Integracion metilación ADN y expresión microARNs

Belén Mollá Moliner

21/06/2021

Contents

1 Integración por POSICIONAMIENTO GENÓMICO

Partimos de los mirnas expresados diferencialmente y anotados con la versión hg38 del genoma humano

1.1 Conversión de la anotación hg38 a la versión hg19 del genoma humano

```
> df <- read.csv("mirna_comp4_hg38_primary_completo.csv", fileEncoding = "utf-8",
+     header = T, stringsAsFactors = F, skipNul = T, row.names = 1)
> library(rtracklayer)
> library(liftOver)
> library(IRanges)
>
> chain <- import.chain("hg38ToHg19.over.chain")
>
> library(GenomicRanges)
> gr <- makeGRangesFromDataFrame(df, strand.field = "strand", TRUE)
> humcon <- liftOver(gr, chain)
>
> library(biovizBase)
> humcon = flatGrl(humcon)
>
> library(Repitools)
```

```
> humcon = annoGR2DF(humcon)
> write.csv(humcon, file = "mirna_comp4_hg19_primary_completo.csv")
>
> # añado a los datos
> df_hg19 = merge(df, humcon, by = "mirna_primary")
> write.csv(df_hg19, file = "mirna_comp4_hg38_hg19_primary_completo.csv")
```

1.2 Búsqueda de las posiciones metiladas dentro de la secuencia de los mirnas

Busco en todas las comparaciones para saber si hay alguna coincidencia o no entre la metilación y el mirnoma!!:

NC_limma metiloma vs comp2 mirnoma:

```
> rm(list = ls())
> gc()
> library(data.table)
>
> metilacion_NC_limma = read.csv("limma_NC_0.1.csv", fileEncoding = "utf-8",
+     header = T, stringsAsFactors = F, skipNul = T)
> mirna_comp2 <- read.csv("mirna_comp2_hg19_primary.csv", fileEncoding = "utf-8",
+     header = T, stringsAsFactors = F, skipNul = T)
>
> Y_NC_limma = data.table(a = metilacion_NC_limma$pos)
> Y_NC_limma
> range_comp2_limma = data.table(mirna_comp2$start, mirna_comp2$end)
> range_comp2_limma
> Y_NC_limma[a %inrange% range_comp2_limma]
```

SC_limma metiloma vs comp4 mirnoma:

```
> rm(list = ls())
> gc()
> library(data.table)
>
> metilacion_SC_limma = read.csv("limma_SC_0.1.csv", fileEncoding = "utf-8",
+     header = T, stringsAsFactors = F, skipNul = T)
> dim(metilacion_SC_limma)
> names(metilacion_SC_limma)
> mirna_comp4 <- read.csv("mirna_comp4_hg19_primary_completo.csv",
+     fileEncoding = "utf-8", header = T, stringsAsFactors = F,
+     skipNul = T)
>
> Y_SC_limma = data.table(a = metilacion_SC_limma$pos)
> Y_SC_limma
> range_comp2_limma = data.table(mirna_comp4$start, mirna_comp4$end)
> range_comp2_limma
> Y_SC_limma[a %inrange% range_comp2_limma]
```

2 Integración MIXOMICS

2.1 preparación de los Datos metiloma

Seleccionamos 5 controles para hacer la mediana de sus valores beta

```
> library(minfi)
> load("GRsetF_integracion_0.1.RData")
> data.frame(pData(GRsetF_integracion_0.1))
> str(GRsetF_integracion_0.1)
> B <- getBeta(GRsetF_integracion_0.1)
> dim(B)
> # control <- subset(B, pheno$Sample_Group=='C' &
+ # pheno$Sample_Name==c('Pac49','Pac37','Pac54','Pac47','Pac55'))
+ # str(control)
> head(B)
> control = B[, c("203922140017_R01C01", "203922140020_R04C01",
+     "203922140020_R07C01", "204081740067_R04C01", "204081740067_R07C01")]
> str(control)
> head(control)
> control_median = data.frame(apply(control, 1, median))
> str(control_median)
> head(control_median)
> class(control_median)
> dim(control_median)
> def = data.frame(control, control_median)
> str(def)
> head(def)
> dim(def)
> # consultamos si hay NAs
> library(tidyr)
> is.na(def) %>%
+     table()
> # eliminamos los casos perdidos
> defnew = na.omit(def)
> dim(defnew)
> names(defnew)
>
> # prueba del cálculo de la mediana de la primera fila
> x = c(0.898569, 0.8786214, 0.8959296, 0.8883907, 0.9027819)
> median(x)
```

Seleccionamos todas las muestras coincidentes entre metiloma y mirnoma:

```
> # metadatos
> pheno = data.frame(pData(GRsetF_integracion_0.1))
> pheno
>
> # seleccionamos 2 muestras control Pac2 y la mediana de las
+ # 5 muestras control
>
> C = data.frame(B[, c("203922140017_R04C01")], control_median)
> colnames(C) = c("Pac2", "Pool")
> head(C)
> dim(C)
>
> # seleccionamos 7 muestras Nosocomial
>
> N = data.frame(B[, c("203922140017_R03C01", "203922140017_R05C01",
+     "203922140020_R01C01", "203922140020_R05C01", "204081740067_R01C01",
```

```
+     "204081740067_R05C01", "204081740067_R06C01")])
> colnames(N) = c("Pac4", "Pac20", "Pac7", "Pac23", "Pac19", "Pac14",
+     "Pac25")
> head(N)
> dim(N)
>
> # seleccionamos 6 muestras Vertical
>
> V = data.frame(B[, c("203922140017_R02C01", "203922140017_R06C01",
+     "203922140020_R02C01", "203922140020_R06C01", "204081740067_R02C01")])
> colnames(V) = c("Pac8", "Pac17", "Pac30", "Pac35", "Pac32")
> head(V)
> dim(V)
>
> # data.frame mixomics 15 muestras
>
> met = data.frame(C, N, V)
> head(met)
> dim(met)
> names(met)
> write.csv(met, file = "met.csv")
```

2.2 preparación de los datos mirnoma

Lo mismo con los datos filtrados y normalizados del mirnoma:

```
> # library(readxl)
> library(xlsx)
> # library(openxlsx)
> datos <- read_xlsx("norm&filt_counts_mirnome.xlsx")
> meta <- read_xlsx("metadata4grupos.xlsx")
> names(datos)
> head(datos$...1)
>
> # seleccionamos 6 controles para hacer el pool
> control_mir = datos[, c("P3_T1", "P10_T1", "P12_T1", "P13_T1",
+     "P18_T1", "P33_T1")]
> control_mir_median = data.frame(apply(control_mir, 1, median))
> def_mir = data.frame(control_mir, control_mir_median)
> is.na(def_mir)
>
> # eliminamos los casos perdidos
> defnew_dir = na.omit(def_mir)
> dim(defnew_dir)
> names(defnew_dir)
```

Seleccionamos todas las muestras coincidentes entre metiloma y mirnoma:

```
> # seleccionamos 2 muestras control Pac2 y la mediana de las
+ # 5 muestras control
>
> C_mir = data.frame(datos[, c("...1", "P2_T1")], control_mir_median)
> colnames(C_mir) = c("mirna", "Pac2", "Pool")
> head(C_mir)
> dim(C_mir)
```

```
>
> # seleccionamos 7 muestras Nosocomial
>
> N_mir = data.frame(datos[, c("P4_T1", "P20_T1", "P7_T1", "P23_T1",
+     "P19_T1", "P14_T1", "P25_T1")])
> colnames(N_mir) = c("Pac4", "Pac20", "Pac7", "Pac23", "Pac19",
+     "Pac14", "Pac25")
> head(N_mir)
> dim(N_mir)
>
> # seleccionamos 6 muestras Vertical
>
> V_mir = data.frame(datos[, c("P8_T1", "P17_T1", "P30_T1", "P35_T1",
+     "P32_T1")])
> colnames(V_mir) = c("Pac8", "Pac17", "Pac30", "Pac35", "Pac32")
> head(V_mir)
> dim(V_mir)
>
> # data.frame mixomics 15 muestras
>
> mir = data.frame(C_mir, N_mir, V_mir)
> write.csv(mir, file = "mir.csv")
```

2.3 Integración mixOMICS

Abrimos los datos de metilación y mirnoma. Establecemos el nombre de los cgs y el nombre de los mirnas como ID.

```
> # datos metilacion
> met <- read.csv(file = "met.csv", header = TRUE, dec = ".", row.names = 1)
> met = as.data.frame(t(met))
>
> # datos mirnoma
> mir = read.csv(file = "mir.csv", header = TRUE, row.names = 1)
> mir = as.data.frame(t(mir))
>
> # metadatos
> subtype <- read.csv(file = "metadataintegracion.csv", header = TRUE,
+     dec = ".")
> subtype
>
> # juntamos en un objeto Large List como en el ejemplo de
+ # miXOmics
> data.train_ <- list(metilacion = data.frame(met), mirna = data.frame(mir),
+     subtype = data.frame(subtype))
>
> # check dimension
> lapply(data.train_, dim)
> data.train_$subtype$Groups
```

Según el caso práctico del mixOmics los datos se dividen en data.test y data.train. En nuestro caso sólo vamos a hacer data.train, ya que tenemos muy pocas muestras:

```
> library(mixOmics)
> # extract training data
```

```
> data_ = list(metilacion = data.train_$metilacion, miRNA = data.train_$mirna)
>
>
> # check dimension
> lapply(data_, dim)
>
> # outcome
> Y_ = as.factor(data.train_$subtype$Groups)
> length(Y_)
> summary(Y_)
> Y_
```

2.3.1 Parameter choice

```
> design_ = matrix(0.1, ncol = length(data_), nrow = length(data_),
+     dimnames = list(names(data_), names(data_)))
> diag(design_) = 0
>
> design_
```

2.3.1.1 Design

```
> library(mixOmics)
> sgccda.res_ = block.splsda(X = data_, Y = Y_, ncomp = 5, design = design_)
>
> set.seed(123)  # for reproducibility, only when the `cpus' argument is not used
> # this code takes a couple of min to run
> perf.diablo = perf(sgccda.res_, validation = "Mfold", folds = 10,
+     nrepeat = 10)
> save(perf.diablo, file = "perf.diablo.RData")
>
> # perf.diablo # lists the different outputs
> plot(perf.diablo)
> dev.print(pdf, "perf.diablo.pdf")  # guardo la grafica
> dev.off()
> # the optimal number od components for the final DIABLO
+ # model
> perf.diablo$choice.ncomp$WeightedVote
> ncomp = perf.diablo$choice.ncomp$WeightedVote["Overall.BER",
+     "centroids.dist"]
```

2.3.1.2 Tuning the number of components

```
> set.seed(123)  # for reproducibility, only when the `cpus' argument is not used
> test.keepX = list(metilacion = c(5:9, seq(10, 18, 2), seq(20,
+     30, 5)), miRNA = c(5:9, seq(10, 18, 2), seq(20, 30, 5)))
> t1 = proc.time()
> tune.SEPSIS = tune.block.splsda(X = data_, Y = Y_, ncomp = ncomp,
+     test.keepX = test.keepX, design = design_, validation = "Mfold",
+     folds = 10, nrepeat = 1, dist = "centroids.dist")
> t2 = proc.time()
```

```
> running_time = t2
> running_time
>
> save(tune.SEPSIS, file = "tune.SEPSIS.RData")
>
> # the number of features select on each component
> list.keepX = tune.SEPSIS$choice.keepX
> list.keepX
```

2.3.1.3 Tuning keepX

2.3.2 The final model

```
> sgccda.res = block.splsda(X = data_, Y = Y_, ncomp = ncomp, keepX = list.keepX,
+     design = design_)
> # sgccda.res # list the different functions of interest
+ # related to that object
>
> sgccda.res$design
>
> # metilacion variables selected on component 1
> posiciones_metilacion = selectVar(sgccda.res, block = "metilacion",
+     comp = 1)$metilacion$name
> write.csv(posiciones_metilacion, file = "posiciones_metilacion.csv")
>
> # mirnas variables selected on component 1
> mirnas = selectVar(sgccda.res, block = "miRNA", comp = 1)$miRNA$name
> write.csv(mirnas, file = "mirnas.csv")
>
> # anoto las posiciones de metilacion
> pos_met <- read.csv(file = "posiciones_metilacion.csv", header = TRUE,
+     dec = ".", row.names = 2)
> library(IlluminaHumanMethylationEPICanno.ilm10b4.hg19)
> annEPIC <- getAnnotation(IlluminaHumanMethylationEPICanno.ilm10b4.hg19)
> annEPICsub <- annEPIC[match(rownames(pos_met), annEPIC$Name),
+     c(1:4, 12, 18, 22)]
> annEPICsub
> write.csv(annEPICsub, file = "posiciones_metilacion_anotadas.csv")
>
>
> # anotación de los mirnas con
> ann_limma_1_sig <- select(hsa_MTI_v22.csv, keys = rownames(limma_1_sig),
+     keytype = "ENSEMBL", columns = c("Target.Gene", "Entrez.Gene.ID."))
```

2.3.3 Sample plots

```
> plotDiablo(sgccda.res, ncomp = 1)
> dev.print(pdf, "plotDiablo.pdf")  # guardo la grafica
> dev.off()
```

```
> plotIndiv(sgccda.res, ind.names = FALSE, legend = TRUE, title = "DIABLO")
> dev.print(pdf, "plotIndiv.pdf")  # guardo la grafica
> dev.off()
```

```
> plotArrow(sgccda.res, ind.names = FALSE, legend = TRUE, title = "DIABLO")
```

2.3.4 Variable plots

```
> plotVar(sgccda.res, var.names = FALSE, style = "graphics", legend = TRUE,
+     pch = c(16, 17), cex = c(2, 2), col = c("darkorchid", "brown1"))
```

```
> circosPlot(sgccda.res, cutoff = 0.7, line = TRUE, color.blocks = c("darkorchid",
+     "brown1"), color.cor = c("chocolate3", "grey20"), size.labels = 1.5)
> dev.print(pdf, "circosPlot.pdf")  # guardo la grafica
> dev.off()
```

The network can be saved in a .gml format to be input into the software Cytoscape, using the R package igraph

```
> # not run
> library(igraph)
> my.network = network(sgccda.res, blocks = c(1, 2), color.node = c("darkorchid",
+     "brown1"), cutoff = 0.4)
> write.graph(my.network$gR, file = "myNetwork.gml", format = "gml")
> dev.print(pdf, "my.network.pdf")  # guardo la grafica
> dev.off()
```

```
> plotLoadings(sgccda.res, comp = 1, contrib = "max", method = "median")
> dev.print(pdf, "plotLoadings.pdf")  # guardo la grafica
> dev.off()
```

```
> cimDiablo(sgccda.res)
> dev.print(pdf, "cimDiablo.pdf")  # guardo la grafica
> dev.off()
```

```
> save.image(file = "all_results_mixOmics.RData")
```

2.3.5 Performance of the model

```
> set.seed(123)  # for reproducibility, only when the `cpus' argument is not used
> perf.diablo = perf(sgccda.res, validation = "Mfold", M = 10,
+     nrepeat = 10, dist = "centroids.dist")
>
> save(perf.diablo, file = "perf.diablo_performance.RData")
>
>
>
> # perf.diablo # lists the different outputs
>
> # Performance with Majority vote
> perf.diablo$MajorityVote.error.rate
>
> # Performance with Weighted prediction
> perf.diablo$WeightedVote.error.rate
>
> auc.splsda = auroc(sgccda.res, roc.block = "miRNA", roc.comp = 2)  # no imprime la gráfica
>
> auc.splsda = auroc(sgccda.res, roc.block = "miRNA", roc.comp = 1)
>
```

```
> dev.print(pdf, "auc.splsda.pdf")  # guardo la grafica
> dev.off()
>
> auc.splsda = auroc(sgccda.res, roc.block = "metiloma", roc.comp = 1)
>
> dev.print(pdf, "auc.splsda_metiloma.pdf")  # guardo la grafica
> dev.off()
```

```
> save.image(file = "all_results_mixOmics.RData")
```

Integration with DIABLO for N-ingretaion with low sample size:

https://mixomics-users.discourse.group/t/integration-with-diablo-for-n-ingretaion-with-low-sample-size/136

kimanh.lecao: parameters tuning (use leave one out cross validation). You can use the usual DIABLO vignette, but basically remove the tune() and perf() functions from your analyses (those are the ones using cross validation).