OverGFM : simulated examples

Jinyu Nie, Zhilong Qin, Wei Liu

2023-08-09

Load GFM package

The package can be loaded with the following command:

```
library("GFM")
```

```
## Loading required package: doSNOW

## Loading required package: foreach

## Loading required package: iterators

## Loading required package: snow

## Loading required package: parallel

##
## Attaching package: 'parallel'

## The following objects are masked from 'package:snow':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, clusterSplit, makeCluster, parApply,
##     parCapply, parLapply, parRapply, parSapply, splitIndices,
##     stopCluster

## GFM :  Generalized factor model is implemented for ultra-high dimensional data with mixed-type varial
## Two algorithms, variational EM and alternate maximization, are designed to implement the generalized
## respectively. The factor matrix and loading matrix together with the number of factors can be well es
## This model can be employed in social and behavioral sciences, economy and finance, and  genomics,
## to extract interpretable nonlinear factors. More details can be referred to
## Wei Liu, Huazhen Lin, Shurong Zheng and Jin Liu. (2021) <doi:10.1080/01621459.2021.1999818>.   Check
```

Load rrpack and PCAmixdata packages for other methods

The rrpack package can be loaded with the following command:

```
library("rrpack")
```

The PCAmixdata package can be loaded with the following command:

```
library("PCAmixdata")
```

Introduction to the data generation mechanisms

We define the function with details for the data generation mechanism.

```
gendata_s2 <- function (seed = 1, n = 500, p = 500,
                        type =  c('homonorm', 'heternorm', 'pois', 'bino', 'norm_pois',
                                  'pois_bino', 'npb'),
                        q = 6, rho = c(0.05, 0.2, 0.1), n_bin=1, sigma_eps=0.1){
  library(MASS)
  Diag <- GFM:::Diag
  cor.mat <- GFM:::cor.mat
  type <- match.arg(type)
  rho_gauss <- rho[1]
  rho_pois <- rho[2]
  rho_binary <- rho[3]
  set.seed(seed)
  Z <- matrix(rnorm(p * q), p, q)
  if (type == "homonorm") {
    g1 <- 1:p
    Z <- rho_gauss * Z
  }else if (type == "heternorm"){
    g1 <- 1:p
    Z <- rho_gauss * Z
  }else if(type == "pois"){
    g1 <- 1:p
    Z <- rho_pois * Z
  }else if(type == 'bino'){
    g1 <- 1:p
    Z <- rho_binary * Z
  }else if (type == "norm_pois") {
    g1 <- 1:floor(p/2)
    g2 <- (floor(p/2) + 1):p
    Z[g1, ] <- rho_gauss * Z[g1, ]
    Z[g2, ] <- rho_pois * Z[g2, ]
  }else if (type == "pois_bino") {
    g1 <- 1:floor(p/2)
    g2 <- (floor(p/2) + 1):p

    Z[g1, ] <- rho_pois * Z[g1, ]
    Z[g2, ] <- rho_binary * Z[g2, ]
  }else if(type == 'npb'){
    g1 <- 1:floor(p/3)
    g2 <- (floor(p/3) + 1):floor(p*2/3)
    g3 <- (floor(2*p/3) + 1):p
    Z[g1, ] <- rho_gauss * Z[g1, ]
```

```
  Z[g2, ] <- rho_pois * Z[g2, ]
  Z[g3, ] <- rho_binary * Z[g3, ]
}
svdZ <- svd(Z)
B1 <- svdZ$u %*% Diag(svdZ$d[1:q])
B0 <- B1 %*% Diag(sign(B1[1, ]))
mu0 <- 0.4 * rnorm(p)
Bm0 <- cbind(mu0, B0)

set.seed(seed)
H <- mvrnorm(n, mu = rep(0, q), cor.mat(q, 0.5))
svdH <- svd(cov(H))
H0 <- scale(H, scale = F) %*% svdH$u %*% Diag(1/sqrt(svdH$d)) %*%
  svdH$v
if (type == "homonorm") {
  X <- H0 %*% t(B0) + matrix(mu0, n, p, byrow = T) + mvrnorm(n,
                                                             rep(0, p), sigma_eps*diag(p))
  group <- rep(1, p)
  XList <- list(X)
  types <- c("gaussian")

}else if (type == "heternorm") {
  sigmas = sigma_eps*(0.1 + 4 * runif(p))
  X <- H0 %*% t(B0) + matrix(mu0, n, p, byrow = T) + mvrnorm(n,
                                                             rep(0, p), diag(sigmas))
  group <- rep(1, p)

  XList <- list(X)
  types <- c("gaussian")

}else if (type == "pois") {


  Eta <- H0 %*% t(B0) + matrix(mu0, n, p, byrow = T) + mvrnorm(n,rep(0, p),
                                                               sigma_eps*diag(p))
  mu <- exp(Eta)
  X <- matrix(rpois(n * p, lambda = mu), n, p)
  group <- rep(1, p)
  XList <- list(X[,g1])
  types <- c("poisson")
}else if(type == 'bino'){

  Eta <- cbind(1, H0) %*% t(Bm0[g1, ]) + mvrnorm(n,rep(0, p), sigma_eps*diag(p))
  mu <- 1/(1 + exp(-Eta))
  X <- matrix(rbinom(prod(dim(mu)), n_bin, mu), n, p)
  group <- rep(1, p)

  XList <- list(X[,g1])
  types <- c("binomial")
}else if (type == "norm_pois") {

  Eps <- mvrnorm(n,rep(0, p), sigma_eps*diag(p))
  mu1 <- cbind(1, H0) %*% t(Bm0[g1, ]) + Eps[, g1]
```

```
    mu2 <- exp(cbind(1, H0) %*% t(Bm0[g2, ])+ Eps[, g2])
    X <- cbind(matrix(rnorm(prod(dim(mu1)), mu1, 1), n, floor(p/2)),
               matrix(rpois(prod(dim(mu2)), mu2), n, ncol(mu2)))
    group <- c(rep(1, length(g1)), rep(2, length(g2)))

    XList <- list(X[,g1], X[,g2])
    types <- c("gaussian", "poisson")

  }else if (type == "pois_bino") {

    Eps <- mvrnorm(n,rep(0, p), sigma_eps*diag(p))
    mu1 <- exp(cbind(1, H0) %*% t(Bm0[g1, ])+ Eps[,g1])
    mu2 <- 1/(1 + exp(-cbind(1, H0) %*% t(Bm0[g2, ])- Eps[,g2]))
    X <- cbind(matrix(rpois(prod(dim(mu1)), mu1), n, ncol(mu1)),
               matrix(rbinom(prod(dim(mu2)), n_bin, mu2), n, ncol(mu2)))
    group <- c(rep(1, length(g1)), rep(2, length(g2)))
    XList <- list(X[,g1], X[,g2])
    types <- c("poisson", 'binomial')
  }else if(type == 'npb'){

    Eps <- mvrnorm(n,rep(0, p), sigma_eps*diag(p))
    mu11 <- cbind(1, H0) %*% t(Bm0[g1, ]) + Eps[,g1]
    mu1 <- exp(cbind(1, H0) %*% t(Bm0[g2, ]) +  Eps[,g2])
    mu2 <- 1/(1 + exp(-cbind(1, H0) %*% t(Bm0[g3, ])-  Eps[,g3]))
    X <- cbind(matrix(rnorm(prod(dim(mu11)),mu11, 1), n, ncol(mu11)),
               matrix(rpois(prod(dim(mu1)), mu1), n, ncol(mu1)),
               matrix(rbinom(prod(dim(mu2)), n_bin, mu2), n, ncol(mu2)))
    group <- c(rep(1, length(g1)), rep(2, length(g2)), rep(3, length(g3)))
    XList <- list(X[,g1], X[,g2], X[,g3])
    types <- c("gaussian", "poisson", 'binomial')
  }

  return(list(X=X, XList = XList,  types= types, B0 = B0, H0 = H0, mu0 = mu0))
}
```

Brief description of other methods

GFM method capable of handling mixed-type data is implemented in the R package **GFM**.

MRRR method which is implemented in the R package **rrpack** also handles mixed-type data through reduced-rank regression model, and we redefine the function of this method and modify its output results for comparison conveniently.

```
Diag <- GFM:::Diag
## Compare with MRRR
mrrr_run <- function(Y, rank0,family=list(poisson()),
                     familygroup,  epsilon = 1e-4, sv.tol = 1e-2,lambdaSVD=0.1, maxIter = 2000, trace=T
  # epsilon = 1e-4; sv.tol = 1e-2; maxIter = 30; trace=TRUE

  require(rrpack)

  n <- nrow(Y); p <- ncol(Y)
```

```
  X <- cbind(1, diag(n))


  svdX0d1 <- svd(X)$d[1]
  init1 = list(kappaC0 = svdX0d1 * 5)
  offset = NULL
  control = list(epsilon = epsilon, sv.tol = sv.tol, maxit = maxIter,
                 trace = trace, gammaC0 = 1.1, plot.cv = TRUE,
                 conv.obj = TRUE)
  res_mrrr <- mrrr(Y=Y, X=X[,-1], family = family, familygroup = familygroup,
                   penstr = list(penaltySVD = "rankCon", lambdaSVD = lambdaSVD),
                   control = control, init = init1, maxrank = rank0)

  hmu <- res_mrrr$coef[1,]
  hTheta <- res_mrrr$coef[-1,]
  #print(dim(hTheta))
  # Matrix::rankMatrix(hTheta)
  svd_Theta <- svd(hTheta, nu=rank0,nv =rank0)
  hH <- svd_Theta$u
  hB <- svd_Theta$v %*% Diag(svd_Theta$d[1:rank0])
  #print(dim(svd_Theta$v))
  #print(dim(Diag(svd_Theta$d)))
  return(list(hH=hH, hB=hB, hmu= hmu))
}
```

PCAmix method which uses Principal component analysis for data with mix of qualitative and quantitative variables is implemented in the R package **PCAmixdata**.

LFM method which handles linear factor model is implemented in the R package **GFM**, and we define the function of this method based on R package **GFM**.

```
factorm <- function(X, q=NULL){

  signrevise <- GFM:::signrevise
  if ((!is.null(q)) && (q < 1))
    stop("q must be NULL or other positive integer!")
  if (!is.matrix(X))
    stop("X must be a matrix.")
  mu <- colMeans(X)
  X <- scale(X, scale = FALSE)
  n <- nrow(X)
  p <- ncol(X)
  if (p > n) {
    svdX <- eigen(X %*% t(X))
    evalues <- svdX$values
    eigrt <- evalues[1:(21 - 1)]/evalues[2:21]
    if (is.null(q)) {
      q <- which.max(eigrt)
    }
    hatF <- as.matrix(svdX$vector[, 1:q] * sqrt(n))
    B2 <- n^(-1) * t(X) %*% hatF
    sB <- sign(B2[1, ])
    hB <- B2 * matrix(sB, nrow = p, ncol = q, byrow = TRUE)
    hH <- sapply(1:q, function(k) hatF[, k] * sign(B2[1,
```

```
    ])[k])
  }
  else {
    svdX <- eigen(t(X) %*% X)
    evalues <- svdX$values
    eigrt <- evalues[1:(21 - 1)]/evalues[2:21]
    if (is.null(q)) {
      q <- which.max(eigrt)
    }
    hB1 <- as.matrix(svdX$vector[, 1:q])
    hH1 <- n^(-1) * X %*% hB1
    svdH <- svd(hH1)
    hH2 <- signrevise(svdH$u * sqrt(n), hH1)
    if (q == 1) {
      hB1 <- hB1 %*% svdH$d[1:q] * sqrt(n)
    }
    else {
      hB1 <- hB1 %*% diag(svdH$d[1:q]) * sqrt(n)
    }
    sB <- sign(hB1[1, ])
    hB <- hB1 * matrix(sB, nrow = p, ncol = q, byrow = TRUE)
    hH <- sapply(1:q, function(j) hH2[, j] * sB[j])
  }
  sigma2vec <- colMeans((X - hH %*% t(hB))^2)
  res <- list()
  res$hH <- hH
  res$hB <- hB
  res$mu <- mu
  res$q <- q
  res$sigma2vec <- sigma2vec
  res$propvar <- sum(evalues[1:q])/sum(evalues)
  res$egvalues <- evalues
  attr(res, "class") <- "fac"
  return(res)
}
```

OverGFM can handle overdispersed mixed-type data

First, we generate a simulated data for three mixed-type variables based on the aforementioned data generate function. We fix $(n, p) = (500, 500)$,$\sigma^2 = 0.7$ and the signal strength $(\rho_1, \rho_2, \rho_3) = (0.05, 0.2, 0.1)$. Additionally, we set the number of factor q is 6. The details for the data setting is following :

```
  q <- 6
  datList <- gendata_s2(seed = 1, type= 'npb', n=500, p=500, q=q,
                        rho= c(0.05, 0.2, 0.1) ,sigma_eps = 0.7)
```

Second, we define the trace statistic to assess the performance.

```
trace_statistic_fun <- function(H, H0){

  tr_fun <- function(x) sum(diag(x))
```

```
  mat1 <- t(H0) %*% H %*% ginv(t(H) %*% H) %*% t(H) %*% H0

  tr_fun(mat1) / tr_fun(t(H0) %*% H0)

}
```

Then we use OverGFM to fit model.

```
  gfm_over <- overdispersedGFM(datList$XList, types=datList$types, q=q)
```

```
## Starting the varitional EM algorithm for overdispersed GFM model...

## iter = 2, ELBO= -255470.336139, dELBO=0.999881
## iter = 3, ELBO= -251437.484067, dELBO=0.015786
## iter = 4, ELBO= -250001.740028, dELBO=0.005710
## iter = 5, ELBO= -249597.585171, dELBO=0.001617
## iter = 6, ELBO= -249686.604392, dELBO=0.000357
## iter = 7, ELBO= -250027.944411, dELBO=0.001367
## iter = 8, ELBO= -250494.823372, dELBO=0.001867
## iter = 9, ELBO= -251010.366388, dELBO=0.002058
## iter = 10, ELBO= -251524.940647, dELBO=0.002050
## iter = 11, ELBO= -252012.647290, dELBO=0.001939
## iter = 12, ELBO= -252470.022945, dELBO=0.001815
## iter = 13, ELBO= -252904.416681, dELBO=0.001721
## iter = 14, ELBO= -253321.012616, dELBO=0.001647
## iter = 15, ELBO= -253719.117636, dELBO=0.001572
## iter = 16, ELBO= -254095.125499, dELBO=0.001482
## iter = 17, ELBO= -254445.628575, dELBO=0.001379
## iter = 18, ELBO= -254768.607205, dELBO=0.001269
## iter = 19, ELBO= -255063.426639, dELBO=0.001157
## iter = 20, ELBO= -255330.471964, dELBO=0.001047
## iter = 21, ELBO= -255570.784829, dELBO=0.000941
## iter = 22, ELBO= -255785.792318, dELBO=0.000841
## iter = 23, ELBO= -255977.123606, dELBO=0.000748
## iter = 24, ELBO= -256146.490700, dELBO=0.000662
## iter = 25, ELBO= -256295.612219, dELBO=0.000582
## iter = 26, ELBO= -256426.165451, dELBO=0.000509
## iter = 27, ELBO= -256539.756978, dELBO=0.000443
## iter = 28, ELBO= -256637.905602, dELBO=0.000383
## iter = 29, ELBO= -256722.033490, dELBO=0.000328
## iter = 30, ELBO= -256793.462834, dELBO=0.000278

## Finish the varitional EM algorithm...
```

```
  OverGFM_H <- trace_statistic_fun(gfm_over$hH, datList$H0)
  OverGFM_G <- trace_statistic_fun(cbind(gfm_over$hmu,gfm_over$hB),
                                   cbind(datList$mu0,datList$B0))
```

Other methods poorly handle overdispersed mixed-type data

We use other methods to fit model.

```
  lfm <- factorm(datList$X, q=q)
  gfm_am <- gfm(datList$XList, types=datList$types, q=q, algorithm = "AM",
                maxIter = 15)
```

```
## Starting the alternate maximization algorithm...

## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=1, dB=1, dH=0.7822,dc=1, c=1.2602

## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=2, dB=0.562, dH=0.8449,dc=6e-04, c=1.2594

## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=3, dB=0.1306, dH=0.1765,dc=0.0033, c=1.2635

## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=4, dB=0.091, dH=0.1317,dc=0.0023, c=1.2664

## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=5, dB=0.4994, dH=0.8212,dc=0.0016, c=1.2684

## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=6, dB=0.0593, dH=0.0925,dc=0.0012, c=1.27

## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=7, dB=0.0517, dH=0.0819,dc=9e-04, c=1.2711
```

```
## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=8, dB=0.0461, dH=0.0737,dc=7e-04, c=1.2721

## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=9, dB=0.0415, dH=0.0669,dc=6e-04, c=1.2728

## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=10, dB=0.0376, dH=0.0609,dc=5e-04, c=1.2734

## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=11, dB=0.0342, dH=0.0554,dc=4e-04, c=1.2738

## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=12, dB=0.031, dH=0.0506,dc=3e-04, c=1.2742

## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=13, dB=0.0283, dH=0.0462,dc=3e-04, c=1.2746

## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=14, dB=0.0258, dH=0.0424,dc=2e-04, c=1.2749

## ---------- B updation is finished!---------

## ---------- H updation is finished!---------

## Iter=15, dB=0.0237, dH=0.0392,dc=2e-04, c=1.2751

## Finish the alternate maximization algorithm...
```

```
familygroup <- lapply(1:length(datList$types), function(j) rep(j,
res_mrrr <- mrrr_run(datList$X, rank0=q, family=list(gaussian(), poisson(),
                                                     binomial()),familygroup =
                        unlist(familygroup), maxIter=2000)
```

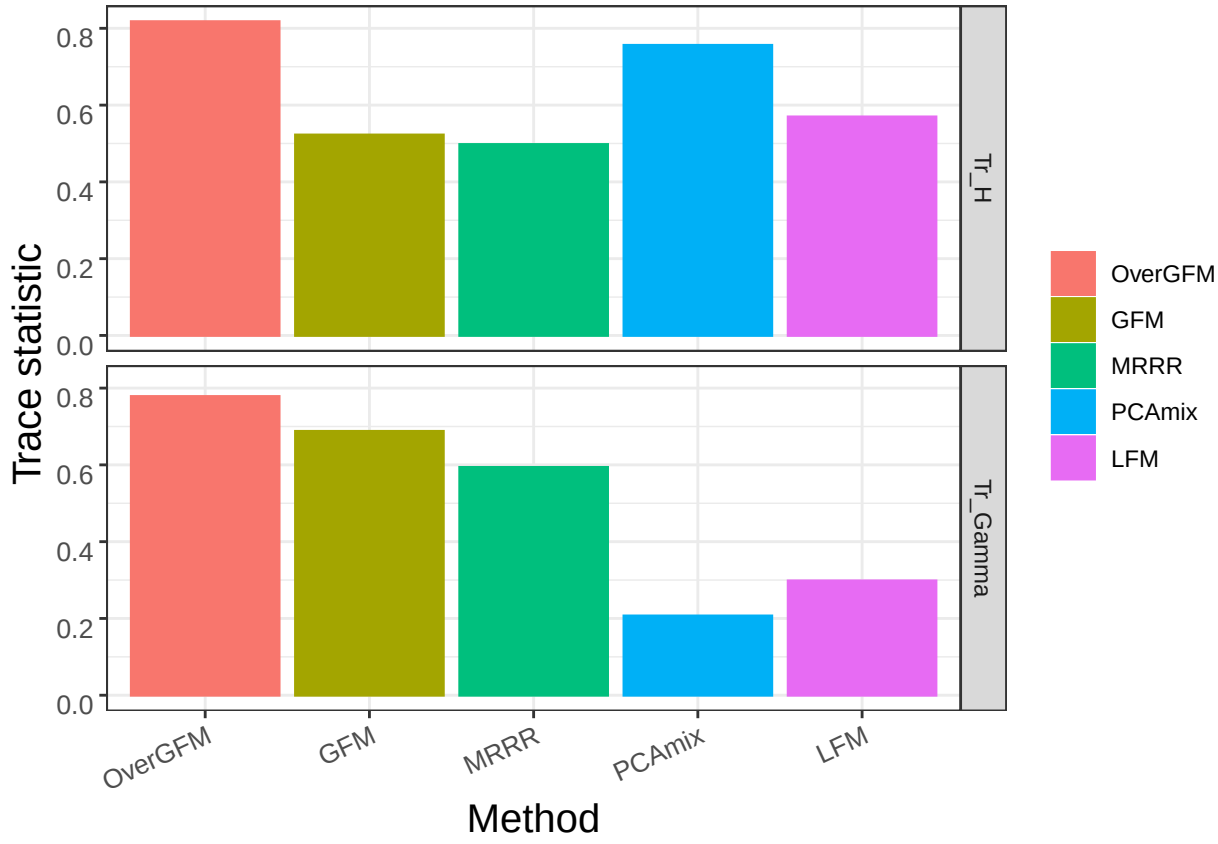
```
## Doing generalized PCA...
## iter = 1 obj/Cnorm_diff = 446069.9
## iter = 2 obj/Cnorm_diff = 429754
## iter = 3 obj/Cnorm_diff = 425045.5
## iter = 4 obj/Cnorm_diff = 419875.6
## iter = 5 obj/Cnorm_diff = 414380
## iter = 6 obj/Cnorm_diff = 408797.6
## iter = 7 obj/Cnorm_diff = 403477.8
## iter = 8 obj/Cnorm_diff = 398830.2
## iter = 9 obj/Cnorm_diff = 395191.2
## iter = 10 obj/Cnorm_diff = 392657.3
## iter = 11 obj/Cnorm_diff = 391042.6
## iter = 12 obj/Cnorm_diff = 390021.5
## iter = 13 obj/Cnorm_diff = 389310.1
## iter = 14 obj/Cnorm_diff = 388745.4
## iter = 15 obj/Cnorm_diff = 388261.5
## iter = 16 obj/Cnorm_diff = 387837.4
## iter = 17 obj/Cnorm_diff = 387460.3
## iter = 18 obj/Cnorm_diff = 387111.6
## iter = 19 obj/Cnorm_diff = 386767.8
## iter = 20 obj/Cnorm_diff = 386405.4
## iter = 21 obj/Cnorm_diff = 386006.8
## iter = 22 obj/Cnorm_diff = 385683.7
## iter = 22 obj/Cnorm_diff = 385683.7
## iter = 23 obj/Cnorm_diff = 385612.4
## iter = 24 obj/Cnorm_diff = 385560.5
## iter = 25 obj/Cnorm_diff = 385521.8
```

```
dat_bino <- as.data.frame(datList$XList[[3]])
for(jj in 1:ncol(dat_bino)) dat_bino[,jj] <- factor(dat_bino[,jj])
dat_norm <- as.data.frame(cbind(datList$XList[[1]],datList$XList[[2]]))
res_pcamix <- PCAmix(X.quanti = dat_norm, X.quali = dat_bino,rename.level=TRUE, ndim=q,
                     graph=F)
reslits <- lapply(res_pcamix$coef, function(x) x[c(seq(2, ncol(dat_norm)+1, by=1),
                                                   seq(ncol(dat_norm)+3,
                                                       nrow(res_pcamix$coef[[1]]), by=2)),])
loadings <- Reduce(cbind, reslits)
GFM_H <- trace_statistic_fun(gfm_am$hH, datList$H0)
GFM_G <- trace_statistic_fun(cbind(gfm_am$hmu,gfm_am$hB),
                                  cbind(datList$mu0,datList$B0))
MRRR_H <- trace_statistic_fun(res_mrrr$hH, datList$H0)
MRRR_G <- trace_statistic_fun(cbind(res_mrrr$hmu,res_mrrr$hB),
                                   cbind(datList$mu0,datList$B0))
PCAmix_H <- trace_statistic_fun(res_pcamix$ind$coord, datList$H0)
PCAmix_G <- trace_statistic_fun(loadings, cbind(datList$mu0,datList$B0))
LFM_H <- trace_statistic_fun(lfm$hH, datList$H0)
LFM_G <- trace_statistic_fun(cbind(lfm$mu,lfm$hB), cbind(datList$mu0,datList$B0))
```

Visualization

We visualize the comparison of the trace statistic of $\mathbf{H}$ and $\boldsymbol{\Upsilon}$ for each methods

```
library(ggplot2)
value <- c(OverGFM_H,OverGFM_G,GFM_H,GFM_G,MRRR_H,MRRR_G,PCAmix_H,PCAmix_G,LFM_H,LFM_G)
df <- data.frame(Value = value,
                 Methods = factor(rep(c("OverGFM","GFM","MRRR","PCAmix","LFM"), each = 2),
                                  levels = c("OverGFM","GFM","MRRR","PCAmix","LFM")),
                 Trace = factor(rep(c("Tr_H","Tr_Gamma"), times = 5),levels = c("Tr_H","Tr_Gamma")))
ggplot(data = df,aes(x = Methods, y = Value, colour = Methods, fill=Methods)) +
  geom_bar(stat="identity") +
  facet_grid(Trace ~ .,drop = TRUE, scales = "free_x") + theme_bw() +
  theme(axis.text.x = element_text(size = 10,angle = 25, hjust = 1, vjust = 1),
        axis.text.y = element_text(size = 10, hjust = 1, vjust = 1),
        axis.title.x = element_text(size = 15),
        axis.title.y = element_text(size = 15),
        legend.title=element_blank())+
  labs( x="Method", y = "Trace statistic ")
```



Session information

```r
sessionInfo()
```

```
## R version 4.3.1 (2023-06-16)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu Mantic Minotaur (development branch)
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.11.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.11.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## time zone: Etc/UTC
## tzcode source: system (glibc)
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] ggplot2_3.4.2    MASS_7.3-60      PCAmixdata_3.1   rrpack_0.1-13
## [5] GFM_1.2.1        doSNOW_1.0.20    snow_0.4-4       iterators_1.0.14
## [9] foreach_1.5.2
##
## loaded via a namespace (and not attached):
##  [1] glmnet_4.1-7      Matrix_1.6-0      gtable_0.3.3      highr_0.10
##  [5] dplyr_1.1.2       compiler_4.3.1    tidyselect_1.2.0  Rcpp_1.0.11
##  [9] splines_4.3.1     scales_1.2.1      yaml_2.3.7        fastmap_1.1.1
## [13] lattice_0.21-8    R6_2.5.1          labeling_0.4.2    generics_0.1.3
## [17] shape_1.4.6       knitr_1.43        tibble_3.2.1      munsell_0.5.0
## [21] pillar_1.9.0      rlang_1.1.1       utf8_1.2.3        xfun_0.39
## [25] cli_3.6.1         withr_2.5.0       magrittr_2.0.3    digest_0.6.33
## [29] grid_4.3.1        rstudioapi_0.15.0 irlba_2.3.5.1     lifecycle_1.0.3
## [33] vctrs_0.6.3       evaluate_0.21     glue_1.6.2        farver_2.1.1
## [37] codetools_0.2-19  survival_3.5-5    fansi_1.0.4       colorspace_2.1-0
## [41] rmarkdown_2.23    tools_4.3.1       pkgconfig_2.0.3   htmltools_0.5.5
```