

Appendix 2A: code

```
#####  
#  
#design for Diagnostic18.CIBERSORT  
#  
#####  
#' CIBERSORT R script v1.03  
#' Note: Signature matrix construction is not currently available; use java version for full  
functionality.  
#' Author: Aaron M. Newman, Stanford University (amnewman@stanford.edu)  
#' Requirements:  
#'      R v3.0 or later. (dependencies below might not work properly with earlier versions)  
#'      install.packages('e1071')  
#'      install.packages('parallel')  
#'      install.packages('preprocessCore')  
#'      if preprocessCore is not available in the repositories you have selected, run the  
following:  
#'      source("http://bioconductor.org/biocLite.R")  
#'      biocLite("preprocessCore")  
#' Windows users using the R GUI may need to Run as Administrator to install or update  
packages.  
#' This script uses 3 parallel processes. Since Windows does not support forking, this script  
will run  
#' single-threaded in Windows.  
#'  
#' Usage:  
#'      Navigate to directory containing R script  
#'  
#' In R:  
#'      source('CIBERSORT.R')  
#'      results <- CIBERSORT('sig_matrix_file.txt','mixture_file.txt', perm, QN)  
#'  
#'      Options:  
#'      i) perm = No. permutations; set to >=100 to calculate p-values (default = 0)  
#'      ii) QN = Quantile normalization of input mixture (default = TRUE)  
#'  
#' Input: signature matrix and mixture file, formatted as specified at  
http://cibersort.stanford.edu/tutorial.php  
#' Output: matrix object containing all results and tabular data written to disk  
'CIBERSORT-Results.txt'  
#' License: http://cibersort.stanford.edu/CIBERSORT\_License.txt  
#' Core algorithm  
#' @param X cell-specific gene expression
```

```

#' @param y mixed expression per sample
#' @export
CoreAlg <- function(X, y){

  #try different values of nu
  svn_itor <- 3

  res <- function(i){
    if(i==1){nus <- 0.25}
    if(i==2){nus <- 0.5}
    if(i==3){nus <- 0.75}
    model<-svm(X,y,type="nu-regression",kernel="linear",nu=nus,scale=F)
    model
  }

  if(Sys.info()['sysname'] == 'Windows') out <- mclapply(1:svn_itor, res, mc.cores=1) else
    out <- mclapply(1:svn_itor, res, mc.cores=svn_itor)

  nusvm <- rep(0,svn_itor)
  corrv <- rep(0,svn_itor)

  #do cibersort
  t <- 1
  while(t <= svn_itor) {
    weights = t(out[[t]]$coefs) %*% out[[t]]$SV
    weights[which(weights<0)]<-0
    w<-weights/sum(weights)
    u <- sweep(X,MARGIN=2,w, '*')
    k <- apply(u, 1, sum)
    nusvm[t] <- sqrt((mean((k - y)^2)))
    corrv[t] <- cor(k, y)
    t <- t + 1
  }

  #pick best model
  rmsees <- nusvm
  mn <- which.min(rmsees)
  model <- out[[mn]]

  #get and normalize coefficients
  q <- t(model$coefs) %*% model$SV
  q[which(q<0)]<-0
  w <- (q/sum(q))

```

```

mix_rmse <- rmses[mn]
mix_r <- corrv[mn]

newList <- list("w" = w, "mix_rmse" = mix_rmse, "mix_r" = mix_r)

}

#' do permutations
#' @param perm Number of permutations
#' @param X cell-specific gene expression
#' @param y mixed expression per sample
#' @export
doPerm <- function(perm, X, Y){
  itor <- 1
  Ylist <- as.list(data.matrix(Y))
  dist <- matrix()

  while(itor <= perm){
    #print(itor)

    #random mixture
    yr <- as.numeric(Ylist[sample(length(Ylist),dim(X)[1])])

    #standardize mixture
    yr <- (yr - mean(yr)) / sd(yr)

    #run CIBERSORT core algorithm
    result <- CoreAlg(X, yr)

    mix_r <- result$mix_r

    #store correlation
    if(itor == 1) {dist <- mix_r}
    else {dist <- rbind(dist, mix_r)}

    itor <- itor + 1
  }
  newList <- list("dist" = dist)
}

#' Main functions
#' @param sig_matrix file path to gene expression from isolated cells
#' @param mixture_file heterogenous mixed expression
#' @param perm Number of permutations

```

```

#' @param QN Perform quantile normalization or not (TRUE/FALSE)
#' @export
CIBERSORT <- function(sig_matrix, mixture_file, perm=0, QN=TRUE){
  library(e1071)
  library(parallel)
  library(preprocessCore)

  #read in data
  X <- read.table(sig_matrix,header=T,sep="\t",row.names=1,check.names=F)
  Y <- read.table(mixture_file, header=T, sep="\t", row.names=1,check.names=F)

  X <- data.matrix(X)
  Y <- data.matrix(Y)

  #order
  X <- X[order(rownames(X)),]
  Y <- Y[order(rownames(Y)),]

  P <- perm #number of permutations

  #anti-log if max < 50 in mixture file
  if(max(Y) < 50) {Y <- 2^Y}

  #quantile normalization of mixture file
  if(QN == TRUE){
    tmpc <- colnames(Y)
    tmpr <- rownames(Y)
    Y <- normalize.quantiles(Y)
    colnames(Y) <- tmpc
    rownames(Y) <- tmpr
  }

  #intersect genes
  Xgns <- row.names(X)
  Ygns <- row.names(Y)
  YintX <- Ygns %in% Xgns
  Y <- Y[YintX,]
  XintY <- Xgns %in% row.names(Y)
  X <- X[XintY,]

  #standardize sig matrix
  X <- (X - mean(X)) / sd(as.vector(X))

  #empirical null distribution of correlation coefficients

```

```

if(P > 0) {nulldist <- sort(doPerm(P, X, Y)$dist)}

#print(nulldist)

header <- c('Mixture',colnames(X),"P-value","Correlation","RMSE")
#print(header)

output <- matrix()
itor <- 1
mixtures <- dim(Y)[2]
pval <- 9999

#iterate through mixtures
while(itor <= mixtures){

  y <- Y[,itor]

  #standardize mixture
  y <- (y - mean(y)) / sd(y)

  #run SVR core algorithm
  result <- CoreAlg(X, y)

  #get results
  w <- result$w
  mix_r <- result$mix_r
  mix_rmse <- result$mix_rmse

  #calculate p-value
  if(P > 0) {pval <- 1 - (which.min(abs(nulldist - mix_r)) / length(nulldist))}

  #print output
  out <- c(colnames(Y)[itor],w,pval,mix_r,mix_rmse)
  if(itor == 1) {output <- out}
  else {output <- rbind(output, out)}

  itor <- itor + 1

}

#save results
write.table(rbind(header,output), file="CIBERSORT-Results.txt", sep="\t", row.names=F,
col.names=F, quote=F)

```

```

#return matrix object containing all results
obj <- rbind(header,output)
obj <- obj[,-1]
obj <- obj[-1,]
obj <- matrix(as.numeric(unlist(obj)),nrow=nrow(obj))
rownames(obj) <- colnames(Y)
colnames(obj) <- c(colnames(X),"P-value","Correlation","RMSE")
obj
}

```

```

#####
#design for Diagnostic18.run
#
#####
#install.packages('e1071')

```

```

#if (!requireNamespace("BiocManager", quietly = TRUE))
#   install.packages("BiocManager")
#BiocManager::install("preprocessCore")

```

```

inputFile="normalize.txt"          #输入文件
setwd("C:\\Users\\90933\\Desktop\\18.CIBERSORT")      #设置工作目录
source("Diagnostic18.CIBERSORT.R")      #引用包

```

```

#免疫细胞浸润分析
outTab=CIBERSORT("ref.txt", inputFile, perm=1000, QN=TRUE)

```

```

#对免疫浸润结果过滤，并且保存免疫细胞浸润结果
outTab=outTab[outTab[,"P-value"]<0.05,]
outTab=as.matrix(outTab[,1:(ncol(outTab)-3)])
outTab=rbind(id=colnames(outTab),outTab)
write.table(outTab, file="CIBERSORT-Results.txt", sep="\t", quote=F, col.names=F)

```