MA461-Ass4-HaplotypeInference

Vishvesh Karthik

29 March 2019

```
library(stringr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
iter_count <<- 1
final.theta.est <<- c()
final.g <<- list()
final.freq_df <<- c()
previous_logLL <<- 1
logLL <<- 0
# for plotting
allLL <<- c()
# results<-list() LAZY verbosity change
verbose <- TRUE
precision <- 3

generate_theta <- function(no_col) {
    no_row <- 2^(no_col)
    tmp <- matrix(rep(0, no_col * (no_row)), nrow = no_row)
    iteration <- 0
    for (i in rev(1:no_col)) {
        skip <- 2^iteration
        indices <- seq(from = skip + 1, to = no_row, by = skip +
            skip)
        # print('i') print(indices) print('r')
        rep_length <- (skip - 1)
        for (n in indices) {
            tmp[seq(n, n + rep_length), i] = 1
        }
        iteration <- iteration + 1
    }
```

```
    return(as.data.frame(tmp))
}

deconstruct_genotypes <- function(Gi, theta) {
    # read each row from Gi matrix give priority for
    # homozygosity(0's and 2's)
    selection <- list()
    for (row in 1:nrow(Gi)) {
        selection[[row]] <- list()
        for (col in 1:ncol(Gi)) {
            switch(as.character(Gi[row, col]), `0` = selection[[row]][[col]] <- (rownames(theta[theta[,
                col] == 0, ])), `1` = selection[[row]][[col]] <- (rownames(theta)),
                `2` = selection[[row]][[col]] <- (rownames(theta[theta[,
                  col] == 1, ])))

        }
        if (verbose) {
            print(selection[[row]])
        }
    }
    sel.unique <- list()
    for (g in 1:length(selection)) {
        sel.str <- c()
        for (i in selection[[g]]) {
            sel.str <- c(sel.str, i)
        }
        sel.unique[g] <- list(sort(unique(sel.str)))
    }
    # return(sel.unique) select rows from theta which match Gi
    # Have two iterations of rows, one moving slower than the
    # other and check if the rows add up to g
    sel.theta <- list()
    for (g in 1:length(sel.unique)) {
        sel.theta[[g]] <- list()
        for (slower_it in 1:length(sel.unique[[g]])) {
            ## print('slow:') print(slower_it) print('fast:')
            for (faster_it in 1:length(sel.unique[[g]])) {
                # colsums == Gi[g]? print('fast:') print(faster_it)
                if (slower_it != faster_it) {
                  sel.dfs <- rbind(theta[sel.unique[[g]][slower_it],
                    ], theta[sel.unique[[g]][faster_it], ])
                  if (all(colSums(sel.dfs) == Gi[g, ])) {
                    sel.theta[[g]][[slower_it]] <- rownames(sel.dfs)
                    print(sel.theta[[g]][[slower_it]])
                  }

                }
                # print('...')
            }
        }
    }
    # Remove NULLs
    sel.theta <- list.rmNulls(sel.theta)
```

```
    ## remove duplicates(transversions)
    final.sel <- list()
    for (ele in sel.theta) {
        # print(ele)
        print("...")
        # for(sub.list in 1:length(ele)){
        dups <- which(duplicated(lapply(ele, function(x) sort(x))))
        final.sel[[length(final.sel) + 1]] <- ele[-c(dups)]
        if (verbose) {
            print("Dups")
            print(ele[c(dups)])
            # ele[c(dups)]<-NULL
            print("final")
            print(final.sel[[length(final.sel)]])
        }
        # }
    }
    final.sel <- list.rmNulls(final.sel)
    return(final.sel)
}

# estimation maximization
EM <- function(N, sel.theta, theta, iters) {
    init_freq <- 1/nrow(theta)
    g <- list()
    elements <- c()
    for (init_i in 1:length(sel.theta)) {
        g[[init_i]] <- list()
        row_str <- c()
        for (init_j in 1:length(sel.theta[[init_i]])) {
            g[[init_i]][[init_j]] <- init_freq
            tmp_name <- getCString(sel.theta, init_i, init_j)
            rowName <- sel.theta[[init_i]][[init_j]][1]
            colName <- sel.theta[[init_i]][[init_j]][2]
            elements <- c(elements, rowName, colName)
            row_str <- c(row_str, tmp_name)
        }
        names(g[[init_i]]) <- row_str
    }
    names(g) <- paste("g", 1:length(sel.theta), sep = "")
    theta.est <- data.frame(e = rep(init_freq, length(unique(elements))),
        row.names = unique(elements))

    ## POPULATE theta associations with g MAKE SURE to check
    ## strictly because '2' might be mistaken to be in '12'
    ## grepl(y,names(x),fixed = TRUE) which('2'==c('2','12','24'))
    ## freq_list<-lapply(rownames(theta.est), function(y){
    ## lapply(g,function(x){
    ## lapply(splitGString(names(x)),function(z)
    ## {which(as.numeric(z)==as.numeric(y))})})}) GREP is a mofo,
    ## thinks '2' is same as '12' and creates issues, fixed with
    ## reg expression
    freq_list <- lapply(rownames(theta.est), function(y) {
```

```
        lapply(g, function(x) {
            which(grepl(paste("\\b", y, "\\b", sep = ""), names(x)))
        })
    })
    names(freq_list) <- rownames(theta.est)
    freq_df <- data.frame(row.names = c(paste("g", 1:length(g),
        sep = "")))

    ## sub.list =3 ##remove later
    for (sub.list in 1:length(freq_list)) {
        tmp <- data.frame(unlist(freq_list[[sub.list]], recursive = T))
        col <- rownames(theta.est[])[sub.list]
        colnames(tmp) <- col
        missing_rows <- setdiff(rownames(freq_df), rownames(tmp))
        ## REMEMBER this because it was a pain in the ass Also future
        ## reference for merging dataframes(This code fills 0 on
        ## missing rows)

        tmp_colnames <- colnames(tmp)
        for (var in missing_rows) {
            new_row = 0
            names(new_row) <- var
            new_row_df <- data.frame(new_row, row.names = var)
            colnames(new_row_df) <- tmp_colnames
            tmp <- rbind(tmp, new_row_df)
        }
        tmp <- data.frame(tmp[order(row.names(tmp)), ], row.names = rownames(freq_df))
        colnames(tmp) <- tmp_colnames
        freq_df <- cbind(freq_df, tmp)
    }
    knitr::kable(print.data.frame(freq_df))
    knitr::kable(print.data.frame(theta.est))

    # initialization done

    # E step
    e_step(N, g, theta.est, freq_df, sel.theta, iters)
    print("Iteration Ended")
    knitr::kable(print.data.frame(final.freq_df))
    knitr::kable(print.simple.list(final.g))
    knitr::kable(print.data.frame(final.theta.est))
    if (previous_logLL == logLL) {
        print(str_c("EM algorithm converges at iteration", iter_count -
            1, "with logLL value:", logLL, sep = " "))

    }
    plot(allLL, xlab = "Iteration Count", ylab = "LogLL", main = "Convergance plot",
        type = "b")
}

calculate_logLL <- function(final.g, final.theta.est, final.freq_df) {
    # for each row in freq_df{
    # sum(2*(theta.est[colname(h1)]*theta.est[colname(h2)])) } ?
```

```
    ll <- 0
    for (gN in 1:nrow(final.freq_df)) {
        t_sum <- 0
        for (l in levels(factor(final.freq_df[gN, which(final.freq_df[gN,
            ] != 0)]))) {
            # final.freq_df[gN,which((factor(final.freq_df[gN,]))==l)]
            t_sum <- sum(t_sum, prod(final.theta.est[colnames(final.freq_df[gN,
                which((factor(final.freq_df[gN, ])) == l)]),
                ]) * 2)
        }
        ll <- sum(ll, log(t_sum))
    }
    return(round(ll, digits = precision))
}

e_step <- function(N, g, theta.est, freq_df, sel.theta, iters) {

    index_g <- 4  ##rem
    ## q is theta.est g is haplotype combos PARALLELIZE FOLLOWING
    ## FOR LOOP
    print("----------------------------------")
    tmp_g <- g
    print(str_c("Iteration :", iter_count))
    for (gN in 1:nrow(freq_df)) {
        # freq_df[gN,which(freq_df[gN,]!=0)]
        combos <- as.numeric(levels(factor(freq_df[gN, which(freq_df[gN,
            ] != 0)])))
        for (c in combos) {
            ## which(freq_df[gN,]!=0)
            current_combo <- as.numeric(colnames(freq_df[gN,
                which(freq_df[gN, ] == c)]))  ##colnames of the combos
            ## which(freq_df[gN,]!=0)
            other_combos <- as.numeric(colnames(freq_df[gN, which(freq_df[gN,
                ] != c)]))
            denominator <- 0
            numerator <- 0
            est_value <- 0
            numerator <- prod(theta.est[row.names(theta.est) %in%
                current_combo, ])  ##numerator
            level_of_current_combo <- as.numeric(levels(factor(freq_df[gN,
                which(freq_df[gN, ] != 0)]))[which(levels(factor(freq_df[gN,
                which(freq_df[gN, ] != 0)])) %in% freq_df[gN,
                colnames(freq_df) %in% current_combo])])
            total_sum <- 0
            combo_product <- 0
            if (length(combos) > 1) {
                for (l in as.numeric(levels(factor(freq_df[gN,
                  which(freq_df[gN, ] != 0)])))) {
                  combo_product <- prod(theta.est[row.names(theta.est) %in%
                    as.numeric(colnames(freq_df[gN, freq_df[gN,
                      ] %in% l])), ])
                  total_sum <- sum(total_sum, combo_product)
                }
```

```
                denominator <- total_sum
            } else {
                denominator <- prod(theta.est[row.names(theta.est) %in%
                  current_combo, ])
            }
            est_value <- numerator/denominator
            g[[gN]][[getCString(sel.theta, gN, c)]] <- (est_value)

        }

    }
    final.g <<- g
    if (any(lapply(g, function(x) any(is.na(x))))) {
        if (verbose) {
            print(tmp_g)
            print(theta.est)
            print(freq_df)
        }
        final.freq_df <<- freq_df
        final.g <<- tmp_g
        final.theta.est <<- theta.est
    } else {
        if (iter_count <= iters && previous_logLL != logLL) {
            if (verbose) {
                print(g)
            }
            # final.g<<-g
            m_step(N, g, theta.est, freq_df, sel.theta, iters)
        } else {
            print(g)
            print(theta.est)
            print(freq_df)
        }
    }
}

m_step <- function(N, g, theta.est, freq_df, sel.theta, iters) {
    # lapply(g,function(x) names(x)) PARALLELIZE THIS M Step
    # updates theta.est table
    for (col in 1:ncol(freq_df)) {
        total <- 0

        for (var_g in rownames(freq_df[col])) {
            if (freq_df[col][var_g, ] != 0) {
                # print('-------------------------') print('g')
                # print(g[[var_g]][[freq_df[var_g,col]]])
                total <- sum(total, g[[var_g]][[freq_df[var_g,
                  col]]])
                # print('total') print(total)
                # print('-------------------------')
            }
        }
        # print('total/2N') print(total/(2*N))
```

```
        theta.est[colnames(freq_df[col]), ] = (total/(2 * N))
    }

    ## Calculate logLL
    previous_logLL <<- logLL
    logLL <<- calculate_logLL(g, theta.est, freq_df)
    allLL <<- c(allLL, logLL)
    print(str_c("logLL:", logLL, sep = " "))

    iter_count <<- iter_count + 1
    if (iter_count <= iters && previous_logLL != logLL) {
        if (verbose) {
            print(theta.est)
        }
        e_step(N, g, theta.est, freq_df, sel.theta, iters)
    } else {
        if (verbose) {
            print(theta.est)
            print(freq_df)
        }
        final.freq_df <<- freq_df
        final.theta.est <<- theta.est
        ## OFFICIAL ENDPOINT OF EM ITERATIONS that's why I'm setting
        ## final values here
    }
    ## return(g,theta.est,iter_count)

}

## HELPER FUNCTIONS

cbind.fill <- function(...) {
    nm <- list(...)
    nm <- lapply(nm, as.matrix)
    n <- max(sapply(nm, nrow))
    do.call(cbind, lapply(nm, function(x) rbind(x, matrix(, n -
        nrow(x), ncol(x)))))
}

getGString <- function(value) {
    return(paste("g", value, sep = ""))
}
splitGString <- function(value) {
    return(strsplit(value, ","))
}
getCString <- function(sel.theta, i, j) {
    return(str_c(sel.theta[[i]][[j]], collapse = ","))
}
is.NullOb <- function(x) is.null(x) | all(sapply(x, is.null))
list.rmNulls <- function(x) {
    x <- Filter(Negate(is.NullOb), x)
    lapply(x, function(x) if (is.list(x))
        list.rmNulls(x) else x)
```

```
}
```

```
library(kableExtra)
# read g(i) from file
Gi <- read.csv(file = "MA461_Ass4_data.csv", header = TRUE, sep = ",")
Gi <- data.frame(Gi[2:ncol(Gi)], row.names = Gi[, 1])
# generate theta (haplotype table) Construct theta table
# based on the number of columns in g(i)
theta <- generate_theta(ncol(Gi))
knitr::kable(Gi)
```

	n1	n2	n3	n4
g1	0	1	1	2
g2	1	0	2	2
g3	0	1	2	2
g4	1	1	2	2
g5	1	0	2	2

```
knitr::kable(theta)
```

V1	V2	V3	V4
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

```
# dceonstruct genotypes into multiple haplotype combinations
sel.theta <- deconstruct_genotypes(Gi, theta)
```

```
## [[1]]
## [1] "1" "2" "3" "4" "5" "6" "7" "8"
##
## [[2]]
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
## [15] "15" "16"
##
## [[3]]
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
## [15] "15" "16"
##
## [[4]]
## [1] "2"  "4"  "6"  "8"  "10" "12" "14" "16"
##
## [[1]]
```

```
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
## [15] "15" "16"
##
## [[2]]
## [1] "1"  "2"  "3"  "4"  "9"  "10" "11" "12"
##
## [[3]]
## [1] "3"  "4"  "7"  "8"  "11" "12" "15" "16"
##
## [[4]]
## [1] "2"  "4"  "6"  "8"  "10" "12" "14" "16"
##
## [[1]]
## [1] "1" "2" "3" "4" "5" "6" "7" "8"
##
## [[2]]
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
## [15] "15" "16"
##
## [[3]]
## [1] "3"  "4"  "7"  "8"  "11" "12" "15" "16"
##
## [[4]]
## [1] "2"  "4"  "6"  "8"  "10" "12" "14" "16"
##
## [[1]]
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
## [15] "15" "16"
##
## [[2]]
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
## [15] "15" "16"
##
## [[3]]
## [1] "3"  "4"  "7"  "8"  "11" "12" "15" "16"
##
## [[4]]
## [1] "2"  "4"  "6"  "8"  "10" "12" "14" "16"
##
## [[1]]
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
## [15] "15" "16"
##
## [[2]]
## [1] "1"  "2"  "3"  "4"  "9"  "10" "11" "12"
##
## [[3]]
## [1] "3"  "4"  "7"  "8"  "11" "12" "15" "16"
##
## [[4]]
## [1] "2"  "4"  "6"  "8"  "10" "12" "14" "16"
##
## [1] "2" "8"
## [1] "4" "6"
```

```
## [1] "6" "4"
## [1] "8" "2"
## [1] "12" "4"
## [1] "4"  "12"
## [1] "4" "8"
## [1] "8" "4"
## [1] "12" "8"
## [1] "16" "4"
## [1] "4"  "16"
## [1] "8"  "12"
## [1] "12" "4"
## [1] "4"  "12"
## [1] "..."
## [1] "Dups"
## [[1]]
## [1] "6" "4"
##
## [[2]]
## [1] "8" "2"
##
## [1] "final"
## [[1]]
## [1] "2" "8"
##
## [[2]]
## [1] "4" "6"
##
## [1] "..."
## [1] "Dups"
## [[1]]
## [1] "4"  "12"
##
## [1] "final"
## [[1]]
## [1] "12" "4"
##
## [1] "..."
## [1] "Dups"
## [[1]]
## [1] "8" "4"
##
## [1] "final"
## [[1]]
## [1] "4" "8"
##
## [1] "..."
## [1] "Dups"
## [[1]]
## [1] "4"  "16"
##
## [[2]]
## [1] "8"  "12"
##
## [1] "final"
```

```
## [[1]]
## [1] "12" "8"
##
## [[2]]
## [1] "16" "4"
##
## [1] "..."
## [1] "Dups"
## [[1]]
## [1] "4"  "12"
##
## [1] "final"
## [[1]]
## [1] "12" "4"
```

```
## EM pipeline
EM(nrow(Gi), sel.theta, theta, iters = 100)
```

```
##    2 8 4 6 12 16
## g1 1 1 2 2  0  0
## g2 0 0 1 0  1  0
## g3 0 1 1 0  0  0
## g4 0 1 2 0  1  2
## g5 0 0 1 0  1  0
##         e
## 2  0.0625
## 8  0.0625
## 4  0.0625
## 6  0.0625
## 12 0.0625
## 16 0.0625
## [1] "---------------------------------"
## [1] "Iteration :1"
## $g1
## $g1$`2,8`
## [1] 0.5
##
## $g1$`4,6`
## [1] 0.5
##
##
## $g2
## $g2$`12,4`
## [1] 1
##
##
## $g3
## $g3$`4,8`
## [1] 1
##
##
## $g4
## $g4$`12,8`
## [1] 0.5
##
```

```
## $g4$`16,4`
## [1] 0.5
##
##
## $g5
## $g5$`12,4`
## [1] 1
##
##
## [1] "logLL: -9.831"
##       e
## 2  0.05
## 8  0.20
## 4  0.40
## 6  0.05
## 12 0.25
## 16 0.05
## [1] "-----------------------------------"
## [1] "Iteration :2"
## $g1
## $g1$`2,8`
## [1] 0.3333333
##
## $g1$`4,6`
## [1] 0.6666667
##
##
## $g2
## $g2$`12,4`
## [1] 1
##
##
## $g3
## $g3$`4,8`
## [1] 1
##
##
## $g4
## $g4$`12,8`
## [1] 0.7142857
##
## $g4$`16,4`
## [1] 0.2857143
##
##
## $g5
## $g5$`12,4`
## [1] 1
##
##
## [1] "logLL: -9.624"
##             e
## 2  0.03333333
## 8  0.20476190
```

```
## 4  0.39523810
## 6  0.06666667
## 12 0.27142857
## 16 0.02857143
## [1] "----------------------------------"
## [1] "Iteration :3"
## $g1
## $g1$`2,8`
## [1] 0.2057416
##
## $g1$`4,6`
## [1] 0.7942584
##
##
## $g2
## $g2$`12,4`
## [1] 1
##
##
## $g3
## $g3$`4,8`
## [1] 1
##
##
## $g4
## $g4$`12,8`
## [1] 0.8311292
##
## $g4$`16,4`
## [1] 0.1688708
##
##
## $g5
## $g5$`12,4`
## [1] 1
##
##
## [1] "logLL: -9.503"
##             e
## 2  0.02057416
## 8  0.20368708
## 4  0.39631292
## 6  0.07942584
## 12 0.28311292
## 16 0.01688708
## [1] "----------------------------------"
## [1] "Iteration :4"
## $g1
## $g1$`2,8`
## [1] 0.117491
##
## $g1$`4,6`
## [1] 0.882509
##
```

```
##
## $g2
## $g2$`12,4`
## [1] 1
##
##
## $g3
## $g3$`4,8`
## [1] 1
##
##
## $g4
## $g4$`12,8`
## [1] 0.896012
##
## $g4$`16,4`
## [1] 0.103988
##
##
## $g5
## $g5$`12,4`
## [1] 1
##
##
## [1] "logLL: -9.43"
##           e
## 2  0.0117491
## 8  0.2013503
## 4  0.3986497
## 6  0.0882509
## 12 0.2896012
## 16 0.0103988
## [1] "-----------------------------------"
## [1] "Iteration :5"
## $g1
## $g1$`2,8`
## [1] 0.06300619
##
## $g1$`4,6`
## [1] 0.9369938
##
##
## $g2
## $g2$`12,4`
## [1] 1
##
##
## $g3
## $g3$`4,8`
## [1] 1
##
##
## $g4
## $g4$`12,8`
```

```
## [1] 0.9336264
##
## $g4$`16,4`
## [1] 0.0663736
##
##
## $g5
## $g5$`12,4`
## [1] 1
##
##
## [1] "logLL: -9.388"
##              e
## 2  0.006300619
## 8  0.199663259
## 4  0.400336741
## 6  0.093699381
## 12 0.293362640
## 16 0.006637360
## [1] "-----------------------------------"
## [1] "Iteration :6"
## $g1
## $g1$`2,8`
## [1] 0.0324484
##
## $g1$`4,6`
## [1] 0.9675516
##
##
## $g2
## $g2$`12,4`
## [1] 1
##
##
## $g3
## $g3$`4,8`
## [1] 1
##
##
## $g4
## $g4$`12,8`
## [1] 0.956604
##
## $g4$`16,4`
## [1] 0.04339603
##
##
## $g5
## $g5$`12,4`
## [1] 1
##
##
## [1] "logLL: -9.364"
##              e
```

```
## 2  0.003244840
## 8  0.198905237
## 4  0.401094763
## 6  0.096755160
## 12 0.295660397
## 16 0.004339603
## [1] "-----------------------------------"
## [1] "Iteration :7"
## $g1
## $g1$`2,8`
## [1] 0.01635894
##
## $g1$`4,6`
## [1] 0.9836411
##
##
## $g2
## $g2$`12,4`
## [1] 1
##
##
## $g3
## $g3$`4,8`
## [1] 1
##
##
## $g4
## $g4$`12,8`
## [1] 0.9712532
##
## $g4$`16,4`
## [1] 0.02874684
##
##
## $g5
## $g5$`12,4`
## [1] 1
##
##
## [1] "logLL: -9.351"
##              e
## 2  0.001635894
## 8  0.198761210
## 4  0.401238790
## 6  0.098364106
## 12 0.297125316
## 16 0.002874684
## [1] "-----------------------------------"
## [1] "Iteration :8"
## $g1
## $g1$`2,8`
## [1] 0.008171162
##
## $g1$`4,6`
```

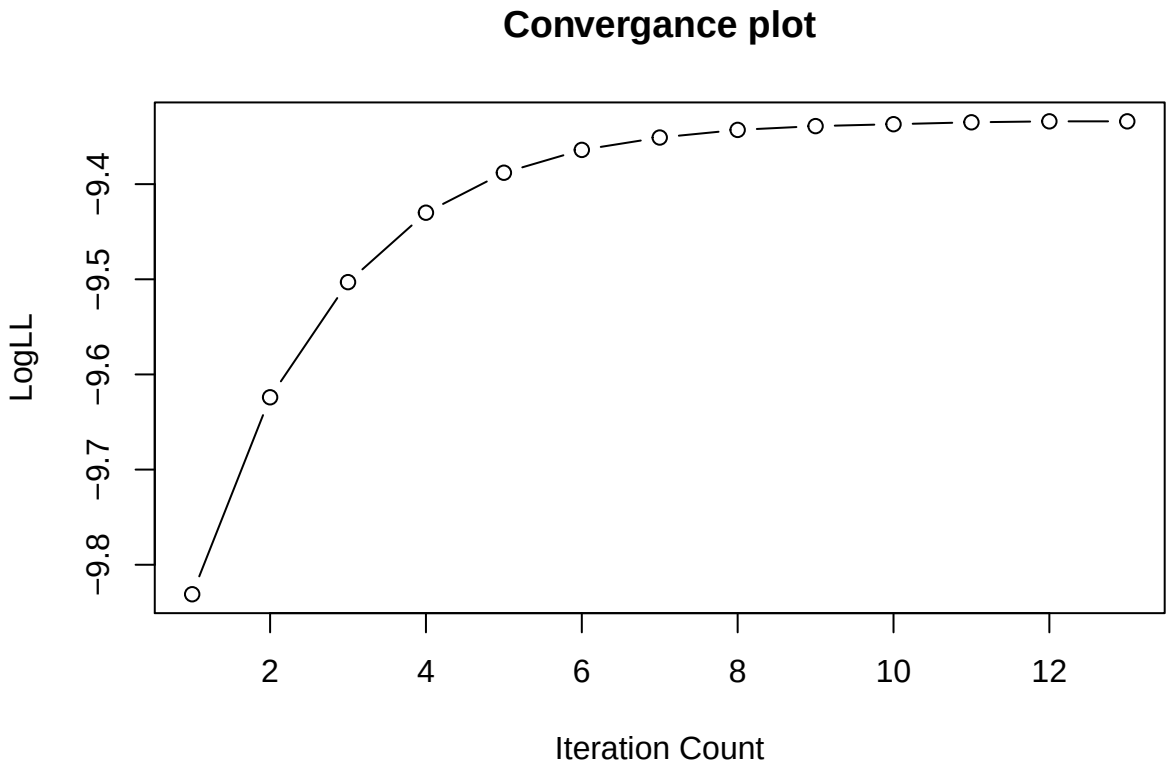
```
## [1] 0.9918288
##
##
## $g2
## $g2$`12,4`
## [1] 1
##
##
## $g3
## $g3$`4,8`
## [1] 1
##
##
## $g4
## $g4$`12,8`
## [1] 0.9808433
##
## $g4$`16,4`
## [1] 0.01915673
##
##
## $g5
## $g5$`12,4`
## [1] 1
##
##
## [1] "logLL: -9.343"
##                e
## 2  0.0008171162
## 8  0.1989014434
## 4  0.4010985566
## 6  0.0991828838
## 12 0.2980843272
## 16 0.0019156728
## [1] "-----------------------------------"
## [1] "Iteration :9"
## $g1
## $g1$`2,8`
## [1] 0.004068771
##
## $g1$`4,6`
## [1] 0.9959312
##
##
## $g2
## $g2$`12,4`
## [1] 1
##
##
## $g3
## $g3$`4,8`
## [1] 1
##
##
```

```
## $g4
## $g4$`12,8`
## [1] 0.9872061
##
## $g4$`16,4`
## [1] 0.01279391
##
##
## $g5
## $g5$`12,4`
## [1] 1
##
##
## [1] "logLL: -9.339"
##               e
## 2  0.0004068771
## 8  0.1991274865
## 4  0.4008725135
## 6  0.0995931229
## 12 0.2987206093
## 16 0.0012793907
## [1] "-----------------------------------"
## [1] "Iteration :10"
## $g1
## $g1$`2,8`
## [1] 0.002025249
##
## $g1$`4,6`
## [1] 0.9979748
##
##
## $g2
## $g2$`12,4`
## [1] 1
##
##
## $g3
## $g3$`4,8`
## [1] 1
##
##
## $g4
## $g4$`12,8`
## [1] 0.9914516
##
## $g4$`16,4`
## [1] 0.008548395
##
##
## $g5
## $g5$`12,4`
## [1] 1
##
##
```

```
## [1] "logLL: -9.337"
##               e
## 2  0.0002025249
## 8  0.1993476854
## 4  0.4006523146
## 6  0.0997974751
## 12 0.2991451605
## 16 0.0008548395
## [1] "-----------------------------------"
## [1] "Iteration :11"
## $g1
## $g1$`2,8`
## [1] 0.001008705
##
## $g1$`4,6`
## [1] 0.9989913
##
##
## $g2
## $g2$`12,4`
## [1] 1
##
##
## $g3
## $g3$`4,8`
## [1] 1
##
##
## $g4
## $g4$`12,8`
## [1] 0.9942895
##
## $g4$`16,4`
## [1] 0.005710471
##
##
## $g5
## $g5$`12,4`
## [1] 1
##
##
## [1] "logLL: -9.335"
##               e
## 2  0.0001008705
## 8  0.1995298234
## 4  0.4004701766
## 6  0.0998991295
## 12 0.2994289529
## 16 0.0005710471
## [1] "-----------------------------------"
## [1] "Iteration :12"
## $g1
## $g1$`2,8`
## [1] 0.0005028305
```

```
##
## $g1$`4,6`
## [1] 0.9994972
##
##
## $g2
## $g2$`12,4`
## [1] 1
##
##
## $g3
## $g3$`4,8`
## [1] 1
##
##
## $g4
## $g4$`12,8`
## [1] 0.9961869
##
## $g4$`16,4`
## [1] 0.003813127
##
##
## $g5
## $g5$`12,4`
## [1] 1
##
##
## [1] "logLL: -9.334"
##                e
## 2  5.028305e-05
## 8  1.996690e-01
## 4  4.003310e-01
## 6  9.994972e-02
## 12 2.996187e-01
## 16 3.813127e-04
## [1] "----------------------------------"
## [1] "Iteration :13"
## $g1
## $g1$`2,8`
## [1] 0.0002508548
##
## $g1$`4,6`
## [1] 0.9997491
##
##
## $g2
## $g2$`12,4`
## [1] 1
##
##
## $g3
## $g3$`4,8`
## [1] 1
```

```
##
##
## $g4
## $g4$`12,8`
## [1] 0.9974548
##
## $g4$`16,4`
## [1] 0.002545156
##
##
## $g5
## $g5$`12,4`
## [1] 1
##
##
## [1] "logLL: -9.334"
##               e
## 2  2.508548e-05
## 8  1.997706e-01
## 4  4.002294e-01
## 6  9.997491e-02
## 12 2.997455e-01
## 16 2.545156e-04
##    2 8 4 6 12 16
## g1 1 1 2 2  0  0
## g2 0 0 1 0  1  0
## g3 0 1 1 0  0  0
## g4 0 1 2 0  1  2
## g5 0 0 1 0  1  0
## [1] "Iteration Ended"
##    2 8 4 6 12 16
## g1 1 1 2 2  0  0
## g2 0 0 1 0  1  0
## g3 0 1 1 0  0  0
## g4 0 1 2 0  1  2
## g5 0 0 1 0  1  0
##                   _
## g1.2,8  0.0002508548
## g1.4,6  0.9997491452
## g2.12,4 1.0000000000
## g3.4,8  1.0000000000
## g4.12,8 0.9974548444
## g4.16,4 0.0025451556
## g5.12,4 1.0000000000
##               e
## 2  2.508548e-05
## 8  1.997706e-01
## 4  4.002294e-01
## 6  9.997491e-02
## 12 2.997455e-01
## 16 2.545156e-04
## [1] "EM algorithm converges at iteration 13 with logLL value: -9.334"
```




```
# iterate through Estep and Mstep Converge at maxima Need to
# add initial parameter randomization and finding whether
# it's global or local maxima
```