BIOINF 545

Real dataset tutorial - Notebook

iTFSC - An R package for robust transcription factor evaluation in single-cell RNA-seq data

Gondal, Mahnoor
4-9-2023

```
---
title: "R Notebook"
output: html_notebook
editor_options:
  chunk_output_type: inline
---



## reading in the libraries
```{r}
library(Seurat)
library(SeuratDisk)
library(SCENIC)
library(BITFAM)
library(dorothea)
library(piano)
library(ggplot2)
library(dplyr)
library(tidyr)
library(AUCell)
library(RcisTarget)
library(GENIE3)
library(base)
library(tibble)
library(ComplexHeatmap)
library(ggVennDiagram)
```


Step 1: Reading the scRNA-seq data
```

```{r}

#' this is case studies data - you can download this RDS file on your local machine using this url: https://drive.google.com/drive/folders/1WL0TxDAQpPGzmGy8gltT-x-ezSw6Ndh1?ths=true

#' please update the path to match your directory

#'

Qian_merged <- readRDS("/mctp/share/users/gondal/DC_Jenny/03_output/Lung_Breast_Colon_Qian/version_01_02_06_23/Qian_merged.RDS")

```

Step 2: Select the tissue (using existing case study)

```{r}

#' Selecting tissue

#'

#' @description

#' allows the user to select the tissue of interest from the case studies.

#'

#' @details

#' We have provided 4 case studies for this package.

#' Users can choose to employ the case studies of their choice or use their own data

#' In this function, users can choose which case study they want to work on

#' The case studies include: lung cancer 2 -> breast cancer 3 -> ovarian cancer 4 -> colon cancer

#'

#' @param in

#' this requires a seurat object, for which the data can be downloaded from here: https://drive.google.com/drive/folders/1WL0TxDAQpPGzmGy8gltT-x-ezSw6Ndh1?ths=true

#'

#' @param out

#' this will return the seurat object containing the case study of interest from the four provided
```

```
#'
Tissue_selection <- function(seurat_obj) {

  # asking users to select the case study of interest, please enter the number corresponding to the case study of interest
  tissue_name <- readline(prompt = paste("Enter the tissue of interest from; 1 -> lung cancer 2 -> breast cancer 3 -> ovarian cancer 4 -> colon cancer: "))
  if (tissue_name == 1)
    seurat_subset <- subset(seurat_obj, subset = TumorType_detail == "Lung_Cancer")

  if (tissue_name == 2)
    seurat_subset <- subset(seurat_obj, subset = TumorType_detail == "Breast_Cancer")

  if (tissue_name == 3)
    seurat_subset <- subset(seurat_obj, subset = TumorType_detail == "Ovarian_Cancer")

  if (tissue_name == 4)
    seurat_subset <- subset(seurat_obj, subset = TumorType_detail == "Colorectal_Cancer")

    return (seurat_subset)
}
Qian_merged_interest <- Tissue_selection(Qian_merged)
```

Step 3: Performing data QC

```{r}
#' Perfroming data quality control on normalized data slot
#'
#' @description
```

```
#' to run the subsequent functions, we need to know if the seurat data contains normalized counts in the right location
#'
#' @details
#' this function makes a 200 by 200 matrix of normalized counts and converts the matrix into a dataframe
#' the values are then checked if they are decimal or integer to check if normalized counts are present
#'
#' @param in
#' this requires a seurat object
#'
#' @param out
#' it will tell the user is the normalized data is present and in the right location
#'
#'
Normalization_check <- function(seurat_obj) {

  seurat_obj_data_slot <- seurat_obj@assays$RNA@data[1:200, 1:200]


  seurat_obj_data_slot <- as.data.frame(summary(seurat_obj_data_slot))


  if (any(round(seurat_obj_data_slot$x) != seurat_obj_data_slot$x) == TRUE)
#  print("data is normalized, good please proceed")
  return ("data is normalized, good please proceed")


  if (any(round(seurat_obj_data_slot$x) != seurat_obj_data_slot$x) == FALSE)
#  print("data is not normalized please normalize data first")
  return ("data is not normalized please normalize data first")
}
```

```
Normalization_check(Qian_merged_interest)


#' Perfroming data quality control on raw data slot
#'
#' @description
#' to run the subsequent functions, we need to know if the seurat data contains raw counts in the right location
#'
#' @details
#' this function makes a 200 by 200 matrix of raw counts and converts the matrix into a dataframe
#' the values are then checked if they are decimal or integer to check if raw counts are present
#'
#' @param in
#' this requires a seurat object
#'
#' @param out
#' it will tell the user is the raw counts is present and in the right location
#'
#'


Rawcount_check <- function(seurat_obj) {


  seurat_obj_counts_slot <- seurat_obj@assays$RNA@counts[1:200, 1:200]


  seurat_obj_counts_slot <- as.data.frame(summary(seurat_obj_counts_slot))


  if (any(round(seurat_obj_counts_slot$x) != seurat_obj_counts_slot$x) == TRUE)
 # print("raw counts not present, please provide raw counts for accurate analysis")
  return ("raw counts not present, please provide raw counts for accurate analysis")
```

```
 if (any(round(seurat_obj_counts_slot$x) != seurat_obj_counts_slot$x) == FALSE)
# print("raw counts present, good please proceed")
 return ("raw counts present, good please proceed")

}

Rawcount_check(Qian_merged_interest)
```

Step 4: Downsample seurat object (this would work with existing case study or new data)

```{r}

#' Downsampling the data
#'
#' @description
#' single-cell objects are usually very large to handle especially for testing purposes therefore this function will downsample the data to an ident users is most interested in
#'
#' @details
#' this function will ask the user for the ident of interest and the number of cells they want each ident to contain
#'
#' @param in
#' this requires a seurat object, ident name and number of cells to downsample
#'
#' @param out
#' it will output the downsampled seurat object
#'
#'
```

```
Down_sample <- function(seurat_obj, ident_name, cell_count) {

  seurat_obj <- SetIdent(seurat_obj, value = ident_name)

  down.sample <- subset(seurat_obj,  downsample = cell_count)

  return (down.sample)
}

Qian_merged_interest_downsample <- Down_sample(Qian_merged_interest, "CellType_updated", 10)
```

Step 5a: BITFAM

```{r}

#' Running BITFAM
#'
#' @description
#' BITFAM [PMID: 34193535] is one of the methods that we want to implement for this tool and it estimates the activity of the transcription factor from single-cell data
#'
#' @details
#' this function will use the downsampled object (since it takes a long time to run ~ 2hours) and run BITFAM on it
#'
#' @param in
#' this requires a seurat object, and path to save results
```

```
#'

#' @param out

#' it will output a matrix file computed for the activity of each TF in each cell

#'

#'

Runnin_BITFAM <- function(seurat_obj, path_to_save_res) {


  print("The time for running this code depends on how many cells users decieded to downsample on in the previous function, for testing purposes, I would recommend a very small number eg 2-5 - it might still take 30-45 mins")


  seurat_obj_counts <- GetAssayData(seurat_obj, slot = "counts", assay = "RNA")


  data_matrix_normalized <- BITFAM_preprocess(raw_data = seurat_obj_counts)


  BITFAM_res <- BITFAM(data = data_matrix_normalized, species = "human", scATAC_obj = NA, ncores = parallel::detectCores())


  BITFAM_activities_res <- BITFAM_activities(BITFAM_res)


  write.csv(BITFAM_activities_res, path_to_save_res, row.names = TRUE)


}


Runnin_BITFAM(Qian_merged_interest_downsample, "/mctp/share/users/gondal/Classes/BIOINF576/BITFAM_activities_res_01.csv")
```


Dorothea

```{r}
Runnin_Dorothea <- function(seurat_obj, ident_name, path_to_save_res_file,
path_to_save_res_heatmap) {

dorothea_regulon_human <- get(data("dorothea_hs", package = "dorothea"))

down.sample_data.Dorothea <- dorothea::run_viper(seurat_obj, dorothea_regulon_human,
        options = list(method = "scale", minsize = 4,
                eset.filter = FALSE, cores = 1,
                verbose = FALSE))

DefaultAssay(object = down.sample_data.Dorothea) <- "dorothea"
down.sample_data.Dorothea <- ScaleData(down.sample_data.Dorothea)
down.sample_data.Dorothea <- RunPCA(down.sample_data.Dorothea, features =
rownames(down.sample_data.Dorothea), verbose = FALSE)
down.sample_data.Dorothea <- FindNeighbors(down.sample_data.Dorothea, dims = 1:10, verbose =
FALSE)
down.sample_data.Dorothea <- FindClusters(down.sample_data.Dorothea, resolution = 0.5, verbose =
FALSE)

down.sample_data.Dorothea <- RunUMAP(down.sample_data.Dorothea, dims = 1:10, umap.method =
"uwot", metric = "cosine")

down.sample_data.Dorothea.markers <- FindAllMarkers(down.sample_data.Dorothea, only.pos = TRUE,
min.pct = 0.25,
        logfc.threshold = 0.25, verbose = FALSE)

down.sample_data.Dorothea <- SetIdent(down.sample_data.Dorothea, value =
down.sample_data.Dorothea@meta.data$CellType_updated)
DimPlot(down.sample_data.Dorothea, reduction = "umap", label = TRUE, pt.size = 0.5) + NoLegend()
```

```
## We transform Viper scores, scaled by seurat, into a data frame to better
## handling the results
viper_scores_df <- GetAssayData(down.sample_data.Dorothea, slot = "data",
                                assay = "dorothea") %>%
  as.data.frame() %>%
  t()

## We create a data frame containing the cells and their clusters
CellsClusters <- data.frame(cell = names(Idents(down.sample_data.Dorothea)),
                            cell_type = as.character(Idents(down.sample_data.Dorothea)),
                            stringsAsFactors = FALSE)

## We create a data frame with the Viper score per cell and its clusters
viper_scores_clusters <- viper_scores_df  %>%
  as.data.frame() %>%
  rownames_to_column("cell") %>%
  gather(tf, activity, -cell) %>%
  inner_join(CellsClusters)



write.csv(viper_scores_df, path_to_save_res_file, row.names = TRUE)

}
```

```
Runnin_Dorothea(Qian_merged_interest_downsample, CellType_updated,
        "/mctp/share/users/gondal/Classes/BIOINF576/Dorothea_activities_res_01.csv",
        "/mctp/share/users/gondal/Classes/BIOINF576/Dorothea_activities_res_01_heatmap.png")
```

Running SCENIC

```{r}

Runnin_SCENIC <- function(seurat_obj, path_to_download, path_to_save_res_file) {

seurat_obj <- SetIdent(seurat_obj, value = seurat_obj@meta.data$CellType)

exprMat <- as.matrix(seurat_obj@assays$RNA@counts)
cellInfo <- data.frame(seuratCluster=Idents(seurat_obj))

saveRDS(cellInfo, file = paste0(path_to_download, "cellInfo.RDS"))
saveRDS(exprMat, file =  paste0(path_to_download, "exprMat.RDS"))

org <- "hgnc" # or hgnc, or dmel
dbDir <- "/mctp/share/users/gondal/01_scHLA/02_processing/colon_pelka/SCENIC" # RcisTarget databases location
myDatasetTitle <- "SCENIC example TS MHC1" # choose a name for your analysis
data(defaultDbNames)
dbs <- defaultDbNames[[org]]
scenicOptions <- initializeScenic(org=org, dbDir=dbDir, dbs=dbs, datasetTitle=myDatasetTitle, nCores=15)
```

```
scenicOptions@inputDatasetInfo$cellInfo <-  paste0(path_to_download, "int/cellInfo.Rds")
scenicOptions@inputDatasetInfo$colVars <-  paste0(path_to_download, "int/colVars.Rds")
saveRDS(scenicOptions, file =  paste0(path_to_download, "scenicOptions_TS.RDS"))

genesKept <- geneFiltering(exprMat, scenicOptions=scenicOptions,
                           minCountsPerGene=3*.01*ncol(exprMat),
                           minSamples=ncol(exprMat)*.01)

exprMat_filtered <- exprMat[genesKept, ]
dim(exprMat_filtered)



runCorrelation(exprMat_filtered, scenicOptions)
saveRDS(scenicOptions, file =  paste0(path_to_download, "scenicOptions_TS.RDS"))

exprMat_filtered <- log2(exprMat_filtered+1)
runGenie3(exprMat_filtered, scenicOptions)

saveRDS(scenicOptions, file =  paste0(path_to_download, "scenicOptions_TS.RDS"))



scenicOptions@settings$verbose <- TRUE
scenicOptions@settings$nCores <- 15
scenicOptions@settings$seed <- 123

scenicOptions@settings$dbs <- scenicOptions@settings$dbs



scenicOptions <- runSCENIC_1_coexNetwork2modules(scenicOptions)
```

```
saveRDS(scenicOptions, file = paste0(path_to_download, "scenicOptions_TS.RDS"))

scenicOptions <- runSCENIC_2_createRegulons(scenicOptions)  #** Only for toy run!!

saveRDS(scenicOptions, file =  paste0(path_to_download, "scenicOptions_TS.RDS"))

scenicOptions <- runSCENIC_3_scoreCells(scenicOptions, exprMat_filtered)
saveRDS(scenicOptions, file =  paste0(path_to_download, "scenicOptions_TS.RDS"))

scenicOptions <- readRDS( paste0(path_to_download, "scenicOptions_TS.RDS"))

aucellApp <- plotTsne_AUCellApp(scenicOptions, exprMat_filtered)
savedSelections <- shiny::runApp(aucellApp)



regulonAUC <- loadInt(scenicOptions, "aucell_regulonAUC")
regulonAUC <- regulonAUC[onlyNonDuplicatedExtended(rownames(regulonAUC)),]
regulonActivity_byCellType <- sapply(split(rownames(cellInfo), cellInfo$seuratCluster),
                                     function(cells) rowMeans(getAUC(regulonAUC)[,cells]))
regulonActivity_byCellType_Scaled <- t(scale(t(regulonActivity_byCellType), center = T, scale=T))

png(file= paste0(path_to_download, "scenic.png"), width=1000, height=4000)
ComplexHeatmap::Heatmap(regulonActivity_byCellType_Scaled, name="Regulon activity")
dev.off()

write.csv(regulonActivity_byCellType_Scaled, path_to_save_res_file, row.names = TRUE)

}
```

```
Runnin_SCENIC(Qian_merged_interest_downsample,
              "/mctp/share/users/gondal/Classes/BIOINF576/",
              "/mctp/share/users/gondal/Classes/BIOINF576/SCENIC_activities_res_01.csv")
```

```

## Combining analysis

```{r}

Combining_res <- function(seurat_obj, path_to_download, path_to_save_res_file) {

  SCENIC <- read.csv("/mctp/share/users/gondal/Classes/BIOINF576/SCENIC_activities_res_01.csv",
         header = TRUE,
         sep = ",")
  SCENIC <- as.data.frame(SCENIC[,c(1, 7)])
  colnames(SCENIC) <- c("TF", "SCENIC")
  SCENIC_up <- filter(SCENIC, SCENIC > 0)
SCENIC_up$TF <- gsub("\\(.*","", SCENIC_up$TF)
SCENIC_up$TF <- gsub("_.*","", SCENIC_up$TF)
SCENIC_up$TF <- gsub("\\ .*","", SCENIC_up$TF)

  BITFAM <- read.csv("/mctp/share/users/gondal/Classes/BIOINF576/BITFAM_activities_res_01.csv",
         header = TRUE,
         sep = ",")
  Qian_merged_interest_md <- Qian_merged_interest@meta.data
```
```

```
Qian_merged_interest_md$X <- rownames(Qian_merged_interest_md)

Qian_merged_interest_md_BITFAM <- merge(Qian_merged_interest_md,

                        BITFAM,

                        by = "X")

Qian_merged_interest_md_BITFAM_cancer <- filter(Qian_merged_interest_md_BITFAM, CellType ==
"Cancer")

BITFAM <- t(Qian_merged_interest_md_BITFAM_cancer)

BITFAM <- as.data.frame(BITFAM[-c(1:18),])

BITFAM_up <- as.data.frame(sapply(BITFAM, as.numeric))

BITFAM_up$BITFAM <- rowMeans(BITFAM_up, na.rm = TRUE)

BITFAM_up$TF <- rownames(BITFAM)

BITFAM_up <- BITFAM_up[,-c(1:2)]

BITFAM_up <- filter(BITFAM_up, BITFAM > 0)


Dorothea <- read.csv("/mctp/share/users/gondal/Classes/BIOINF576/Dorothea_activities_res_01.csv",

    header = TRUE,

    sep = ",")

Qian_merged_interest_md_Dorothea <- merge(Qian_merged_interest_md,

                        Dorothea,

                        by = "X")

Qian_merged_interest_md_Dorothea_cancer <- filter(Qian_merged_interest_md_Dorothea, CellType
== "Cancer")

Dorothea <- t(Qian_merged_interest_md_Dorothea_cancer)

Dorothea <- as.data.frame(Dorothea[-c(1:18),])

Dorothea_up <- as.data.frame(sapply(Dorothea, as.numeric))

Dorothea_up$Dorothea <- rowMeans(Dorothea_up, na.rm = TRUE)

Dorothea_up$TF <- rownames(Dorothea)

Dorothea_up <- Dorothea_up[,-c(1:20)]

Dorothea_up <- filter(Dorothea_up, Dorothea > 0)
```

```
 ####
 SCENIC_up_TF <- SCENIC_up$TF
 Dorothea_up_TF <- Dorothea_up$TF
 BITFAM_up_TF <- BITFAM_up$TF

 xrepressors <- list(
 SCENIC = SCENIC_up_TF,
 Dorothea = Dorothea_up_TF,
 BITFAM = BITFAM_up_TF
 )

 common <- intersect(intersect(SCENIC_up_TF,
                 Dorothea_up_TF),
                 BITFAM_up_TF)

venn <- Venn(xrepressors)
data <- process_data(venn)

ggVennDiagram(xrepressors, label_alpha = 0, color = 1, lwd = 0.2, lty = 1 , category.names = c("SCENIC",
                          "Dorothea",
                          "BITFAM"))+ scale_fill_gradient(low = "#fafafa", high = "#fafafa") + theme(legend.title
= element_text(size=12))+
 theme(legend.text = element_text(size=12)) + theme(text = element_text(size = 15)) +
 geom_sf(size = 1,  color = "black", data = venn_setedge(data), show.legend = F)  +
theme(legend.position = "none")

ggsave("/mctp/share/users/gondal/Classes/BIOINF576/common6.png", width =8, height=6)
}
```

```
```