Integrate analysis of pancreatic islet scRNA-seq dataset

Weixu Wang

In this tutorial, I will demonstrate ICAnet's ability to integrate scRNA-seq dataset from different sequencing platforms. We also want to show that using our SVD based cell embedding could improve batch mixing. Here, I will introduce the workflow to integrate different batches.

Stage I: Preprocessing, Normalization, and Scaling

This tutorial is for users who need to integrate more than three batches of scRNA-seq datasets with multiple cell types. To deal with this, ICAnet need to perform preprocessing (denoise) on each dataset. Here, I use three pancreatic islet-derived scRNA-seq datasets produced by different labs (named as Baron et al, Muraro et al, and Segerostople et al, all datasets can be download from https://hemberg-lab.github.io/scRNA.seq.datasets/ (https://hemberg-lab.github.io/scRNA.seq.datasets/)) to illustrate how ICAnet perform batch-effect correction on these datasets. First, loading the datasets with the commands below.

```
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)

library(SingleCellExperiment)
```

```
## Loading required package: SummarizedExperiment
```

```
## Loading required package: GenomicRanges
```

```
## Loading required package: stats4
```

```
## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colMeans,
##     colnames, colSums, dirname, do.call, duplicated, eval, evalq,
##     Filter, Find, get, grep, grepl, intersect, is.unsorted, lapply,
##     lengths, Map, mapply, match, mget, order, paste, pmax, pmax.int,
##     pmin, pmin.int, Position, rank, rbind, Reduce, rowMeans, rownames,
##     rowSums, sapply, setdiff, sort, table, tapply, union, unique,
##     unsplit, which, which.max, which.min
```

```
## Loading required package: S4Vectors
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following object is masked from 'package:base':
##
##     expand.grid
```

```
## Loading required package: IRanges
```

```
## Loading required package: GenomeInfoDb
```

```
## Loading required package: Biobase
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
## Loading required package: DelayedArray
```

```
## Loading required package: matrixStats
```

```
##
## Attaching package: 'matrixStats'
```

```
## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians
```

```
## Loading required package: BiocParallel
```

```
##
## Attaching package: 'DelayedArray'
```

```
## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges
```

```
## The following objects are masked from 'package:base':
##
##     aperm, apply
```

```
##
## Attaching package: 'SummarizedExperiment'
```

```
## The following object is masked from 'package:Seurat':
##
##     Assays
```

```
baron <- readRDS("/mnt/data4/weixu/MCTA/immuneAtlas/Pancreas/baron-human.rds")
muraro <- readRDS("/mnt/data4/weixu/MCTA/immuneAtlas/Pancreas/muraro.rds")
segerstolpe <- readRDS("/mnt/data4/weixu/MCTA/immuneAtlas/Pancreas/segerstolpe.rds")
baron_exp <- counts(baron)
segerstolpe_exp <- counts(segerstolpe)
muraro_exp <- normcounts(muraro)
geneID <- rownames(muraro_exp)
geneID_str <- strsplit(geneID,split="_")
for(i in 1:length(geneID)){
  geneID[i] <- geneID_str[[i]][1]
}
rownames(muraro_exp) <- geneID
geneID <- c(rownames(baron_exp),rownames(muraro_exp),rownames(segerstolpe_exp))
geneID <- names(table(geneID))[table(geneID)==3]
baron_exp <- baron_exp[geneID,]
segerstolpe_exp <- segerstolpe_exp[geneID,]
muraro_exp <- muraro_exp[geneID,]
```

Next, we define *batch* variable and *cell type* variable, and create Seurat object with the commands below.

```
batch <- c(baron$human, rep("segerstolpe",(ncol(segerstolpe_exp))),rep("muraro",ncol(muraro_ex
p)))
celltype <- c(as.character(as.matrix(baron$cell_type1)),
              as.character(as.matrix(segerstolpe$cell_type1)),
              as.character(as.matrix(muraro$cell_type1)))
pancreas <- CreateSeuratObject(cbind(baron_exp,segerstolpe_exp,muraro_exp))
pancreas$batch <- batch
pancreas$celltype <- celltype
```

Then, like Seurat V3(CCA) preprocessing, we perform normalization on each dataset and select highly variable genes (HVGs) in each batch to merge these datasets. We select highly variable genes because some researches point out that selection of highly variable genes improves the performance of data integration [1].

```
pancreas.list <- SplitObject(pancreas, split.by="batch")
for(i in 1:length(pancreas.list)) {
   pancreas.list[[i]] <- NormalizeData(pancreas.list[[i]],verbose=FALSE)
   pancreas.list[[i]] <- FindVariableFeatures(pancreas.list[[i]],selection.method="vst",nfeatur
e=8000,verbose=FALSE)
}
```

** Feature selection for integrative analysis of multiple datasets**:

- The users could use different ways to select variable features:
- *1):* Select consensus HVGs across different batches;
- *2):* Select HVGs of multiple datasets through inspecting their ranks in each dataset. This selection could be implemented through using the function *SelectIntegrationFeatures* in *Seurat*.

```
##Select integrate features
pancreas.feature <- SelectIntegrationFeatures(object.list = pancreas.list,nfeatures = 3000)
###integrate pancreas list
pancreas.all <- pancreas.list[[1]]
pancreas.all <- GetAssayData(pancreas.all)[pancreas.feature,]
for(i in 2:length(table(pancreas$batch))){
  pancreas.set <- pancreas.list[[i]]
  pancreas.set <- GetAssayData(pancreas.set)[pancreas.feature,]
  pancreas.all <- cbind(pancreas.all,pancreas.set[pancreas.feature,])
}
pancreas[['Consensus.RNA']] <- CreateAssayObject(pancreas.all)
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from consensus.rna_ to consensusrna_
```

```
DefaultAssay(pancreas) <- "Consensus.RNA"
```

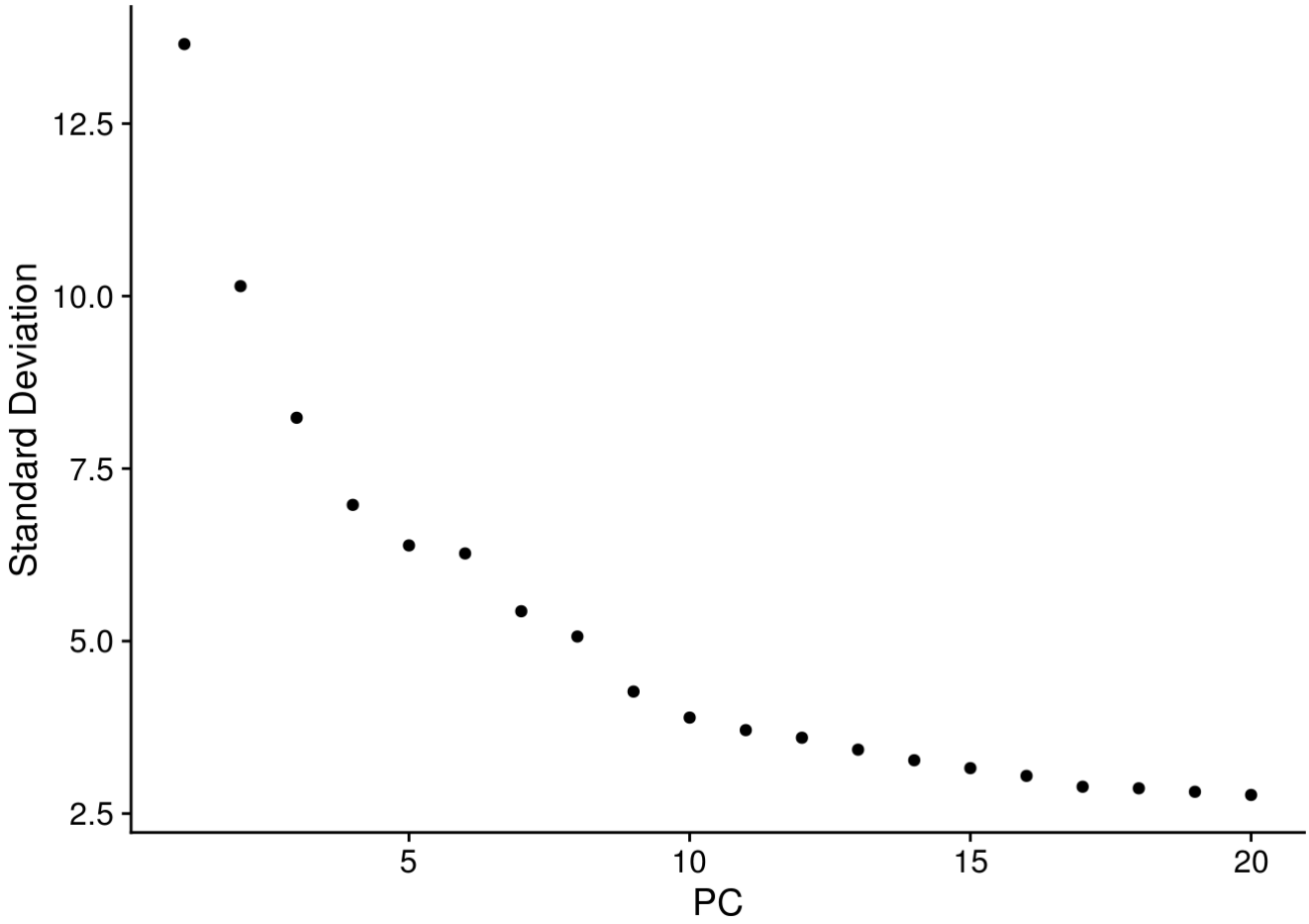
Stage II: Using ICAnet to learn expression programs on each dataset

We further use ICAnet to learn expression programs to validate its performance. Original workflow was applied to analyze pancreas dataset for comparison.

```
pancreas <- ScaleData(pancreas)
```

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

```
pancreas <- RunPCA(pancreas, npcs =50,verbose=FALSE,features=rownames(pancreas))
ElbowPlot(pancreas,reduction = "pca")
```



This Figure indicates the data variation when they projected on each PCs, and the variation is represented through Standard Deviation.

```
pancreas <- RunUMAP(pancreas, reduction = "pca", dims = 1:20, reduction.name = "umap",  reduction.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 'correlation'
## This message will be shown once per session
```

```
## 20:48:04 UMAP embedding parameters a = 0.9922 b = 1.112
```

```
## 20:48:04 Read 14209 rows and found 20 numeric columns
```

```
## 20:48:04 Using Annoy for neighbor search, n_neighbors = 30
```

```
## 20:48:04 Building Annoy index with metric = cosine, n_trees = 50
```

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

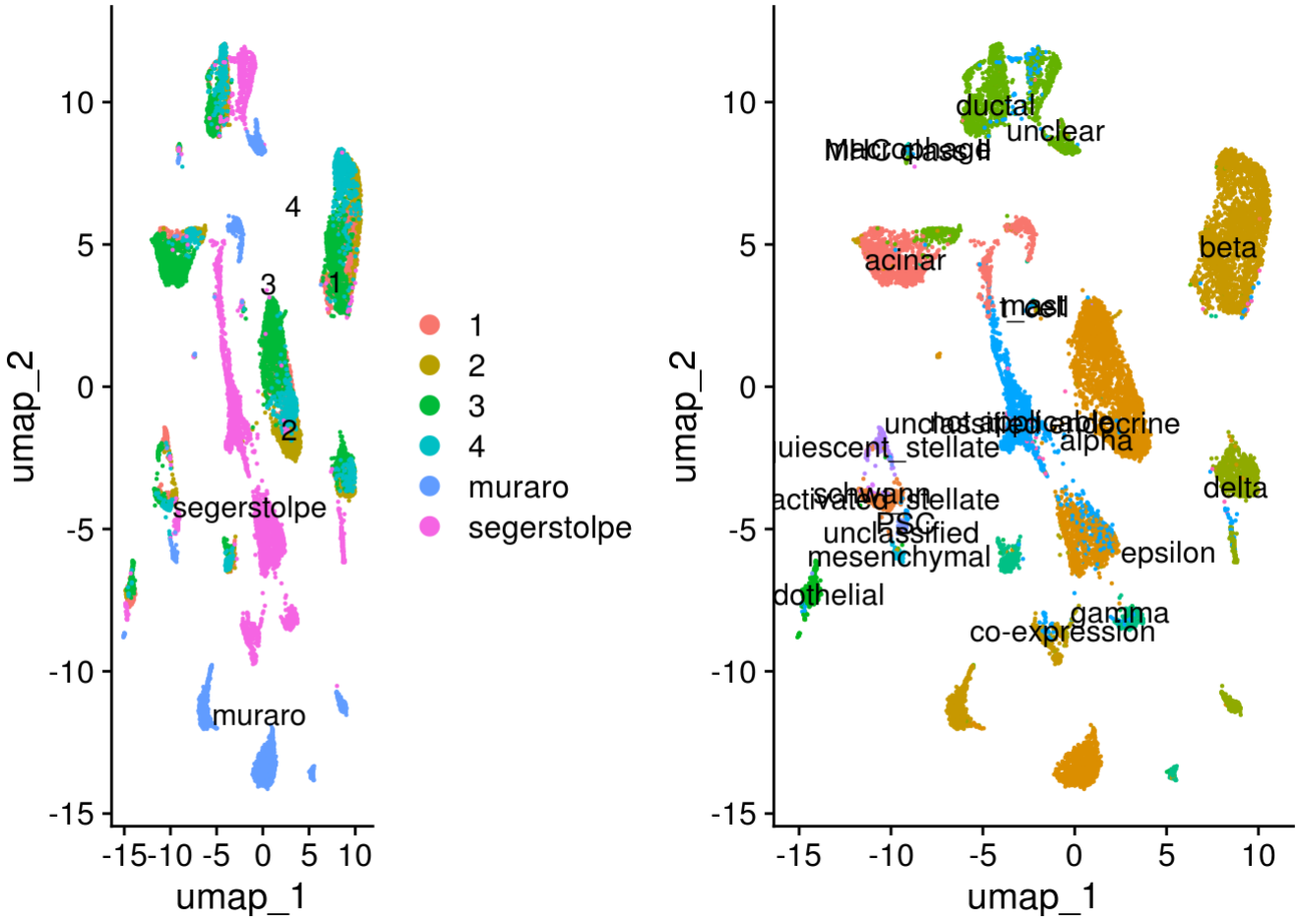
```
## [----|----|----|----|----|----|----|----|----|----|
```

```
## ************************************************|
## 20:48:09 Writing NN index file to temp file /tmp/RtmpQy3cfg/file368f2a9fc215
## 20:48:09 Searching Annoy index using 1 thread, search_k = 3000
## 20:48:14 Annoy recall = 100%
## 20:48:15 Commencing smooth kNN distance calibration using 1 thread
## 20:48:16 Initializing from normalized Laplacian + noise
## 20:48:18 Commencing optimization for 200 epochs, with 595264 positive edges
## 20:48:36 Optimization finished
```

```
p1 <- DimPlot(pancreas, reduction = 'umap', group.by = 'batch',label=1)
```

```
## 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(pancreas, reduction = 'umap', group.by = 'celltype',label=1)+NoLegend()
plot_grid(p1,p2)
```



The left panel of this figure represents the visualization of the cells, and each dot is colored according to its batch label. Also, each dot in right panel is colored according to its cell type label.

As we can see from the above figure, even after preprocessing step, strong batch effect still exists in these datasets. ICAnet aims to identify the shared gene co-expression pattern across different batches, therefore can be applied to correct batch effect. We therefore decompose each batch of scRNA-seq dataset using ICA with the command below.

```
ica.pancreas <- ICAcomputing(pancreas, ICA.type="JADE",two.stage = FALSE,global.mode=FALSE,center=FALSE,scale=TRUE)
```

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

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

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

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

```
##
## Attaching package: 'coop'
```

```
## The following object is masked from 'package:DelayedArray':
##
##     sparsity
```

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

```
## [1] "RMT estimate 19 expression programm"
## [1] "batch 2 Indepdent Component Analysis"
## [1] "emmm...scaling..."
```

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

```
## [1] "Done scaling"
## [1] "Using RMT to estimate number of module"
## [1] "RMT estimate 18 expression programm"
## [1] "batch 3 Indepdent Component Analysis"
## [1] "emmm...scaling..."
```

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

```
## [1] "Done scaling"
## [1] "Using RMT to estimate number of module"
## [1] "RMT estimate 22 expression programm"
## [1] "batch 4 Indepdent Component Analysis"
## [1] "emmm...scaling..."
```

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

```
## [1] "Done scaling"
## [1] "Using RMT to estimate number of module"
## [1] "RMT estimate 15 expression programm"
## [1] "batch 5 Indepdent Component Analysis"
## [1] "emmm...scaling..."
```

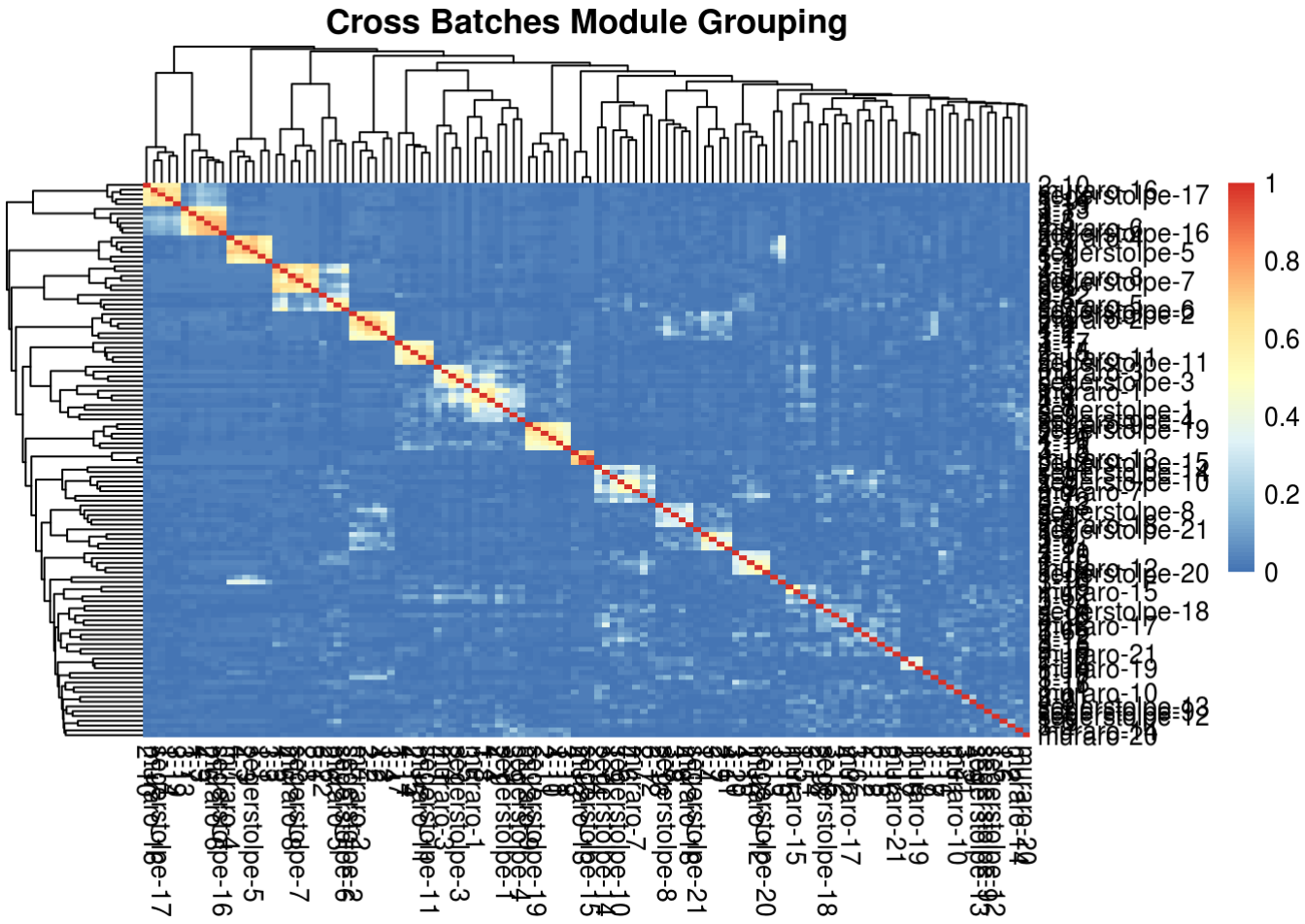
```
## Centering and scaling data matrix
```

```
## [1] "Done scaling"
## [1] "Using RMT to estimate number of module"
## [1] "RMT estimate 21 expression programm"
## [1] "batch 6 Indepdent Component Analysis"
## [1] "emmm...scaling..."
```

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

```
## [1] "Done scaling"
## [1] "Using RMT to estimate number of module"
## [1] "RMT estimate 21 expression programm"
```

```
ica.filter <- CrossBatchGrouping(ica.pancreas$ica.pooling,cor="spearman",Unique.Preservation=FALSE)
```




```
## Identify26 patterns
```

This figure indicates the correlation heatmap of the independent components from all batches.

Important parameters are as follows:

- *Unique.Preservation*: This boolean variable is used to determine whether to select the cluster having more than one component. If this parameter is FALSE, then clusters with only one component are removed.

Stage III: Running ICAnet to perform batch-effect correction

Before running ICAnet, the user can use the function *getPPI_Strings* provided by RSCORE packages to get the PPI network.

```
library(RSCORE)
```

```
##
```

```
## Warning: replacing previous import 'clusterProfiler::simplify' by
## 'igraph::simplify' when loading 'RSCORE'
```

```
##
```

```
## Warning: replacing previous import 'igraph::simplify' by 'propr::simplify' when
## loading 'RSCORE'
```

```
## Warning: replacing previous import 'igraph::get.vertex.attribute' by
## 'network::get.vertex.attribute' when loading 'RSCORE'
```

```
hs_network_matrix <- getPPI_String(pancreas,species = 9606)
```

```
## Warning:  we couldn't map to STRING 0% of your identifiers
```

Then, we run ICAnet to convert gene expression matrix into module activity matrix. Here we use non-scaled matrix to evaluate module activity.

```
pancreas <- RunICAnet(pancreas,ica.filter$ica.filter, PPI.net = hs_network_matrix, W.top=2,aucM
axRank = 300,ModuleSignificance=FALSE,scale=FALSE)
```

```
## [1] 2043
## [1] "num:553"
```

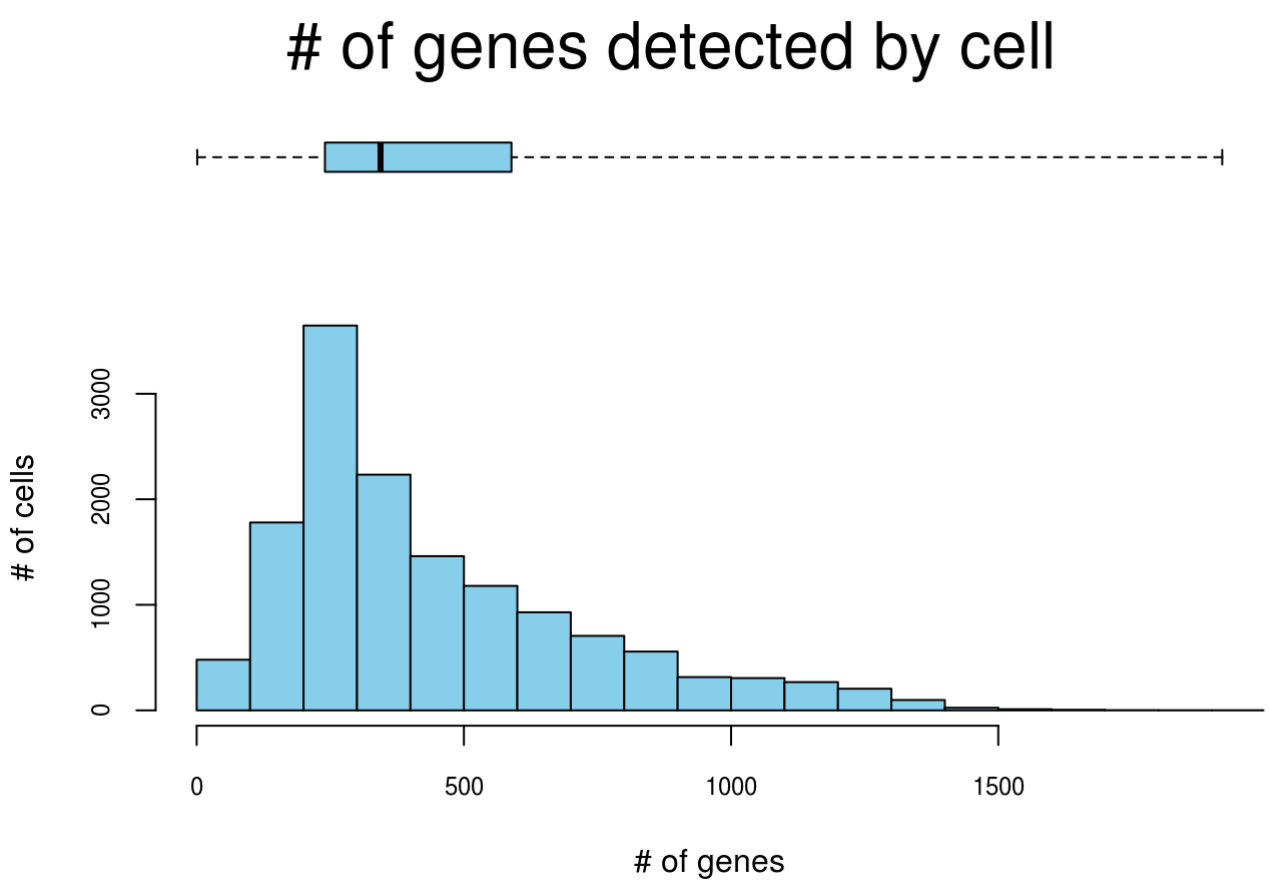
```
## Quantiles for the number of genes detected by cell:
## (Non-detected genes are shuffled at the end of the ranking. Keep it in mind when choosing th
e threshold for calculating the AUC).
```

```
##     min      1%      5%     10%     50%    100%
##    1.00   39.08  122.00  164.00  344.00 1919.00
```

```
## 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.

To perform dimensional reductions, here we provided a SVD-based feature reduction methods which could be used to improve batch mixing.

```
pancreas <- RunModuleSVD(pancreas,nu=30,power=0.2)
```

```
## Warning: No assay specified, setting assay as RNA by default.
```

```
## Warning: No columnames present in cell embeddings, setting to 'svd_1:30'
```

Important parameters are as follows:

- *power* This interger value is used to control the signal smoothness over each components.

The SVD based dimensional reduction is stored in pancreas@reductions (mailto:pancreas@reductions) [['Module_SVD']], which represent the SVD-based cell embedding vectors calculated from module activity values.

Further, we could test different *power* value to see how this parameter influences the batch mixing performance.

```
pancreas <- RunModuleSVD(pancreas,nu=30,power=0.1)
```

```
## Warning: No assay specified, setting assay as RNA by default.
```

```
## Warning: No columnames present in cell embeddings, setting to 'svd_1:30'
```

```
pancreas <- RunUMAP(pancreas, reduction = "Module_SVD", dims = 1:20, reduction.name = "umap_pow
er_0.1",  reduction.key = "umap_",verbose=FALSE)
```

```
## Warning: Cannot add objects with duplicate keys (offending key: umap_), setting
## key to 'umap_power_0.1_'
```

```
pancreas <- RunModuleSVD(pancreas,nu=30,power=0.2)
```

```
## Warning: No assay specified, setting assay as RNA by default.

## Warning: No columnames present in cell embeddings, setting to 'svd_1:30'
```

```
pancreas <- RunUMAP(pancreas, reduction = "Module_SVD", dims = 1:20, reduction.name = "umap_pow
er_0.2",  reduction.key = "umap_",verbose=FALSE)
```

```
## Warning: Cannot add objects with duplicate keys (offending key: umap_), setting
## key to 'umap_power_0.2_'
```

```
pancreas <- RunModuleSVD(pancreas,nu=30,power=0.35)
```

```
## Warning: No assay specified, setting assay as RNA by default.

## Warning: No columnames present in cell embeddings, setting to 'svd_1:30'
```

```
pancreas <- RunUMAP(pancreas, reduction = "Module_SVD", dims = 1:20, reduction.name = "umap_pow
er_0.35",  reduction.key = "umap_",verbose=FALSE)
```

```
## Warning: Cannot add objects with duplicate keys (offending key: umap_), setting
## key to 'umap_power_0.35_'
```

```
pancreas <- RunModuleSVD(pancreas,nu=30,power=0.5)
```

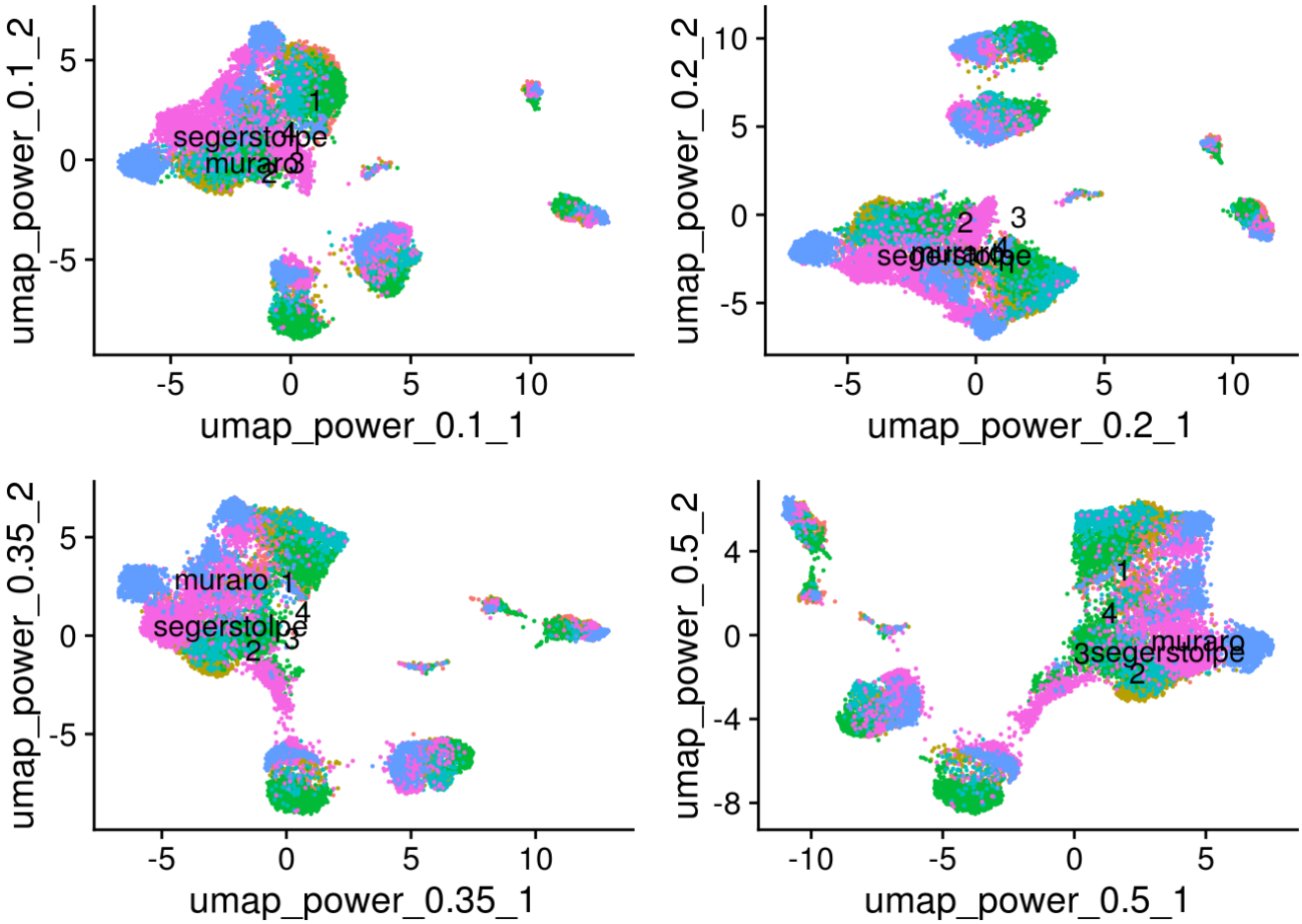
```
## Warning: No assay specified, setting assay as RNA by default.

## Warning: No columnames present in cell embeddings, setting to 'svd_1:30'
```

```
pancreas <- RunUMAP(pancreas, reduction = "Module_SVD", dims = 1:20, reduction.name = "umap_pow
er_0.5",  reduction.key = "umap_",verbose=FALSE)
```

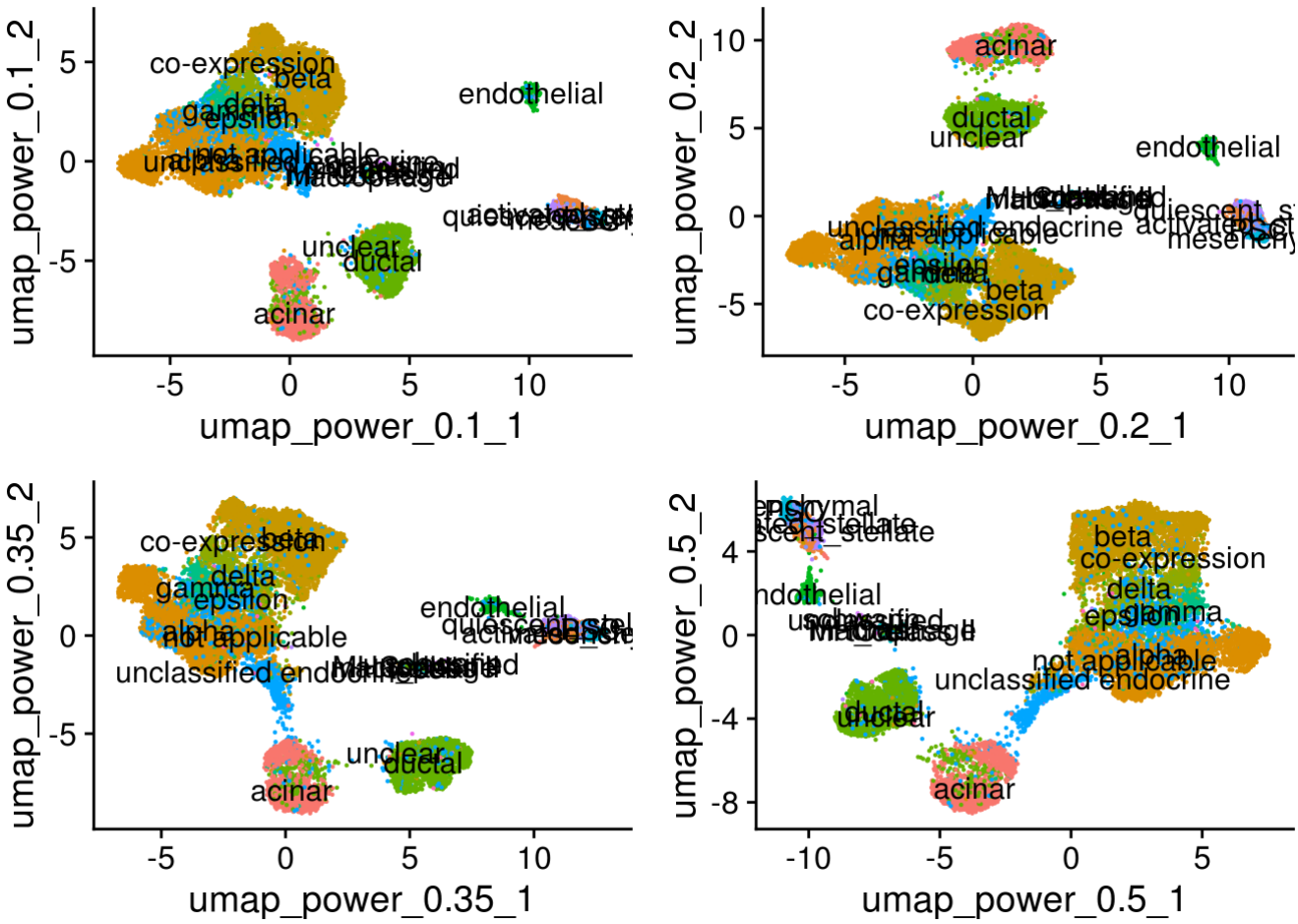
```
## Warning: Cannot add objects with duplicate keys (offending key: umap_), setting
## key to 'umap_power_0.5_'
```

```
p1 <- DimPlot(pancreas, reduction = 'umap_power_0.1', group.by = 'batch',label=1)+NoLegend()
p2 <- DimPlot(pancreas, reduction = 'umap_power_0.2', group.by = 'batch',label=1)+NoLegend()
p3 <- DimPlot(pancreas, reduction = 'umap_power_0.35', group.by = 'batch',label=1)+NoLegend()
p4 <- DimPlot(pancreas, reduction = 'umap_power_0.5', group.by = 'batch',label=1)+NoLegend()
plot_grid(p1,p2,p3,p4,nrow=2,ncol=2)
```



The figure represents the visualization of the cells, and each dot is colored according to its batch label.

```
p1 <- DimPlot(pancreas, reduction = 'umap_power_0.1', group.by = 'celltype',label=1)+NoLegend()
p2 <- DimPlot(pancreas, reduction = 'umap_power_0.2', group.by = 'celltype',label=1)+NoLegend()
p3 <- DimPlot(pancreas, reduction = 'umap_power_0.35', group.by = 'celltype',label=1)+NoLegend
()
p4 <- DimPlot(pancreas, reduction = 'umap_power_0.5', group.by = 'celltype',label=1)+NoLegend()
plot_grid(p1,p2,p3,p4,nrow=2,ncol=2)
```



The figure represents the visualization of the cells, and each dot is colored according to its cell type label.

We find that with the increasing value of the power, the batch mixing effect is also improved, while it can also influence the cell type identification (which means cells from some cell types are no longer grouped together). After running ICAnet, we find that most cells are largely grouped together according to their cell identity. To further improve batch effect correction, we can scale gene expression matrix of each batch to further improve integrations. It has been validated by recent work indicating that scaling could improve batch correction [1].

```
DefaultAssay(pancreas) <- "Consensus.RNA"
pancreas <- RunICAnet(pancreas, ica.filter$ica.filter, PPI.net = hs_network_matrix, W.top=2, aucM
axRank = 300, scale=TRUE)
```

```
##
## Running Scaling on each batch(Improve Batch Corrections)
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.1_ to umappower01_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.2_ to umappower02_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.35_ to umappower035_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.5_ to umappower05_
```

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

```
##
## Done Batch 1
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.1_ to umappower01_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.2_ to umappower02_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.35_ to umappower035_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.5_ to umappower05_
```

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

```
##
## Done Batch 2
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.1_ to umappower01_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.2_ to umappower02_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.35_ to umappower035_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.5_ to umappower05_
```

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

```
##
## Done Batch 3
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.1_ to umappower01_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.2_ to umappower02_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.35_ to umappower035_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.5_ to umappower05_
```

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

```
##
## Done Batch 4
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.1_ to umappower01_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.2_ to umappower02_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.35_ to umappower035_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.5_ to umappower05_
```

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

```
##
## Done Batch muraro
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.1_ to umappower01_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.2_ to umappower02_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.35_ to umappower035_
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from umap_power_0.5_ to umappower05_
```

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

```
##
## Done Batch segerstolpe
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from scale.data_ to scaledata_
```

```
## [1] 2043
## [1] 2043
## [1] "num:553"
```

```
## 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).
```

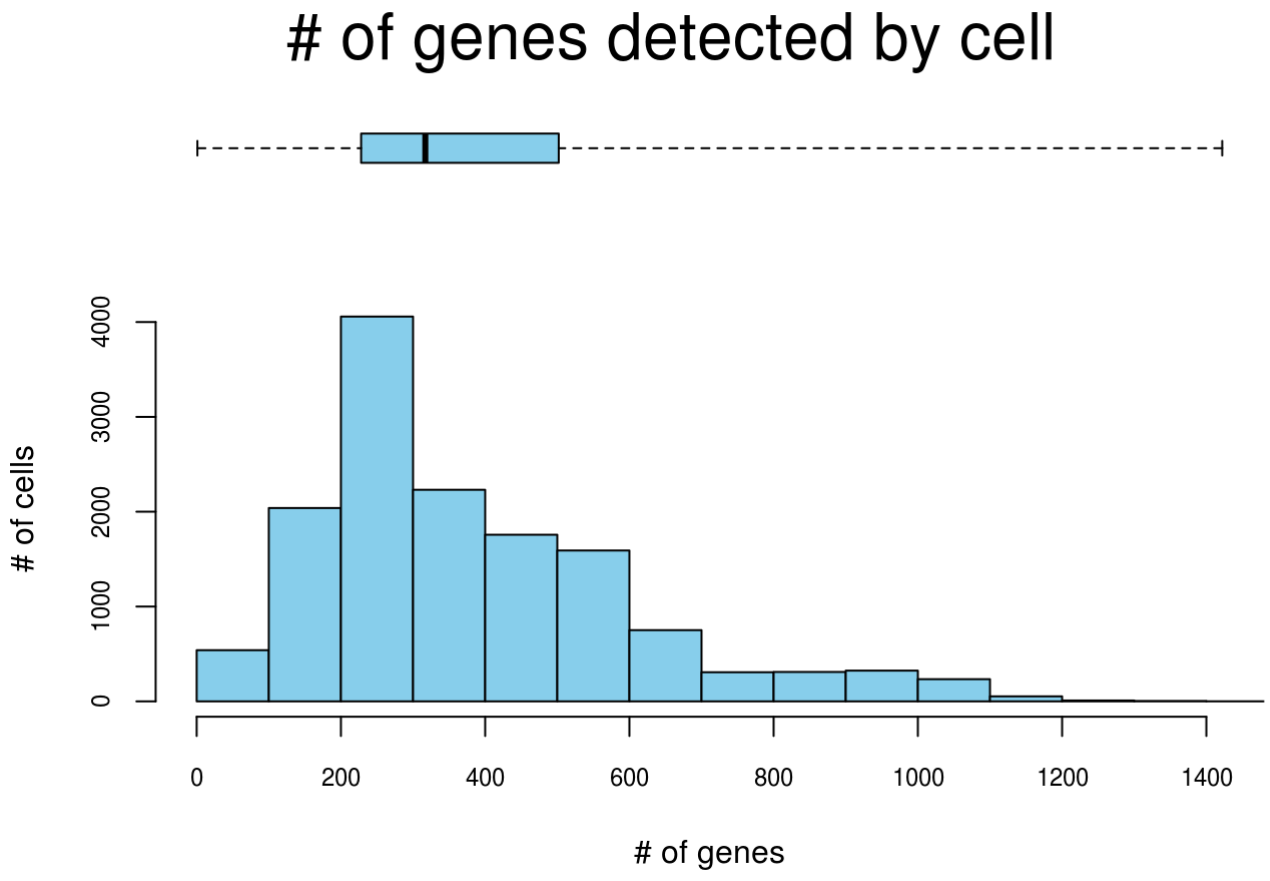
```
##  min   1%   5%  10%  50% 100%
##    1   38  114  155  317 1422
```

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

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

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

```
## Warning in `Misc<-.Seurat`(`*tmp*`, slot = "IcaNet_geneSets", value =
## final_geneSets): Overwriting miscellanous data for IcaNet_geneSets
```



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.

```
pancreas <- RunModuleSVD(pancreas,nu=30,power=0.2)
```

```
## Warning: No assay specified, setting assay as RNA by default.
```

```
## Warning: No columnames present in cell embeddings, setting to 'svd_1:30'
```

```
pancreas <- RunUMAP(pancreas, reduction = "Module_SVD", dims = 1:20, reduction.name = "umap",
 reduction.key = "umap_")
```

```
## 21:32:30 UMAP embedding parameters a = 0.9922 b = 1.112
```

```
## 21:32:30 Read 14209 rows and found 20 numeric columns
```

```
## 21:32:30 Using Annoy for neighbor search, n_neighbors = 30
```

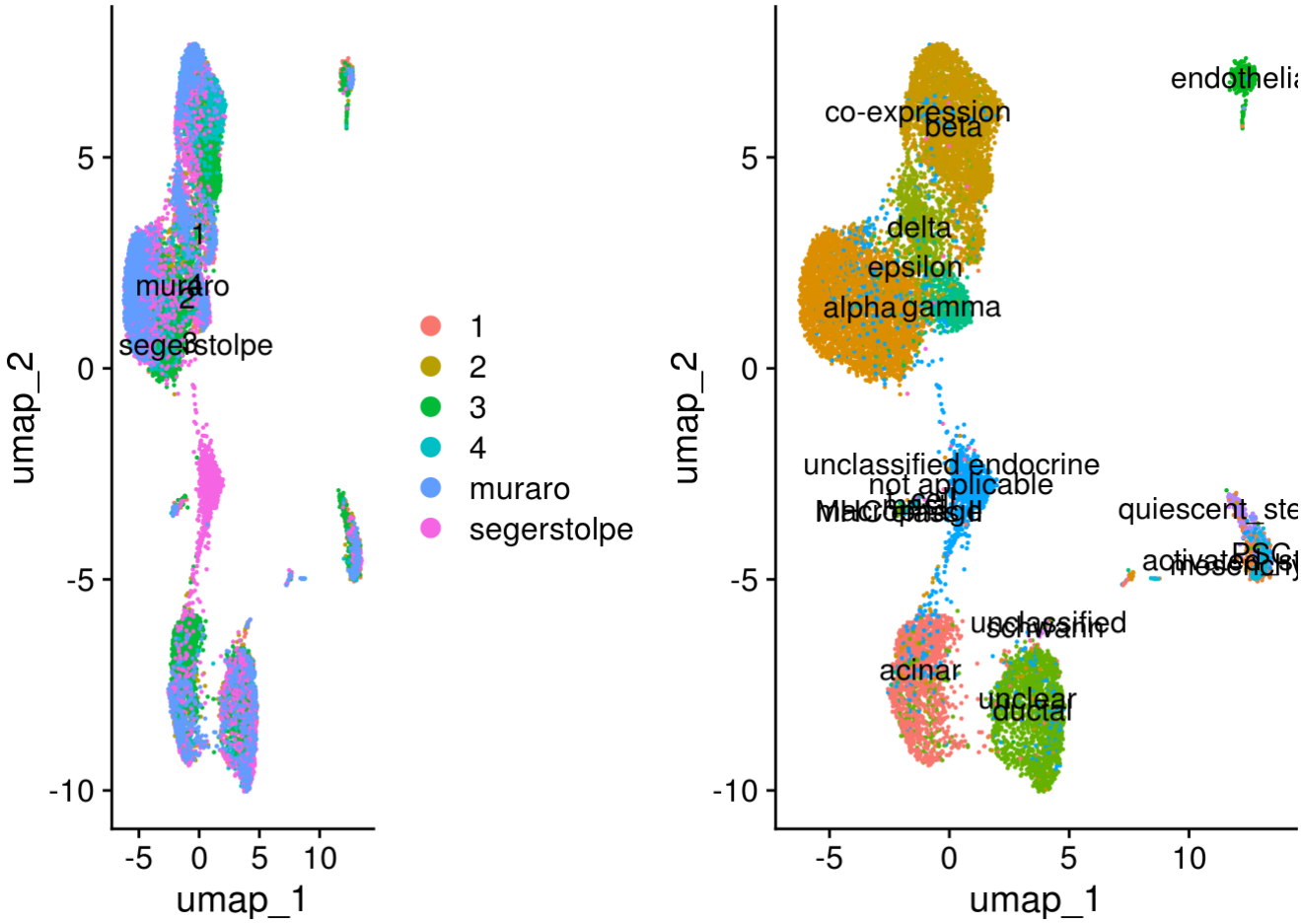
```
## 21:32:30 Building Annoy index with metric = cosine, n_trees = 50
```

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

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

```
## **************************************************|
## 21:32:34 Writing NN index file to temp file /tmp/RtmpQy3cfg/file368f6fda7e2a
## 21:32:34 Searching Annoy index using 1 thread, search_k = 3000
## 21:32:40 Annoy recall = 100%
## 21:32:42 Commencing smooth kNN distance calibration using 1 thread
## 21:32:44 Initializing from normalized Laplacian + noise
## 21:32:45 Commencing optimization for 200 epochs, with 591090 positive edges
## 21:33:03 Optimization finished
```

```
p1 <- DimPlot(pancreas, reduction = 'umap', group.by = 'celltype',label=1)+NoLegend()
p2 <- DimPlot(pancreas, reduction = 'umap', group.by = 'batch',label=1)
plot_grid(p2,p1)
```



This figure represents the visualization of the cells, and each dot is colored according to its batch label. Also, each dot in right panel is colored according to its cell type label.

Stage IV: Visualize statistical significance module(sub-network) activity pattern and network structure.

ICAnet can also visualize module statistical significance through comparing its conditional number of gene members' expression matrix with null (emperical) distribution generated from randomized modules. Firstly, we need to select cell type specific modules:

```
pancreas@active.ident <- as.factor(pancreas$celltype)
modules <- FindMarkerModule(pancreas, identity="beta")
head(modules)
```

	my... <dbl>	avg_diff <dbl>	power <dbl>	pct.1 <dbl>	pct.2 <dbl>	Module_significance <dbl>
ICAnet-17-2-11	0.955	0.2554782	0.910	0.993	0.577	0
ICAnet-17-38-9	0.943	0.2349114	0.886	0.987	0.524	0
ICAnet-22-23-5	0.922	0.2986747	0.844	0.955	0.344	0
ICAnet-22-7-6	0.918	0.2509845	0.836	0.971	0.535	0
ICAnet-5-8-5	0.916	0.2373001	0.832	0.977	0.629	0
ICAnet-22-39-11	0.909	0.1708048	0.818	0.990	0.660	0

6 rows

Important parameters are as follows:

- *identity* Character values used to specify the cell type ID.
- *MQC* Boolean variable used to define whether to perform module quality control (filter the module which its pvalue > threshold).
- *pvalue* Decimal value used to define the significant modules. The default p-value is 0.05.

Additional Informations

- *module barcode* Each module will be assigned a “module barcode” to indicate their specific information, encoded as the “ICAnet-from x th component-x th module-the number of genes in the module”

The FindMarkerModule function will use AUC based metric to evaluate each module’s differential activation and rank the module according their AUC value. It also could return module significance values, so that the user could select the interesting cell type specific modules with statistical significant.

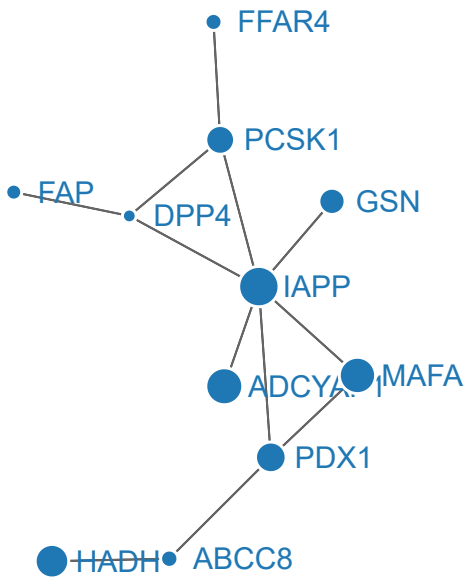
We can further visualize the module of interest using the following functions. For example, among gamma cell specific modules, supposing we are interested in the top1 significant module.

```
library(networkD3)
```

```
##
## Attaching package: 'networkD3'
```

```
## The following object is masked from 'package:Seurat':
##
##     JS
```

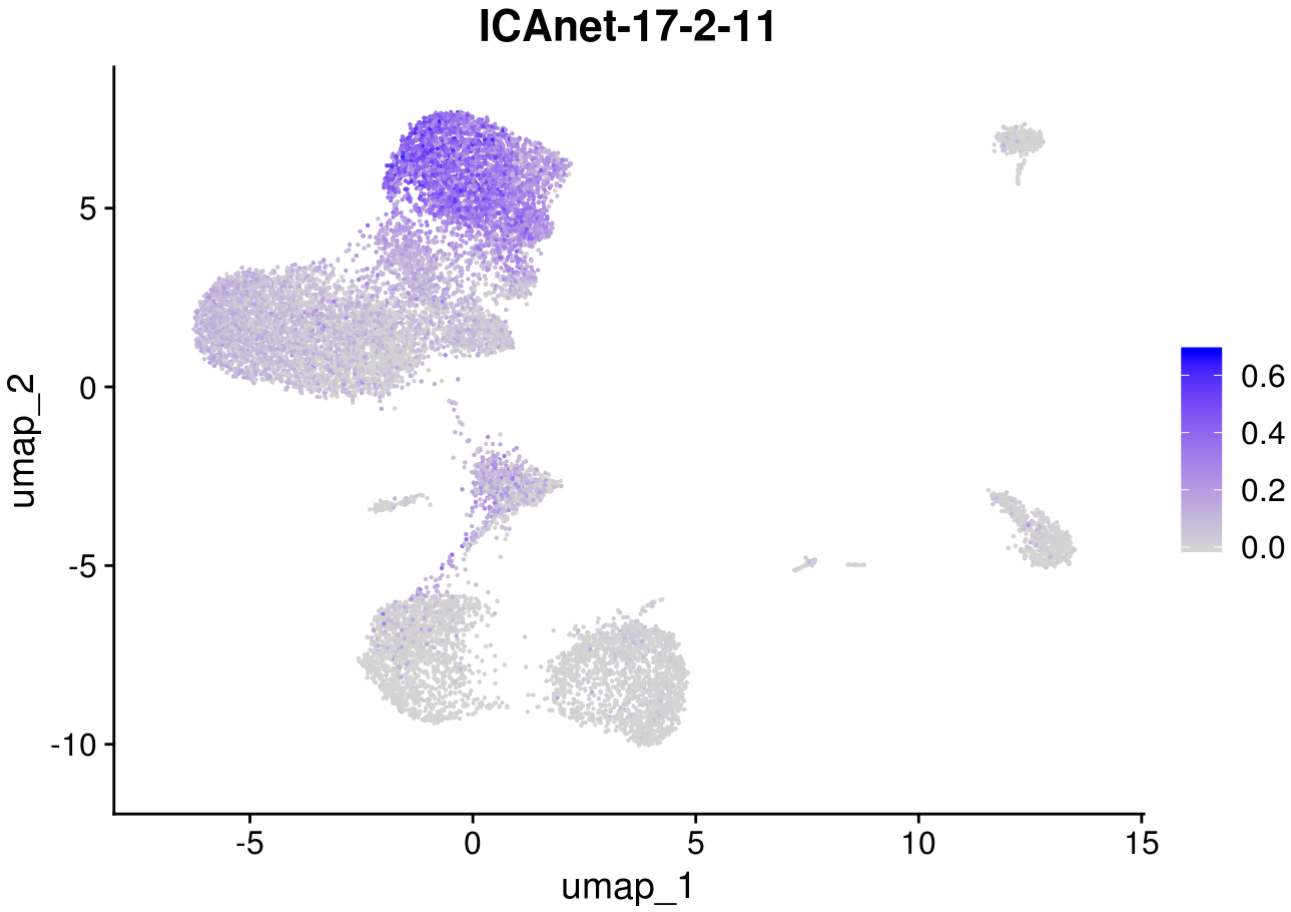
```
plot_module(gene_sets = pancreas@misc$IcaNet_geneSets[['ICAnet-17-2-11']],network=hs_network_matrix,nodeSize="ica.score",ica.score = abs(ica.filter$ica.filter[,17]),size_scale = 1.5)
```



In this subnetwork, we find that the gene IAPP has the highest degree and ica.score. IAPP encodes islet amyloid polypeptide hormone, which is released from pancreatic beta cell following food intake to regulate blood glucose levels.

Further, we can visualize the module activity pattern on UMAP plot as shown below:

```
FeaturePlot(pancreas,"ICAnet-17-2-11", reduction="umap")
```



As we could see from the above figure that this module is highly activated in the beta cells, suggesting its possible role with beta cell function.

Reference

[1] Luecken M D, Buttner M, Chaichoompu K, et al. Benchmarking atlas-level data integration in single-cell genomics[J]. BioRxiv, 2020.