PACSI Tutorial

Chonghui Liu
Guohua Wang

2022-08-05

Introduction

PACSI, a novel network-based method to identify cell subpopulations associated with disease phenotypes of interest.

```
library(PACSI)
```

PACSI example

The input of PACSI are: a single-cell expression matrix, a bulk expression matrix, a ppi data and a phenotype label data. The phenotype label of each bulk sample should be binary. In this tutorial, we use a example to show how to execute PACSI in real applications.

Prepare the scRNA-seq data

We load in the HNSC scRNA-seq data:

```
data("case1_sc_data_1to100")
sc_data <- case1_sc_data_1to100
```

In this scRNA-seq data, each row represents a gene and each column represents a cell.The single-cell expression matrix should be TPM/FPKM-normalized. The dimensions of the single-cell data are:

```
dim(sc_data)
```

```
## [1] 5000  100
```

which indicates there are 5,000 genes and 100 cells in total. For the scRNA-seq data used in PACSI, We can use the functions in the Seurat package to preprocess this data:

```
library(Seurat)
```

```
## Attaching SeuratObject
```

```
  sc_data2 <- Seurat::CreateSeuratObject(as.matrix(sc_data))
  sc_data2 <- Seurat::FindVariableFeatures(object = sc_data2, verbose = F, nfeatures=2000)
  sc_data2 <- Seurat::ScaleData(object = sc_data2, verbose = F)
  sc_data2 <- Seurat::RunPCA(object = sc_data2, features = Seurat::VariableFeatures(sc_data2),
 verbose = TRUE)
```

```
## PC_ 1
## Positive:  KRT5, TACSTD2, KRT18, PKP3, DSP, LAMB3, KRT15, IRF6, CBLC, ST6GALNAC2
##     S100A14, KRT8, TNFSF10, LAD1, DUOXA1, AKIRIN1, SLC35F2, GRB7, PTTG1, FZD6
##     GNL2, ELF3, NME1, SRD5A1, PYCARD, GLRX3, TM4SF1, BRIX1, NDUFA12, HDAC1
## Negative:  EGR1, FOS, FHL1, SERPING1, ZFP36, TXNIP, SLC2A3, C4B, GPX3, HSPB6
##     FHL5, RGS16, MCAM, ITGA7, LGI4, MRVI1, DPYSL2, MT1A, UGGT2, RNASE4
##     RCAN2, GGT5, CYR61, C11orf96, SERPINF1, ZNF331, LTBP4, HEYL, PLCE1, MYOCD
## PC_ 2
## Positive:  RNASE7, SPRR2D, EPS8L1, SDCBP2, PRSS22, AGPAT9, TMEM125, S100P, TMPRSS11F, RNF223
##     ERO1L, IL1RN, KLK7, RAET1G, ATG9B, B3GNT3, TMEM40, USP6NL, GABRP, ZSCAN12P1
##     CERS3, PSCA, TMEM253, SPACA4, GRB7, NCCRP1, MFSD9, CSTA, KLK14, LINC00857
## Negative:  EFEMP1, MMP2, IFITM3, C3, LTBP2, CTSK, SPON1, PLTP, SERPINF1, TMEM119
##     IGFBP6, FOS, LUM, SRPX, CAPN6, MXRA5, TSKU, PLA2G2A, STAT1, ATP1B3
##     ARAP1, ROR2, CFD, MGST1, EPHA3, SUPT20HL1, AOX1, TMEM176B, ARL2, IL33
## PC_ 3
## Positive:  CT45A5, PRODH, CLDN19, KRT8, ALG1L, CT45A3, ZFP69B, HIST1H2AG, MAGEA9, HLA-DOA
##     BCAM, EMR1, CTAG2, IL2RG, GAGE4, KRTCAP3, NUDT16P1, DPH2, C11orf92, IL20RA
##     LCP1, TNF, SOSTDC1, PARM1, FAM211B, PPIL1, LAMB3, ABCC5, FAM3B, MKRN3
## Negative:  ECM1, MMP2, MALL, VAT1, EFEMP1, CTSK, PPIC, RNASE7, SPRR2D, C3
##     LTBP2, FIBIN, TMEM119, EPS8L1, SPON1, RNF223, HEBP2, SLC9A5, CAPN6, KLHL20
##     RHOT1, SDCBP2, PLA2G2A, PPL, NDUFS4, PRSS22, AGPAT9, SRPX, MFAP5, S100P
## PC_ 4
## Positive:  MCAM, SNHG15, HSPB1, FHL5, SLC38A5, MRVI1, TINAGL1, C4B, SLC38A11, ACAT1
##     CAMK2G, TMEM54, TUBB6, MAPRE2, SLC2A4, PTPLA, CTNNAL1, IFITM3, GCDH, PLCE1
##     RARS, C11orf96, PSMG2, RASSF1, ITGA7, MYOCD, STXBP3, SLC39A13, GUCY1B3, RPUSD2
## Negative:  MGST1, LUM, CLDN1, CD55, PRODH, ABCA6, ELF3, MUC4, LTBP4, ZNF85
##     ZFP69B, HLA-DRA, IGSF11, CFD, LTBP2, ABCC5, PIR, ALG1L, KRT8, CAPN6
##     HLA-DOA, FXYD6, CT45A5, SERPINA3, LCP1, LOC100506233, C3, STOX2, MT1H, AOX1
## PC_ 5
## Positive:  DMKN, IER2, FAM83A, CYP2S1, VSNL1, KRT6C, SLC38A5, TMEM171, CKMT1B, CBLC
##     GRHL3, LAD1, DEPDC7, KRT5, FOS, MMP13, C6orf141, S100A14, CXCL1, SERPINB2
##     IGFL1, KRT15, DUOXA1, LAPTM4B, ZFP36, LOC154761, TM4SF1, ZNF133, RCAN3, NCCRP1
## Negative:  LMOD2, CSRP3, ACTA1, MLIP, TNNT3, ACTN2, CAV3, TNNI2, AQP7P1, SNX29P1
##     C1orf145, PLA1A, PPP1R3A, NEDD8-MDP1, C8G, RASGRP3, GNLY, ZFP37, SLC13A3, DDIT4L
##     AGPAT9, GPKOW, CXorf40A, POPDC3, EDA2R, MYBPC2, IL21R, IP6K3, ZNF230, RAPSN
```

```
  sc_data2 <- Seurat::FindNeighbors(object = sc_data2, dims = 1:5, verbose = TRUE)
```

```
## Computing nearest neighbor graph
```

```
## Computing SNN
```

```
  sc_data2 <- Seurat::FindClusters( object = sc_data2, resolution = 0.6, verbose = TRUE)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 100
## Number of edges: 2232
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.6320
## Number of communities: 3
## Elapsed time: 0 seconds
```

```
  sc_data2 <- Seurat::RunUMAP(object = sc_data2, dims = 1:5, verbose = TRUE)
```

```
## 08:01:05 UMAP embedding parameters a = 0.9922 b = 1.112
```

```
## 08:01:05 Read 100 rows and found 5 numeric columns
```

```
## 08:01:05 Using Annoy for neighbor search, n_neighbors = 30
```

```
## 08:01:05 Building Annoy index with metric = cosine, n_trees = 50
```

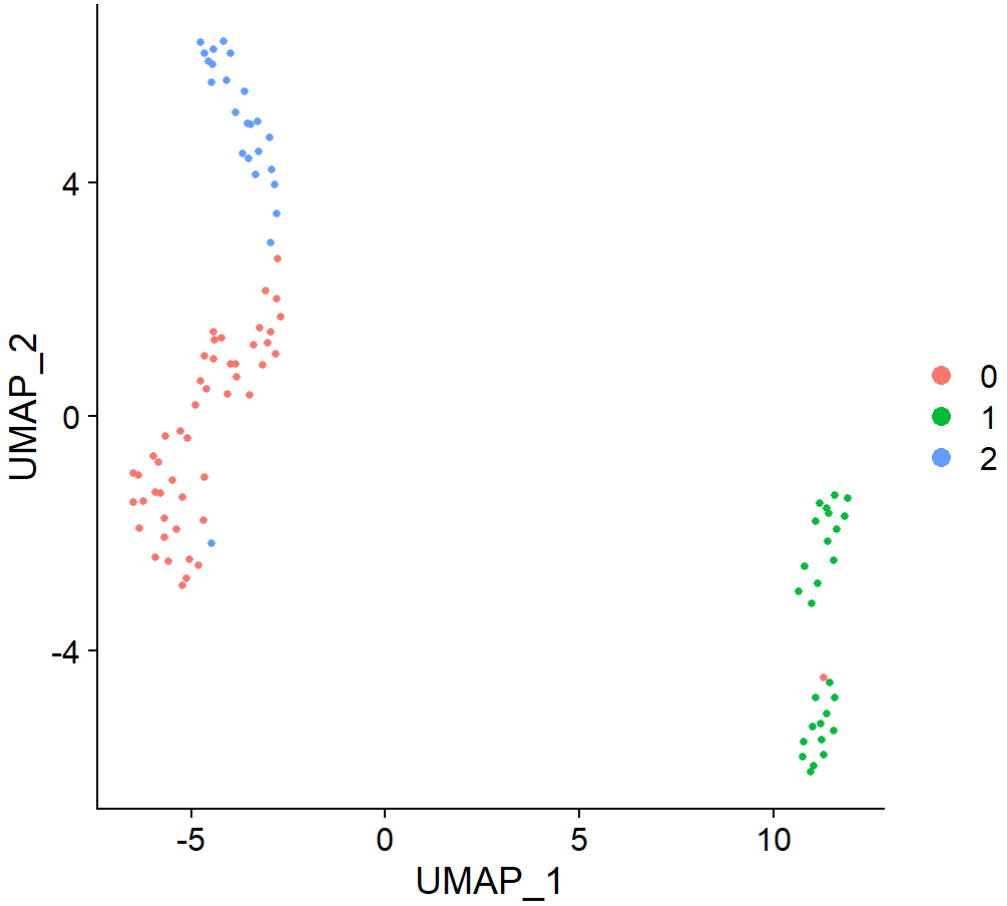
```
## 0%   10   20   30   40   50   60   70   80   90   100%
```

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

```
## **************************************************|
## 08:01:05 Writing NN index file to temp file C:\Users\NODE5\AppData\Local\Temp\RtmpGKdRjq\fil
e46dc22b1d87
## 08:01:05 Searching Annoy index using 1 thread, search_k = 3000
## 08:01:05 Annoy recall = 100%
## 08:01:05 Commencing smooth kNN distance calibration using 1 thread
## 08:01:06 Initializing from normalized Laplacian + noise
## 08:01:06 Commencing optimization for 500 epochs, with 2618 positive edges
## 08:01:06 Optimization finished
```

we can visualize these 100 cells using the UMAP coordinates:

```
Seurat::DimPlot(sc_data2, reduction = 'umap', label.size = 5)
```



Prepare the bulk data, phenotype labels and PPI data

We load in the preprocessed HNSC bulk expression matrix and the corresponding phenotype data.And then we locate the PPI data.

```
data("case1_bulk_data_200to300")
bulk_data <- case1_bulk_data_200to300
data("case1_sample_pheno_labels_200to300")
phenotype_data <- case1_sample_pheno_labels_200to300
ppi_location <- system.file("extdata", "ppi.csv", package = "PACSI")
```

In this bulk data, each row represents a gene and each column represents a sample. The bulk expression matrix should be TPM/FPKM-normalized. The dimensions of the bulk data are:

```
dim(bulk_data)
```

```
## [1] 5000  101
```

It shows this data has 5000 genes and 101 samples in total. Users do not need to keep the common genes between the single-cell and bulk samples. Besides, all of these samples have phenotype labels:

```
head(phenotype_data)
```

```
## TCGA-CR-7388-01A TCGA-CR-7389-01A TCGA-CR-7390-01A TCGA-CR-7391-01A
##                1                1                1                1
## TCGA-CR-7392-01A TCGA-CR-7393-01A
##                1                1
```

The location of PPI data is:

```
ppi_location
```

```
## [1] "D:/software/R/R-4.1.1/library/PACSI/extdata/ppi.csv"
```

Execute PACSI to select the cells associated with disease phenotype

Given the above inputs, we can use PACSI to select the phenotype-associated cell subpopulations:

```
PACSI_result <- PACSI(bulk_data, phenotype_data, sc_data, ppi_data = ppi_location,times = 10,nc
ores = 2)
```

```
## [1] "build the graph"
## [1] "Data preprocessing"
## [1] "Calculate the cell signatures and sample signatures"
```

```
## 载入需要的程辑包：org.Hs.eg.db
```

```
## 载入需要的程辑包：AnnotationDbi
```

```
## 载入需要的程辑包：stats4
```

```
## 载入需要的程辑包：BiocGenerics
```

```
## 载入需要的程辑包：parallel
```

```
##
## 载入程辑包：'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, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min
```

```
## 载入需要的程辑包：Biobase
```

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

```
## 载入需要的程辑包：IRanges
```

```
## 载入需要的程辑包：S4Vectors
```

```
##
## 载入程辑包：'S4Vectors'
```

```
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
```

```
##
## 载入程辑包：'IRanges'
```

```
## The following object is masked from 'package:grDevices':
##
##     windows
```

```
##
```

```
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
```

```
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
```

```
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
```

```
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
```

```
## [1] "Calculate the distance between cells and the phenotype"
## [1] "permutation"
## [1] "calculate p value"
```

```
head(PACSI_result$result)
```

	cells_scores <dbl>	p.value <dbl>	FDR <dbl>	cell_phenotype_labels <dbl>
HN28_P6_G05_S173_comb	2.041540	0.0	0.0000000	1
HN25_P5_E10_S58_comb	2.038839	0.0	0.0000000	1
HN26_P6_B06_S18_comb	2.169671	0.0	0.0000000	1
HN25_P5_C05_S29_comb	2.177160	0.4	0.4081633	0
HN28_P6_D09_S141_comb	2.150787	0.1	0.1086957	0
HN26_P5_H01_S85_comb	2.126974	0.0	0.0000000	1

6 rows

In total, PACIS identifies 72 cells associated with HNSC:

```
table(PACSI_result$result$cell_phenotype_labels)
```

```
##
##  0  1
## 22 78
```

We can visualize the Scissor selected cells:

```
library(Seurat)
  sc_data2 <- Seurat::CreateSeuratObject(as.matrix(sc_data))
  sc_data2 <- Seurat::FindVariableFeatures(object = sc_data2, verbose = F, nfeatures=2000)
  sc_data2 <- Seurat::ScaleData(object = sc_data2, verbose = F)
  sc_data2 <- Seurat::RunPCA(object = sc_data2, features = Seurat::VariableFeatures(sc_data2),
 verbose = TRUE)
```

```
## PC_ 1
## Positive:  KRT5, TACSTD2, KRT18, PKP3, DSP, LAMB3, KRT15, IRF6, CBLC, ST6GALNAC2
##     S100A14, KRT8, TNFSF10, LAD1, DUOXA1, AKIRIN1, SLC35F2, GRB7, PTTG1, FZD6
##     GNL2, ELF3, NME1, SRD5A1, PYCARD, GLRX3, TM4SF1, BRIX1, NDUFA12, HDAC1
## Negative:  EGR1, FOS, FHL1, SERPING1, ZFP36, TXNIP, SLC2A3, C4B, GPX3, HSPB6
##     FHL5, RGS16, MCAM, ITGA7, LGI4, MRVI1, DPYSL2, MT1A, UGGT2, RNASE4
##     RCAN2, GGT5, CYR61, C11orf96, SERPINF1, ZNF331, LTBP4, HEYL, PLCE1, MYOCD
## PC_ 2
## Positive:  RNASE7, SPRR2D, EPS8L1, SDCBP2, PRSS22, AGPAT9, TMEM125, S100P, TMPRSS11F, RNF223
##     ERO1L, IL1RN, KLK7, RAET1G, ATG9B, B3GNT3, TMEM40, USP6NL, GABRP, ZSCAN12P1
##     CERS3, PSCA, TMEM253, SPACA4, GRB7, NCCRP1, MFSD9, CSTA, KLK14, LINC00857
## Negative:  EFEMP1, MMP2, IFITM3, C3, LTBP2, CTSK, SPON1, PLTP, SERPINF1, TMEM119
##     IGFBP6, FOS, LUM, SRPX, CAPN6, MXRA5, TSKU, PLA2G2A, STAT1, ATP1B3
##     ARAP1, ROR2, CFD, MGST1, EPHA3, SUPT20HL1, AOX1, TMEM176B, ARL2, IL33
## PC_ 3
## Positive:  CT45A5, PRODH, CLDN19, KRT8, ALG1L, CT45A3, ZFP69B, HIST1H2AG, MAGEA9, HLA-DOA
##     BCAM, EMR1, CTAG2, IL2RG, GAGE4, KRTCAP3, NUDT16P1, DPH2, C11orf92, IL20RA
##     LCP1, TNF, SOSTDC1, PARM1, FAM211B, PPIL1, LAMB3, ABCC5, FAM3B, MKRN3
## Negative:  ECM1, MMP2, MALL, VAT1, EFEMP1, CTSK, PPIC, RNASE7, SPRR2D, C3
##     LTBP2, FIBIN, TMEM119, EPS8L1, SPON1, RNF223, HEBP2, SLC9A5, CAPN6, KLHL20
##     RHOT1, SDCBP2, PLA2G2A, PPL, NDUFS4, PRSS22, AGPAT9, SRPX, MFAP5, S100P
## PC_ 4
## Positive:  MCAM, SNHG15, HSPB1, FHL5, SLC38A5, MRVI1, TINAGL1, C4B, SLC38A11, ACAT1
##     CAMK2G, TMEM54, TUBB6, MAPRE2, SLC2A4, PTPLA, CTNNAL1, IFITM3, GCDH, PLCE1
##     RARS, C11orf96, PSMG2, RASSF1, ITGA7, MYOCD, STXBP3, SLC39A13, GUCY1B3, RPUSD2
## Negative:  MGST1, LUM, CLDN1, CD55, PRODH, ABCA6, ELF3, MUC4, LTBP4, ZNF85
##     ZFP69B, HLA-DRA, IGSF11, CFD, LTBP2, ABCC5, PIR, ALG1L, KRT8, CAPN6
##     HLA-DOA, FXYD6, CT45A5, SERPINA3, LCP1, LOC100506233, C3, STOX2, MT1H, AOX1
## PC_ 5
## Positive:  DMKN, IER2, FAM83A, CYP2S1, VSNL1, KRT6C, SLC38A5, TMEM171, CKMT1B, CBLC
##     GRHL3, LAD1, DEPDC7, KRT5, FOS, MMP13, C6orf141, S100A14, CXCL1, SERPINB2
##     IGFL1, KRT15, DUOXA1, LAPTM4B, ZFP36, LOC154761, TM4SF1, ZNF133, RCAN3, NCCRP1
## Negative:  LMOD2, CSRP3, ACTA1, MLIP, TNNT3, ACTN2, CAV3, TNNI2, AQP7P1, SNX29P1
##     C1orf145, PLA1A, PPP1R3A, NEDD8-MDP1, C8G, RASGRP3, GNLY, ZFP37, SLC13A3, DDIT4L
##     AGPAT9, GPKOW, CXorf40A, POPDC3, EDA2R, MYBPC2, IL21R, IP6K3, ZNF230, RAPSN
```

```
  sc_data2 <- Seurat::FindNeighbors(object = sc_data2, dims = 1:5, verbose = TRUE)
```

```
## Computing nearest neighbor graph
```

```
## Computing SNN
```

```
  sc_data2 <- Seurat::FindClusters( object = sc_data2, resolution = 0.6, verbose = TRUE)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 100
## Number of edges: 2232
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.6320
## Number of communities: 3
## Elapsed time: 0 seconds
```

```
  sc_data2 <- Seurat::RunUMAP(object = sc_data2, dims = 1:5, verbose = TRUE)
```

```
## 08:03:22 UMAP embedding parameters a = 0.9922 b = 1.112
```

```
## 08:03:22 Read 100 rows and found 5 numeric columns
```

```
## 08:03:22 Using Annoy for neighbor search, n_neighbors = 30
```

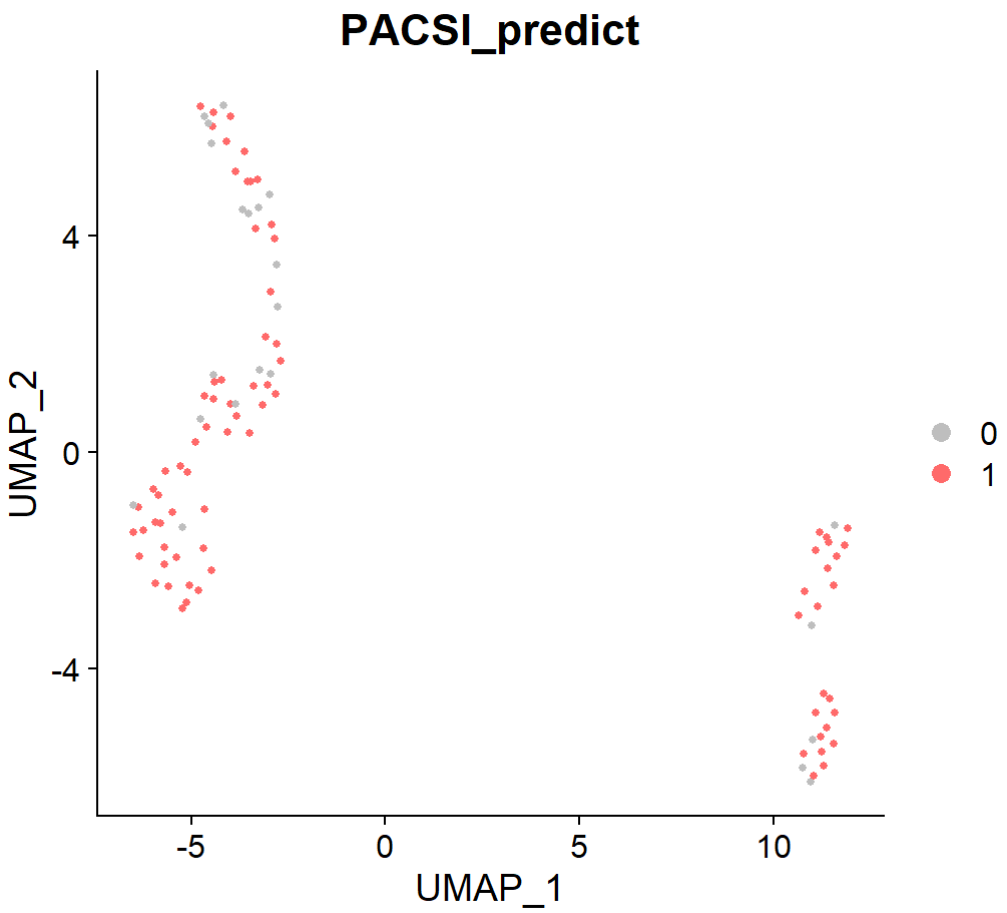
```
## 08:03:22 Building Annoy index with metric = cosine, n_trees = 50
```

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

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

```
## **************************************************|
## 08:03:22 Writing NN index file to temp file C:\Users\NODE5\AppData\Local\Temp\RtmpGKdRjq\fil
e46dc4e3d24bf
## 08:03:22 Searching Annoy index using 1 thread, search_k = 3000
## 08:03:22 Annoy recall = 100%
## 08:03:22 Commencing smooth kNN distance calibration using 1 thread
## 08:03:23 Initializing from normalized Laplacian + noise
## 08:03:23 Commencing optimization for 500 epochs, with 2618 positive edges
## 08:03:23 Optimization finished
```

```
  sc_data3 <- AddMetaData(sc_data2, metadata = PACSI_result[["result"]]$cell_phenotype_labels,
 col.name = "PACSI_predict")
  DimPlot(sc_data3, reduction = 'umap', group.by = 'PACSI_predict', cols = c('grey','indianred
1'))
```



At last, we can get access to the parameters using:

```
PACSI_result[["parameter"]]
```

```
## $ppi_data
## [1] "D:/software/R/R-4.1.1/library/PACSI/extdata/ppi.csv"
##
## $permutation.times
## [1] 10
##
## $ncores
## [1] 2
##
## $sc_VariableFeatures
## [1] 2000
##
## $sample_signature_threshold
## [1] 150
##
## $cell_signature_threshold
## [1] 150
##
## $adjacent.matrix_p_threshold
## [1] 0.05
##
## $cell_population_FDR_threshold
## [1] 0.05
```