User guide

All R codes to reproduce the entire analysis of characterization the non-response to drugs for LN.

Article: Immune and molecular landscape behind non-response to Mycophenolate Mofetil and Azathioprine in lupus nephritis therapy.

Authors: Raúl López-Domínguez, Juan Antonio Villatoro-García, Concepción Marañón, Daniel Goldman, Michelle Petri, Pedro Carmona-Sáez, Marta E. Alarcón-Riquelme and Daniel Toro-Domínguez.

Correspondence: Daniel Toro-Domínguez; email: danieltorodominguez@gmail.com, Daniel.toro@genyo.es; Phone: +34 958715500-143.

Installing basic requirements

All analyses have been performed using R (R version 4.3.1) and tested in 2 different operating systems (Windows 10 Pro, version 22H2 and Ubuntu 20.04).

Install R and R Studio

For Windows, download installers from: https://cran.r-project.org/bin/windows/base/ and y https://posit.co/download/rstudio-desktop/ and follow installation steps.

For Ubuntu, follow the following instruction:

- R

  Open a terminal. Before installing R, we need to update the system package list.

  ```
  $ sudo apt-get update
  ```

  Install the dependencies necessAary to add a new repository over HTTPS

  ```
  $ sudo apt install dirmngr gnupg apt-transport-https ca-certificates software-properties-common
  ```

  Add the CRAN repository to your system sources' list

```
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
E298A3A825C0D65DFD57CBB651716619E084DAB9
$ sudo add-apt-repository 'deb https://cloud.r-project.org/bin/linux/ubuntu focal-
cran40/'
```

Install R by typing

```
$ sudo apt install r-base
```

- R Studio

  Download R Studio installer from the official site (https://posit.co/downloads/), choose the version according to the characteristics of your device and download the file. Then, open a terminal, go to the folder where the downloaded file is located and run:

  ```
  $ sudo dpkg -i rstudio-version-amd64.deb
  ```

  Note: the name of the file (.deb) can change based on the version downloaded

Clone GitHub folder locally

Access the GitHub link: https://github.com/dtordom/LNtherapy and clone the repository locally. To do this, click on the "code" button and then on "Download ZIP" (Fig. 1). Finally, unzip the files to the local path you want. A folder called "R" should appear among the downloaded files, which will contain all the necessary codes.

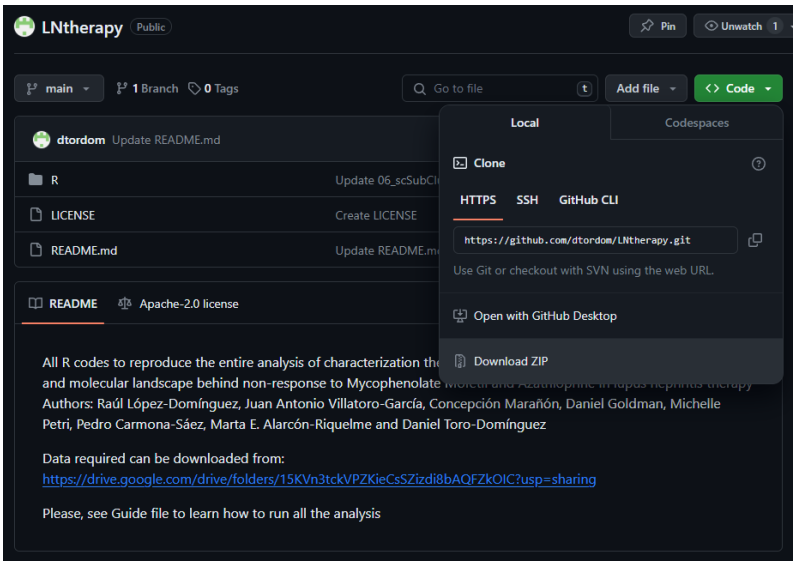

Figure 1: Clone GitHub folder in a local folder

Download data

Data required can be downloaded from:
https://drive.google.com/drive/folders/15KVn3tckVPZKieCsSZizdi8bAQFZkOIC?usp=sharing

Download and store the files in a local folder. The files are:

Dataset.RData: R environment containing the expression matrix "*data*" (genes in rows and patients in columans) and "*metadata*" with clinical and demographical information of patients (patients in rows and clinical variables in columns). These data are also public at NCBI GEO (ID: GSE224705): https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE224705.

Dataset_affy.RData: R environment containing Dataset.RData files and the raw expression matriz "*affy*" (probesets (from Affymetrix HT HG-U133+ PM Array Plate) in rows and patients in columns).

ClinicalData.RData: R environment containing matrices with different clinical variables for each patient/sample, including disease activity indexes, serological measurements, treatments and demographical data.

sysdata.rda: List of functional databases with biological functions and associated genes.

MMF_resp2.rds: List with gene expression matrix and metadata with response/non-response information based on protein/creatinine ratio in urine. This file is used only to compare results obtained using different response metrics.

cibersort.R: Code with R functions used to impute cell percentages based on gene expression data.

LM22.txt: Gene expression matrix from different blood cell types used as the reference mixture file for Cibersort analysis (imputation of cell percentage based on gene expression data).

Install required R packages

In this section, you can find all R packages used and versions. Open R Studio and run the following code:

```
## Chech installed packages
check.packages <- function(pkg){
  new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
  if (length(new.pkg)>0){
    print(paste0("Installing ", paste(new.pkg,collapse = ", ")))
    if (!requireNamespace("BiocManager", quietly = TRUE)){
      install.packages("BiocManager")
    }
    bioconductor_packages <- BiocManager::available()
    bioconductor_packages <- bioconductor_packages[bioconductor_packages
%in% new.pkg]
    if (length(bioconductor_packages) > 0){
      BiocManager::install(bioconductor_packages, dependencies =
TRUE,ask=FALSE)
    }
    new.pkg = new.pkg[!(new.pkg %in% bioconductor_packages)]
    if (length(new.pkg)>0){
      install.packages(new.pkg, dependencies = TRUE)
    }
  }
  res <- lapply(pkg,load.packages)
}

## Load packages
load.packages <- function(pkg){
  for(packages in pkg){
    cat(paste0("\nLoading ",packages,"..."))
    suppressMessages(require(packages,character.only = T))
  }
}

# Install and load packages from Cran and Bioconductor
check.packages(c("Matrix","dplyr","pheatmap","e1071","parallel","tidyvers
e","preprocessCore","fgsea","ggplot2","stringr","UpSetR","ggrepel","ggpub
r","cowplot","ComplexHeatmap","qusage","tmod","stringi","doParallel","Seu
rat","rafalib","biomaRt","optparse","utils","matrixStats","patchwork","Si
ngleCellExperiment","scales","batchelor","clustree","RColorBrewer","vegan
","ineq","igraph","sva","scran","scater","fields","data.table","scDblFind
er","harmony","ggsci","tidyr","tibble","reshape","NMF","future","hipathia
","limma","caret","devtools"))

## Some packages must be installed from GitHub
devtools::install_github("sqjin/CellChat")
```

```
library("CellChat")
devtools::install_github("jordimartorell/pathMED")
library("pathMED")
devtools:: install_version("dbplyr", version = "2.3.4")
library("dbplyr")

## This process can take some time (2 hours if all packages and
dependencies must be installed)
```

Once the data is downloaded and all the R packages are installed, we will only need to execute the codes contained in the "R" folder to carry out the different analyzes described in the article, in order, as we will explain below.

Package versions:

Matrix (1.6-4), dplyr (1.1.4), pheatmap (1.0.12), e1071 (1.7-14), parallel (4.3.1), tidyverse (2.0.0), preprocessCore (1.62.1), fgsea (1.26.0), ggplot2 (3.4.4), stringr (1.5.1), UpSetR (1.4.0), ggrepel (0.9.4), ggpubr (0.6.0), cowplot (1.1.2), ComplexHeatmap (2.16.0), qusage (2.34.0), tmod (0.50.13), stringi (1.8.3), doParallel (1.0.17), Seurat (5.0.1), rafalib (1.0.0), biomaRt (2.56.1), optparse (1.7.3), utils (4.3.1), matrixStats (1.2.0), patchwork (1.1.3), SingleCellExperiment (1.22.0), scales (1.3.0), batchelor (1.16.0), clustree (0.5.1), RColorBrewer (1.1-3), vegan (2.6-4), ineq (0.2-13), igraph (1.6.0), sva (3.48.0), scran (1.28.2), scater (1.28.0), fields (15.2), data.table (1.14.10), scDblFinder (1.14.0), harmony (1.2.0), ggsci (3.0.0), tidyr (1.3.0), tibble (3.2.1) reshape (0.8.9), NMF (0.26), future (1.33.1), hipathia (3.0.2), limma (3.56.2), caret (6.0-94), devtools (2.4.5), Cellchat (1.6.1), pathMED (0.1.23)

01_GetDEGs.R

The first step, as described in the article, is to perform differential expression analysis between responders and non-responders to each drug (mycophenolate mofetil (MMF), azathioprine (AZA), standard of care drugs (SOC) and hydroxychloroquine (HC)). That analysis is performed using the code *01_GetDEGs.R*. Open the code in R Studio and run the different sections to obtain the results. Pay special attention to the comments of the code, they explain what each part does, and they also indicate where modifications should be made (for example, the paths where the

files used are located must be changed). Below we will briefly describe the different sections of the code:

Step 1: For each drug, samples are selected based on treatment used and differentially expressed genes (DEGs) were obtained using the function *limma.DEG*.
DEGS.RData object is the main output from this part, an R environment contains gene signatures obtained for each drug (list of up and down DEGs for each drug), gene-expression matrices and metadata required for future analyses.

Step 2: From DEGs obtained, upset plot, volcano plots, heatmap and boxplots were obtained

Step 3: Up and down DEGs are compared between drugs using Gene Set Enrichment Analysis (GSEA). DEGs for a drug (comparing response and non-response) are sorted into the ranked list of genes (by fold change) of other drug (comparing response and non-response). If up-DEGs of a drug are in the top and the down-DEGs are in the bottom of the ranked list of genes of the other drug, a positive similarity score is obtained (and a negative similarity score is obtained in the opposite case).
GSEA was also used to compare DEGs obtained using 2 different metrics to measure drug response.

Step 4: Functional analysis were performed comparing responder and non-responder samples using qusage R package and a database of immunological functions (obtained from tmod R package).

02_MLtop10.R

This code was used to built machine learning based model to predict response or non-response to each drug using the expression of top 10 genes most differentially expressed.

Step 1: Load gene-expression data of patients, DEGs information for each drug and clinical information (response or non-response to the drugs).

Step 2: A 5-nested 4-fold cross validation repeated 10 times was performed to predict response/ non response to each drug using *getML* R function from pathMED package. More info about *getML* function at: https://github.com/jordimartorell/pathMED.

03_Cells.R

With this code, cell percentages were imputed using cibersort software (source codes).

Step 1: A blood cell mixture file (LM22.txt) was used to impute cell percentage for each sample from gene-expression data using *CIBERSORT* function.

Step 2: Comparison of cell proportions between responder and non-responder samples to each drug using Mann-Whitney-Wilcoxon test. Boxplots were obtained for significant results.

Step 3: In this section, patients were labeled as poor or rich for each cell population and proportion tests were performed comparing label proportions between responder and non-responder patients.

04_ClinicalAnalysis.R

Here, statistical analyses were performed comparing different clinical variables between responder and non-responder patients using *GetStats* R function (from utils.R). Clinical variables were analyzed by sample (variables that can change between visits in the same patient) or by patient (the variable is always the same for the same patient). Clinical variables can be numerical or categorical. These parameters can be controlled by *GetStats* function.

The Wilcoxon Mann-Whitney and Fisher's exact tests were used to identify significant associations between response/non-response in continuous and categorical clinical variables, respectively.

05_scRNASeq_mainCells.R

This script contains all steps for single cell data analysis, from raw data to cluster cells. First, parameter values are set (using *opt* object to store all parameters using during the analysis).

Step 1: Raw data (*cellranger* outputs) are loaded into R using *Seurat* R package.

Step 2: Different quality controls are applied to the data, including quality control by gene and by sample (sample filtering by percentage of mitochondrial and ribosomal genes, by diversity measurements, number of counts and UMIT, number of protein coding genes and doublets per sample. Mitochondrial, ribosomal and non-protein coding genes were also removed). Finally, data was log normalized.

Step 3: All samples were integrated using *harmony* R package expression data was corrected by cell cycle states.

Step 4: Clustering resolution calculation and cluster the cells using base functions from *Seurat* R package.

Step 5: Functional annotation of clusters based on gene-expression of cell types markers previously defined in the literature.

Step 6: Save data from each cluster into a separate file.

06_scSubClustering.R

This script is used for each cell cluster (clusters of major cell types) in order to subdivide cells from a specific cell type into subclusters and increase granularity of results.

Step 1: Load single cell data of a specific cluster previously defined and run a second clustering process using. The resolution values were manually selected based on cluster stabilities (see plot obtained using *clustree* R function).

Step 2: Get DEGs between clusters using *FindAllMarker*s function (from Seurat R package, and contained into *getDEG* function (utils.R).

Step 3: *AddModuleScore* function from Seurat was used to get gene expression scores for different gene signatures (including up and down-DEGs to each drug, and/or specific markers and functional associated genes)

07_CellCellComm.R

In this script, gene expression data and subcluster labs for cells from single cell data was loaded and then, CellChat R package was used to impute communication network between cluster of cells. Chord plots was generated using *netVisual_circle* R function. The output of this section includes a list of regulatory networks between cell clusters related to the non-response to each drug and their target genes.

08_ApplyHipathia.R

This script use hipathia R package to infer transcriptional changes driven by gene-expression modifications (i.e. inhibition of the expression of drug targets) using mechanistic models.

Step1: Gene expression data from patients and healthy controls, clinical information (including the information of response /non-response to each drug) and the output of 07_CellCellCom.

Step2: For each drug (MMF, AZA, SOC and HC), a matrix was created including healthy samples and samples after and before target inhibition of genes from each regulatory network obtained in 07_CellCellCom. Then, we imputed circuits activity using *hipathia* R function at sample level.

Step3: Finally, a responsiveness score was calculated for each patient for the inhibition of each regulatory network comparing circuit activity after and before target inhibitions. Patients were labeled as good or bad responders to the inhibition of each regulatory network.

utils.R

This code contains different R functions to be used by the other codes.

limma.DEG: Function to apply linear models using *limma.*

orderHeatmap: Function to order samples based on gene-expression similarities.

GetStats: Function to perform clinical association analysis.

markerDots: Function to plot expression of markers in single cell clusters.

getDEG: Function to obtain DEG between clusters of cells (single cell data).