ICAnet tutorial

Weixu Wang

ICAnet can run on both single-batch and multi-batch datasets. For each dataset, we run ICA independently. Further, we cluster all the independent components, find shared and specific expression programs, and combine with PPI to perform clustering. In this tutorial, I am gonna use cell line dataset provided by 10X Genomics as an example dataset to show how to use ICAnet to perform batch effect correction and clustering. The source datasets (including gene expression dataset, cell annotation, and PPI network) can be download from https://github.com/WWXkenmo/MouseGerm (https://github.com/WWXkenmo/MouseGerm).

We load the dataset at first, as shown below.

```
setwd("/mnt/data4/weixu/MCTA/MouseSperm")
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
library(Seurat)
library(cowplot)
```

```
##
## ********************************************************
```

```
## Note: As of version 1.0.0, cowplot does not change the
```

```
##   default ggplot2 theme anymore. To recover the previous
```

```
##   behavior, execute:
##   theme_set(theme_cowplot())
```

```
## ********************************************************
```

```
library(ica)
library(ICAnet)

PPI <- readRDS("PPI_for_cell_line.RDS")
all.data <- readRDS("cell_line_exp.RDS")
ID <- read.table("ID_for_cell_line.txt",header=TRUE,row.names=1,stringsAsFactors = FALSE)
```

Next, check the dimension for cell line dataset:

```
dim(all.data)
```

```
## [1] 16602  8783
```

Create Seurat Object

ICAnet requires user to save their scRNA-seq dataset into Seurat object. Meanwhile, the Seurat object also need to have ‘batch’ and ‘celltype’ ID. In this tutorial, the scRNA-seq dataset has been preprocessed.

```
head(ID)
```

	cell.type <chr>	**batch** <chr>
AAACATACACTGGT	293t cell	293t cell
AAACATACAGACTC	293t cell	293t cell
AAACATTGACCAAC	293t cell	293t cell
AAACATTGAGGCGA	293t cell	293t cell
AAACATTGGGACTT	293t cell	293t cell
AAACCGTGATTTCC	293t cell	293t cell

6 rows

```
all.data <- CreateSeuratObject(counts = all.data, project = "cell.all", min.cells = 0)
all.data$celltype <- ID$cell.type
all.data$batch <- ID$batch
all.data <- NormalizeData(all.data, normalization.method = "LogNormalize", scale.factor = 10000
)
```

We first run the original workflow using PCA to perform embedding at first, and then use UMAP to visualize the results.

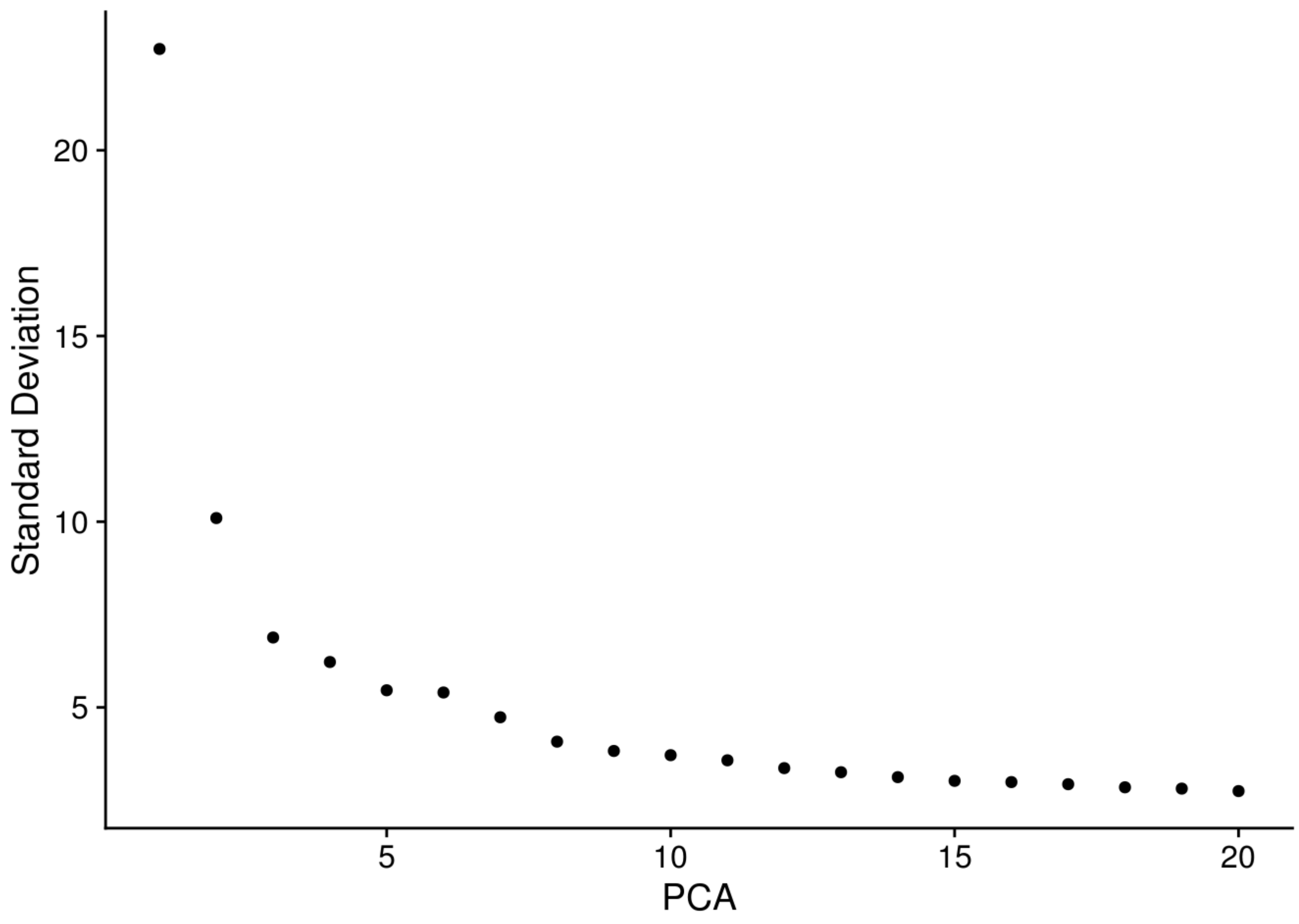
```
all.data <- ScaleData(all.data)
```

```
## Centering and scaling data matrix
```

```
all.data <- RunPCA(all.data, features = rownames(all.data), npcs = 20, reduction.name = "PCA",
 reduction.key = "PCA_", verbose = FALSE)
```

According to ElbowPlot, we selected the top15 PCs.

```
ElbowPlot(all.data,reduction = "PCA")
```



```
all.data <- RunUMAP(all.data, reduction = "PCA", dims = 1:15, reduction.name = "UMAP",  reducti
on.key = "UMAP_")
```

```
## Warning: The default method for RunUMAP has changed from calling Python UMAP via reticulate
to the R-native UWOT using the cosine metric
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric to 'correlatio
n'
## This message will be shown once per session
```

```
## 16:23:13 UMAP embedding parameters a = 0.9922 b = 1.112
```

```
## 16:23:13 Read 8783 rows and found 15 numeric columns
```

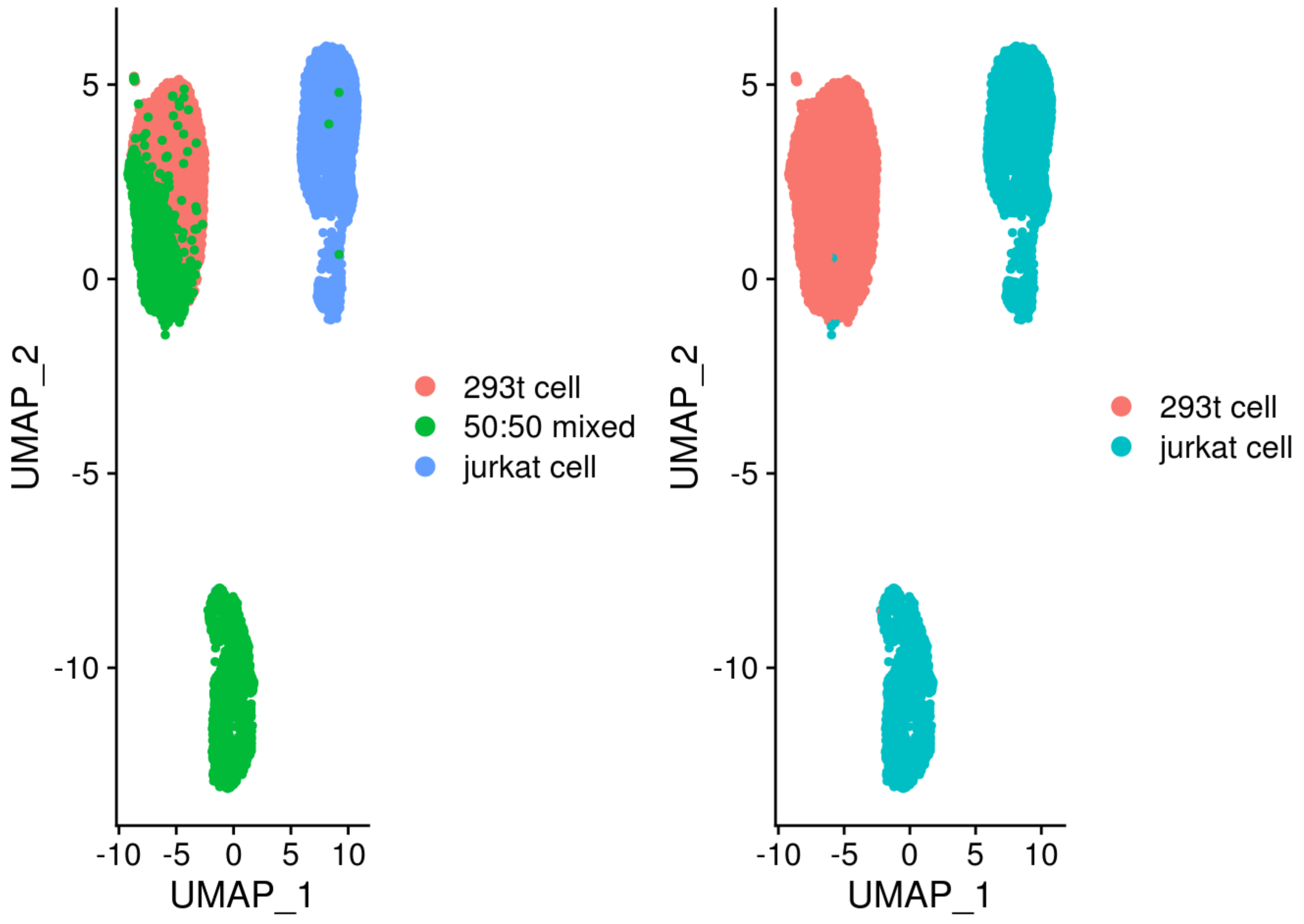
```
## 16:23:13 Using Annoy for neighbor search, n_neighbors = 30
```

```
## 16:23:13 Building Annoy index with metric = cosine, n_trees = 50
```

```
## 0%   10   20   30   40   50   60   70   80   90   100%
```

```
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
## 16:23:16 Writing NN index file to temp file /tmp/RtmpQmxx00/file7921765c77bd
## 16:23:16 Searching Annoy index using 1 thread, search_k = 3000
## 16:23:20 Annoy recall = 100%
## 16:23:21 Commencing smooth kNN distance calibration using 1 thread
## 16:23:23 Initializing from normalized Laplacian + noise
## 16:23:23 Commencing optimization for 500 epochs, with 353924 positive edges
## 16:23:55 Optimization finished
```

```
p1 <- DimPlot(all.data, reduction = 'UMAP', pt.size = 1, group.by = 'batch')
p2 <- DimPlot(all.data, reduction = 'UMAP', pt.size = 1, group.by = 'celltype')
plot_grid(p1, p2)
```



As you can see from the above figure, a part of cells from “50:50 mixed” are separated as an independent group (left panel), while this part of cells are actually belong to jukart cells (right panel). Therefore batch effect exists in the original dataset.

Run ICAnet to correct batch-effect

ICAnet is consistent of three steps. The first step is to decompose each batch of single-cell gene expression matrix into a number of independent components, wherein the number of components was chosen based on random matrix theory.

```
ica <- ICAcomputing(obj = all.data,RMT=TRUE,ICA.type = "JADE")
```

```
## [1] "batch 1 Indepdent Component Analysis"
## [1] "emmm...centering..."
```

```
## Centering data matrix
```

```
## [1] "Done Centering"
## [1] "Using RMT to estimate number of module"
```

```
## Loading required package: coop
```

```
## Loading required package: rARPACK
```

```
## [1] "RMT estimate 6 expression programm"
## [1] "Running 2th-step of RMT..."
```

```
## [1] "Two step RMT estimate 3 expression programm"
## [1] "batch 2 Indepdent Component Analysis"
## [1] "emmm...centering..."
```

```
## Centering data matrix
```

```
## [1] "Done Centering"
## [1] "Using RMT to estimate number of module"
## [1] "RMT estimate 7 expression programm"
## [1] "Running 2th-step of RMT..."
```

```
## [1] "Two step RMT estimate 2 expression programm"
## [1] "batch 3 Indepdent Component Analysis"
## [1] "emmm...centering..."
```

```
## Centering data matrix
```

```
## [1] "Done Centering"
## [1] "Using RMT to estimate number of module"
## [1] "RMT estimate 8 expression programm"
## [1] "Running 2th-step of RMT..."
```

```
## [1] "Two step RMT estimate 3 expression programm"
```

```
head(ica$ica.pooling)
```

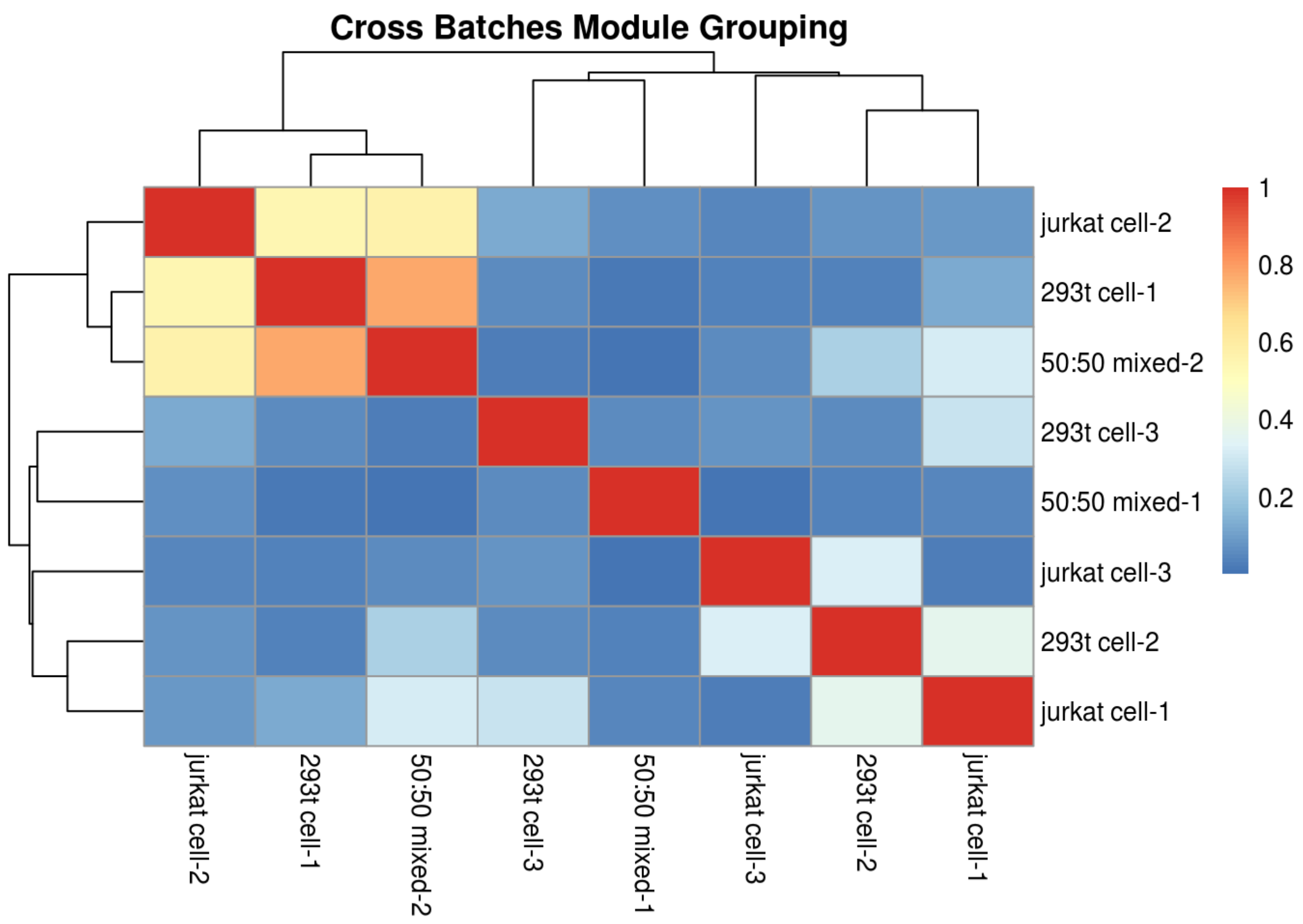
```
##          293t cell-1 293t cell-2 293t cell-3 50:50 mixed-1 50:50 mixed-2
## 7SK.2    -0.18068420 -0.08114175  0.03944666   -0.01833748     0.2412057
## A1BG     -1.86723333 -0.62728335 -0.20187972   -1.58745373     2.1830933
## A1BG-AS1 -0.35676662 -0.15649454  0.15070714   -0.01155930     0.2564616
## A2M-AS1  -0.16014356 -0.10913912  0.05514712   -0.01526717     0.2752598
## A4GALT   -0.05843392  0.01175534  0.11732078    0.04645104     0.1957072
## AAAS      1.49244298 -0.17560096 -0.11327307   -0.03039927    -1.4349816
##          jurkat cell-1 jurkat cell-2 jurkat cell-3
## 7SK.2       -0.1580621    0.18584382    0.05559095
## A1BG        -3.1460765    1.39732805   -0.21896769
## A1BG-AS1    -0.5255411    0.09140577   -0.07379551
## A2M-AS1     -0.1415866    0.21466110    0.09347620
## A4GALT      -0.1415866    0.21466110    0.09347620
## AAAS         0.9239377   -0.72184542    0.05644844
```

Important parameters need to note:

- *ICA.type* Character value specifying the type of ICA used in decomposed expression matrix. We provided three types of ICA: 1) JADE; 2) fastICA; 3) MineICA. MineICA means the multiple running of fastICA, and selecting out the consensus ICs, therefore it requires a longer running time to complete the whole running. We recommend to use JADE for independent component analysis.
- *nc.vec* Integer value specifying the number of independent components for each batch. Higher nc.vec is recommended for datasets with more substructures. Because this is an unsupervised and exploratory analysis step, there is no “right” values for nc.vec. The users could use random matrix theory to estimate the number of components.
- *global.mode* Boolean value indicating whether to perform independent component analysis on whole datasets. It only requires to specify the parameter *nc.global.mode*. Using global mode can correct both continual and discrete technique effects.
- *nc.global.mode* Integer value specifying the number of independent components for the whole scRNA-seq dataset.
- *two.stage* blooean: Value indicating whether to use ICA+RMT (two-step RMT) to estimate the number of latent independent components, as shown in our manuscript. Using RMT+ICA would give better estimation of the number of cell types compared with merely using RMT when facing high dimensional dataset. We recommend to use this parameter when performing batch-effect correction on dataset with sample size > 5000.
- *two.stage.ICA* Character values indicating the type of ICA used in the two-stage estimation. We recommend to use “JADE” when sample size is not so large (<2000 cells). When encountering dataset with large samples (cells), we recommend to use “FastICA”.

The result is a number of independent components of the whole gene expression matrix on each batch. To learn the basal components from these components, we use Partitioning Around Medoids clustering method to cluster the components and select the resulted medoids as the “basal components”.

```
Ica.filter <- CrossBatchGrouping(ica$ica.pooling)
```



```
## Identify3 patterns
```

This figure indicates the correlation heatmap of the independent components from three batches

Important parameters are as follows:

- *cor* Character value indicating the type of correlation used to cluster source ICs from different batches. We provide two type of correlations: Pearson and Spearman.
- *W.top* The weight threshold to define “activated” genes whose attribute values are used for similarity comparison between different ICs.

Ica.filter is a list object that contains the basal programs and the number of it. Each program is represented by a vector, and the specific value of a gene in a program vector represents its association with the specific expression program. Further, we could learn ‘activated’ modules through combining ICA-score and PPI network. The user could download PPI network through the function getPPI_String provided by the R package SCORE. For convenience, we have prepared the pre-processed STRING PPI network ready to use in ICAnet by default. The STRING PPI networks is generated through filtering out interactions whose combined score is below 600.

```
cat('Running ICAnet')
```

```
## Running ICAnet
```

```
all.data <- RunICAnet(all.data,Ica.filter$ica.filter,PPI.net = PPI,scale=FALSE,ModuleSignifican
ce = FALSE)
```

```
## [1] 551
## [1] "num:303"
```

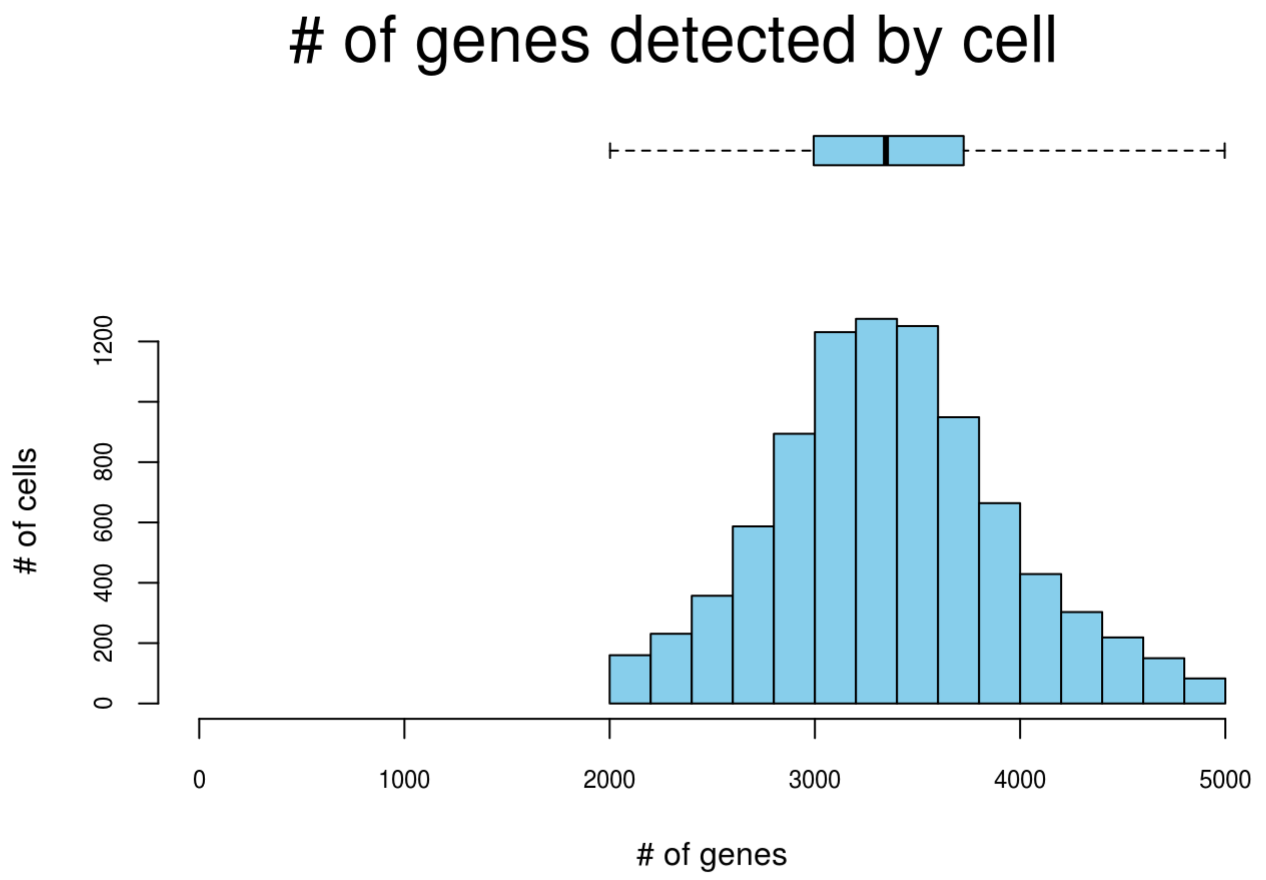
```
## Quantiles for the number of genes detected by cell:
## (Non-detected genes are shuffled at the end of the ranking. Keep it in mind when choosing th
e threshold for calculating the AUC).
```

```
##     min      1%      5%     10%     50%    100%
## 2002.00 2112.82 2438.00 2654.20 3346.00 4997.00
```

```
## Using 6 cores.
```

```
## Using 6 cores.
```

```
## Centering and scaling data matrix
```



This figure is the histogram plot returned by AUCell. It gives a density visualization of the distribution of the number of genes detected in each cell.

Important parameters are as follows:

- *W.top* Integer value specifying the threshold used to define "activated" genes. The genes whose attribute values are larger than the W.top*derivation from the mean will be selected as activated genes.
- *PPI.net* Character value denoting the protein-protein interaction matrix. If no specific PPI matrix (PPI.net=NULL) was specified, ICAnet would downloaded "String" or "BioGrid" PPI network. These functions are dependent on R.SCORE packages.

- *small.size* Integer value. Modules whose gene number is smaller than *small.size* will be filtered out.
- *nMC* Integer value specifying the number of bootstrapping when calculating the statistical significance for each module.
- *scale* boolean variable, perform scaling on each batch of dataset if scale=TRUE. We recommend user to use this parameters when the batches are from different sequencing platforms.
- *ModuleSignificance* boolean variable, evaluate each module statistical significance if ModuleSignificance = TRUE, consider its time consuming, users could omic this step if they don't need to known the module statistical significance.

ICAnet would assign an assay named as "ICAnet" on the Seurat obj, which represents the module-cell activity score, with the row and column denote module and cell, respectively. The statistical significance score for each module is stored in *all.data@misc$modN.score (mailto:all.data@misc$modN.score)*, which measures whether the co-expression of member genes in a module is significantly higher than combination of randomly select genes. The module which presents significant 'co-express' tends to enrich more biological relevant pathways, which has been validated in our research. The gene set information of each module is stored in *all.data@misc$ICAnet_geneSets (mailto:all.data@misc$ICAnet_geneSets)*.

Now, we could perform PCA and UMAP on this module-cell activity matrix.

```
all.data <- RunPCA(all.data, features = rownames(all.data), npcs = 20, reduction.name = "IcaNet
PCA", reduction.key = "IcaNetPCA_", verbose = F)
all.data <- RunUMAP(all.data, reduction = "IcaNetPCA", dims = 1:15, reduction.name = "IcaNetUMA
P",  reduction.key = "IcaNetUMAP_")
```

```
## 16:30:52 UMAP embedding parameters a = 0.9922 b = 1.112
```

```
## 16:30:52 Read 8783 rows and found 15 numeric columns
```

```
## 16:30:52 Using Annoy for neighbor search, n_neighbors = 30
```

```
## 16:30:52 Building Annoy index with metric = cosine, n_trees = 50
```

```
## 0%   10   20   30   40   50   60   70   80   90   100%
```

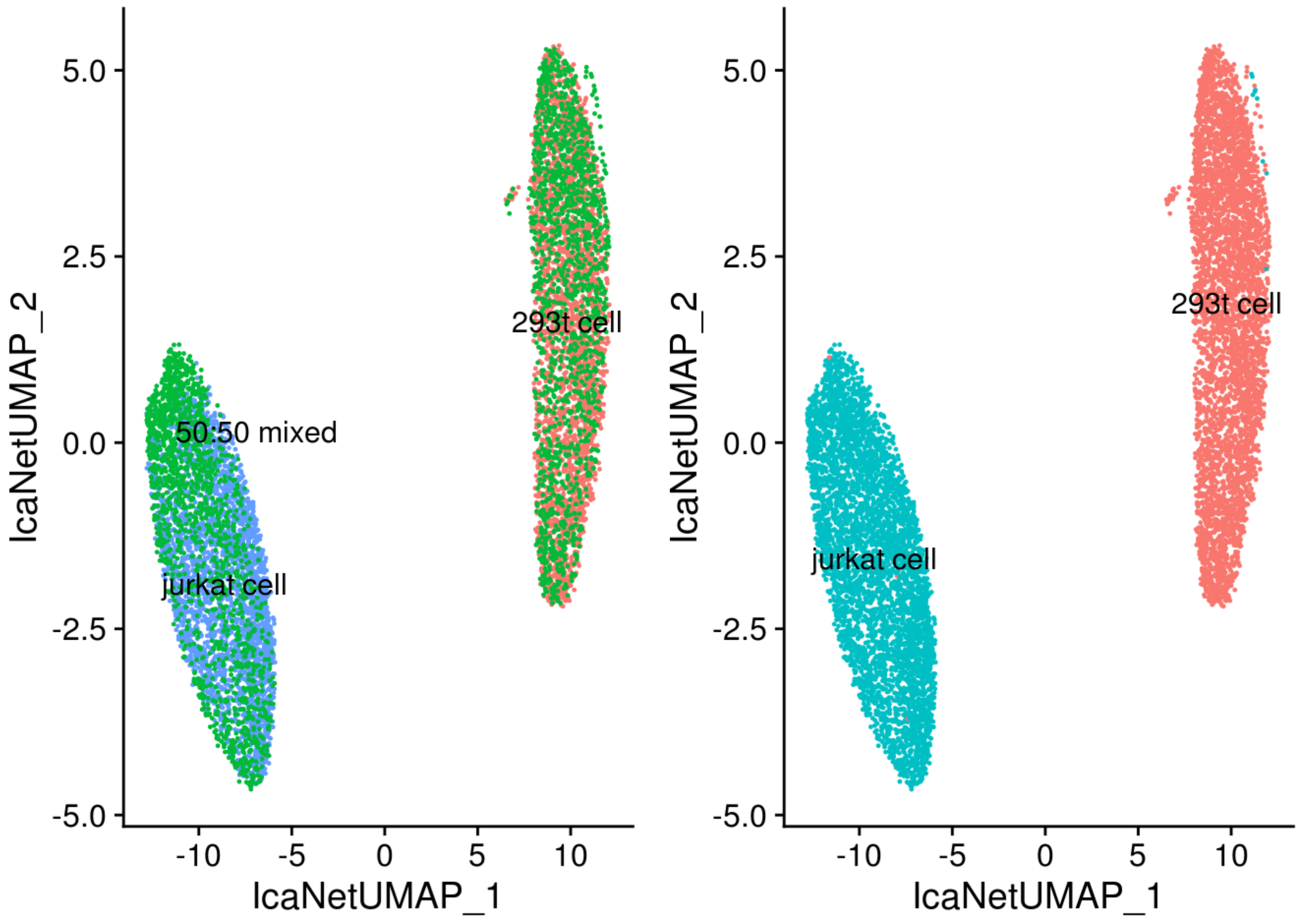
```
## [----|----|----|----|----|----|----|----|----|----|
```

```
## **************************************************|
## 16:30:55 Writing NN index file to temp file /tmp/RtmpQmxx00/file7921e9806f5
## 16:30:55 Searching Annoy index using 1 thread, search_k = 3000
## 16:30:59 Annoy recall = 100%
## 16:31:01 Commencing smooth kNN distance calibration using 1 thread
## 16:31:04 Initializing from normalized Laplacian + noise
## 16:31:04 Commencing optimization for 500 epochs, with 370714 positive edges
## 16:31:40 Optimization finished
```

```
p1 <- DimPlot(all.data, reduction = 'IcaNetUMAP', group.by = 'batch',label=1)+NoLegend()
```

```
## Warning: Using `as.character()` on a quosure is deprecated as of rlang 0.3.0.
## Please use `as_label()` or `as_name()` instead.
## This warning is displayed once per session.
```

```
p2 <- DimPlot(all.data, reduction = 'IcaNetUMAP', group.by = 'celltype',label=1)+NoLegend()
plot_grid(p1, p2)
```



As you can see from the above figure, using module-cell activity matrix inferred from ICAnet can efficiently correct the existing batch effect.