haoharryfeng / ISLET

Code Issues Pull requests Actions Projects Security Insights

ISLET / R / islet.est.R

haoharryfeng 0.99.6 8b8e137 · 2 years ago History

Code Blame 213 lines (170 loc) · 7.11 KB

```r

###function to implement EM algorithm by block of genes in ISLET algorithm
#function here for unix and windows, using lapply, no parallel computing
#Y is a GxN matrix for gene expression
islet.solve.block<-function(Y, datuse){
    #exp_case = as.matrix(datuse@exp_case)
    #exp_ctrl = as.matrix(datuse@exp_ctrl)
    X <- datuse@X
    A <- datuse@A
    K <- datuse@K
    NU <- datuse@NU
    NS <- datuse@NS
#    para<-datuse@para

    #initialization of parameters parameter estimation storage
    B_est<-NULL
    Sig0_est <- NULL
    SigU_est <- NULL
    E_U_est <- NULL
    llk <- NULL
    ##
```

$$X = \begin{pmatrix} \theta_{111} & \theta_{112} & \dots & \theta_{11K} & z_1\theta_{111} & z_1\theta_{112} & \dots & z_1\theta_{11K} \\ \theta_{121} & \theta_{122} & \dots & \theta_{12K} & z_1\theta_{121} & z_1\theta_{122} & \dots & z_1\theta_{12K} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \theta_{1T_11} & \theta_{1T_12} & \dots & \theta_{1T_1K} & z_1\theta_{1T_11} & z_1\theta_{1T_12} & \dots & z_1\theta_{1T_1K} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \theta_{J11} & \theta_{J12} & \dots & \theta_{J1K} & z_J\theta_{J11} & z_J\theta_{J12} & \dots & z_J\theta_{J1K} \\ \theta_{J21} & \theta_{J22} & \dots & \theta_{J2K} & z_J\theta_{J21} & z_J\theta_{J22} & \dots & z_J\theta_{J2K} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \theta_{JT_J1} & \theta_{JT_J2} & \dots & \theta_{JT_JK} & z_J\theta_{JT_J1} & z_J\theta_{JT_J2} & \dots & z_J\theta_{JT_JK} \end{pmatrix}_{N\times 2K}$$

$$\Sigma_u = \begin{bmatrix} \sigma_1^2 & & & & & & \\ & \sigma_1^2 & & & & & \\ & & \ddots & & & & \\ & & & \sigma_1^2 & & & \\ & & & & \sigma_2^2 & & \\ & & & & & \ddots & \\ & & & & & & \sigma_K^2 \end{bmatrix} \in \mathbb{R}^+ \quad JKt\{$$

$\dim = (NU\times K)\times(NU\times K)$

β

σ_0

Σ_u

300×300

$\dim(A) = NS \times (NU\times K)$

$\dim(u) = (NU\times K)\times 1$

mization (EM) algorithm, although other viable approaches exist. Here, to facilitate the setup of the EM algorithm, we first define the "observed" and the "missing" data: $\boldsymbol{w} = (\boldsymbol{y},\boldsymbol{u}) := (\boldsymbol{w}_{obs},\boldsymbol{w}_{mis})$, where $\boldsymbol{w}_{obs} := \boldsymbol{y}$ is the observed data of admixed gene expression, and $\boldsymbol{w}_{mis} := \boldsymbol{u}$ is the missing data of individual-level deviance from the group mean. Then, we have the conditional distribution $\boldsymbol{w}_{obs}|\boldsymbol{w}_{mis} = \boldsymbol{y}|\boldsymbol{u} \sim N(\boldsymbol{X\beta}+\boldsymbol{Au},\sigma_0^2\boldsymbol{I})$ and marginal distribution $\boldsymbol{w}_{mis} = \boldsymbol{u} \sim N(\boldsymbol{0},\boldsymbol{\Sigma}_u)$. Here, $\boldsymbol{\Sigma}_u$ is a block-diagonal matrix $\boldsymbol{\Sigma}_u = diag(\sigma_1^2\boldsymbol{I}_J,\sigma_2^2\boldsymbol{I}_J,\cdots,\sigma_K^2\boldsymbol{I}_J)$. By calculating the variance-covariance matrix of $\boldsymbol{w}_{mis}$ and $\boldsymbol{w}_{obs}$, we have the following multivariate normal distribution:

$$\begin{pmatrix}\boldsymbol{w}_{obs}\\ \boldsymbol{w}_{mis}\end{pmatrix} = N\left[\begin{pmatrix}\boldsymbol{X\beta}\\ \boldsymbol{0}\end{pmatrix},\begin{pmatrix}\boldsymbol{A\Sigma}_u\boldsymbol{A}'+\sigma_0^2\boldsymbol{I} & \boldsymbol{A\Sigma}_u\\ \boldsymbol{\Sigma}_u'\boldsymbol{A}' & \boldsymbol{\Sigma}_u\end{pmatrix}\right] \tag{6}$$

The EM algorithm calculation will then follow naturally.

E-step:

$$E[\boldsymbol{u}|\boldsymbol{w}_{obs}=\boldsymbol{y}] = \boldsymbol{\Sigma}_u\boldsymbol{A}'\boldsymbol{V}^{-1}\underbrace{(\boldsymbol{y}-\boldsymbol{X\beta})}_{}\quad \text{sig}$$

$$E\left[\boldsymbol{s}'\boldsymbol{s}|\boldsymbol{w}_{obs}=\boldsymbol{y}\right] = tr(\boldsymbol{A\Sigma}_p\boldsymbol{A}') + (\boldsymbol{A\mu}_p+\boldsymbol{X\beta}-\boldsymbol{y})'(\boldsymbol{A\mu}_p+\boldsymbol{X\beta}-\boldsymbol{y})$$

$$E\left[\boldsymbol{u}_k'\boldsymbol{u}_k|\boldsymbol{w}_{obs}=\boldsymbol{y}\right] = tr(\boldsymbol{\Sigma}_{p_k}) + \boldsymbol{\mu}_{p_k}'\boldsymbol{\mu}_{p_k}$$

Here, $\boldsymbol{s} = \boldsymbol{Au}+\boldsymbol{X\beta}-\boldsymbol{y}$, $\boldsymbol{V} := \boldsymbol{A\Sigma}_u\boldsymbol{A}'+\sigma_0^2\boldsymbol{I}$, $\boldsymbol{\Sigma}_{p_k}$ is the kth diagonal block of matrix $\boldsymbol{\Sigma}_p$, and $\boldsymbol{\mu}_{p_k}$ is the kth sub-vector in $\boldsymbol{\mu}_p$.

```r
    Y<-t(Y)
    G<-ncol(Y)
    #  Y=log2(Y+1)

    ####1. Initialization of parameters
    #1.1 cell type profiles AND csDE B parameters
    #B_0 = solve(X,Y)
    B_0 <- Matrix::tcrossprod( Matrix::tcrossprod(solve( Matrix::crossprod(X)), X), t(Y))

    #1.2 error terms
    # sig <- mean((Y-X%*%B_0)^2)
    sig <- colMeans((Y-X%*%B_0)^2)
    #sig <- 20

    #1.3 missing values
    U_0 <- rep(0, NU*K)

    B_t <- B_0
    #sig_t = rep(sig, 7)
    U_t <- U_0
    #sig0_t <- rep(sig, G)
    #sigK_t <- rep(sig, K)
    sig0_t <- sig #rep(sig, G)
    sigK_t <- matrix(rep(sig, each=K), nrow=K)


    iem <- 1
    diff1 <- 100
    diff2 <- 100
    pp <- 1


    #Sig_U = diag(rep(sigK_t, each = NU))
    Sig_p<-lapply(seq_len(G), function(x, A, sig0_t, sigK_t, NU, Y, X, B_t){
        invSig_U<-Matrix::bdiag(diag(rep(1/sigK_t[, x], each=NU)))
```

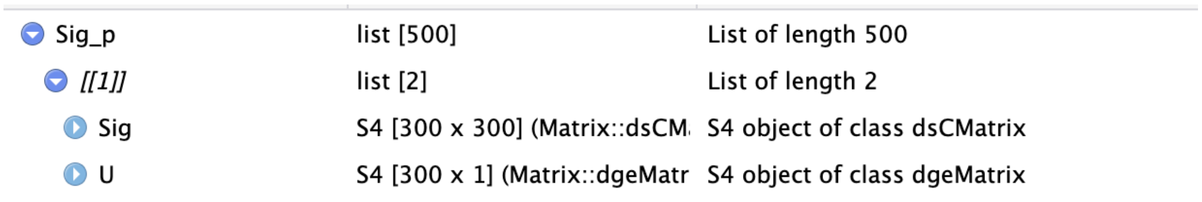
after this step, dim(Y) = N×G

NU : # of subject

50×6 = 300

$$\underset{N\times G}{Y} = \overset{N\times 2K\ (4K\text{ for slope})}{X}\underset{2K\times G}{\beta} + \overset{N\times Q\ (Q=JK)}{A}\underset{Q\times G}{u} + \varepsilon$$

$$u = \begin{array}{cc} & \text{gene1} \quad \cdots \quad \text{gene } G \\ \text{celltype1 for subject } 1\sim 50 & \left[\begin{array}{c} u_{11} \\ u_{21} \\ \vdots \\ u_{J1} \\ u_{12} \\ \vdots \\ u_{JK} \end{array}\right] \end{array}$$



```
Sig_p      list [500]                              List of length 500
  [[1]]    list [2]                                List of length 2
    Sig    S4 [300 x 300] (Matrix::dsCM            S4 object of class dsCMatrix
    U      S4 [300 x 1] (Matrix::dgeMatr           S4 object of class dgeMatrix
```

1个list, 有G个element.

$X \in [0,1]^{NS\times 2K}$

for each gene g of subject j: $u \in \mathbb{R}^{K\times 1}$

Σ_u^{-1}

$$\boldsymbol{\Sigma}_u = diag(\sigma_1^2\boldsymbol{I}_J, \sigma_2^2\boldsymbol{I}_J, \cdots, \sigma_K^2\boldsymbol{I}_J).$$

$\Sigma = \left(\frac{A^TA}{\sigma_0^2} + \Sigma_u^{-1}\right)^{-1}$ $A^TA/\sigma_0^2 + \Sigma_u^{-1}$

crossprod(x,y) = X^Ty

tcrossprod(x,y) = xy^T

NxJK

$A \in (0,1)$

$u = \Sigma_u A^T(A\Sigma_u A^T + \sigma_0^2 I_N)^{-1}(y - X\beta)$

$= (\sigma_0^2)^{-1}\Sigma_u A^T\left(\frac{A\Sigma_u A^T}{\sigma_0^2} + I_N\right)^{-1}(y - X\beta)$

1-50: CT1 for subject 1-50

```r
        Sig<-solve( Matrix::crossprod(A)/sig0_t[x]+invSig_U)
        U<- Matrix::tcrossprod( Matrix::tcrossprod(Sig, A),
                                   BiocGenerics::t(Y[, x] - Matrix::tcrossprod(X,
                                  BiocGenerics::t(B_t[, x])))
                                  )/sig0_t[x]
        return(list(Sig=Sig, U=U))
    }, A, sig0_t, sigK_t, NU, Y, X, B_t)
    E_Up<-do.call(cbind, lapply(Sig_p, function(x)x$U))


    while(iem<15){
#        cat("iteration=", iem, "\n")
        iem <- iem + 1
        ####2. E-step
        #observed data COV(Y) = V

        #V = A%*%Sig_U%*%t(A) + diag(rep(sig0_t, 5*600))

        #2.1 E[U|Y]: missing data [U|Y] given observed data
        #invV = solve(V)
        # E_U = mu_p = t(Sig_U) %*% t(A) %*% invV %*% (Y - X %*% B_t)
        # Sig_p = Sig_U - crossprod(Sig_U,t(A)) %*% invV %*% A %*% Sig_U

        # Estimate from last iteration

        E_U <- E_Up
        mu_p <- E_Up
        E_U_frame <- as.data.frame(as.matrix(E_U))
        #2.2 E[t(S)S|Y]
        E_StS <- lapply(seq_len(G), function(x, A, Sig_p, mu_p, X, B_t, Y){
           sum( Matrix::diag(Matrix::tcrossprod( Matrix::tcrossprod(A, Sig_p[[x]]$Sig), A))) +
                sum(( Matrix::tcrossprod(A, BiocGenerics::t(mu_p[, x])) +
                          Matrix::tcrossprod(X, BiocGenerics::t(B_t[, x])) - Y[, x])^2)},
           A, Sig_p, mu_p, X, B_t, Y)
        E_StS <-unlist(E_StS)
```

JK G

```
E_Up        S4 [300 x 500] (Matrix::dgeM…  S4 object of class dgeMatrix
  Dim       integer [2]                    300 500
  Dimnames  list [2]                       List of length 2
  x         double [150000]                -97.90 -303.88 -79.14 243.73 -2.02 -79.58 ...
  factors   list [0]                       List of length 0
```

E-step:

$$E[\boldsymbol{u}|\boldsymbol{w}_{obs} = \boldsymbol{y}] = \boldsymbol{\Sigma}_u \boldsymbol{A}'\boldsymbol{V}^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$$

$$E\left[\boldsymbol{s}'\boldsymbol{s}|\boldsymbol{w}_{obs} = \boldsymbol{y}\right] = tr(\boldsymbol{A}\boldsymbol{\Sigma}_p\boldsymbol{A}') + (\boldsymbol{A}\boldsymbol{\mu}_p + \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y})'(\boldsymbol{A}\boldsymbol{\mu}_p + \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y})$$

$$E\left[\boldsymbol{u}_k'\boldsymbol{u}_k|\boldsymbol{w}_{obs} = \boldsymbol{y}\right] = tr(\boldsymbol{\Sigma}_{p_k}) + \boldsymbol{\mu}_{p_k}'\boldsymbol{\mu}_{p_k}$$

Here, $\boldsymbol{s} = \boldsymbol{A}\boldsymbol{u} + \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}$, $\boldsymbol{V} := \boldsymbol{A}\boldsymbol{\Sigma}_u\boldsymbol{A}' + \sigma_0^2\boldsymbol{I}$, $\boldsymbol{\Sigma}_{p_k}$ is the kth diagonal block of matrix $\boldsymbol{\Sigma}_p$, and $\boldsymbol{\mu}_{p_k}$ is the kth sub-vector in $\boldsymbol{\mu}_p$.

```r

        #2.3 E[U_k^T U_k|Y]
        mutra_split <- lapply(Sig_p, function(x){
            sig_p<-split(diag(x$Sig), ceiling(seq_len(NU*K)/NU))
            tra<-unlist(lapply(sig_p, sum))
            return(tra)
        })
        mu_split <- split(E_U_frame, ceiling(seq_along(E_U_frame[, 1])/NU))

        E_UkTUk <- do.call('cbind', mutra_split) + do.call('rbind', lapply(mu_split, colss))

        ####3. M-step
        #3.1 B
```

$(X^TX)^{-1}$

```r
        B_tp <-  Matrix::tcrossprod( Matrix::tcrossprod(solve( Matrix::crossprod(X)), X),
                                     BiocGenerics::t(Y- Matrix::tcrossprod(A, BiocGenerics::t(E_U))) )

        #make correction in case B[1:K]<0 or B_tp[(K+1):2K]<0
        #important to bound the estimation to positive values

        # B_tp[1:K,]=ifelse(B_tp[1:K,]<0,0,B_tp[1:K,])
        # B_tp[-(1:K),]=ifelse(B_tp[1:K,]+B_tp[-(1:K),]<0,-B_tp[1:K,],B_tp[-(1:K),])



        #3.2 sigma_0^2
        sig0_tp <- E_StS/(NS)

        #3.3 sigma_k^2
        sigK_tp <- E_UkTUk/(NU)

        ####4. Stopping criteria
        diff1 <- sum(abs(B_tp - B_t)) + abs(sig0_tp - sig0_t) + sum(abs(sigK_tp - sigK_t))

        n1 <- sum(abs(B_tp - B_t))/length(B_tp)
        n2 <- sum(abs(B_tp))/length(B_tp)
        pp <- n1/n2
```

M-step:

For the $(t+1)^{th}$ iteration given the t^{th} iteration:

$$\hat{\boldsymbol{\beta}}^{(t+1)} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'(\boldsymbol{y} - AE_{\eta^{(t)}}(\boldsymbol{u}^{(t)}))$$

```r
#        cat("B_sum_val=", n2, "\n")
#        cat("B_change_val=", n1, "\n")
#        cat("B_change_prop=", pp*100,"% \n")

        ####5. Update params
        B_t<-B_tp
        sig0_t <- sig0_tp
        sigK_t <- sigK_tp

        Sig_p<-lapply(seq_len(G), function(x, A, sig0_t, sigK_t, NU, Y, X, B_t){
            Sig_U<- Matrix::bdiag(diag(rep(sigK_t[, x], each=NU)))
            invSig_U<-Matrix::bdiag(diag(rep(1/sigK_t[, x], each=NU)))
            Sig<-solve( Matrix::crossprod(A)/sig0_t[x]+invSig_U)
            U<- Matrix::tcrossprod(Matrix::tcrossprod(Sig, A),
                                   t(as.matrix(Y[, x] -
                                    Matrix::tcrossprod(X, BiocGenerics::t(B_t[, x]))))
                                   )/sig0_t[x]
            return(list(Sig_U=Sig_U, Sig=Sig, U=U))
        }, A, sig0_t, sigK_t, NU, Y, X, B_t)
        Sig_p_all<-NULL
        E_Up<-do.call(cbind, lapply(Sig_p, function(x)x$U))

        diff2 <- sum(abs(E_Up - E_U))/(length(E_U)*mean(colMeans(Y))^2)
#        cat("Random effect diff2=", diff2, "\n")

    }
    # Estimate of fixed effect
    B_est<-cbind(B_est, B_t)
    # Estimate of random effect
    E_U_est<-cbind(E_U_est, E_Up)
    # Estimate of variance Sigma_U,  Sigma_0
    Sig0_est <- cbind(Sig0_est, sig0_t)
    SigU_est <- cbind(SigU_est, sigK_t)

    #calculate LLK
```

(log-likelihood)

$$\hat{\sigma}_0^{2(t+1)} = \frac{E_{\eta^{(t)}}\left[\boldsymbol{s}'\boldsymbol{s}|\boldsymbol{w}_{obs} = \boldsymbol{y}\right]}{N}$$

$$\hat{\sigma}_k^{2(t+1)} = \frac{E_{\eta^{(t)}}\left[\boldsymbol{u}_k'\boldsymbol{u}_k|\boldsymbol{w}_{obs} = \boldsymbol{y}\right]}{J}$$

The E-step and M-step above are repeated until convergence. The details of modeling and algorithm is available in Additional file 5.

```r
    llk<-lapply(seq_len(G), function(x){
        Sig<- Matrix::tcrossprod( Matrix::tcrossprod(A, Sig_p[[x]]$Sig_U),  A)+
            Matrix::bdiag(diag(sig0_t[x], nrow = nrow(A)))
        l<- Matrix::determinant(Sig)$modulus+
            Matrix::tcrossprod(Matrix::crossprod(Y[, x]-
                      Matrix::tcrossprod(X, BiocGenerics::t(B_t[, x])), solve(Sig)),
                      BiocGenerics::t(Y[, x]-
                           Matrix::tcrossprod(X, BiocGenerics::t(B_t[, x]))) )
        return(-as.numeric(l))
    })

    llk<-unlist(llk)

```

$m_k + \alpha_k$ β_k : fixed group effect

```r
    #compile return list
    case.m <- B_est[seq_len(K), ]+B_est[K+seq_len(K), ]
    ctrl.m <- B_est[seq_len(K), ]

    #(2) the individual value for case and control, for all cell types. 2 matrices of NU by K.
    rel <- split(as.data.frame(as.matrix(E_U_est)), ceiling(seq_along(E_U_est[, 1])/NU))

```

u

```r
    case.indv <- lapply(seq_len(K), function(k){rel[[k]][seq_len(datuse@case_num), ] +
            matrix(rep(case.m[k, ], each=datuse@case_num), nrow=datuse@case_num)})
    ctrl.indv <- lapply(seq_len(K), function(k){rel[[k]][-seq_len(datuse@case_num), ] +
            matrix(rep(ctrl.m[k, ], each=datuse@ctrl_num), nrow=datuse@ctrl_num)})
    names(case.indv) <- names(rel)
    names(ctrl.indv) <- names(rel)

    #(3) Variance for K cell types. 1 vector of length K.
    #'SigU_est' is already to be rendered.

    #(4) Variance for grand residuals. 1 scalar.
    #'Sig0_est' is already to be rendered.

```

β

case_num:
# of case subject

```r
        #(5) the model likelihood. 1 scalar.
        #'llk' is already to be rendered.

        #compile return list
        rval <- list(
            case.m=case.m,
            ctrl.m=ctrl.m,
            case.indv=case.indv,
            ctrl.indv=ctrl.indv,
            var.k=SigU_est,
            var.0=Sig0_est,
            LLK=llk)
        return(rval)

        message("Complete: parameter estimation from ISLET is complete.")
    }
```

haoharryfeng / **ISLET**

Type / to search

Code Issues Pull requests Actions Projects Security Insights

ISLET / R / islet.solve.R

haoharryfeng 0.99.6 8b8e137 · 2 years ago History

Code Blame 70 lines (58 loc) · 2.38 KB Raw

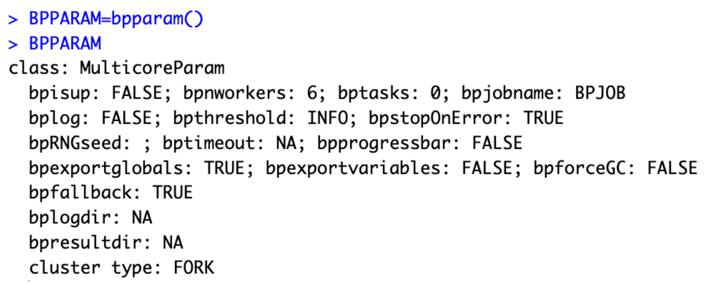
```r

###function to run ISLET, using parallel computing

isletSolve<-function(input, BPPARAM=bpparam() ){
    # islet.solve only runs on the model without age effect.
    if(input@type == 'slope'){
        stop('Input should be prepared by dataPrep()')
    }


    #make Yall a list
    G <- nrow(input@exp_case)
    Yall<-as.matrix(cbind(input@exp_case, input@exp_ctrl))
    aval.nworkers<-BPPARAM$workers
    block.size<-max(ceiling(G/aval.nworkers), 5)
    Yall.list <- split(as.data.frame(Yall), ceiling(seq_len(G)/block.size))

#    if(.Platform$OS.type == "unix") {
    ## do some parallel computation under Unix
#        multicoreParam <- MulticoreParam(workers = ncores)
     res <- bplapply(X=Yall.list, islet.solve.block, datuse=input, BPPARAM=BPPARAM)
```

$G \times NS$

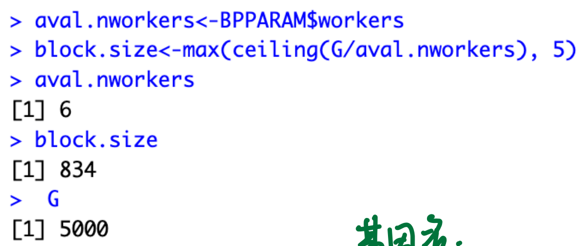
NS: # of sample

NU: # of subject

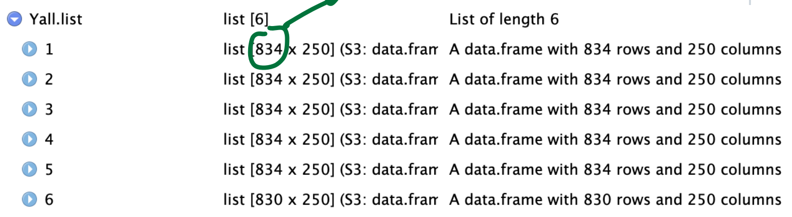
见page 1-7



```
> BPPARAM=bpparam()
> BPPARAM
class: MulticoreParam
  bpisup: FALSE; bpnworkers: 6; bptasks: 0; bpjobname: BPJOB
  bplog: FALSE; bpthreshold: INFO; bpstopOnError: TRUE
  bpRNGseed: ; bptimeout: NA; bpprogressbar: FALSE
  bpexportglobals: TRUE; bpexportvariables: FALSE; bpforceGC: FALSE
  bpfallback: TRUE
  bplogdir: NA
  bpresultdir: NA
  cluster type: FORK
```



```
> aval.nworkers<-BPPARAM$workers
> block.size<-max(ceiling(G/aval.nworkers), 5)
> aval.nworkers
[1] 6
> block.size
[1] 834
>  G
[1] 5000
```



Yall.list	list [6]	List of length 6
1	list [834 x 250] (S3: data.fram	A data.frame with 834 rows and 250 columns
2	list [834 x 250] (S3: data.fram	A data.frame with 834 rows and 250 columns
3	list [834 x 250] (S3: data.fram	A data.frame with 834 rows and 250 columns
4	list [834 x 250] (S3: data.fram	A data.frame with 834 rows and 250 columns
5	list [834 x 250] (S3: data.fram	A data.frame with 834 rows and 250 columns
6	list [830 x 250] (S3: data.fram	A data.frame with 830 rows and 250 columns

基因名.

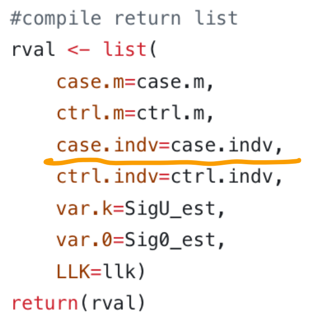
1.2.3.4.5.6 的行名均不同

```
#  }
#  else {
     ## This will be windows
     ## Use serial param or do not use any parallel functions, just use ‘lapply’
     ## result should be of the same “type” from both the if and else statements.
#     nworkers<-length(Yall.list)
#     cl <- makeCluster(nworkers)

     ## Remove clusterExport(), clusterEvalQ() if use devtools::install() to build package
#     clusterExport(cl,list('colss'))
#     clusterEvalQ(cl,{
#         library(Matrix)
#         library(BiocGenerics)})

#     res <-parLapply(cl, X=Yall.list, islet.solve.block, datuse=input)
#     stopCluster(cl)
#
#  }

  # Organize estimated individual reference
  K<-input@K
```

c(1,2,···,50)

```
  SubjectID<-unique(input@SubjectID)
  case_num<-input@case_num
  case.indv.merge<-lapply(seq_len(K), function(k){
      case.indv.all<-lapply(res, '[[', 3)
```

提取res的第3个元素，相当于[[3]]



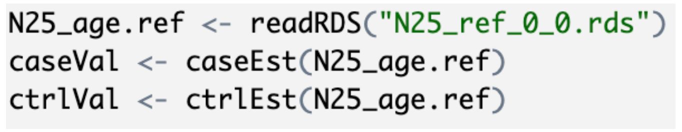
```
#compile return list
rval <- list(
    case.m=case.m,
    ctrl.m=ctrl.m,
    case.indv=case.indv,
    ctrl.indv=ctrl.indv,
    var.k=SigU_est,
    var.0=Sig0_est,
    LLK=llk)
return(rval)
```

```
      case.indv.ctk<-t(do.call(cbind, lapply(case.indv.all, '[[', k)))
      case.indv.ctk<-ifelse(as.matrix(case.indv.ctk)<0, 0, case.indv.ctk)
      dimnames(case.indv.ctk)<-list(rownames(input@exp_case), SubjectID[seq_len(case_num)])
      return(case.indv.ctk)
```

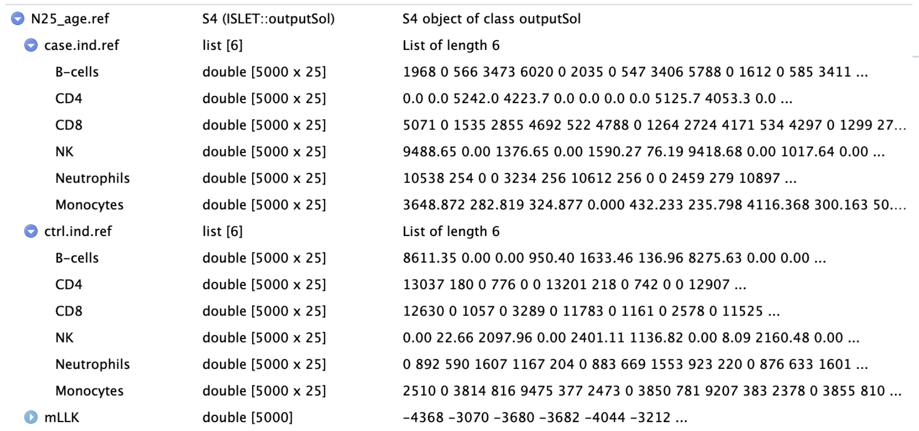
gene_names

```
  })
  ctrl.indv.merge<-lapply(seq_len(K), function(k){
      ctrl.indv.all<-lapply(res, '[[', 4)
      ctrl.indv.ctk<-t(do.call(cbind, lapply(ctrl.indv.all, '[[', k)))
      ctrl.indv.ctk<-ifelse(as.matrix(ctrl.indv.ctk)<0, 0, ctrl.indv.ctk)
      dimnames(ctrl.indv.ctk)<-list(rownames(input@exp_ctrl), SubjectID[-seq_len(case_num)])
```

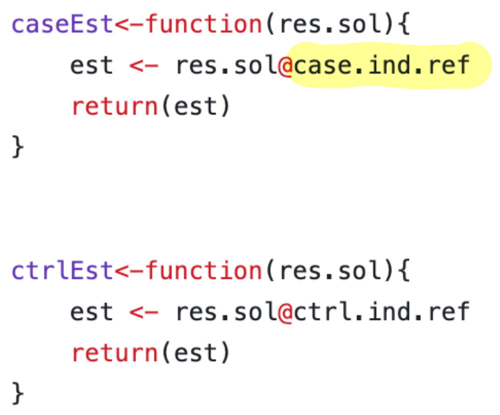
```r
            return(ctrl.indv.ctk)
        })

        names(case.indv.merge)<-input@CT
        names(ctrl.indv.merge)<-input@CT
        llk<-unlist(lapply(res, '[[', 7))

        rval<-outputSol(case.ind.ref=case.indv.merge,
                        ctrl.ind.ref=ctrl.indv.merge,
                        mLLK=llk)

        return(rval)
    }
```



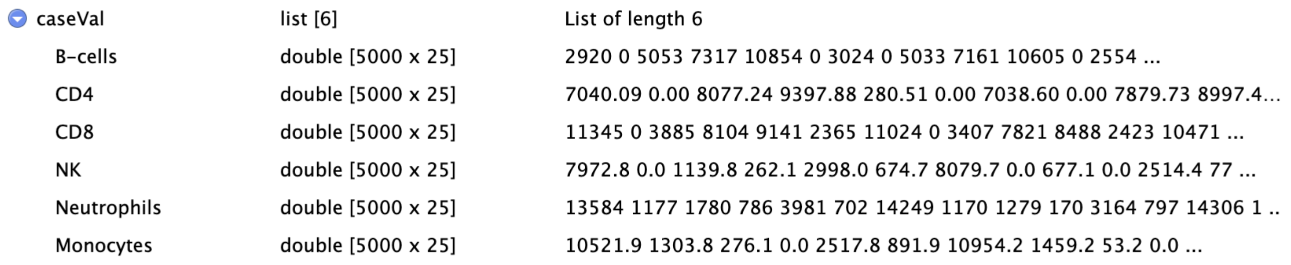
```r
N25_age.ref <- readRDS("N25_ref_0_0.rds")
caseVal <- caseEst(N25_age.ref)
ctrlVal <- ctrlEst(N25_age.ref)
```



```
N25_age.ref        S4 (ISLET::outputSol)   S4 object of class outputSol
  case.ind.ref     list [6]                List of length 6
    B-cells        double [5000 x 25]      1968 0 566 3473 6020 0 2035 0 547 3406 5788 0 1612 0 585 3411 ...
    CD4            double [5000 x 25]      0.0 0.0 5242.0 4223.7 0.0 0.0 0.0 0.0 5125.7 4053.3 0.0 ...
    CD8            double [5000 x 25]      5071 0 1535 2855 4692 522 4788 0 1264 2724 4171 534 4297 0 1299 27...
    NK             double [5000 x 25]      9488.65 0.00 1376.65 0.00 1590.27 76.19 9418.68 0.00 1017.64 0.00 ...
    Neutrophils    double [5000 x 25]      10538 254 0 0 3234 256 10612 256 0 0 2459 279 10897 ...
    Monocytes      double [5000 x 25]      3648.872 282.819 324.877 0.000 432.233 235.798 4116.368 300.163 50....
  ctrl.ind.ref     list [6]                List of length 6
    B-cells        double [5000 x 25]      8611.35 0.00 0.00 950.40 1633.46 136.96 8275.63 0.00 0.00 ...
    CD4            double [5000 x 25]      13037 180 0 776 0 0 13201 218 0 742 0 0 12907 ...
    CD8            double [5000 x 25]      12630 0 1057 0 3289 0 11783 0 1161 0 2578 0 11525 ...
    NK             double [5000 x 25]      0.00 22.66 2097.96 0.00 2401.11 1136.82 0.00 8.09 2160.48 0.00 ...
    Neutrophils    double [5000 x 25]      0 892 590 1607 1167 204 0 883 669 1553 923 220 0 876 633 1601 ...
    Monocytes      double [5000 x 25]      2510 0 3814 816 9475 377 2473 0 3850 781 9207 383 2378 0 3855 810 ...
  mLLK             double [5000]           -4368 -3070 -3680 -3682 -4044 -3212 ...
```



```r
caseEst<-function(res.sol){
    est <- res.sol@case.ind.ref
    return(est)
}


ctrlEst<-function(res.sol){
    est <- res.sol@ctrl.ind.ref
    return(est)
}
```



```
caseVal          list [6]             List of length 6
  B-cells        double [5000 x 25]   2920 0 5053 7317 10854 0 3024 0 5033 7161 10605 0 2554 ...
  CD4            double [5000 x 25]   7040.09 0.00 8077.24 9397.88 280.51 0.00 7038.60 0.00 7879.73 8997.4...
  CD8            double [5000 x 25]   11345 0 3885 8104 9141 2365 11024 0 3407 7821 8488 2423 10471 ...
  NK             double [5000 x 25]   7972.8 0.0 1139.8 262.1 2998.0 674.7 8079.7 0.0 677.1 0.0 2514.4 77 ...
  Neutrophils    double [5000 x 25]   13584 1177 1780 786 3981 702 14249 1170 1279 170 3164 797 14306 1 ..
  Monocytes      double [5000 x 25]   10521.9 1303.8 276.1 0.0 2517.8 891.9 10954.2 1459.2 53.2 0.0 ...
```