Statistics 243 Final project

Renata Barreto-Montenegro, Aniket Kesari, Aziz Khiyami, AbdulRahman Kreidieh

June 15, 2016

0.1 Main Function

```
mfa <- function(data, sets, ncomps = NULL, center = TRUE, scale = TRUE) {
  if (class(data)!="matrix" && class(data)!="data.frame") {
    stop('data must be of class "matrix" or "data.frame"')
  }

  # remove NA's
  data = na.omit(data)

  indeces = c()
  for (i in 1:length(sets)) {
    indeces = c(indeces, sets[[i]])
  }
  dat = data[,indeces]

  # center and scale if requested
  if (scale) {
    if (center) {
      dat = scale(dat, center = TRUE, scale = FALSE)
      dat = apply(dat, 2 , function(x){x/sqrt(sum(x^2))})
    }
    else {
      dat = scale(dat, center = FALSE, scale = apply(dat, 2, sd, na.rm = TRUE))
    }
  }
  if (center) {
    dat = scale(dat, center = TRUE, scale = FALSE)
  }


  # Step 1: PCA of Each Data Table
  F_partial = list()
  a = c()
  K = length(sets)
  J = c()
  indx = 1
  for (i in 1:length(sets)) {

    # break up data into each assessor
    Xi = dat[,indx:(indx+length(sets[[i]])-1)]
    J = c(J, length(sets[[i]]))

    # compute SVD
    SVD = svd(Xi)
    U = SVD$u
```

```
  D = diag(SVD$d)
  V = SVD$v

  # alfa weights
  alfa_1 = D[1,1]^-2
  a = c(a, rep(alfa_1,length(sets[[i]])))

  # partial factor scores (step 1)
  F_partial[[i]] = K*alfa_1*Xi

  indx = indx + length(sets[[i]])
}


# Step 2: Generalized SVD of X
m = rep(1/dim(dat)[1], dim(dat)[1])

# compute GSVD
GSVD = svd(diag(m^(1/2)) %*% dat %*% diag(a^(1/2)))
Q = t(GSVD$v) %*% diag(a^(-1/2))  # factor loadings

# eigenvalues
eigenvalues = GSVD$d^2

# common factor scores
F_common = dat %*% diag(a) %*% t(Q)

# partial factor scores (step 2)
indx = 1
for (i in 1:length(sets)) {
  F_partial[[i]] = F_partial[[i]] %*% t(Q)[indx:(indx+length(sets[[i]])-1),]
  indx = indx + length(sets[[i]])
}

# in order to extract number of requested components
if (is.null(ncomps)) {
  ncomps = length(eigenvalues)
}

for (i in 1:length(F_partial)) {
  F_partial[[i]] = F_partial[[i]][,1:ncomps]
}

# placing results into a list and setting the class of the list as "mfa"
res <- list(
  alfa_weights = a,  # ask, maybe remove
  Jk = J,  # ask, maybe remove
  eigenvalues = eigenvalues[1:ncomps],
  common_factor_scores = F_common[,1:ncomps],
  partial_factor_scores = F_partial,
  factor_loadings = t(Q)[,1:ncomps]
)
class(res) <- "mfa"
```

```
  return(res)
}
```

0.2 Printing and Plotting

```
# print function
print.mfa <- function(x, ...) {
  print("Hello world")
}

# plot function
plot.mfa <- function(x, factor_text = NULL, load_text = NULL, table_text = NULL, ...) {
  # bar chart for the eigenvalues
  barplot(x$eigenvalues, main = 'Eigenvalues', xlab = 'Components', ylab = "Eigenvalue of Component")

  # scatterplot for common factor scores
  plot(x$common_factor_scores[,1], x$common_factor_scores[,2], main = "Common Factor Scores",
       xlab = "F_common[,1]", ylab = "F_common[,2]", type = "n")

  if (is.null(factor_text)) {
    factor_text = c()
    for (i in 1:dim(x$common_factor_scores)[1]) {
      factor_text = c(factor_text, paste('Sample',i))
    }
  }

  text(x$common_factor_scores[,1], x$common_factor_scores[,2], labels = factor_text)

  # scatterplot for partial factor scores
  plot(x$common_factor_scores[,1], x$common_factor_scores[,2], main = "Partial Factor Scores",
       xlab = "F_partial[,1]", ylab = "F_partial[,2]", type = "n")

  text(x$common_factor_scores[,1], x$common_factor_scores[,2], labels = factor_text)

  for (i in 1:length(x$partial_factor_scores)) {
    points(x$partial_factor_scores[[i]][,1], x$partial_factor_scores[[i]][,2],
           pch = 16, col = "black")
  }

  # scatterplot for loadings
  if (is.null(load_text)) {
    load_text = c()
    for (i in 1:dim(x$factor_loadings)[1]) {
      load_text = c(load_text, paste('Feature',i))
    }
  }

  plot(x$factor_loadings[,1], x$factor_loadings[,2], main = "Factor Loads",
       xlab = "Load[,1]", ylab = "Load[,2]", type = "n")
  text(x$factor_loadings[,1], x$factor_loadings[,2], labels = load_text)
}
```

0.3 Summaries of Eigenvalues

```
summaries_of_eigenvalues <- function(object, ...) UseMethod('summaries_of_eigenvalues')

summaries_of_eigenvalues.mfa <- function(object) {
  # variables of interest to be placed in table
  eigenvalues = object$eigenvalues
  singularvalues = eigenvalues^(1/2)
  cumulative_eigenvalues = cumsum(eigenvalues)
  inertia = eigenvalues/sum(eigenvalues) * 100
  cumulative_interia = cumsum(inertia)
  tbl = as.data.frame(rbind(singularvalues, eigenvalues, cumulative_eigenvalues, inertia, cumulative_in

  # printing the table
  tbl
}
```

0.4 Contributions

0.4.1 Contribution of Observation to a Given Dimension

```
contribution_of_observation <- function(object, ...) UseMethod(contribution_of_observation)

contribution_of_observation.mfa <- function(object, observation_num, dim_num) {
  # the mass of each observation is equal to 1/(number of observers)
  m = 1/length(object$partial_factor_scores)
  f = object$common_factor_scores[observation_num, dim_num]
  lambda = object$eigenvalues[dim_num]

  return(m*f/lambda)
}
```

0.4.2 Contribution of Variable to a Given Dimension

```
contribution_of_variable <- function(object, ...) UseMethod(contribution_of_variable)

contribution_of_variable.mfa <- function(object, variable_num, dim_num) {
  a = object$alfa_weights[variable_num]
  q = object$factor_loadings[variable_num, dim_num]

  return(a*q)
}
```

0.4.3 Contribution of Table to a Given Dimension

```
contribution_of_table <- function(object, ...) UseMethod(contribution_of_table)

contribution_of_table.mfa <- function(object, table_num, dim_num) {
  res = 0
```

```
  for (i in 1:object$Jk[table_num]) {
    res = res + contribution_of_variable(object, i, dim_num)
  }

  return(res)
}
```

0.5 Coefficients

0.5.1 RV Coefficient

```
Rv_coefficient <- function(dataset, sets) {
  return(RV_table(object, sets))
}

RV <- function(table1, table2) {
  X_k_k = (table1 %*% t(table1)) %*% (table1 %*% t(table1))
  X_k_kp = (table1 %*% t(table1)) %*% (table2 %*% t(table2))
  X_kp_kp = (table2 %*% t(table2)) %*% (table2 %*% t(table2))

  res = sum(diag(X_k_kp))/sqrt(sum(diag(X_k_k)) * sum(diag(X_kp_kp)))
  return(res)
}

RV_table <- function(dataset, sets) {
  res = matrix(rep(0,length(sets)^2), nrow = length(sets), ncol = length(sets))

  for (i in 1:length(sets)) {
    for (j in 1:i) {
      res[i,j] = RV(dataset[,sets[[i]]], dataset[,sets[[j]]])
      res[j,i] = res[i,j]
    }
  }

  return(res)
}
```

0.5.2 Lg Coefficient

```
Lg_coefficient <- function(dataset, sets) {
  return(Lg_table(dataset, sets))
}

Lg <- function(table1, table2, alfa) {
  X_k_kp = (table1 %*% t(table1)) %*% (table2 %*% t(table2))

  SVD1 = svd(table1)
  alfa_k = SVD1$d[1]^-2

  SVD2 = svd(table2)
```

```
  alfa_kp = SVD2$d[1]^-2

  res = sum(diag(X_k_kp)) * alfa_k * alfa_kp
  return(res)
}

Lg_table <- function(dataset, sets) {
  res = matrix(rep(0,length(sets)^2), nrow = length(sets), ncol = length(sets))

  for (i in 1:length(sets)) {
    for (j in 1:i) {
      res[i,j] = Lg(dataset[,sets[[i]]], dataset[,sets[[j]]])
      res[j,i] = res[i,j]
    }
  }

  return(res)
}
```