L1KProcs: a pipeline for L1K data preprocessing and compound signature discovery

Chenglin Liu[1,2], Jing Su[1], Jinwen Ma[2], and Xiaobo Zhou[1]

November 17, 2013

[1]Department of Diagnostic Radiology, Wake Forest School of Medicine, Winston-Salem, NC, 27157, USA
[2]Department of Information Science, School of Mathematical Sciences and LMAM, Peking University, Beijing 100871, China

xizhou@wakehealth.edu

Contents

1 Introduction

1.1 Background

L1000 platform is a new gene expression assay for 1000-plex transcription response measurement proposed by Broad LINCS Center. The Library of Integrated Network-Based Cellular Signatures (LINCS) program (http://www.lincscloud.org/) generated millions of gene expression profiles related to thousands of compounds and genetic perturbagens from tens of cell lines using L1000 platform. Currently, no 'offical' are proposed for the preprocessing and analysis of L1000 data.

L1KProcs is a `R/Bioconductor` package which can systematically preprocess and analysis the L1000 data. The functions include a L1000 data preprocessing pipeline using GMM peak calling method, the qualitication of compounds and genes connectivities using EGEM matrix, and the compound signature discovery using csNMF method. Details can be referred in **?**.

1.2 Installation

The *L1KProcs* depends on several packages including *preprocessCore*, *stats* and *prada* for preprocessing, *SeqGSEA* for EGEM score computing, *gplots* for visualization, and *doParallel* for parallel computing. The package also requires the annotation package *L1KAnno*.Please make sure these packages are properly installed before installing L1KProcs.

2 L1000 preprocessing pipeline

2.1 The *PlateInfo* class

The information of processed raw data are stored to different *PlateInfo* classes according to different plate names. The recored information includes the name of the plate, location of the output files, the target used for quantile normalization, well qualities, and control wells. The preprocessed data by `l1kpreprocs` are stored to a list of *PlateInfo* classes. Type `?PlateInfo` for details.

2.2 Preprocessing pipeline with multiple cores

The preprocessing procedure includes five steps: Guassian mixture peak calling method, gene expresssion quantile normalization, plate based and well based quality control, log fold change between treatment wells and control wells, and the converting from landmark gene expression to whole-genome expression. The pipeline uses *doParallel* for parallel computing.

Only `datapath` as the locations of raw data is required for the pipeline `l1kpreprocs`. Either the top folder containing the raw data or a file listing the detailed path of each raw data works. To demonstrate the functionality of `l1kpreprocs`, we use it to preprocess the raw data from three pairs of duplicated wells using 6 cores. The processed data are of .gct format **?** located in folder 'l1kdata'.

```
> library(L1KProcs)
> datapath <- system.file("test_data",package="L1KProcs")
> outpath <- "l1kdata"
> lstPlateInfo <- l1kpreprocs(datapath,outpath,target=FALSE,
+                 ifAll=FALSE,nthread=6,plot=FALSE)
> show(lstPlateInfo[[1]])

PlateName: CPC001_PC3_24H_X1_B3_DUO52HI53LO
Location of expression files: l1kdata
Control Wells: BO, PCRNEG, DMSO, DMSO, ...
Major control: DMSO
Duplicates: CPC001_PC3_24H_X1_B3_DUO52HI53LO, CPC001_PC3_24H_X2_B3_DUO52HI53LO
Threshold: Plate: 0.5 Well: 0.6
Source of target for normalization: default
Number of bad analytes: 0, 0, 0
Correlation between controls: 1, 0.962039262923282, 0.962039262923282, 1
Correlation between wells with its duplicate: FALSE, TRUE, TRUE
If pass quality control: Yes
Good wells: CPC001_PC3_24H_X1_B3_DUO52HI53LO_A17, CPC001_PC3_24H_X1_B3_DUO52HI53LO_B17
```

2.3 Data quality visualization

During the preprocessing process, the function `QualityControl` fulfills the quality control function. The qualities of each of the wells are stored in the return value of `l1kpreprocs`. They can be visualized by using this return value to `PlatePlot` and `WellPlot`. One example is shown in Figure 1.

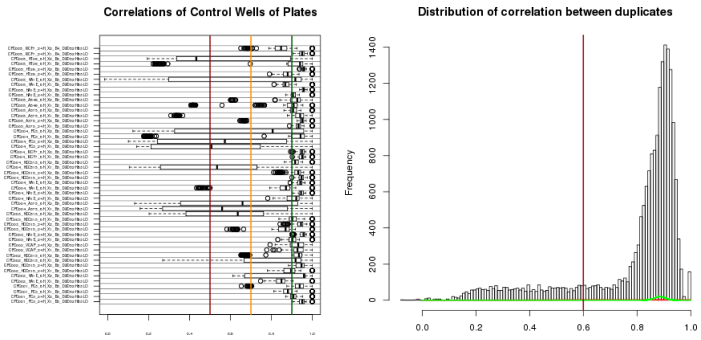

Figure 1: Left: correlations of control wells of Plates. Plate with low correlations fail the quality control (default: 0.5). Right: distribution of correlations between duplicate wells. Wells with low correlaton with their duplicates fail the quality control (default: 0.6).

```
> library(L1KProcs)
> load(system.file("test_data","lstPlateInfo.rda", package="L1KProcs"))
> outfile <- file.path("CorPlates.png")
> PlatePlot(lstPlateInfo,outfile)
> outfile <- file.path("CorWells.png")
> WellPlot(lstPlateInfo,outfile)
```

2.4 The plotting of peak calling results

Each well profiles 1000 genes by 500 analytes. The peak calling method is to distinguish each of the two genes of the analyte. To further investigate the deconvolution performance, these peak calling results can be plotted. You can either plot all 500 analytes or specify the interesting ones using `analyteID`. The return value is the number of poor quality analytes without enough supporting bead sizes. The plot of one analyte is shown in Figure 2.

```
> library(L1KProcs)
> wellname <- "CPC001_PC3_24H_X1_B3_DUO52HI53LO_A01"
> filename <- file.path(system.file("test_data",package="L1KProcs"),paste(wellname,"lxb",sep="."))
> dPeak(outpath="l1kdata",filename,wellname,plot=TRUE,analyteID=c(11,393))

[1] 0
```

3 EGEM matrix: the relations of compounds and genes

Enrichment of gene effects to the molecule (EGEM) score is to describing the relations of compounds and genes with the view of their perturbagens to the gene expresion pattern alteration. Gene expression data after compounds and gene knockdown perturbagens from a same cell line is required for EGEM score computing. You can use knockdown gene expression data of the LINCS library. The heatmap of EGEM matrix and the distribution of all EGEM scores can be obtained by `egem.plot`, as shown in Figure 3. Interesting genes those have similar gene expression effects as the compounds are detected by `egem.analyze`.

An example based on LINCS library is as follows:

```
> library(L1KProcs)
> load(system.file("test_data","cpdata.rda",package="L1KProcs"))
> egem.info <- egem(cpdata,nthread=8,LINCS=TRUE,lib.name="HA1E_96H")
> names(egem.info)
```

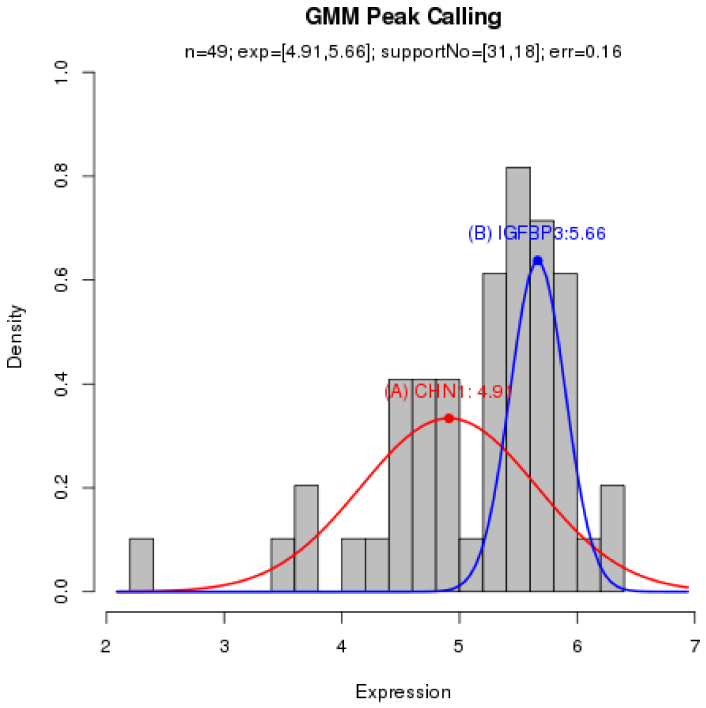

Figure 2: Peak calling of one analyte from the test well. The red and blue curves indicate the distributions of Gene H (gene with larger bead size) and Gene L (gene with smaller bead size). The gene names, expression levels and supported bead number are listed top of the figure. Err indicates the quality of the peak calling. The small, the better.

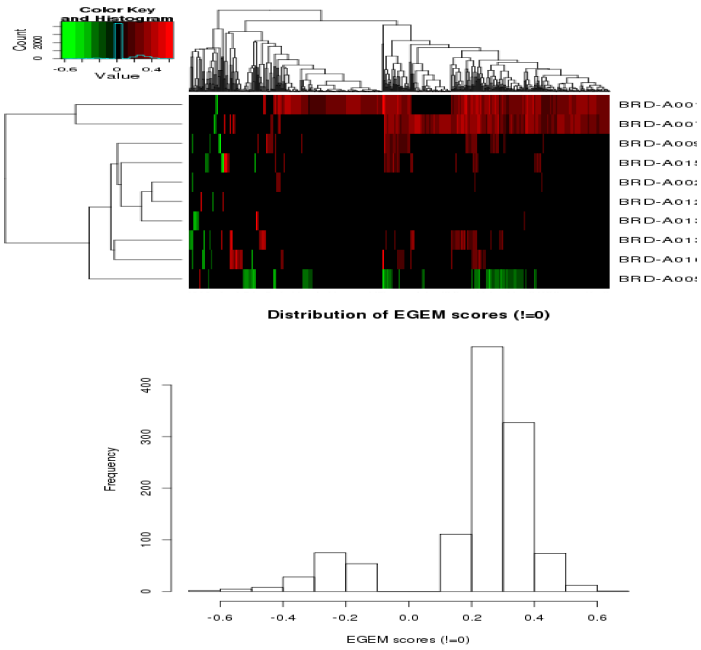

Figure 3: Top: the heatmap of EGEM matrix. Rows are genes, colunm are compounds. Red regions indicate similar effects between genes and compounds, while green regions mean reverse. Bottom: the histplot of non-zero EGEM scores.

```
[1] "egem"        "origin.EGEM" "p.value"     "mean.perm"   "sd.perm"
[6] "DEGs"        "CNames"      "PNames"
```

```
> sgenes <- egem.analyze(egem.info$egem,th=0.05,print=FALSE)
> str(sgenes[[1]])
```

```
List of 2
 $ same   : chr [1:62] "TRCN0000019130" "TRCN0000019267" "TRCN0000019320" "TRCN0000019896" ...
 $ reverse: chr [1:4] "TRCN0000003176" "TRCN0000004986" "TRCN0000060676" "TRCN0000359421"
```

```
> egem.plot(egem.info$egem)
```

4 Compound signature discovery

After constructing EGEM matrix, compounds with similar related genes can be detected using csNMF. These compounds have demonstrated similar functions or behaviours, we call, signatures. csNMF method cannot determine the number of signatures aumomatedly. Hence, in order to detect the compound signatures, we need to provide a list of protential numbers to `pNo` (default: 5-20). Enough `repeatNo` guarentees the stability of the detection (default: 30). The return value includes the genes and compounds of each signature, potential related transcription factors. A heatmap displaying the signature is also provided. An example is shown as follows, with the heatmap shown in Figure 4.

```
> library(L1KProcs)
> data("libEGEM",package="L1KAnno")
> egem.info <- libEGEM[["MCF7_CP_24H_KD_96H"]]
> egem <- egem.info[["egem"]][1:50,1:50]
```

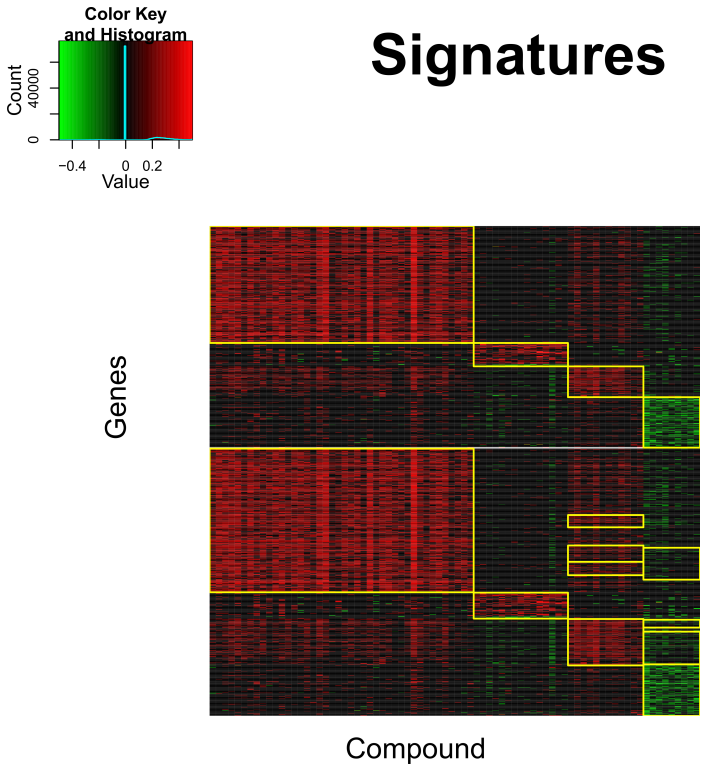

Figure 4: Compounds of a same signature are highlighted using the yellow rectangular. Red regions are genes of similar gene expression effects as compounds, while green regions mean the reverse.

```
> PNames <- egem.info[["PNames"]][1:50]
> CNames <- egem.info[["CNames"]][1:50]
> pNo <- c(3:4)
> repeatNo <- 30
> var <- SigDetect(egem.info,outpath="csNMFresult",pNo=pNo,
+ repeatNo=repeatNo,nthread=16,keep=FALSE,print=FALSE,TF=FALSE)
> names(var)

[1] "k"          "SigCNames"  "SigPNames1" "SigPNames2"
```

5 Session information

```
>   sessionInfo()

R version 3.0.0 (2013-04-03)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
```

```
attached base packages:
[1] grid      parallel  stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
 [1] L1KProcs_0.99      doParallel_1.0.3   iterators_1.0.6    gplots_2.11.3
 [5] MASS_7.3-29        KernSmooth_2.23-10 caTools_1.14       gdata_2.13.2
 [9] gtools_3.1.0       SeqGSEA_1.0.2      biomaRt_2.16.0     DESeq_1.12.1
[13] lattice_0.20-23    locfit_1.5-9.1     Biobase_2.20.1     BiocGenerics_0.6.0
[17] foreach_1.4.1

loaded via a namespace (and not attached):
 [1] annotate_1.38.0       AnnotationDbi_1.22.6  bitops_1.0-6
 [4] codetools_0.2-8       compiler_3.0.0        DBI_0.2-7
 [7] genefilter_1.42.0     geneplotter_1.38.0    IRanges_1.18.4
[10] mvtnorm_0.9-9996      pcaPP_1.9-49          prada_1.36.1
[13] preprocessCore_1.22.0 RColorBrewer_1.0-5    RCurl_1.95-4.1
[16] robustbase_0.9-10     rrcov_1.3-4           RSQLite_0.11.4
[19] splines_3.0.0         stats4_3.0.0          survival_2.37-4
[22] tools_3.0.0           XML_3.98-1.1          xtable_1.7-1
```