

```

load("C:/.../MouseDivData.RData")

library(MouseDivGeno)

library(HiddenMarkov)

install.packages("HiddenMarkov")

library(affyio)

install.packages("affyio")

library("preprocessCore") # required for normalize.quantiles.use.target function

install.packages("preprocessCore")


setwd("C:/.../target folder")


##### simpleCNV: #####

#---- Originally: -----

#> simpleCNV <- function (snpProbeInfo, snpInfo, snpReferenceDistribution = NULL,
#  invariantProbeInfo, invariantProbesetInfo, invariantReferenceDistribution = NULL,
#  celFiles = expandCelFiles(getwd()),referenceCelFile, chromosomes = c(1:19,"X", "Y",
#"M"),
#  verbose = FALSE, cluster = NULL, summaryOutputFile = NULL)
#{...}


#-----Rewritten: -----

celFiles = expandCelFiles("C:/.../folder with cel.files")

chromosomes = c(1:19,"X", "Y", "M")

verbose = FALSE

cluster = NULL

```

```
summaryOutputFile = NULL
```

```
#-----
```

```
snpCount <- nrow(snpInfo)
```

```
if (!inherits(snpProbeInfo, "data.frame") || !all(c("probeIndex",  
  "isAAAllele", "snpld") %in% names(snpProbeInfo))) {  
  stop("You must supply a \"snpProbeInfo\" data frame parameter which has ",  
    "at a minimum the \"probeIndex\", \"isAAAllele\" and \"snpld\" ",  
    "components. Please see the help documentation for more details.")  
}
```

```
snpProbeInfo$snpld <- as.factor(snpProbeInfo$snpld)
```

```
if (!inherits(snpInfo, "data.frame") || !all(c("snpld", "chrId",  
  "positionBp") %in% names(snpInfo))) {  
  stop("You must supply a \"snpInfo\" data frame parameter which has ",  
    "at a minimum the \"snpld\", \"chrId\" and \"positionBp\" ",  
    "components. Please see the help documentation for more details.")  
}
```

```
snpInfo$snpld <- as.factor(snpInfo$snpld)
```

```
combinedInfo <- cbind(snpInfo[, c("snpld", "chrId", "positionBp")],  
  0)
```

```
colnames(combinedInfo) <- c("ID", "chrId", "positionBp",  
  "type")
```

```
invariantProbeInfo <- MouseDivGeno:::.listify(invariantProbeInfo)
```

```
invariantProbesetInfo <- MouseDivGeno:::.listify(invariantProbesetInfo)
```

```
invariantReferenceDistribution <- MouseDivGeno:::.listify(invariantReferenceDistribution)
```

```
invariantGroupCount <- length(invariantProbeInfo)
```

```

if (length(invariantReferenceDistribution) == 0) {
  for (i in 1:invariantGroupCount) {
    invariantReferenceDistribution[[i]] <- NULL
  }
}

if (invariantGroupCount != length(invariantProbesetInfo) ||
    invariantGroupCount != length(invariantReferenceDistribution)) {
  stop("there is a mismatch between the \"invariantProbeInfo\", \"",
       "\"invariantProbesetInfo\", \"invariantReferenceDistribution\"")
}

if (!is.null(summaryOutputFile)) {
  if (is.character(summaryOutputFile)) {
    summaryOutputFile <- file(summaryOutputFile, "wt")
  }

  invStrs <- paste("Number of exon", 1:invariantGroupCount,
                  sep = "")

  invStrs <- paste(invStrs, " sets", sep = "")

  summaryHeader <- c("Name", "Status", "Chromosome", "StartPosition",
                    "EndPosition", "Number of Probe sets", "Size", "Number of SNP probe sets",
                    invStrs, "Mean intensity of reference sample", "Mean intensity of sample")

  write.table(rbind(summaryHeader), file = summaryOutputFile,
             row.names = FALSE, col.names = FALSE)
}

for (i in 1:invariantGroupCount) {
  if (!inherits(invariantProbeInfo[[i]], "data.frame") ||
      !all(c("probeIndex", "probesetId") %in% names(invariantProbeInfo[[i]]))) {

```

```

stop("You must supply a \"invariantProbeInfo\" data frame parameter which has ",
     "at a minimum the \"probeIndex\", and \"probesetId\" ",
     "components. Please see the help documentation for more details.")
}

invariantProbeInfo[[i]]$probesetId <- as.factor(invariantProbeInfo[[i]]$probesetId)

if (!inherits(invariantProbesetInfo[[i]], "data.frame") ||
    !all(c("probesetId", "chrId", "positionBp") %in%
         names(invariantProbesetInfo[[i]]))) {
  stop("You must supply a \"invariantProbesetInfo\" data frame parameter which has
",
       "at a minimum the \"probesetId\", \"chrId\" and \"positionBp\" ",
       "components. Please see the help documentation for more details.")
}

invariantProbesetInfo[[i]]$probesetId <- as.factor(invariantProbesetInfo[[i]]
$probesetId)

newInfo <- cbind(invariantProbesetInfo[[i]][, c("probesetId",
        "chrId", "positionBp")], i)

colnames(newInfo) <- c("ID", "chrId", "positionBp", "type")

combinedInfo <- rbind(combinedInfo, newInfo)

if (!is.null(invariantReferenceDistribution[[i]]) &&
    !is.numeric(invariantReferenceDistribution[[i]])) {
  stop("The \"invariantReferenceDistribution\" should either be ",
       "numeric or NULL")
}
}

if (is.list(ceFiles)) {
  if (!("fileName" %in% names(ceFiles))) {

```

```

    stop("failed to find \"fileName\" component in the \"celFiles\" list")
  }

  celFiles <- celFiles$fileName
}

sampleCount <- length(celFiles)

chromosomes <- toupper(as.character(chromosomes))

snpChromosomes <- unique(snpInfo$chrId)

if (!all(chromosomes %in% snpChromosomes)) {

  warning("SNP data for the following requested chromosomes are not available: ",
    paste(setdiff(chromosomes, snpChromosomes), collapse = ", "),
    ". These chromosomes will be skipped.")
}

chromosomes <- intersect(chromosomes, snpChromosomes)

rm(snpChromosomes)

invariantChromosomes <- as.character(unique(unlist(lapply(invariantProbesetInfo,
  function(x) {
    x$chrId
  }), use.names = F)))

if (!all(chromosomes %in% invariantChromosomes)) {

  warning("Invariant data for the following requested chromosomes are not available: ",
    paste(setdiff(chromosomes, invariantChromosomes),
      collapse = ", "), ". These chromosomes will be skipped.")
}

chromosomes <- unique(chromosomes, invariantChromosomes)

rm(invariantChromosomes)

if (length(chromosomes) == 0) {

```

```

    stop("Stopping CNV analysis. There are no chromosomes to process")
}

if (verbose) {
  cat("processing CEL files\n")
}

combinedInfoByChr <- list()

for (chr in chromosomes) {
  info <- combinedInfo[combinedInfo$chrId == chr, , drop = FALSE]

  info <- info[order(info$positionBp, as.character(info$ID)),
    , drop = FALSE]

  combinedInfoByChr[[chr]] <- info
}

rm(combinedInfo)

#----- Rewritten from here on: -----

unlistVectorsKeepNames <- function(x)
{
  y <- NULL
  for(z in x)
  {
    y <- c(y,z)
  }
  y
}

```

```

#Combine all probe infos in one table:

IGPex1 <- invariantProbesetInfo$exon1  #saves probe info IGP (exon1) in IGPex1
IGPex2 <- invariantProbesetInfo$exon2  # saves probe info IGP (Exon2) in IGPex2

SNP <- snpInfo[,c("snpId","chrId","positionBp")]  #saves probe position info of SNP probes
                                                # in SNP

names(SNP) <- c("probesetId","chrId","positionBp") #renames SNPs, identical to IGPex1 and
                                                #IGPex2

allProbes <- rbind(IGPex1,IGPex2,SNP)

allProbesSorted <- with(allProbes, allProbes[order(chrId,positionBp),]) #sorted by Chr, then
                                                                    # byPosition

print(head(allProbesSorted))

print(dim(allProbesSorted))


for (currCelFile in celFiles) {

  normIntensitiesMatrix <- NULL


##### .normalizeForSimpleCNV: #####

#function (celFileName, chromosomes, snpProbeInfo, snpInfo, snpReferenceDistribution,
#  invariantProbeInfo, invariantProbesetInfo, invariantReferenceDistribution,
#  verbose)
#{ ... }


#----- Rewritten: -----

celFileName <- currCelFile

```

```

#-----
if (verbose)

  cat("Reading and normalizing CEL file: ", celFileName,

      "\n", sep = "")

invariantGroupCount <- length(invariantProbeInfo)

celData <- read.celfile(celFileName, intensity.means.only = TRUE)

celData <- log2(as.matrix(celData[["INTENSITY"]][["MEAN"]]))

invY <- list()

for (i in 1:invariantGroupCount) {

  invY[[i]] <- celData[invariantProbeInfo[[i]]$probeIndex,

    , drop = FALSE]

  if (length(invariantProbeInfo[[i]]$correction) > 0)

    invY[[i]] <- invY[[i]] + invariantProbeInfo[[i]]$correction

  if (length(invariantReferenceDistribution[[i]]) > 0)

    invY[[i]] <- normalize.quantiles.use.target(invY[[i]],

      target = invariantReferenceDistribution[[i]])

  invY[[i]] <- subColSummarizeMedian(matrix(invY[[i]],

    ncol = 1), invariantProbeInfo[[i]]$probesetId)

  invY[[i]] <- invY[[i]][, 1, drop = TRUE]

}

snpY <- celData[snpProbeInfo$probeIndex, , drop = FALSE]

if (length(snpProbeInfo$correction) > 0)

  snpY <- snpY + snpProbeInfo$correction

if (length(snpReferenceDistribution) > 0)

  snpY <- normalize.quantiles.use.target(snpY, target = snpReferenceDistribution)

```



```

aProbeInt <- snpY[snpProbeInfo$isAAllele, 1, drop = FALSE]
bProbeInt <- snpY[!snpProbeInfo$isAAllele, 1, drop = FALSE]
aSnplnt <- subColSummarizeMedian(matrix(aProbeInt, ncol = 1),
  snpProbeInfo$snpld[snpProbeInfo$isAAllele])
bSnplnt <- subColSummarizeMedian(matrix(bProbeInt, ncol = 1),
  snpProbeInfo$snpld[!snpProbeInfo$isAAllele])
aSnplds <- rownames(aSnplnt)
bSnplds <- rownames(bSnplnt)
if (!all(aSnplds == bSnplds)) {
  stop("The SNP IDs for the A alleles should match up with the SNP IDs ",
    "for the B alleles but they do not.")
}
aGreaterIds <- aSnplds[aSnplnt > bSnplnt]
aGreaterProbeIndices <- which(snpProbeInfo$snpld[snpProbeInfo$isAAllele] %in%
  aGreaterIds)
highProbeInt <- bProbeInt
highProbeInt[aGreaterProbeIndices, ] <- aProbeInt[aGreaterProbeIndices,
  ]
highProbeInt <- normalize.quantiles.use.target(highProbeInt,
  target = invariantReferenceDistribution[[1]])
highSnplnt <- subColSummarizeMedian(matrix(highProbeInt,
  ncol = 1), snpProbeInfo$snpld[snpProbeInfo$isAAllele])
highSnplnt <- highSnplnt[, 1, drop = TRUE]
if (length(snpInfo$snpld) != length(highSnplnt)) {
  stop("internal error: vector lengths should match but they do not")
}

```

```

snpldOrdering <- match(snpInfo$snpld, names(highSnpInt))
if (any(is.na(snpldOrdering)))
  stop("Failed to match up SNP probe IDs")
highSnpInt <- highSnpInt[snpldOrdering]
for (i in 1:invariantGroupCount) {
  if (length(invariantProbesetInfo[[i]]$probesetId) !=
    length(invY[[i]])) {
    stop("internal error: vector lengths should match but they do not")
  }
  invldOrdering <- match(invariantProbesetInfo[[i]]$probesetId,
    names(invY[[i]]))
  if (any(is.na(invldOrdering)))
    stop("Failed to match up invariant probe IDs")
  invY[[i]] <- invY[[i]][invldOrdering]
}
combinedIntensities <- list()
for (chr in chromosomes) {
  chrInvInt <- NULL
  chrInvPos <- NULL
  for (i in 1:invariantGroupCount) {
    chrInvIndices <- which(invariantProbesetInfo[[i]]$chrId ==
      chr)
    chrInvInt <- c(chrInvInt, invY[[i]][chrInvIndices])
    chrInvPos <- c(chrInvPos, invariantProbesetInfo[[i]]$positionBp[chrInvIndices])
  }
  chrSnpIndices <- which(snpInfo$chrId == chr)

```

```

chrSnpInt <- highSnpInt[chrSnpIndices]

chrSnpPos <- snpInfo$positionBp[chrSnpIndices]

combinedInt <- c(chrInvInt, chrSnpInt)

combinedPos <- c(chrInvPos, chrSnpPos)

combinedInt <- combinedInt[order(combinedPos, names(combinedInt))]

combinedIntensities[[chr]] <- combinedInt
}

# ----- changed again: -----

normIntensities <- combinedIntensities

normIntensities <- unlistVectorsKeepNames(normIntensities)

normIntensitiesMatrix <- matrix(normIntensities, ncol = 1)

rownames(normIntensitiesMatrix) <- names(normIntensities)

# normIntensitiesMatrix <- cbind(normIntensitiesMatrix, normIntensities)

colnames(normIntensitiesMatrix) <- currCelFile    #colnames do have full file path as name

Title <- colnames(normIntensitiesMatrix)    #path is exchanged by only the file name

#----- change here RegularExpressions if needed: -----

Title <- sub("C.+GetNormIntensities/", "", Title, perl=TRUE)

Title <- sub("-GES12.+", "", Title, perl=TRUE)

colnames(normIntensitiesMatrix) <- Title    #colnames (file path) exchanged (by file
name)

```

```

#-----

All <- merge(allProbesSorted,normIntensitiesMatrix,by="row.names",all=TRUE)

      #combine all infos again

All <- All[, -which(names(All) %in% "Row.names")]    # Column Row.names (automatically
                                                    #created by...

                                                    # ...merge(by="row.names") was deleted

All <- with(All, All[order(chrId,positionBp),])    #sorted by Chr and Position

print(Title)

print(dim(All))

#print(head(All))

write.table(All,paste(currCelFile,"_NormIntensities.CEL"))


All <- NULL

}

date()

```