Sheet 7

Carsten Stahl

2023-12-05

Exercise 22

implementation of the bicluster consists a couple of functions first one is the computation of the residuals and H_Y for a given matrix:

```
# computes the heterogeneity-index for a given matrix m
biclustering_residuals <- function(m) {
  # apparently a matrix with one row gets converted to a vector
  if (is.matrix(m) == T) {
    n_col <- length(m[1,])
    n_row <- length(m[,1])
  }else {
    return(NA)
  }
  # basic initialization
  overall_mean <- mean(m)
  column_wise_mean <- c()
  row_wise_mean <- c()

  residuals <- matrix(data = rep(NA, n_col*n_row),
                              nrow = n_row,
                              ncol = n_col)
  # computing column-wise and row-wise mean
  for (i in 1:n_col) {
    column_wise_mean <- append(column_wise_mean, mean(m[,i]) - overall_mean)
  }
  for (j in 1:n_row) {
    row_wise_mean <- append(row_wise_mean, mean(m[j,]) - overall_mean)
  }

  # computing SQUARED residuals
  for (i in 1:n_row) {
    for (j in 1:n_col) {
      residuals[i,j] <- (overall_mean + row_wise_mean[i] + column_wise_mean[j] - m[i,j])^2
    }
  }

  # return sum of residuals
  return(residuals)
}

h_y <- function(m) {
```

```
  return(sum(biclustering_residuals(m)))
}
```

Next we want to implement the biclustering-algorithm of method 2 for one cluster:

```
bicluster <- function(m, delta) {
  if (is.matrix(m) == T) {
    n_col <- length(m[1,])
    n_row <- length(m[,1])
  }else {
    return(m)
  }

  # check if contributions need to be checked
  if (length(m) == 0) {
    return(m)
  }
  h_y_m <- h_y(m)
  if (h_y_m <= delta) {
    return(m)
  }

  # else check contributions
  contr_row <- c()
  contr_col <- c()

  bi_res <- biclustering_residuals(m)

  # determine contributions
  for(i in 1:n_col) {
    contr_col <- append(contr_col, mean(bi_res[,i]))
  }
  for(j in 1:n_row) {
    contr_row <- append(contr_row, mean(bi_res[j,]))
  }

  # check which contribution is the biggest
  rm_col <- which.max(contr_col)
  rm_row <- which.max(contr_row)

  bgst_contr_r <- contr_row[rm_row]
  bgst_contr_c <- contr_col[rm_col]

  # remove the biggest contribution and repeat algorithm
  if (bgst_contr_c >= bgst_contr_r) {
    bicluster(m[,-rm_col], delta)
  } else {
    bicluster(m[-rm_row,], delta)
  }
}
```

Okay so now we can test the implementation for the given data:

```r
data <- matrix(c(1.03, 1.99, 2.98, 4.21, 4.93,
                 1.82, 3.09, 3.95, 4.98, 5.99,
                 6.03, 7.21, 7.95, 9.01, 10.09,
                 7.00, 8.08, 8.98, 10.03, 14.96,
                 2.90, 3.06, 2.88, 3.09, 2.98,
                 4.94, 4.08, 3.13, 1.99, 1.07
                 ), nrow = 6)

bicluster(data, 0.1)
```

```
##      [,1] [,2]  [,3]
## [1,] 1.03 3.09  7.95
## [2,] 1.99 3.95  9.01
## [3,] 2.98 4.98 10.09
```