1] "32 :  -236163.483083582"
## [1] "33 :  -236155.610619587"
## [1] "34 :  -236147.670918584"
## [1] "35 :  -236141.055287054"
## [1] "36 :  -236134.82372023"
## [1] "37 :  -236129.137826613"
## [1] "38 :  -236124.097363394"
## [1] "39 :  -236119.630881741"
## [1] "40 :  -236115.625381525"
## [1] "41 :  -236111.985542327"
## [1] "42 :  -236108.646105146"
## [1] "43 :  -236105.578686563"
## [1] "44 :  -236102.780391712"
## [1] "45 :  -236100.241187451"
## [1] "46 :  -236097.889010336"
## [1] "47 :  -236095.433511752"
## [1] "48 :  -236092.777042435"
## [1] "49 :  -236090.658558697"
## [1] "50 :  -236088.903559937"
## [1] "51 :  -236087.322051471"
## [1] "52 :  -236085.883630121"
## [1] "53 :  -236084.580824245"
## [1] "54 :  -236083.406643206"
## [1] "55 :  -236082.351163833"
## [1] "56 :  -236081.402508835"
## [1] "57 :  -236080.548419978"
## [1] "58 :  -236079.777252126"
```

```
## [1] "59 :  -236079.078421473"
## [1] "60 :  -236078.442558309"
## [1] "61 :  -236077.861530585"
## [1] "62 :  -236077.328414901"
## [1] "63 :  -236076.83743892"
## [1] "64 :  -236076.383895083"
## [1] "65 :  -236075.964018771"
## [1] "66 :  -236075.574830342"
## [1] "67 :  -236075.213952336"
## [1] "68 :  -236074.879424342"
## [1] "69 :  -236074.569540389"
## [1] "70 :  -236074.282726961"
## [1] "71 :  -236074.017467402"
## [1] "72 :  -236073.772267355"
## [1] "73 :  -236073.545649339"
## [1] "74 :  -236073.336163747"
## [1] "75 :  -236073.142406007"
## [1] "76 :  -236072.963033303"
## [1] "77 :  -236072.79677751"
## [1] "78 :  -236072.642453591"
## [1] "79 :  -236072.498963339"
## [1] "80 :  -236072.36529572"
## [1] "81 :  -236072.240524466"
## [1] "82 :  -236072.123803988"
## [1] "83 :  -236072.014364153"
## [1] "84 :  -236071.911504665"
## [1] "85 :  -236071.814589087"
## [1] "86 :  -236071.723039108"
## [1] "87 :  -236071.636328909"
## [1] "88 :  -236071.553979953"
## [1] "89 :  -236071.475556066"
## [1] "90 :  -236071.400658874"
## [1] "91 :  -236071.328923693"
## [1] "92 :  -236071.260015711"
## [1] "93 :  -236071.193626524"
## [1] "94 :  -236071.129470898"
## [1] "95 :  -236071.067283925"
## [1] "96 :  -236071.006818289"
## [1] "97 :  -236070.9478418"
## [1] "98 :  -236070.890135092"
## [1] "99 :  -236070.833489433"
## [1] "100 :  -236070.777704613"
## [1] "101 :  -236070.72258702"
## [1] "102 :  -236070.667947567"
## [1] "103 :  -236070.61359977"
## [1] "104 :  -236070.559357696"
## [1] "105 :  -236070.505033798"
## [1] "106 :  -236070.450436578"
## [1] "107 :  -236070.39536814"
## [1] "108 :  -236070.33962119"
## [1] "109 :  -236070.28297577"
## [1] "110 :  -236070.225195284"
## [1] "111 :  -236070.166021833"
## [1] "112 :  -236070.105170456"
```

```
## [1] "113 :  -236070.042321971"
## [1] "114 :  -236069.977114098"
## [1] "115 :  -236069.909130109"
## [1] "116 :  -236069.837884216"
## [1] "117 :  -236069.762802674"
## [1] "118 :  -236069.683198729"
## [1] "119 :  -236069.59823954"
## [1] "120 :  -236069.506901345"
## [1] "121 :  -236069.407908453"
## [1] "122 :  -236069.299649102"
## [1] "123 :  -236069.180057933"
## [1] "124 :  -236069.046449967"
## [1] "125 :  -236068.895284394"
## [1] "126 :  -236068.721825885"
## [1] "127 :  -236068.519660233"
## [1] "128 :  -236068.280009575"
## [1] "129 :  -236067.990797353"
## [1] "130 :  -236067.635468882"
## [1] "131 :  -236067.191765786"
## [1] "132 :  -236066.63112096"
## [1] "133 :  -236065.920158163"
## [1] "134 :  -236065.026421865"
## [1] "135 :  -236063.929017939"
## [1] "136 :  -236062.629297431"
## [1] "137 :  -236061.151033546"
## [1] "138 :  -236059.523993058"
## [1] "139 :  -236057.755776114"
## [1] "140 :  -236055.791146713"
## [1] "141 :  -236053.423777924"
## [1] "142 :  -236050.100120825"
## [1] "143 :  -236044.879833893"
## [1] "144 :  -236036.99339953"
## [1] "145 :  -236025.601626853"
## [1] "146 :  -236011.166206262"
## [1] "147 :  -235995.745633226"
## [1] "148 :  -235981.288250883"
## [1] "149 :  -235969.30855547"
## [1] "150 :  -235960.440166325"
## [1] "151 :  -235954.442524887"
## [1] "152 :  -235950.606338222"
## [1] "153 :  -235948.201314503"
## [1] "154 :  -235946.675032034"
## [1] "155 :  -235945.667910674"
## [1] "156 :  -235944.965470246"
## [1] "157 :  -235944.44567489"
## [1] "158 :  -235944.040802869"
## [1] "159 :  -235943.714501367"
## [1] "160 :  -235943.44808953"
## [1] "161 :  -235943.231236969"
## [1] "162 :  -235943.056001953"
## [1] "163 :  -235942.914627238"
## [1] "164 :  -235942.79980554"
## [1] "165 :  -235942.705388001"
## [1] "166 :  -235942.626629141"
```

```
## [1] "167 :  -235942.560033585"
## [1] "168 :  -235942.503066591"
## [1] "169 :  -235942.45388051"
## [1] "170 :  -235942.411104529"
## [1] "171 :  -235942.373697122"
## [1] "172 :  -235942.340846555"
## [1] "173 :  -235942.311904766"
## [1] "174 :  -235942.286343312"
## [1] "175 :  -235942.263723632"
## [1] "176 :  -235942.243676455"
## [1] "177 :  -235942.225887345"
## [1] "178 :  -235942.210086065"
## [1] "179 :  -235942.196038803"
## [1] "180 :  -235942.183542051"
## [1] "181 :  -235942.172417929"
## [1] "182 :  -235942.162510417"
## [1] "183 :  -235942.153682276"
## [1] "184 :  -235942.145812463"
## [1] "185 :  -235942.138794112"
## [1] "186 :  -235942.13253264"
## [1] "187 :  -235942.126944321"
## [1] "188 :  -235942.121954886"
## [1] "189 :  -235942.117498447"
## [1] "190 :  -235942.113516497"
## [1] "191 :  -235942.109957055"
## [1] "192 :  -235942.106773943"
## [1] "193 :  -235942.103926078"
## [1] "194 :  -235942.101376923"
## [1] "195 :  -235942.099093976"
## [1] "196 :  -235942.097048305"
## [1] "197 :  -235942.095214198"
## [1] "198 :  -235942.093568739"
## [1] "199 :  -235942.092091503"
## [1] "200 :  -235942.090764354"
## [1] "201 :  -235942.089571121"
## [1] "202 :  -235942.088497414"
## [1] "203 :  -235942.087530386"
## [1] "204 :  -235942.086658645"
## [1] "205 :  -235942.085871998"
## [1] "206 :  -235942.085161392"
## [1] "207 :  -235942.084518759"
## [1] "208 :  -235942.083936879"
## [1] "209 :  -235942.083409382"
## [1] "210 :  -235942.082930529"
## [1] "211 :  -235942.082495235"
## [1] "212 :  -235942.082098972"
## [1] "213 :  -235942.081737668"
## [1] "214 :  -235942.081407736"
## [1] "215 :  -235942.081105938"
## [1] "216 :  -235942.080829419"
## [1] "217 :  -235942.08057559"
## [1] "218 :  -235942.080342173"
## [1] "219 :  -235942.080127126"
## [1] "220 :  -235942.079928608"
```

```
## [1] "221 :  -235942.079744966"
## [1] "222 :  -235942.079574762"
## [1] "223 :  -235942.079416661"
## [1] "224 :  -235942.079269461"
## [1] "225 :  -235942.079132115"
## [1] "226 :  -235942.079003639"
## [1] "227 :  -235942.078883163"
## [1] "228 :  -235942.078769886"
## [1] "229 :  -235942.07866305"
## [1] "230 :  -235942.078561992"
## [1] "231 :  -235942.078466029"
## [1] "232 :  -235942.078374568"
## [1] "233 :  -235942.078286987"
## [1] "234 :  -235942.078202677"
## [1] "235 :  -235942.078121031"
## [1] "236 :  -235942.078041383"
## [1] "237 :  -235942.077963024"
## [1] "238 :  -235942.077885206"
## [1] "239 :  -235942.077807016"
## [1] "240 :  -235942.077727427"
## [1] "241 :  -235942.077645259"
## [1] "242 :  -235942.07755905"
## [1] "243 :  -235942.077467107"
## [1] "244 :  -235942.077367287"
## [1] "245 :  -235942.077257031"
## [1] "246 :  -235942.077133147"
## [1] "247 :  -235942.076991444"
## [1] "248 :  -235942.076821389"
## [1] "249 :  -235942.076512241"
## [1] "250 :  -235942.074129062"
## [1] "251 :  -235942.035034696"
## [1] "252 :  -235941.518770615"
## [1] "253 :  -235939.897998837"
## [1] "254 :  -235939.322831952"
## [1] "255 :  -235939.233318884"
## [1] "256 :  -235939.198571065"
## [1] "257 :  -235939.176028762"
## [1] "258 :  -235939.158100412"
## [1] "259 :  -235939.142635433"
## [1] "260 :  -235939.128768658"
## [1] "261 :  -235939.116017847"
## [1] "262 :  -235939.104036296"
## [1] "263 :  -235939.092533863"
## [1] "264 :  -235939.081249781"
## [1] "265 :  -235939.069944946"
## [1] "266 :  -235939.058403955"
## [1] "267 :  -235939.046443374"
## [1] "268 :  -235939.033924321"
## [1] "269 :  -235939.020766829"
## [1] "270 :  -235939.006962532"
## [1] "271 :  -235938.992581302"
## [1] "272 :  -235938.977768011"
## [1] "273 :  -235938.962727842"
## [1] "274 :  -235938.947701664"
```

```
## [1] "275 :  -235938.932936693"
## [1] "276 :  -235938.918659119"
## [1] "277 :  -235938.905053825"
## [1] "278 :  -235938.892254244"
## [1] "279 :  -235938.880341518"
## [1] "280 :  -235938.869350323"
## [1] "281 :  -235938.859278244"
## [1] "282 :  -235938.850095873"
## [1] "283 :  -235938.841755936"
## [1] "284 :  -235938.834200889"
## [1] "285 :  -235938.827368422"
## [1] "286 :  -235938.821195417"
## [1] "287 :  -235938.815620565"
## [1] "288 :  -235938.810585912"
## [1] "289 :  -235938.806037661"
## [1] "290 :  -235938.80192661"
## [1] "291 :  -235938.798208109"
## [1] "292 :  -235938.794841922"
## [1] "293 :  -235938.791791989"
## [1] "294 :  -235938.789025977"
## [1] "295 :  -235938.786515085"
## [1] "296 :  -235938.784233588"
## [1] "297 :  -235938.782158547"
## [1] "298 :  -235938.780269525"
## [1] "299 :  -235938.778548249"
## [1] "300 :  -235938.776978418"
## [1] "301 :  -235938.775545491"
## [1] "302 :  -235938.774236397"
## [1] "303 :  -235938.773039474"
## [1] "304 :  -235938.771944254"
## [1] "305 :  -235938.770941328"
## [1] "306 :  -235938.770022244"
## [1] "307 :  -235938.76917938"
## [1] "308 :  -235938.768405895"
## [1] "309 :  -235938.767695599"
## [1] "310 :  -235938.767042897"
## [1] "311 :  -235938.766442737"
## [1] "312 :  -235938.765890539"
## [1] "313 :  -235938.765382166"
## [1] "314 :  -235938.764913849"
## [1] "315 :  -235938.764482167"
## [1] "316 :  -235938.764084035"
## [1] "317 :  -235938.763716634"
## [1] "318 :  -235938.763377372"
## [1] "319 :  -235938.763063928"
## [1] "320 :  -235938.762774154"
## [1] "321 :  -235938.762506125"
## [1] "322 :  -235938.762258056"
## [1] "323 :  -235938.762028305"
## [1] "324 :  -235938.761815421"
## [1] "325 :  -235938.761618039"
## [1] "326 :  -235938.761434908"
## [1] "327 :  -235938.761264884"
## [1] "328 :  -235938.761106949"
```

```
## [1] "329 :  -235938.760960142"
## [1] "330 :  -235938.760823593"
## [1] "331 :  -235938.760696489"
## [1] "332 :  -235938.760578103"
## [1] "333 :  -235938.760467762"
## [1] "334 :  -235938.760364825"
## [1] "335 :  -235938.760268761"
## [1] "336 :  -235938.760179027"
## [1] "337 :  -235938.760095135"
## [1] "338 :  -235938.760016649"
## [1] "339 :  -235938.759943165"
## [1] "340 :  -235938.75987431"
## [1] "341 :  -235938.75980974"
## [1] "342 :  -235938.759749138"
## [1] "343 :  -235938.759692205"
## [1] "344 :  -235938.759638676"
## [1] "345 :  -235938.759588304"
## [1] "346 :  -235938.759540851"
## [1] "347 :  -235938.759496132"
## [1] "348 :  -235938.759453936"
## [1] "349 :  -235938.759414072"
## [1] "350 :  -235938.7593764"
## [1] "351 :  -235938.759340762"
## [1] "352 :  -235938.75930701"
## [1] "353 :  -235938.759274999"
## [1] "354 :  -235938.759244647"
## [1] "355 :  -235938.759215816"
## [1] "356 :  -235938.759188402"
## [1] "357 :  -235938.759162323"
## [1] "358 :  -235938.759137476"
## [1] "359 :  -235938.759113807"
## [1] "360 :  -235938.759091214"
## [1] "361 :  -235938.759069642"
## [1] "362 :  -235938.75904902"
## [1] "363 :  -235938.759029293"
## [1] "364 :  -235938.7590104"
## [1] "365 :  -235938.758992297"
## [1] "366 :  -235938.758974931"
## [1] "367 :  -235938.758958261"
## [1] "368 :  -235938.758942246"
## [1] "369 :  -235938.758926848"
## [1] "370 :  -235938.758912021"
## [1] "371 :  -235938.758897756"
## [1] "372 :  -235938.758884008"
## [1] "373 :  -235938.758870735"
## [1] "374 :  -235938.758857938"
## [1] "375 :  -235938.758845576"
## [1] "376 :  -235938.758833635"
## [1] "377 :  -235938.758822084"
## [1] "378 :  -235938.758810908"
## [1] "379 :  -235938.758800093"
## [1] "380 :  -235938.758789605"
## [1] "381 :  -235938.758779444"
## [1] "382 :  -235938.758769593"
```

```
## [1] "iteration : log-likelihood"
## [1] "1 :  -1506709.37329971"
## [1] "2 :  -250354.104649651"
## [1] "3 :  -245935.387621639"
## [1] "4 :  -242775.561034423"
## [1] "5 :  -240766.624791721"
## [1] "6 :  -239693.00370568"
## [1] "7 :  -238740.786650037"
## [1] "8 :  -237770.906779328"
## [1] "9 :  -237416.250750319"
## [1] "10 :  -237305.870480668"
## [1] "11 :  -237233.725039897"
## [1] "12 :  -237175.969632162"
## [1] "13 :  -237127.858620545"
## [1] "14 :  -237086.700952938"
## [1] "15 :  -237050.337575074"
## [1] "16 :  -237017.044804852"
## [1] "17 :  -236986.69337256"
## [1] "18 :  -236959.612182267"
## [1] "19 :  -236935.367231571"
## [1] "20 :  -236913.536505392"
## [1] "21 :  -236893.800442798"
## [1] "22 :  -236875.899613416"
## [1] "23 :  -236859.602376673"
## [1] "24 :  -236844.779824612"
## [1] "25 :  -236831.445835683"
## [1] "26 :  -236819.560488287"
## [1] "27 :  -236808.981999035"
## [1] "28 :  -236799.58998173"
## [1] "29 :  -236791.309174022"
## [1] "30 :  -236784.069327996"
## [1] "31 :  -236777.775564644"
## [1] "32 :  -236772.313555297"
## [1] "33 :  -236767.56450946"
## [1] "34 :  -236763.402823627"
## [1] "35 :  -236759.653987123"
## [1] "36 :  -236756.033176568"
## [1] "37 :  -236752.379663695"
## [1] "38 :  -236749.038362828"
## [1] "39 :  -236746.342070624"
## [1] "40 :  -236744.236940415"
## [1] "41 :  -236742.53875458"
## [1] "42 :  -236741.120504291"
## [1] "43 :  -236739.913547692"
## [1] "44 :  -236738.877186918"
## [1] "45 :  -236737.982907498"
## [1] "46 :  -236737.208431267"
## [1] "47 :  -236736.535407674"
## [1] "48 :  -236735.948156786"
## [1] "49 :  -236735.432339595"
## [1] "50 :  -236734.972357074"
## [1] "51 :  -236734.544938418"
## [1] "52 :  -236734.105428115"
## [1] "53 :  -236733.579106303"
```

```
## [1] "54 :  -236732.923709924"
## [1] "55 :  -236732.225991481"
## [1] "56 :  -236731.574779108"
## [1] "57 :  -236730.921186911"
## [1] "58 :  -236730.209194976"
## [1] "59 :  -236729.546255577"
## [1] "60 :  -236728.88462122"
## [1] "61 :  -236727.981334002"
## [1] "62 :  -236726.658430605"
## [1] "63 :  -236724.931436717"
## [1] "64 :  -236722.996680143"
## [1] "65 :  -236721.072949263"
## [1] "66 :  -236719.29592197"
## [1] "67 :  -236717.715035454"
## [1] "68 :  -236716.323872055"
## [1] "69 :  -236715.087632226"
## [1] "70 :  -236713.960646124"
## [1] "71 :  -236712.895304261"
## [1] "72 :  -236711.845132459"
## [1] "73 :  -236710.76412189"
## [1] "74 :  -236709.603550634"
## [1] "75 :  -236708.306757313"
## [1] "76 :  -236706.80160055"
## [1] "77 :  -236704.989600231"
## [1] "78 :  -236702.730020056"
## [1] "79 :  -236699.81671648"
## [1] "80 :  -236695.947147683"
## [1] "81 :  -236690.693959296"
## [1] "82 :  -236683.534159341"
## [1] "83 :  -236674.091066293"
## [1] "84 :  -236662.694918893"
## [1] "85 :  -236650.704649237"
## [1] "86 :  -236639.773574988"
## [1] "87 :  -236630.796120794"
## [1] "88 :  -236623.76086037"
## [1] "89 :  -236618.24231867"
## [1] "90 :  -236613.806299342"
## [1] "91 :  -236610.146447236"
## [1] "92 :  -236607.072421748"
## [1] "93 :  -236604.465103786"
## [1] "94 :  -236602.243045329"
## [1] "95 :  -236600.344478428"
## [1] "96 :  -236598.719074902"
## [1] "97 :  -236597.324414564"
## [1] "98 :  -236596.124388073"
## [1] "99 :  -236595.08828149"
## [1] "100 :  -236594.190063942"
## [1] "101 :  -236593.407738151"
## [1] "102 :  -236592.72273609"
## [1] "103 :  -236592.119372831"
## [1] "104 :  -236591.584367919"
## [1] "105 :  -236591.106435363"
## [1] "106 :  -236590.675936085"
## [1] "107 :  -236590.284583661"
```

```
## [1] "108 :  -236589.92519386"
## [1] "109 :  -236589.591468777"
## [1] "110 :  -236589.277807726"
## [1] "111 :  -236588.97913727"
## [1] "112 :  -236588.690753862"
## [1] "113 :  -236588.408171984"
## [1] "114 :  -236588.126970635"
## [1] "115 :  -236587.842629757"
## [1] "116 :  -236587.550346406"
## [1] "117 :  -236587.244817723"
## [1] "118 :  -236586.91997364"
## [1] "119 :  -236586.568635825"
## [1] "120 :  -236586.182070301"
## [1] "121 :  -236585.749387141"
## [1] "122 :  -236585.256719305"
## [1] "123 :  -236584.686077906"
## [1] "124 :  -236584.0137212"
## [1] "125 :  -236583.207762466"
## [1] "126 :  -236582.224518418"
## [1] "127 :  -236581.002656829"
## [1] "128 :  -236579.453496054"
## [1] "129 :  -236577.445787343"
## [1] "130 :  -236574.789305638"
## [1] "131 :  -236571.243089413"
## [1] "132 :  -236566.5797989"
## [1] "133 :  -236560.640766541"
## [1] "134 :  -236553.315416105"
## [1] "135 :  -236544.648702793"
## [1] "136 :  -236535.124848147"
## [1] "137 :  -236525.628823447"
## [1] "138 :  -236516.990533534"
## [1] "139 :  -236509.734042407"
## [1] "140 :  -236504.031223003"
## [1] "141 :  -236499.697208358"
## [1] "142 :  -236496.386913326"
## [1] "143 :  -236493.795020764"
## [1] "144 :  -236491.70108029"
## [1] "145 :  -236489.956932974"
## [1] "146 :  -236488.464708144"
## [1] "147 :  -236487.155662955"
## [1] "148 :  -236485.978442051"
## [1] "149 :  -236484.894417801"
## [1] "150 :  -236483.874003878"
## [1] "151 :  -236482.885548089"
## [1] "152 :  -236481.829061507"
## [1] "153 :  -236480.143210458"
## [1] "154 :  -236476.154983345"
## [1] "155 :  -236470.477440546"
## [1] "156 :  -236466.470651481"
## [1] "157 :  -236464.240037995"
## [1] "158 :  -236462.738482386"
## [1] "159 :  -236461.401651685"
## [1] "160 :  -236460.042475207"
## [1] "161 :  -236458.598287562"
```

```
## [1] "162 :  -236457.019447921"
## [1] "163 :  -236455.174396497"
## [1] "164 :  -236452.386353159"
## [1] "165 :  -236445.681756409"
## [1] "166 :  -236431.448279458"
## [1] "167 :  -236414.63160128"
## [1] "168 :  -236400.628997932"
## [1] "169 :  -236389.555743321"
## [1] "170 :  -236379.564649691"
## [1] "171 :  -236368.903480958"
## [1] "172 :  -236355.756576439"
## [1] "173 :  -236337.421205006"
## [1] "174 :  -236308.925590884"
## [1] "175 :  -236261.074806545"
## [1] "176 :  -236184.830326132"
## [1] "177 :  -236098.229472576"
## [1] "178 :  -236041.583773858"
## [1] "179 :  -236015.303281767"
## [1] "180 :  -236001.610426474"
## [1] "181 :  -235992.652328085"
## [1] "182 :  -235985.968281773"
## [1] "183 :  -235980.674363298"
## [1] "184 :  -235976.353920755"
## [1] "185 :  -235972.772694"
## [1] "186 :  -235969.78924178"
## [1] "187 :  -235967.304952146"
## [1] "188 :  -235965.236209815"
## [1] "189 :  -235963.508184834"
## [1] "190 :  -235962.057116211"
## [1] "191 :  -235960.831437312"
## [1] "192 :  -235959.790062576"
## [1] "193 :  -235958.89929826"
## [1] "194 :  -235958.130021311"
## [1] "195 :  -235957.455800715"
## [1] "196 :  -235956.851635832"
## [1] "197 :  -235956.292620911"
## [1] "198 :  -235955.751950489"
## [1] "199 :  -235955.197986935"
## [1] "200 :  -235954.591253542"
## [1] "201 :  -235953.886563453"
## [1] "202 :  -235953.053803556"
## [1] "203 :  -235952.119771359"
## [1] "204 :  -235951.169271376"
## [1] "205 :  -235950.266927546"
## [1] "206 :  -235949.417958588"
## [1] "207 :  -235948.599760447"
## [1] "208 :  -235947.788750386"
## [1] "209 :  -235946.970313209"
## [1] "210 :  -235946.139298156"
## [1] "211 :  -235945.293133787"
## [1] "212 :  -235944.428029619"
## [1] "213 :  -235943.542283479"
## [1] "214 :  -235942.638440289"
## [1] "215 :  -235941.723688927"
```

```
## [1] "216 :  -235940.816921541"
## [1] "217 :  -235939.956289127"
## [1] "218 :  -235939.183658966"
## [1] "219 :  -235938.517589955"
## [1] "220 :  -235937.952791439"
## [1] "221 :  -235937.475734564"
## [1] "222 :  -235937.073043935"
## [1] "223 :  -235936.733110553"
## [1] "224 :  -235936.446021179"
## [1] "225 :  -235936.203352655"
## [1] "226 :  -235935.997988972"
## [1] "227 :  -235935.82394888"
## [1] "228 :  -235935.676222551"
## [1] "229 :  -235935.550622185"
## [1] "230 :  -235935.443650044"
## [1] "231 :  -235935.352384599"
## [1] "232 :  -235935.274384199"
## [1] "233 :  -235935.207605983"
## [1] "234 :  -235935.150338536"
## [1] "235 :  -235935.101145709"
## [1] "236 :  -235935.058819915"
## [1] "237 :  -235935.022343004"
## [1] "238 :  -235934.9908523"
## [1] "239 :  -235934.96360923"
## [1] "240 :  -235934.939964318"
## [1] "241 :  -235934.919305743"
## [1] "242 :  -235934.900958472"
## [1] "243 :  -235934.883950698"
## [1] "244 :  -235934.866436865"
## [1] "245 :  -235934.844279604"
## [1] "246 :  -235934.807862044"
## [1] "247 :  -235934.736896181"
## [1] "248 :  -235934.601252523"
## [1] "249 :  -235934.394476523"
## [1] "250 :  -235934.182098559"
## [1] "251 :  -235934.039369664"
## [1] "252 :  -235933.964391215"
## [1] "253 :  -235933.923291464"
## [1] "254 :  -235933.895607634"
## [1] "255 :  -235933.873665207"
## [1] "256 :  -235933.854945481"
## [1] "257 :  -235933.838547812"
## [1] "258 :  -235933.824052067"
## [1] "259 :  -235933.81119308"
## [1] "260 :  -235933.799768172"
## [1] "261 :  -235933.789608668"
## [1] "262 :  -235933.780569386"
## [1] "263 :  -235933.772523628"
## [1] "264 :  -235933.765360011"
## [1] "265 :  -235933.758980292"
## [1] "266 :  -235933.753297581"
## [1] "267 :  -235933.748234878"
## [1] "268 :  -235933.743723896"
## [1] "269 :  -235933.739703999"
```

```
## [1] "270 :  -235933.736121303"
## [1] "271 :  -235933.732927875"
## [1] "272 :  -235933.730081117"
## [1] "273 :  -235933.727543095"
## [1] "274 :  -235933.725280064"
## [1] "275 :  -235933.723261985"
## [1] "276 :  -235933.721462092"
## [1] "277 :  -235933.719856596"
## [1] "278 :  -235933.718424289"
## [1] "279 :  -235933.717146273"
## [1] "280 :  -235933.71600576"
## [1] "281 :  -235933.714987765"
## [1] "282 :  -235933.714078973"
## [1] "283 :  -235933.713267517"
## [1] "284 :  -235933.712542802"
## [1] "285 :  -235933.711895445"
## [1] "286 :  -235933.711317048"
## [1] "287 :  -235933.710800154"
## [1] "288 :  -235933.710338113"
## [1] "289 :  -235933.709925015"
## [1] "290 :  -235933.70955556"
## [1] "291 :  -235933.709225082"
## [1] "292 :  -235933.708929374"
## [1] "293 :  -235933.708664724"
## [1] "294 :  -235933.708427794"
## [1] "295 :  -235933.708215627"
## [1] "296 :  -235933.708025587"
## [1] "297 :  -235933.707855317"
## [1] "298 :  -235933.707702713"
## [1] "299 :  -235933.707565916"
## [1] "300 :  -235933.70744326"
## [1] "301 :  -235933.70733322"
## [1] "302 :  -235933.707234512"
## [1] "303 :  -235933.707145905"
## [1] "304 :  -235933.707066379"
## [1] "305 :  -235933.706994963"
## [1] "306 :  -235933.706930818"
## [1] "307 :  -235933.706873191"
## [1] "308 :  -235933.706821407"
## [1] "309 :  -235933.706774847"
## [1] "310 :  -235933.706732987"
## [1] "311 :  -235933.706695343"
## [1] "312 :  -235933.706661477"
## [1] "313 :  -235933.706631012"
## [1] "314 :  -235933.706603587"
## [1] "315 :  -235933.706578887"
## [1] "316 :  -235933.706556653"
## [1] "317 :  -235933.706536633"
## [1] "318 :  -235933.70651859"
## [1] "319 :  -235933.706502331"
## [1] "320 :  -235933.70648769"
## [1] "321 :  -235933.706474476"
## [1] "322 :  -235933.706462582"
## [1] "323 :  -235933.706451836"
```

```
## [1] "324 :  -235933.706442156"
## [1] "iteration : log-likelihood"
## [1] "1 :  -1473036.7075682"
## [1] "2 :  -251009.957583791"
## [1] "3 :  -246238.772150397"
## [1] "4 :  -242280.662296301"
## [1] "5 :  -239929.748569942"
## [1] "6 :  -239046.453486385"
## [1] "7 :  -238691.714126371"
## [1] "8 :  -238484.824888306"
## [1] "9 :  -238340.143558767"
## [1] "10 :  -238227.654923943"
## [1] "11 :  -238134.86248727"
## [1] "12 :  -238054.11425031"
## [1] "13 :  -237981.896122477"
## [1] "14 :  -237916.621958256"
## [1] "15 :  -237857.164196433"
## [1] "16 :  -237803.396015428"
## [1] "17 :  -237755.361932756"
## [1] "18 :  -237712.890360447"
## [1] "19 :  -237675.566854711"
## [1] "20 :  -237642.886083222"
## [1] "21 :  -237614.083522648"
## [1] "22 :  -237588.188099418"
## [1] "23 :  -237564.282722891"
## [1] "24 :  -237541.652370069"
## [1] "25 :  -237519.724404599"
## [1] "26 :  -237497.87278557"
## [1] "27 :  -237475.31686672"
## [1] "28 :  -237451.163981068"
## [1] "29 :  -237424.411967514"
## [1] "30 :  -237393.932364328"
## [1] "31 :  -237360.859583551"
## [1] "32 :  -237329.511887756"
## [1] "33 :  -237299.680534736"
## [1] "34 :  -237270.533676121"
## [1] "35 :  -237241.750893923"
## [1] "36 :  -237213.728825789"
## [1] "37 :  -237186.594546602"
## [1] "38 :  -237161.879930509"
## [1] "39 :  -237134.951120831"
## [1] "40 :  -237118.194890774"
## [1] "41 :  -237093.766632951"
## [1] "42 :  -237078.336149127"
## [1] "43 :  -237067.385178975"
## [1] "44 :  -237057.438075688"
## [1] "45 :  -237048.231334882"
## [1] "46 :  -237038.685118923"
## [1] "47 :  -237024.141144402"
## [1] "48 :  -237013.934574637"
## [1] "49 :  -237003.313049946"
## [1] "50 :  -236991.648899592"
## [1] "51 :  -236980.639917545"
## [1] "52 :  -236968.747881109"
```

```
## [1] "53 :  -236955.319606755"
## [1] "54 :  -236941.21126332"
## [1] "55 :  -236926.973298764"
## [1] "56 :  -236912.736447399"
## [1] "57 :  -236899.008653393"
## [1] "58 :  -236885.791993216"
## [1] "59 :  -236872.876407395"
## [1] "60 :  -236860.122102281"
## [1] "61 :  -236847.44942732"
## [1] "62 :  -236834.764507672"
## [1] "63 :  -236821.943948862"
## [1] "64 :  -236808.924327895"
## [1] "65 :  -236795.880358309"
## [1] "66 :  -236783.0484875"
## [1] "67 :  -236770.455031725"
## [1] "68 :  -236755.84054902"
## [1] "69 :  -236738.634303517"
## [1] "70 :  -236726.421222461"
## [1] "71 :  -236715.491573304"
## [1] "72 :  -236705.143250066"
## [1] "73 :  -236695.250801212"
## [1] "74 :  -236685.782047431"
## [1] "75 :  -236676.802510086"
## [1] "76 :  -236665.092654991"
## [1] "77 :  -236631.546284492"
## [1] "78 :  -236621.377395259"
## [1] "79 :  -236614.349136085"
## [1] "80 :  -236608.691798136"
## [1] "81 :  -236604.087268238"
## [1] "82 :  -236600.323594018"
## [1] "83 :  -236597.231150615"
## [1] "84 :  -236594.674523768"
## [1] "85 :  -236592.546823248"
## [1] "86 :  -236590.76398526"
## [1] "87 :  -236589.259772063"
## [1] "88 :  -236587.981739756"
## [1] "89 :  -236586.888097459"
## [1] "90 :  -236585.945282334"
## [1] "91 :  -236585.12608924"
## [1] "92 :  -236584.408234569"
## [1] "93 :  -236583.773270691"
## [1] "94 :  -236583.205794439"
## [1] "95 :  -236582.692910382"
## [1] "96 :  -236582.223915164"
## [1] "97 :  -236581.790157267"
## [1] "98 :  -236581.384997785"
## [1] "99 :  -236581.003761497"
## [1] "100 :  -236580.643552633"
## [1] "101 :  -236580.302849671"
## [1] "102 :  -236579.980897817"
## [1] "103 :  -236579.677040886"
## [1] "104 :  -236579.390192359"
## [1] "105 :  -236579.118585736"
## [1] "106 :  -236578.859805344"
```

```
## [1] "107 :  -236578.610984189"
## [1] "108 :  -236578.369028553"
## [1] "109 :  -236578.130774899"
## [1] "110 :  -236577.893048278"
## [1] "111 :  -236577.652631771"
## [1] "112 :  -236577.406169635"
## [1] "113 :  -236577.150021771"
## [1] "114 :  -236576.880074836"
## [1] "115 :  -236576.591503119"
## [1] "116 :  -236576.278459939"
## [1] "117 :  -236575.933667006"
## [1] "118 :  -236575.547852252"
## [1] "119 :  -236575.10896211"
## [1] "120 :  -236574.601036671"
## [1] "121 :  -236574.002579268"
## [1] "122 :  -236573.284162543"
## [1] "123 :  -236572.40487792"
## [1] "124 :  -236571.307027002"
## [1] "125 :  -236569.908129302"
## [1] "126 :  -236568.088765091"
## [1] "127 :  -236565.673647218"
## [1] "128 :  -236562.400778169"
## [1] "129 :  -236557.868688159"
## [1] "130 :  -236551.449423583"
## [1] "131 :  -236542.190997384"
## [1] "132 :  -236528.89896612"
## [1] "133 :  -236510.754030858"
## [1] "134 :  -236488.667173345"
## [1] "135 :  -236467.43160817"
## [1] "136 :  -236452.288757078"
## [1] "137 :  -236442.704116894"
## [1] "138 :  -236436.423130479"
## [1] "139 :  -236432.059980525"
## [1] "140 :  -236428.894812576"
## [1] "141 :  -236426.53651049"
## [1] "142 :  -236424.750946726"
## [1] "143 :  -236423.385087508"
## [1] "144 :  -236422.332593328"
## [1] "145 :  -236421.516682575"
## [1] "146 :  -236420.88060826"
## [1] "147 :  -236420.381840743"
## [1] "148 :  -236419.988248294"
## [1] "149 :  -236419.675451818"
## [1] "150 :  -236419.424918649"
## [1] "151 :  -236419.222550348"
## [1] "152 :  -236419.057617477"
## [1] "153 :  -236418.921948819"
## [1] "154 :  -236418.809312373"
## [1] "155 :  -236418.714943381"
## [1] "156 :  -236418.635185593"
## [1] "157 :  -236418.567219715"
## [1] "158 :  -236418.508858528"
## [1] "159 :  -236418.458392425"
## [1] "160 :  -236418.414472292"
```

```
## [1] "161 :  -236418.376018166"
## [1] "162 :  -236418.342141171"
## [1] "163 :  -236418.312058655"
## [1] "164 :  -236418.284959276"
## [1] "165 :  -236418.25970857"
## [1] "166 :  -236418.234104082"
## [1] "167 :  -236418.202906527"
## [1] "168 :  -236418.152650272"
## [1] "169 :  -236418.048591933"
## [1] "170 :  -236417.806337964"
## [1] "171 :  -236417.25357879"
## [1] "172 :  -236416.158648781"
## [1] "173 :  -236414.43552716"
## [1] "174 :  -236412.297498753"
## [1] "175 :  -236410.073612268"
## [1] "176 :  -236408.00199637"
## [1] "177 :  -236406.191055053"
## [1] "178 :  -236404.652084013"
## [1] "179 :  -236403.343475395"
## [1] "180 :  -236402.206285615"
## [1] "181 :  -236401.18446953"
## [1] "182 :  -236400.232399517"
## [1] "183 :  -236399.315268378"
## [1] "184 :  -236398.406699904"
## [1] "185 :  -236397.485825062"
## [1] "186 :  -236396.534662326"
## [1] "187 :  -236395.535935391"
## [1] "188 :  -236394.471182275"
## [1] "189 :  -236393.318917636"
## [1] "190 :  -236392.052566509"
## [1] "191 :  -236390.637822069"
## [1] "192 :  -236389.028940831"
## [1] "193 :  -236387.163220214"
## [1] "194 :  -236384.952398255"
## [1] "195 :  -236382.268760076"
## [1] "196 :  -236378.921860134"
## [1] "197 :  -236374.617903615"
## [1] "198 :  -236368.88537972"
## [1] "199 :  -236360.930840103"
## [1] "200 :  -236349.340055267"
## [1] "201 :  -236331.419473803"
## [1] "202 :  -236301.764842373"
## [1] "203 :  -236250.425037454"
## [1] "204 :  -236169.967054096"
## [1] "205 :  -236084.710690408"
## [1] "206 :  -236033.152122223"
## [1] "207 :  -236009.805656158"
## [1] "208 :  -235997.584400857"
## [1] "209 :  -235989.600744695"
## [1] "210 :  -235983.671335107"
## [1] "211 :  -235978.979710592"
## [1] "212 :  -235975.129328171"
## [1] "213 :  -235971.891404329"
## [1] "214 :  -235969.118307581"
```

```
## [1] "215 :  -235966.717593581"
## [1] "216 :  -235964.666074994"
## [1] "217 :  -235962.96453007"
## [1] "218 :  -235961.568322183"
## [1] "219 :  -235960.409769672"
## [1] "220 :  -235959.433282874"
## [1] "221 :  -235958.598851411"
## [1] "222 :  -235957.875733703"
## [1] "223 :  -235957.237700145"
## [1] "224 :  -235956.660362128"
## [1] "225 :  -235956.119162662"
## [1] "226 :  -235955.586974034"
## [1] "227 :  -235955.030924121"
## [1] "228 :  -235954.40984129"
## [1] "229 :  -235953.679355049"
## [1] "230 :  -235952.819235843"
## [1] "231 :  -235951.874070461"
## [1] "232 :  -235950.932665246"
## [1] "233 :  -235950.04538947"
## [1] "234 :  -235949.207090336"
## [1] "235 :  -235948.393230692"
## [1] "236 :  -235947.581818417"
## [1] "237 :  -235946.760867443"
## [1] "238 :  -235945.926634902"
## [1] "239 :  -235945.076293501"
## [1] "240 :  -235944.206186134"
## [1] "241 :  -235943.31566031"
## [1] "242 :  -235942.408267116"
## [1] "243 :  -235941.493115072"
## [1] "244 :  -235940.593742133"
## [1] "245 :  -235939.751853166"
## [1] "246 :  -235939.005432012"
## [1] "247 :  -235938.365908058"
## [1] "248 :  -235937.824544957"
## [1] "249 :  -235937.367434809"
## [1] "250 :  -235936.981606528"
## [1] "251 :  -235936.655891689"
## [1] "252 :  -235936.38076686"
## [1] "253 :  -235936.148152539"
## [1] "254 :  -235935.951233269"
## [1] "255 :  -235935.784286849"
## [1] "256 :  -235935.642523805"
## [1] "257 :  -235935.521941987"
## [1] "258 :  -235935.419199074"
## [1] "259 :  -235935.331503172"
## [1] "260 :  -235935.256520451"
## [1] "261 :  -235935.192297729"
## [1] "262 :  -235935.137197973"
## [1] "263 :  -235935.089846675"
## [1] "264 :  -235935.049086943"
## [1] "265 :  -235935.013941254"
## [1] "266 :  -235934.983577144"
## [1] "267 :  -235934.957271913"
## [1] "268 :  -235934.934366025"
```

```
## [1] "269 :  -235934.914180213"
## [1] "270 :  -235934.895832715"
## [1] "271 :  -235934.877796596"
## [1] "272 :  -235934.856809984"
## [1] "273 :  -235934.825344444"
## [1] "274 :  -235934.766889045"
## [1] "275 :  -235934.653185491"
## [1] "276 :  -235934.464967207"
## [1] "277 :  -235934.245357844"
## [1] "278 :  -235934.078959829"
## [1] "279 :  -235933.986990922"
## [1] "280 :  -235933.938177951"
## [1] "281 :  -235933.907166084"
## [1] "282 :  -235933.883444972"
## [1] "283 :  -235933.863492977"
## [1] "284 :  -235933.846102079"
## [1] "285 :  -235933.8307555"
## [1] "286 :  -235933.817151598"
## [1] "287 :  -235933.805069182"
## [1] "288 :  -235933.794327287"
## [1] "289 :  -235933.784771213"
## [1] "290 :  -235933.776266336"
## [1] "291 :  -235933.768694524"
## [1] "292 :  -235933.761951667"
## [1] "293 :  -235933.755945779"
## [1] "294 :  -235933.75059535"
## [1] "295 :  -235933.745828153"
## [1] "296 :  -235933.741580043"
## [1] "297 :  -235933.737794038"
## [1] "298 :  -235933.734419501"
## [1] "299 :  -235933.731411362"
## [1] "300 :  -235933.728729551"
## [1] "301 :  -235933.726338375"
## [1] "302 :  -235933.724206101"
## [1] "303 :  -235933.72230444"
## [1] "304 :  -235933.720608256"
## [1] "305 :  -235933.719095096"
## [1] "306 :  -235933.717745037"
## [1] "307 :  -235933.71654029"
## [1] "308 :  -235933.715465046"
## [1] "309 :  -235933.714505201"
## [1] "310 :  -235933.713648218"
## [1] "311 :  -235933.712882916"
## [1] "312 :  -235933.712199357"
## [1] "313 :  -235933.71158867"
## [1] "314 :  -235933.711042974"
## [1] "315 :  -235933.710555229"
## [1] "316 :  -235933.710119193"
## [1] "317 :  -235933.70972928"
## [1] "318 :  -235933.709380518"
## [1] "319 :  -235933.709068506"
## [1] "320 :  -235933.708789279"
## [1] "321 :  -235933.708539333"
## [1] "322 :  -235933.708315534"
```

```
## [1] "323 :  -235933.708115106"
## [1] "324 :  -235933.707935536"
## [1] "325 :  -235933.707774631"
## [1] "326 :  -235933.707630407"
## [1] "327 :  -235933.707501087"
## [1] "328 :  -235933.707385121"
## [1] "329 :  -235933.707281073"
## [1] "330 :  -235933.70718772"
## [1] "331 :  -235933.707103922"
## [1] "332 :  -235933.707028676"
## [1] "333 :  -235933.706961118"
## [1] "334 :  -235933.706900409"
## [1] "335 :  -235933.706845861"
## [1] "336 :  -235933.706796844"
## [1] "337 :  -235933.706752769"
## [1] "338 :  -235933.706713142"
## [1] "339 :  -235933.706677489"
## [1] "340 :  -235933.706645414"
## [1] "341 :  -235933.70661655"
## [1] "342 :  -235933.706590575"
## [1] "343 :  -235933.70656718"
## [1] "344 :  -235933.706546109"
## [1] "345 :  -235933.706527129"
## [1] "346 :  -235933.70651003"
## [1] "347 :  -235933.706494624"
## [1] "348 :  -235933.706480739"
## [1] "349 :  -235933.706468212"
## [1] "350 :  -235933.706456925"
## [1] "351 :  -235933.706446741"
## [1] "352 :  -235933.706437553"
```

```
print(accuracy_v)
```

```
## [1] "0.604359069080163" "0.62781677133358"  "0.62781677133358"
```

With these output, we can say that the best performance has the highest accuracy value. we could also run the algorithm more than 3 times. In order to save time, we just use n=3 as an example.

Question 2

(a) Create a sampling function for thompson sampling (bernoulli distribution)

```
## Set Bernoulli the success probability for each arm.
ps <- c(0.6,0.4)

arm <- function(ns,ss) {
## the number of successes a
  a <- 1 + ss
## the number of failures b
  b <- 1 + ns - ss
```

```
## probabilities distribution of two arms by using Beta distributions
  L <- rbeta(1,a[1],b[1])
  R <- rbeta(1,a[2],b[2])

## If L is great than R, and we will pull the arm 1 next time, so the function returns
## the value "1";if R is great than L, and we will choose the arm2 next time, so the
## function returns the value "2"
  if (L > R) {
    return(1)
  } else {
    return(2)
  }
}
```

Create the Thompson sampling function

```
th.be <- function(ps,n) {
## the values of arm played from time 1 to n
  as <- rep(0,n)
## the reward values from time 1 to n
  rs <- rep(0,n)

## initialization that the number of plays and number of successes is 0
## for arm 1 and arm 2
  ns <- rep(0,2); ss <- rep(0,2)

  for (i in 1:n) {
    ## choose one of two arms
    a <- arm(ns,ss)

    ## If we get one random probability which is less than probability of arm 'a',
    ## and r should be 1; if we the probability is great than the probability of
    ## arm 'a', and r will be 0
    r <- ifelse(runif(1) < ps[a], 1, 0)
    ns[a] <- ns[a] + 1
    ss[a] <- ss[a] + r
    as[i] <- a
    rs[i] <- r
  }
  return(list(as=as,rs=rs))
}
```

Run one example of Thompson Sampling

```
## Adding the success parameters and time
thompson.bernoulli.out0 <- th.be(ps=ps,n=100)

## The total reward value we get by using Thompson sampling
sum(thompson.bernoulli.out0$rs)
```

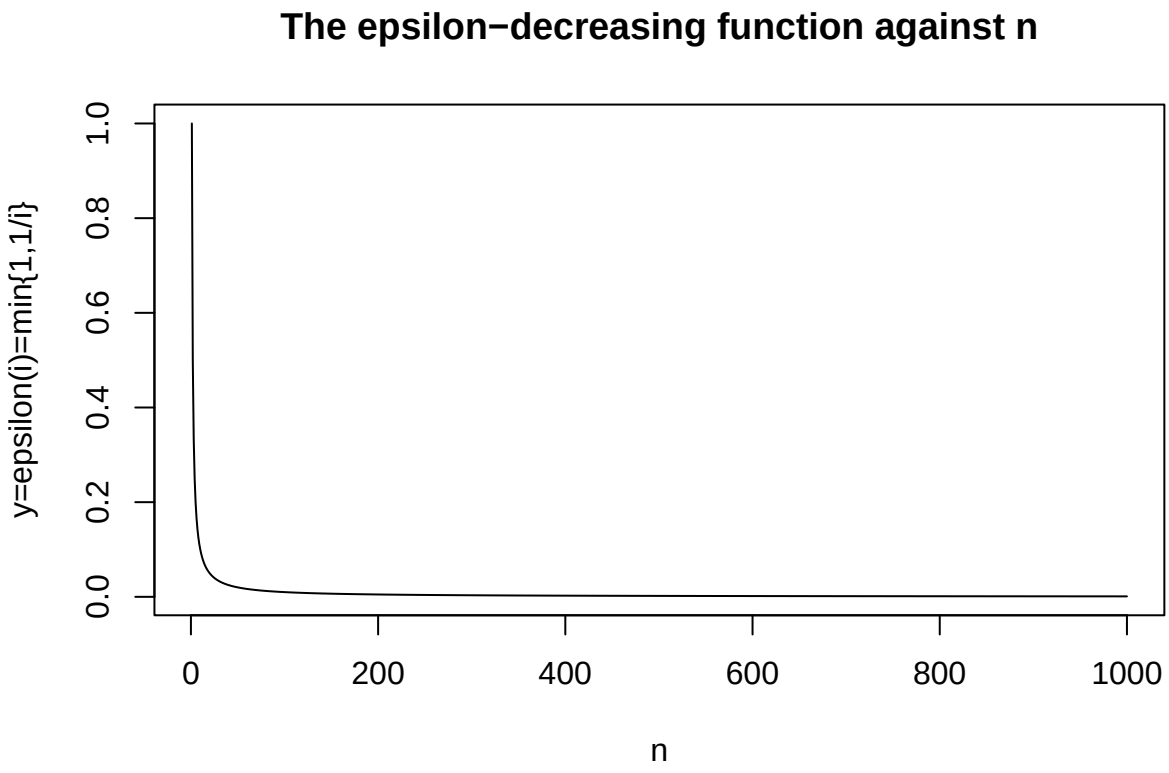
```
## [1] 60
```

Create the epsilon decreasing function $\epsilon(n)$. The decreasing function of ϵ with input the constant value C, the negative value k and time n

```r
ep<-function(C,k,n){
  e<-rep(0,n)

  for(i in 1:n){
  e[i]<-min(1,C*i^k)
  }
  return(e)
}
```

Let C=1, k=-1 and n =1000, we can see the ϵ-decreasing function $\min\{1, i^{-1}\}$ is decreasing from this plot

```r
x<-seq(1,1000,1)
plot(x,ep(C=1,k=-1,1000),type="l",
ylab = "y=epsilon(i)=min{1,1/i}",
xlab="n",
main = "The epsilon-decreasing function against n")
```



Create the ϵ-decreasing function with input ps, n, and ep

```r
epsilon.de <- function(ps,n,ep) {
  ## initialize the list of choosing arms and rewards
  as <- rep(0,n); rs <- rep(0,n)
  ## initialize the numbers of times and successes
  ns <- rep(0,2); ss <- rep(0,2)

  ## Play arm 1 and arm 2 once, record reward, and update number of successes and
```

```
  ## number of plays
  for (i in 1:2) {
    a <- i
    r <- runif(1) < ps[a]
    ns[a] <- ns[a] + 1
    ss[a] <- ss[a] + r
    as[i] <- a
    rs[i] <- r
  }

  ## now follow the epsilon decreasing strategy from the time 3 because we have pulled
  ## two arms from time 1 and 2
  for (i in 3:n) {

    ## with probability of epsilon decreasing function "ep" we have set, we can pick an
    ## arm uniformly at random
   if (runif(1) < ep[i]) {
      a <- sample(2,1)
    } else {
      ## if not, we will choose the best arm so far
      a <- which.max(ss/ns)
    }

     r <- ifelse(runif(1) < ps[a], 1, 0)

    ## update the numbers of plays and successes
     ns[a] <- ns[a] + 1
     ss[a] <- ss[a] + r

    ## record the arm played and the reward received
    as[i] <- a
    rs[i] <- r
  }

  ## print out the record the armed played and the reward received totally
  return(list(as=as,rs=rs))
}
```

(b)

In the part b, the ϵ-Greedy is a simple method to balance exploration and exploitation by choosing between exploration and exploitation randomly. We can select an initial *epsilon* value using the ϵ-decreasing strategy that will gradually drop over time. This makes sure that there is a lot of exploration in the early stages of the experiment, but as time goes on and the best choice becomes more certain, there are fewer iterations dedicated to making less-than-ideal choices.

At first, we will play each arm once, record reward, and update number of successes and number of plays. From time 3 , with prob $\epsilon(3)$, play an arm at random; with prob of $1-\epsilon(3)$, play the one with the best rate of success so far. The $\epsilon_n = \min\{1, Cn^{-1}\}$, and this function is decreasing, so the return value of ϵ is decreasing as the time goes up. This can increasing the prob of choosing the best rate of success so far, so that the value of $1-\epsilon(n)$ is almost close to 1. Now we have $\mu 1 > \mu 2$, and $\exp(sum(r_1 : r_N)/N)$ should be close to our best arm$\mu 1$ which is 0.6 in our question.

Run *epsilon.de* with the given *ps* and $\epsilon_n = \min\{1, Cn^{-1}\}$ for one time, C is a positive constant and k should be negative value

```
## One simple example for 10000 iteration and only 1 run
egd.out1 <- epsilon.de(ps=ps,n=1e4,ep(C=1,k=-1,n=1e4))
sum(egd.out1$rs)
```

```
## [1] 6031
```

```
sum(egd.out1$rs)/length(egd.out1$rs)
```

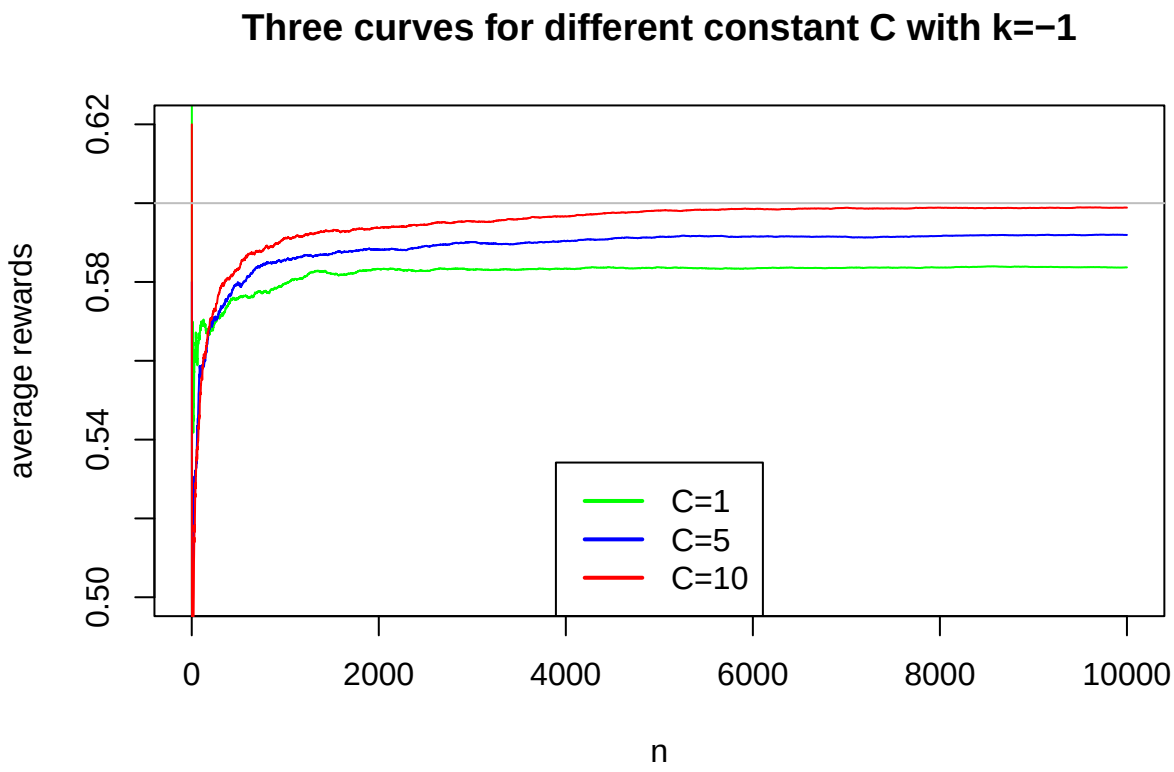
```
## [1] 0.6031
```

We need to run for N times so that we can get the average rewards, so the following function is based on N runs , success probabilities ps, n iterations(times), positive constant C, and negative value k.

```
## Here we create a new function to test epsilon-decreasing algorithm for N runs and n iterations
## and get the value of n rewards

epsilon.t<-function(N,ps,n,C,k){
  reward<-rep(0,n)
  for (i in 1:N) {

    ## we can get the reward for each time and accumulate these rewards as the time increases
    rs<-(epsilon.de(ps,n,ep(C,k,n)))$rs
    reward<-reward+(1/N)*(cumsum(rs)/(1:n))
  }
  return(reward)
}
```

Comparison with different values of constant C, and control other conditions same

```
## With the reward test function, we can change the value of positive constant C, and
## keep other conditions constant, and in this part k=-1
  t1<-epsilon.t(100,ps=c(0.6,0.4),10000,1,-1)
  t2<-epsilon.t(100,ps=c(0.6,0.4),10000,5,-1)
  t3<-epsilon.t(100,ps=c(0.6,0.4),10000,10,-1)

## Using plot function to plot the average reward against time n between three different
## curves and the probability of our best arm of 0.6
plot(n=c(1:10000),t1,type="l",col="green",ylim=c(0.5,0.62),xlab="n",ylab="average rewards",
       main ="Three curves for different constant C with k=-1" )
  lines(n=c(1:10000),t2,type="l",col="blue")
  lines(n=c(1:10000),t3,type="l",col="red")
  abline(h=0.6,col="gray")
  legend('bottom', col=c("green","blue", "red"),
       lty=c(1,1), lwd=c(2,2),
       legend=c("C=1", "C=5","C=10"))
```



Therefore, the plot shows that the higher C can converge to 0.6 faster than smaller C, because smaller C need to run more times than bigger C. The $\epsilon - decreasing$ strategy defined with $\epsilon_n = min\{1, Cn^{-1}\}$ is asymptotically optimal as our graphs with different values of C can converge to 0.6 eventually.

(c) *epsilon.de* with the given *ps* and $\epsilon n = \min\{1, Cn^{-}2\}$ and C is a positive constant

```
## One simple example for 1e4 iteration and only 1 run
egd.out <- epsilon.de(ps=ps,n=1e4,ep(C=1,k=-2,n=1e4))
sum(egd.out$rs)
```

```
## [1] 6020
```

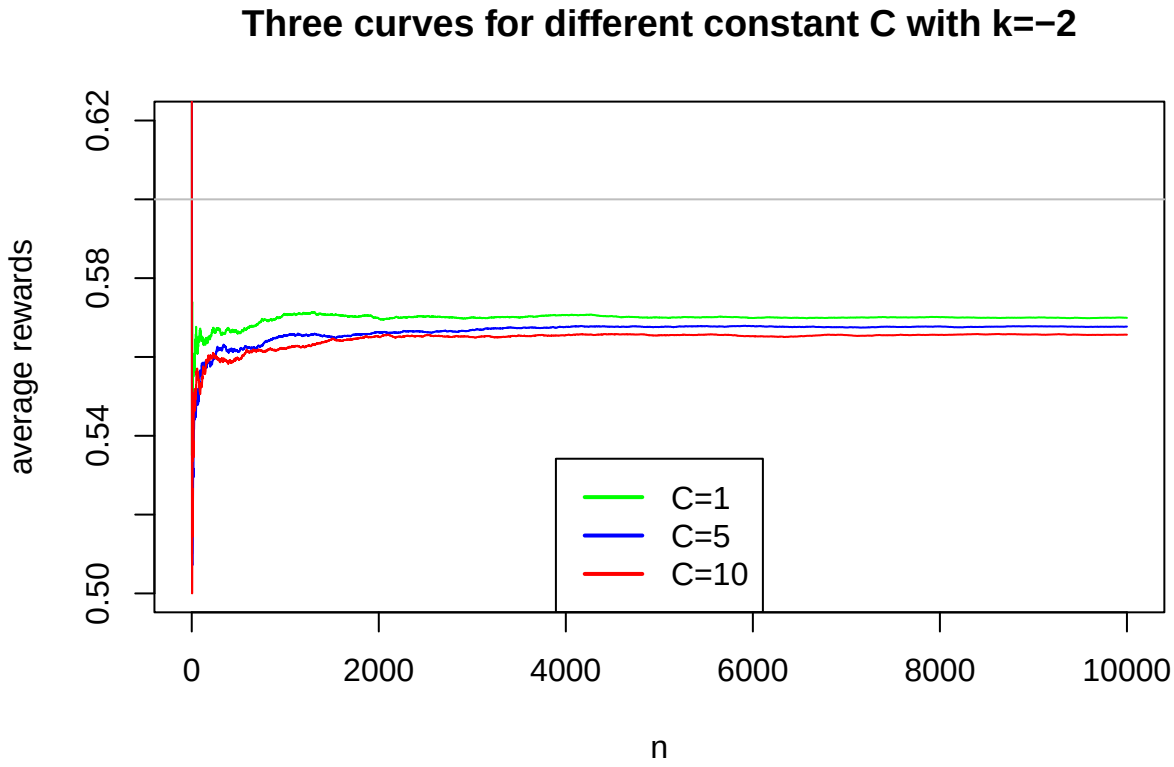
```
sum(egd.out$rs)/length(egd.out$rs)
```

```
## [1] 0.602
```

Comparison with different values of constant C, and control other conditions same

```
## In this part, we will run for 100 times and 10000 iterations for getting the average reward.
  t1<-epsilon.t(100,ps=c(0.6,0.4),10000,1,-2)
  t2<-epsilon.t(100,ps=c(0.6,0.4),10000,5,-2)
  t3<-epsilon.t(100,ps=c(0.6,0.4),10000,10,-2)

plot(n=c(1:10000),t1,type="l",ylim=c(0.5,0.62),col="green",xlab="n",ylab="average rewards",
       main ="Three curves for different constant C with k=-2" )
```

```
  lines(n=c(1:10000),t2,type="l",col="blue")
  lines(n=c(1:10000),t3,type="l",col="red")
  abline(h=0.6,col="gray")
  legend('bottom', col=c("green","blue", "red"),
       lty=c(1,1), lwd=c(2,2),
       legend=c("C=1", "C=5","C=10"))
```



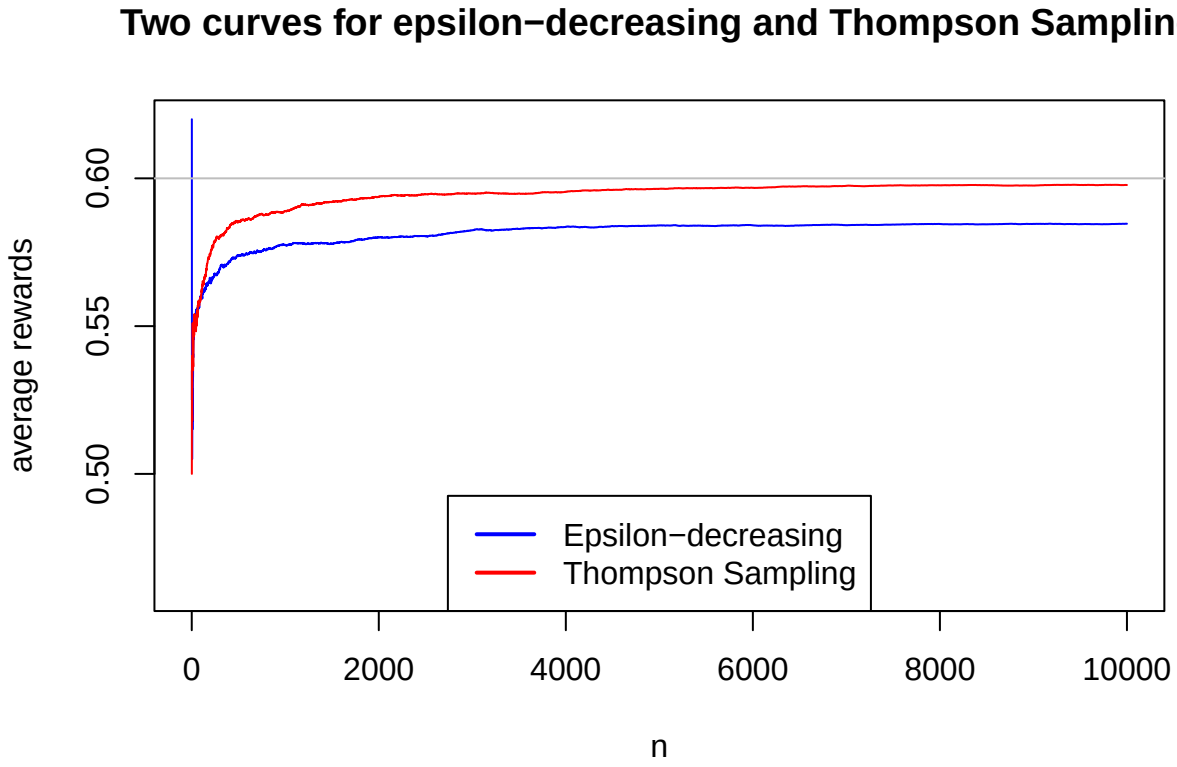
The $\epsilon - decreasing$ strategy defined with $\epsilon_n = min\{1, Cn^{-2}\}$ is not asymptotically optimal because our graphs with the highest and smallest C cannot converge to 0.6 eventually.

(d) part(i) Compare graph of the implementations of ϵ-decreasing and Thompson sampling for this problem

```
## Create a Thompson Sampling reward function
Thompson.t<-function(N,ps,n){
  reward<-rep(0,n)
  for (i in 1:N) {
    #record the reward
    rs<-(th.be(ps,n))$rs
    #accumulate these rewards
    reward<-reward+(1/N)*(cumsum(rs)/(1:n))
  }
  return(reward)
}
```

Compare two curves of asymptotically optimal ϵ-decreasing and Thompson Sampling

```
## Using epsilon-decreasing function that epsilon(i)=min{1,1/i}, and compare this function
## with the Thompson Sampling of the same runs and lengths
e<-epsilon.t(100,ps=c(0.6,0.4),10000,1,-1)
t<-Thompson.t(100,ps=c(0.6,0.4),10000)

plot(n=c(1:10000),e,type="l",ylim=c(0.46,0.62),col="blue",xlab="n",ylab="average rewards",
       main ="Two curves for epsilon-decreasing and Thompson Sampling" )
  lines(n=c(1:10000),t,type="l",col="red")
  abline(h=0.6,col="gray")
  legend('bottom', col=c("blue","red"),
       lty=c(1,1), lwd=c(2,2),
       legend=c("Epsilon-decreasing", "Thompson Sampling"))
```



From this plot, we can see that Thompson Sampling can converge faster to 0.6 than the ϵ-decreasing. So we suppose that Thompson Sampling is better than the ϵ-deceasing.

(d) part(ii)For two asymptotically optimal procedures, there is a bound on how good any procedure can be in terms of regret: Expected lost reward (From the lecture notes).

$$R(n) = \exp[sum_{i=1}^{n}(\mu_1 - R_i)]$$

and our best arm probability $\mu_1 = 0.6$ in our question. From the Optimal asymptotic regret of Thompson Sampling with $\mu_1 > \mu_2$,

we can get

$$\boldsymbol{R(n)} \leq (1+\epsilon)\frac{\mu_1 - \mu_2}{D_{KL}(mu_2||mu_1)}log(n) + O_{\epsilon,\mu_1,\mu_2}(log(n))$$

(From the lecture notes) for any $\epsilon > 0$ and $\mu_1 = 0.6$, and $\mu_2 = 0.4$

```
## Calculate D_kl constant
D_kl<-0.2/(0.6*log(6/4)+0.4*log(4/6))
D_kl
```

```
## [1] 2.466303
```

Therefore we can get

$$\frac{\mu_1 - \mu_2}{D_{KL}(mu_2||mu_1)} = \frac{0.6 - 0.4}{D_{KL}(0.6||0.4)} = \frac{0.2}{0.6log\frac{0.6}{0.4} + (1-0.6)log\frac{1-0.6}{1-0.4}} = 2.466303$$

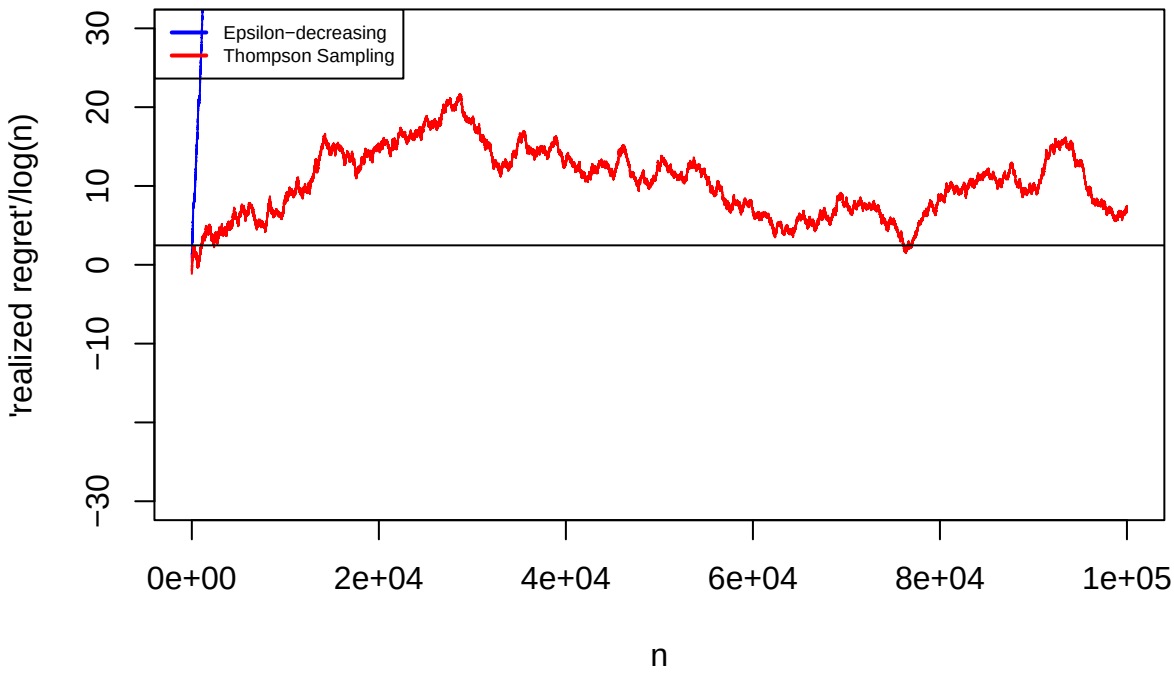
and we take this constant value into the plot of "realized regret"/log(n) against n

```
n<-1e5
## Create the expected lost reward using the above equations for epsilon-decreasing
ep.r<-ps[1]*(1:n)-cumsum((epsilon.de(ps,n,ep(C=1,-1,n)))$rs)
ep.Rn<-ep.r/log(1:n)
## Create the expected lost reward using the above equations for Thompson Sampling
th.r<-ps[1]*(1:n)-cumsum((th.be(ps,n))$rs)
th.Rn<-th.r/log(1:n)
```

Plot the graph of regret/log(n) against n and the constant line

$$\frac{\mu_1 - \mu_2}{D_{KL}(mu_2||mu_1)} = 2.466303$$

```
## Plot two curves and constant
plot(n=c(1:100000),ep.Rn,ylim=c(-30,30),col="blue",type="l",xlab="n",ylab="'realized regret'/log(n)")
  lines(n=c(1:100000),th.Rn,col="red",type="l")
  abline(h=2.466303)
  legend('topleft', col=c("blue","red"),
       lty=c(1,1), lwd=c(2,2), cex=0.6,
       legend=c("Epsilon-decreasing", "Thompson Sampling"))
```



From this plot, it shows that Thompson sampling has smaller regret than ϵ-decreasing with the same times n , a more stable $R(n)/log(n)$, and we have seen that Thomspon Sampling can converge faster than ϵ, so Thompson Sampling is better than ϵ-decreasing strategy. We can usually choose the Thompson Sampling in this problem.

Question 3

(a) we know that using the inverse-distance weighting we could get the predicted values as :

$$\frac{\sum_{i=1}^{k} w_i y_i}{\sum_{i=1}^{k} w_i}$$

with

$$w_i = \frac{1}{k_i}$$

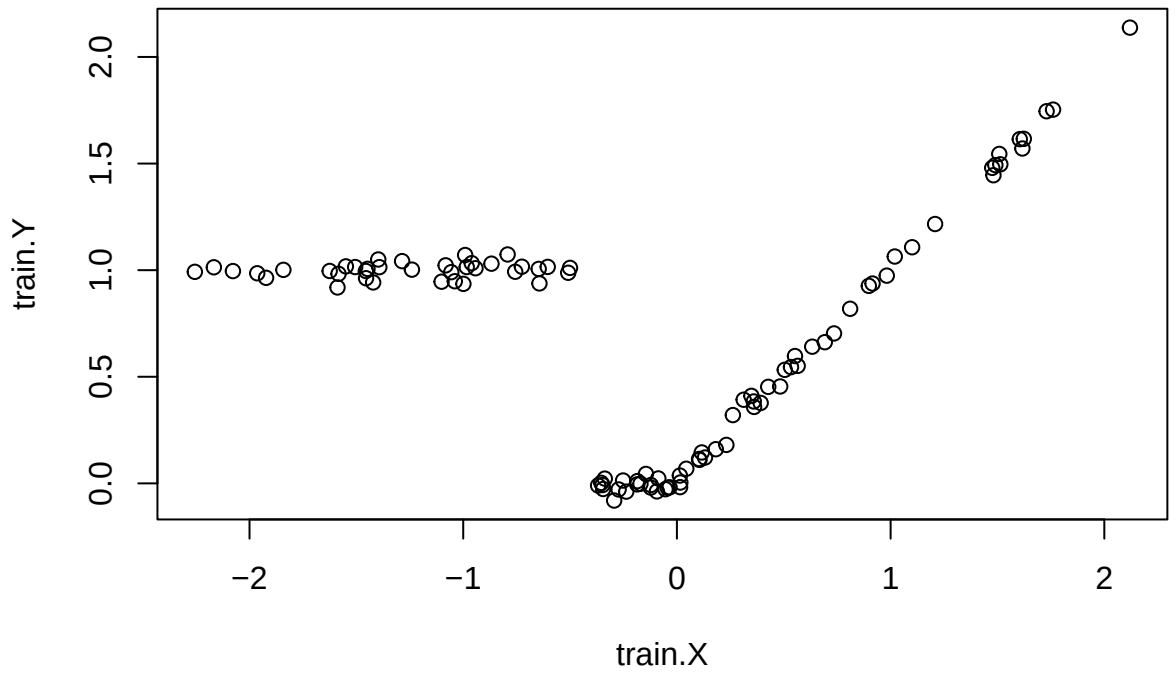
```
knn.regression.test <- function(k, train.X, train.Y, test.X, test.Y, distances) {
 # calculate the distance between trainX and testX
  distances <- distances(train.X,test.X)
 # the weighted sum
 weight_Y <- rep(0,length(test.X[,1]))
weight <- rep(0,length(test.X[,1]))
  for(i in 1:length(test.X[,1])){
  # sort in increasing order
   sort <- sort(distances[,i])
   # for each of the KNN
```

```
    for(x in 1:k){
     weight_Y[i] <- weight_Y[i] + (1/sort[x])*(train.Y[match(sort[x],distances[,i])])
      weight[i] <- weight[i] + (1/sort[x])
    }
  }
  estimates <- weight_Y/weight
 print(sum((test.Y - estimates)^2))
}
```

(b)

Toy data 1

```
distances.l1 <- function(X,W) {
  apply(W,1,function(p) apply(X,1,function(q) sum(abs(p-q))))
}
n <- 100
set.seed(2021)
train.X <- matrix(sort(rnorm(n)),n,1)
train.Y <- (train.X < -0.5) + train.X*(train.X>0)+rnorm(n,sd=0.03)
plot(train.X,train.Y)
```




```
test.X <- matrix(sort(rnorm(n)),n,1)
test.Y <- (test.X < -0.5) + test.X*(test.X>0)+rnorm(n,sd=0.03)
```

```
k <- 2
knn.regression.test(k,train.X,train.Y,test.X,test.Y,distances.l1)
```

```
## [1] 3.91539
```

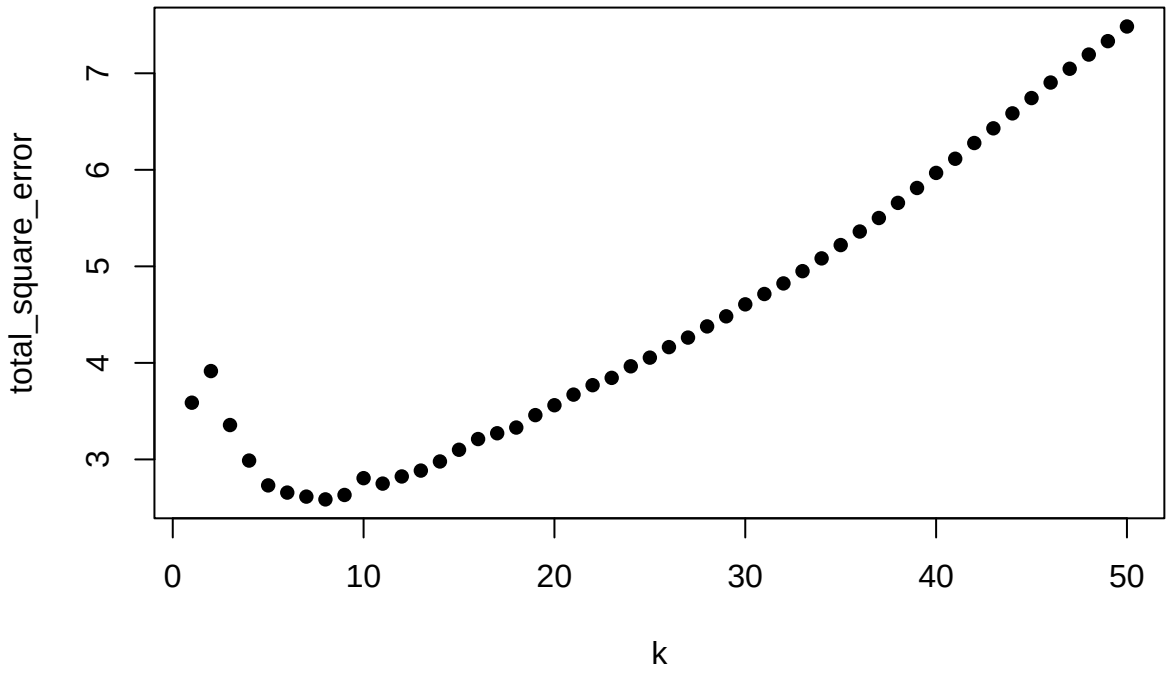
we could get different value for k from 1 to 50:

```
total_square_error<- rep(0,50)
for(i in 1:50) {
total_square_error[i] <- knn.regression.test(i,train.X,train.Y,test.X,test.Y,distances.l1)
}
```

```
## [1] 3.587658
## [1] 3.91539
## [1] 3.356198
## [1] 2.988354
## [1] 2.731105
## [1] 2.656472
## [1] 2.614705
## [1] 2.586138
## [1] 2.631774
## [1] 2.805828
## [1] 2.749701
## [1] 2.823711
## [1] 2.884471
## [1] 2.978788
## [1] 3.100061
## [1] 3.211132
## [1] 3.271182
## [1] 3.330101
## [1] 3.458824
## [1] 3.561326
## [1] 3.671474
## [1] 3.769173
## [1] 3.844918
## [1] 3.964797
## [1] 4.054525
## [1] 4.163418
## [1] 4.262161
## [1] 4.378077
## [1] 4.482333
## [1] 4.606498
## [1] 4.713636
## [1] 4.823312
## [1] 4.950254
## [1] 5.082642
## [1] 5.220597
## [1] 5.360952
## [1] 5.50079
## [1] 5.657626
## [1] 5.812036
## [1] 5.967978
```

```
## [1] 6.114774
## [1] 6.277597
## [1] 6.429928
## [1] 6.584894
## [1] 6.743505
## [1] 6.904092
## [1] 7.046885
## [1] 7.19396
## [1] 7.333013
## [1] 7.484759
```

```
plot(total_square_error,xlab="k",pch=16)
```



From the graph we can see that for k$>$8 total_square_error values increase linearly as k increases, for $2 \leq k \leq 8$ total_square_error values decrease linearly as k increases.

```
min_index <- which.min(total_square_error)
min_value <- total_square_error[min_index]
min_index
```

```
## [1] 8
```

```
min_value
```

```
## [1] 2.586138
```

The minimum total_square_error is 2.586138 when k=8

Toy data 2

```
set.seed(100)
train.X <- matrix(rnorm(200),100,2)
train.Y <- train.X[,1]
test.X <- matrix(rnorm(100),50,2)
test.Y <- test.X[,1]
k <- 3
knn.regression.test(k,train.X,train.Y,test.X,test.Y,distances.l1)
```

```
## [1] 3.232214
```

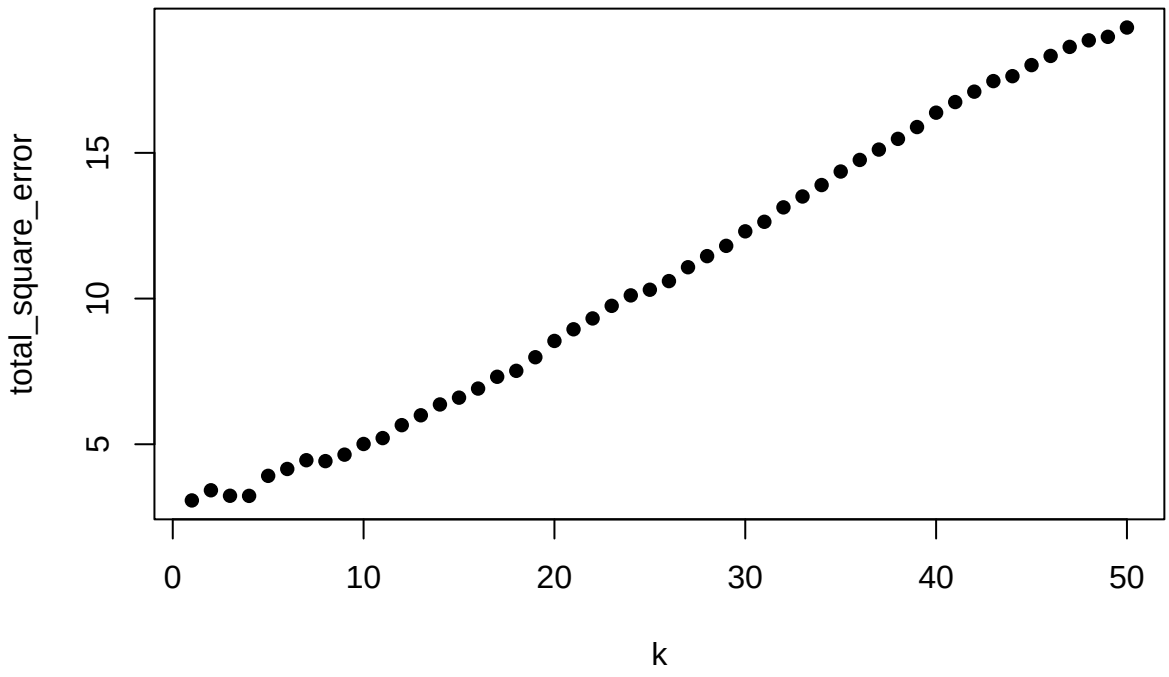
we could get different value for k from 1 to 50:

```
total_square_error<- rep(0,50)
for(i in 1:50) {
total_square_error[i] <- knn.regression.test(i,train.X,train.Y,test.X,test.Y,distances.l1)
}
```

```
## [1] 3.072546
## [1] 3.420534
## [1] 3.232214
## [1] 3.228323
## [1] 3.916005
## [1] 4.152378
## [1] 4.452607
## [1] 4.420877
## [1] 4.643639
## [1] 5.009417
## [1] 5.212056
## [1] 5.655799
## [1] 5.994365
## [1] 6.365525
## [1] 6.599111
## [1] 6.91093
## [1] 7.315493
## [1] 7.519298
## [1] 7.986425
## [1] 8.545205
## [1] 8.945644
## [1] 9.317943
## [1] 9.749311
## [1] 10.1087
## [1] 10.3025
## [1] 10.59918
## [1] 11.07523
## [1] 11.45585
## [1] 11.80846
## [1] 12.30688
## [1] 12.63405
## [1] 13.12897
```

```
## [1] 13.50266
## [1] 13.8949
## [1] 14.35943
## [1] 14.75709
## [1] 15.11221
## [1] 15.47975
## [1] 15.88535
## [1] 16.37828
## [1] 16.74274
## [1] 17.10024
## [1] 17.4621
## [1] 17.63137
## [1] 18.01166
## [1] 18.32617
## [1] 18.63642
## [1] 18.86075
## [1] 18.9816
## [1] 19.30032
```

```
plot(total_square_error,xlab="k",pch=16)
```



From the graph we see tnat for 1≤ k total_square_error values increase linearly as k increases.

```
min_index <- which.min(total_square_error)
min_value <- total_square_error[min_index]
min_index
```

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

```
min_value
```

```
## [1] 3.072546
```

The minimum total_square_error is 3.072546 when k=1

Oevrall from both graphs we see that as k increases,the total_square_error shows an approximately linear relationship with k

(c) We use tibble to show years and predictions combined with part (a)

```
Estimates<- function(k, train.X, train.Y, test.X, test.Y, distances) {
 distances <- distances(train.X,test.X)
 weight_Y <- rep(0,length(test.X[,1]))
  weight <- rep(0,length(test.X[,1]))
  for(i in 1:length(test.X[,1])){
    sort <- sort(distances[,i])
     for(x in 1:k){
      weight_Y[i] <- weight_Y[i] + (1/sort[x])*(train.Y[match(sort[x],distances[,i])])
      weight[i] <- weight[i] + (1/sort[x])
    }
  }
  return( weight_Y/weight)
}
distances.l2 <- function(X,W) {
apply(W,1,function(p) apply(X,1,function(q) sqrt(sum((p-q)^2))))
}
Iowa <- read.delim("~/Downloads/Iowa.txt")
train.X=as.matrix(Iowa[seq(1,33,2),1:9])
train.Y=c(Iowa[seq(1,33,2),10])
test.X=as.matrix(Iowa[seq(2,32,2),1:9])
test.Y=c(Iowa[seq(2,32,2),10])
k <- 5
tibble(year = test.X[,1]) %>%
  mutate(prediction =Estimates(k,train.X,train.Y,test.X,test.Y,distances.l2))
```

```
## # A tibble: 16 x 2
##     year prediction
##    <dbl>      <dbl>
##  1  1931       36.4
##  2  1933       39.6
##  3  1935       47.2
##  4  1937       41.7
##  5  1939       47.4
##  6  1941       46.7
##  7  1943       53.5
##  8  1945       55.0
##  9  1947       55.5
## 10  1949       56.6
## 11  1951       55.5
## 12  1953       56.2
```

```
## 13  1955       59.2
## 14  1957       58.8
## 15  1959       64.4
## 16  1961       60.2
```

(d)

ordinary least squares

```
Iowa <- read.delim("~/Downloads/Iowa.txt")
train <- Iowa[seq(1,33,2),]
test.X <- Iowa[seq(2,32,2),1:9]
test.Y <- Iowa[seq(2,32,2),10]
ols_pridiction <- predict(lm(Yield~., data = train), newdata = test.X)
sum((test.Y - ols_pridiction)^2)
```

```
## [1] 1891.556
```

```
ols_error<-sum((test.Y - ols_pridiction)^2)
ols_error
```

```
## [1] 1891.556
```

Ridge regression

```
train <- Iowa[seq(1,33,2),]
test.X <- Iowa[seq(2,32,2),1:9]
test.Y <- Iowa[seq(2,32,2),10]
ridge_regression <-lm.ridge(Yield~ ., data = train, lambda = 10)
ridge_pridiction <- as.matrix(cbind(const=1, test.X)) %*% coef(ridge_regression)
ridge_error<- sum((test.Y - ridge_pridiction)^2)
ridge_error
```

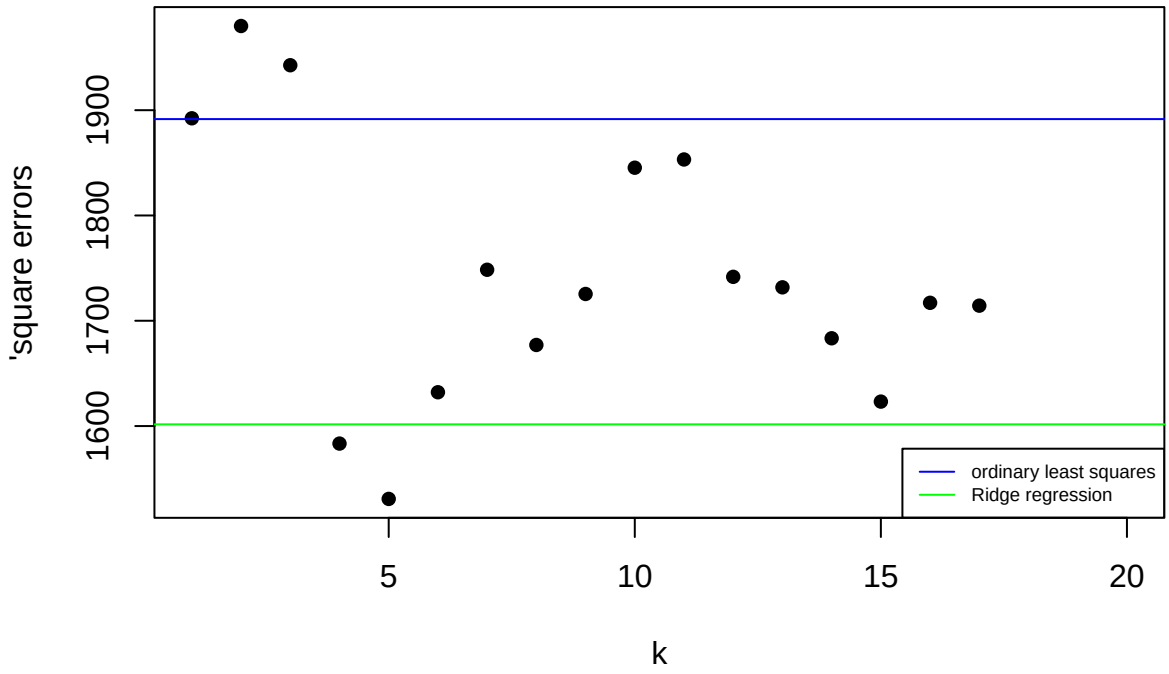
```
## [1] 1601.603
```

```
total_square_error<- rep(0,20)
for(i in 1:20) {
total_square_error[i] <- knn.regression.test(i,train.X,train.Y,test.X,test.Y,distances.l2)
}
```

```
## [1] 1892.31
## [1] 1979.957
## [1] 1942.714
## [1] 1583.353
## [1] 1530.799
## [1] 1632.112
## [1] 1748.446
## [1] 1677.013
## [1] 1725.433
## [1] 1845.36
## [1] 1853.193
```

```
## [1] 1741.733
## [1] 1731.75
## [1] 1683.367
## [1] 1623.224
## [1] 1717.06
## [1] 1714.341
## [1] NA
## [1] NA
## [1] NA
```

```
plot(c(1:20),total_square_error,xlab="k",ylab="'square errors",pch=16)
 abline(h=ols_error,col="blue")
 abline(h=ridge_error,col="green")
 legend('bottomright', col=c("blue","green"),
    lty=c(1,1),lwd=c(1,1),cex=0.6,
    legend=c("ordinary least squares", "Ridge regression"))
```



The blue line is ordinary least squares model and the green line is Ridge regression model.The ordinary least squares model is greater than Ridge regression model.The lowest point in Knn is less than Ridge regression model,but large proportion of the points lies between least squares model and Ridge regression model.So the estimation for Ridge regression model and Knn are better than the estimation for ordinary least squares model.The estimation of Ridge regression model and Knn are depend on the specific value of K in general case.