RAISIN: Regression Analysis in Single-cell RNA-Seq with multiple samples

Zhicheng Ji, Department of Biostatistics and Bioinformatics, Duke University School of Medicine
Wenpin Hou, Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health
Hongkai Ji, Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health

Introductions

RAISIN is a software tool to perform Regression Analysis In SINgle-cell RNA-seq datasets with multiple samples. RAISIN performs DE analysis using a flexible mixed effects regression framework that accounts for both sample-level and cell- level variances (Fig. 1a). The classical linear mixed effects model (LMM) do not consider small sample size or small cell number in rare cell populations, which are common in scRNA-seq studies and can lead to poor variance estimation and reduced statistical power. Fitting mixed effects models to large datasets consisting of many samples and millions of cells is also computationally challenging. To address these issues, RAISIN combines the mixed model with a hierarchical model to regularize variances, and a new model fitting algorithm is developed to efficiently handle large datasets. RAISIN can be applied to identify differential cell type proportions, differential gene expression between cell types, and differential gene expression between sample groups within each cell type.

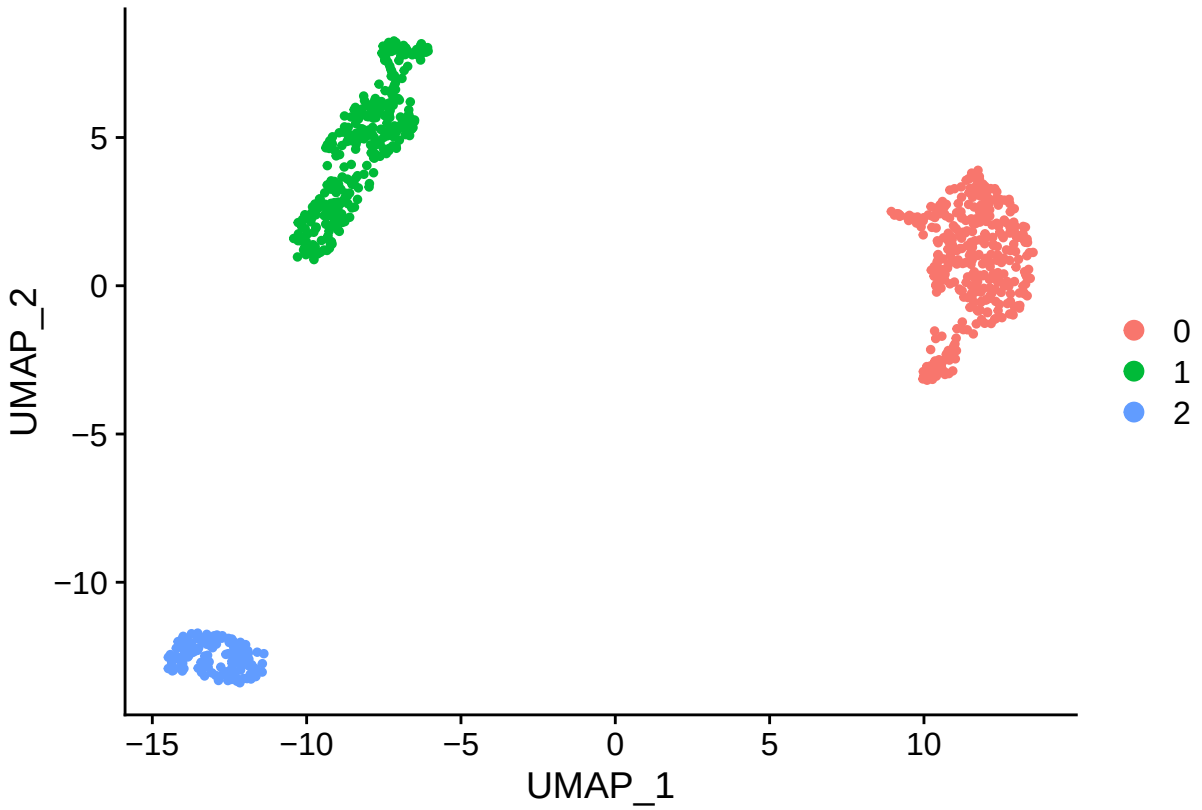
Data loading and exploration

We demonstrate the functions of RAISIN using a small dataset, which is a subset of a single-cell RNA-seq dataset measuring the gene expression profiles of PBMC cells in 2 COVID-19 patients and 2 healthy donors collected from Lee et al. Science Immunology, 2020. The dataset is in the format of a Seurat object. We first read in the dataset and visualize the cell clusters on the umap.

```
library(raisin)
library(Seurat)
```

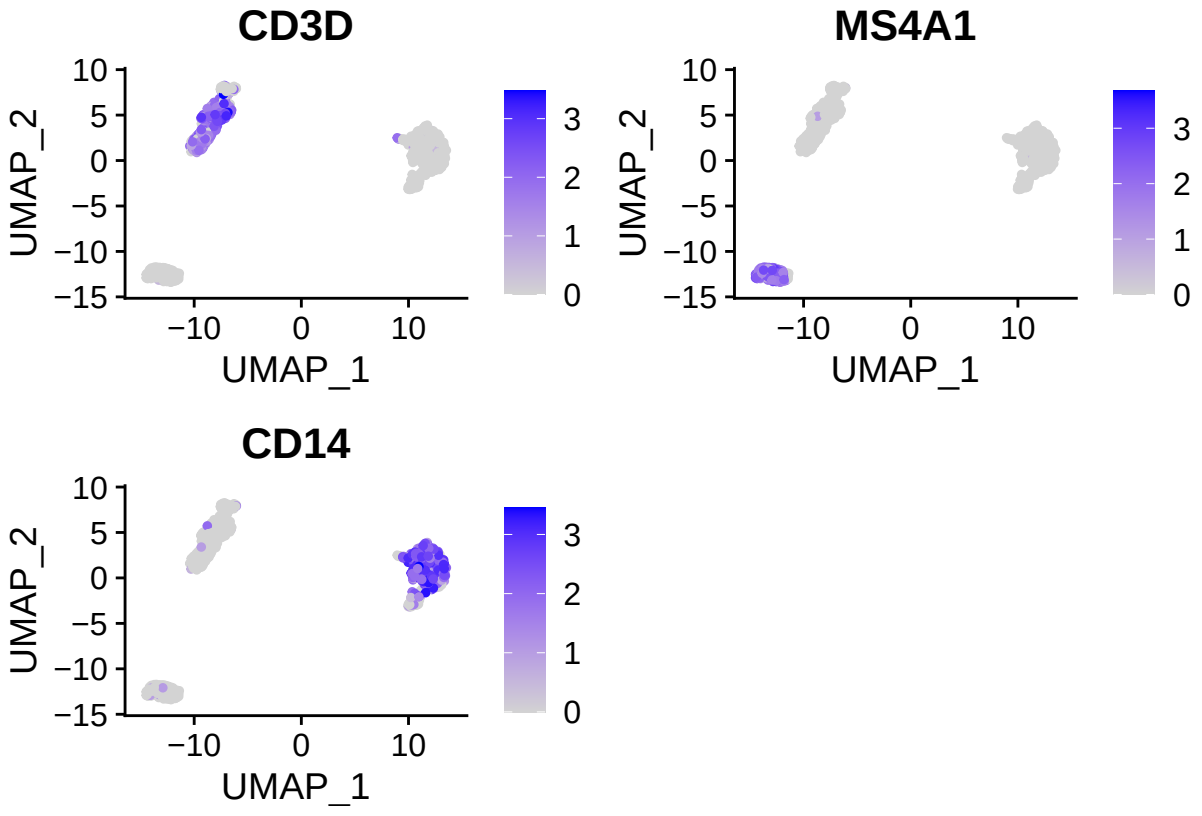
```
## Attaching SeuratObject
```

```
# read in the seurat object
d <- readRDS(paste0(system.file('data',package = 'raisin'),'/exampledata.rds'))
DimPlot(d)
```



There are three clusters. To identify the potential cell types, we visualize the expression of marker genes

```
FeaturePlot(d,c('CD3D','MS4A1','CD14'))
```



Cluster 0 has high expression in CD14 and likely consists of monocytes. Cluster 1 has high expression in CD3D and likely consists of T cells. Cluster 2 has high expression in MS4A1 and likely consists of B cells.

We used the following code to extract the log normalized gene expression matrix, the cell clustering assignments, the individual information from which cells are collected, and the umap coordinates. We paste the cluster and individual together to form a 'sample' variable, which indicates a group of cell with same cell cluster and from the same individual. Note that if imputed gene expression matrix (e.g., imputed by SAVER) is available, it is recommended to be used.

```
mat <- d$RNA@data
cluster <- Idents(d)
individual <- sub('_.*','',colnames(mat))
sample <- paste0(cluster,'_',individual)
umap <- d$umap@cell.embeddings
```

Here H1 and H2 are two healthy donors, and S1 and S2 are two COVID-19 patients with severe symptoms.

Cell type proportion test

The first analysis is to test if different groups of individuals have enriched cell types. For example, if COVID-19 patients have a higher proportion of certain cell types. The test needs the cluster of each cell (cluster), the individual from which each cell is collected (individual), and a design matrix (design). The design matrix should be a numeric matrix with row names being the unique values of individuals. The 'proportiontest' function is used to perform the test.

```
# make the design
design <- data.frame(intercept=1,contrast=c(0,0,1,1),row.names = c('H1','H2','S1','S2'))
# run the test
proportiontest(cluster,individual,design)
```

```
##        logFC   P.Value adj.P.Val
## 1 -1.1461406 0.1571740 0.3482658
## 2  0.8818376 0.2600600 0.3482658
## 0  0.7214135 0.3482658 0.3482658
```

The result of 'proportiontest' is a data.frame. Each row is a cluster. Although none of the cluster shows significant results most probably due to small sample size, there is a trend of enriched B cells and monocytes in COVID patients and enriched T cells in healthy donors.

Differential genes between two cell types

The second analysis is to identify differential genes between two cell types, cell clusters, or groups of cells using a paired differential test. Here we take the cluster 0 and 1 as example and identify the differential genes between the two clusters. We use the paired test since each individual has cells in both clusters. The input is the gene expression matrix, a vector indicating which sample (here is a combination of cell type and individual) each cell comes from, and a design. The design has three columns: sample, individual, and feature.

```
# subset the cells in cluster 0 and 1
cellid <- which(cluster %in% c(0,1))
# make the design matrix.
design <- data.frame(sample=unique(sample[cellid]),individual=sub('.*_','',unique(sample[cellid])),featu
# run the test
head(raisintest(raisinfit(mat[,cellid],sample[cellid],testtype = 'paired',design=design)))
```

```
## [1] "Estimating sigma2 for group: difference"
## [1] "alpha=0.124171673403845 beta=7.92618046238372"
## [1] "Estimating sigma2 for group: individual"
```

```
## [1] "alpha=0.0813907185886269 beta=3.66724040060736"
```

```
##          Foldchange      stat       pvalue         FDR
## CTSS      -2.692201 -33.28677 5.789824e-07 0.002065418
## IL32       2.427594  28.63112 1.207482e-06 0.002065418
## BRI3      -1.816379 -27.14867 1.564639e-06 0.002065418
## SERPINA1  -2.028666 -26.48989 1.763545e-06 0.002065418
## TYROBP    -2.387768 -25.39427 2.166299e-06 0.002065418
## LYZ       -3.922438 -25.13292 2.278189e-06 0.002065418
```

The output is a data.frame. The columns are (log) fold change, statistics, p-value and FDR. Each row is a gene ordered by its p-value.

One common task is to identify the marker genes in each cluster. This can be easily done using the function 'findmarker'. It will iteratively go through each cluster, and compare the cells within and outside the cluster.

```
# run the findmarker function
marker <- findmarker(as.matrix(mat),individual,cluster)
```

```
## [1] "Estimating sigma2 for group: difference"
## [1] "alpha=0.1433926579854 beta=7.83112414893939"
## [1] "Estimating sigma2 for group: individual"
## [1] "alpha=0.119743085645786 beta=7.52266947938212"
## [1] "Estimating sigma2 for group: difference"
## [1] "alpha=0.169827623697361 beta=9.56286846424127"
## [1] "Estimating sigma2 for group: individual"
## [1] "alpha=0.125917844549007 beta=5.60993998689675"
## [1] "Estimating sigma2 for group: difference"
## [1] "alpha=0.178660789633262 beta=8.32719161996407"
## [1] "Estimating sigma2 for group: individual"
## [1] "alpha=0.113134490321299 beta=3.20741938289453"
```

```
# marker for the first cluster '0'
names(marker)
```

```
## [1] "0" "1" "2"
```

```
head(marker[[1]])
```

```
##          Outsidecluster Withincluster Foldchange     stat       pvalue
## FCER1G       0.33433551      2.865837   2.531501 35.23311 5.559539e-07
## SERPINA1     0.02818681      2.070770   2.042583 30.10240 1.179920e-06
## CTSS         0.71146982      3.310028   2.598559 29.45905 1.308221e-06
## BRI3         0.57910039      2.356499   1.777399 29.44471 1.311267e-06
## CST3         0.12694963      3.142645   3.015695 29.41493 1.317622e-06
## TYROBP       0.41655644      2.998465   2.581908 26.94410 2.003318e-06
##                   FDR
## FCER1G   0.001444377
## SERPINA1 0.001444377
## CTSS     0.001444377
## BRI3     0.001444377
## CST3     0.001444377
## TYROBP   0.001762385
```

The result is a list. Each element corresponds to a cluster. Genes are ordered based on the log fold change.

Differential genes between two groups of samples within each cluster

The third analysis is to identify differential genes within each cell cluster between two groups of samples, such as COVID patients and healthy donors. Here we want to identify the differential genes between COVID and healthy in cluster 0. We use unpaired test in this case. The input is similar as before, with the difference that the column 'individual' is not included in the design.

```
cellid <- which(cluster == 0)
design <- data.frame(sample=unique(sample[cellid]),feature=sub('[1-9]$','',sub('.*_','',unique(sample[c
head(raisintest(raisinfit(mat[,cellid],sample[cellid],testtype = 'unpaired',design=design)))
```

```
## [1] "Estimating sigma2 for group: H"
## [1] "alpha=0.110759258232813 beta=33.1249639121366"
## [1] "Estimating sigma2 for group: S"
## [1] "alpha=0.0980989821833064 beta=4.98036189386644"
```

```
##          Foldchange       stat       pvalue          FDR
## XIST     -1.5379123 -14.271479 3.750829e-14 2.584696e-10
## RPS4Y1    1.3391218  10.106073 1.044621e-10 3.246573e-07
## HLA-DPB1 -1.5204409  -9.965671 1.413397e-10 3.246573e-07
## RPS26    -0.9769758  -9.060760 1.054498e-09 1.816636e-06
## TXNIP    -1.1135435  -8.731702 2.248930e-09 3.099475e-06
## CXCL8     1.3543388   8.264405 6.757345e-09 7.760810e-06
```

The output is similar as before.

Session Info

```
sessionInfo()
```

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Mojave 10.14.5
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] SeuratObject_4.0.0 Seurat_4.0.0       raisin_1.0
##
## loaded via a namespace (and not attached):
##   [1] nlme_3.1-148         matrixStats_0.57.0   RcppAnnoy_0.0.18
##   [4] RColorBrewer_1.1-2   httr_1.4.2           sctransform_0.3.2
##   [7] tools_4.0.2          utf8_1.2.1           R6_2.5.0
##  [10] irlba_2.3.3          rpart_4.1-15         KernSmooth_2.23-17
##  [13] uwot_0.1.10          mgcv_1.8-33          DBI_1.1.0
##  [16] lazyeval_0.2.2       colorspace_2.0-1     tidyselect_1.1.0
##  [19] gridExtra_2.3        compiler_4.0.2       plotly_4.9.2.1
##  [22] labeling_0.4.2       scales_1.1.1         spatstat.data_2.0-0
```

```
##  [25] lmtest_0.9-37        ggridges_0.5.2       pbapply_1.4-2
##  [28] goftest_1.2-2        spatstat_1.64-1      stringr_1.4.0
##  [31] digest_0.6.27        spatstat.utils_2.0-0 rmarkdown_2.7
##  [34] pkgconfig_2.0.3      htmltools_0.5.1.1    highr_0.8
##  [37] fastmap_1.0.1        limma_3.45.7         htmlwidgets_1.5.1
##  [40] rlang_0.4.11         shiny_1.5.0          farver_2.1.0
##  [43] generics_0.1.0       zoo_1.8-8            jsonlite_1.7.1
##  [46] ica_1.0-2            dplyr_1.0.2          magrittr_2.0.1
##  [49] patchwork_1.0.1      Matrix_1.2-18        Rcpp_1.0.5
##  [52] munsell_0.5.0        fansi_0.4.2          abind_1.4-5
##  [55] reticulate_1.16      lifecycle_1.0.0      stringi_1.5.3
##  [58] yaml_2.2.1           MASS_7.3-51.6        Rtsne_0.15
##  [61] plyr_1.8.6           grid_4.0.2           parallel_4.0.2
##  [64] listenv_0.8.0        promises_1.1.1       ggrepel_0.8.2
##  [67] crayon_1.4.1         deldir_0.2-10        miniUI_0.1.1.1
##  [70] lattice_0.20-41      cowplot_1.1.0        splines_4.0.2
##  [73] tensor_1.5           knitr_1.33           pillar_1.6.1
##  [76] igraph_1.2.5         future.apply_1.6.0   reshape2_1.4.4
##  [79] codetools_0.2-16     leiden_0.3.3         glue_1.4.2
##  [82] evaluate_0.14        data.table_1.13.2    png_0.1-7
##  [85] vctrs_0.3.8          httpuv_1.5.4         polyclip_1.10-0
##  [88] gtable_0.3.0         RANN_2.6.1           purrr_0.3.4
##  [91] tidyr_1.1.2          scattermore_0.7      future_1.17.0
##  [94] ggplot2_3.3.3        xfun_0.22            mime_0.9
##  [97] xtable_1.8-4         later_1.1.0.1        survival_3.1-12
## [100] viridisLite_0.4.0    tibble_3.1.2         cluster_2.1.0
## [103] globals_0.12.5       statmod_1.4.35       fitdistrplus_1.1-1
## [106] ellipsis_0.3.2       ROCR_1.0-11
```