

# Step 系列：scRNA-seq 基本分析

2024-02-06

LiChuang Huang



@ 立效研究院

# Contents

<b>1 Step 系列共性特征</b>	<b>1</b>
1.1 理念	1
1.2 局限性	2
1.3 泛用方法	2
1.3.1 提取方法	3
1.3.2 存储方法	3
1.3.3 空间管理	3
1.3.4 查看 step 方法的默认参数	3
1.3.5 额外的信息	4
1.4 关于安装配置	4
1.5 关于本文档	4
<b>2 Step 系列: scRNA-seq 基本分析</b>	<b>4</b>
2.1 摘要	4
2.1.1 目的	4
2.1.2 解决的问题	4
2.2 适配性	4
2.3 方法	4
2.4 安装 (首次使用)	5
2.4.1 安装依赖	5
2.4.1.1 一些额外可能需要的系统依赖工具	5
2.4.1.2 安装 Seurat v5	5
2.4.1.3 安装 seurat-wrappers	5
2.4.1.4 安装 monocle3	6
2.4.1.5 安装 CellChat	6
2.4.1.6 安装 SCSA	6
2.4.1.7 其它程序	6
2.4.2 安装主体	7
2.5 使用说明	7
2.6 示例分析	7
2.6.1 数据准备 (从 GEO 的单细胞数据库开始分析)	7
2.6.1.1 快速获取示例数据	7
2.6.1.2 一些补充说明 (上述快速获取数据方式的实用价值)	8
2.6.2 分析流程	9
2.6.2.1 Job-seurat 从 Seurat 开始单细胞数据分析	9
2.6.2.2 Step1 数据质控	9
2.6.2.3 Step2 根据上一步 QC 作图过滤数据并标准化	10
2.6.2.4 Step3 完成降维、聚类和 Marker 筛选	12
2.6.2.5 Step4 使用 SingleR 注释细胞类	13
2.6.2.6 Step5 计算各细胞类的 Marker 基因 (差异分析)	14

2.6.2.7	Step6 使用 SCSA 注释细胞类 . . . . .	15
2.6.2.8	从以下开始拟时分析的示例 . . . . .	16
2.6.2.9	As-job-monocle 对选定的细胞进行拟时分析 . . . . .	16
2.6.2.10	Step1 构建拟时轨迹 . . . . .	18
2.6.2.11	Step2 选择拟时起点 . . . . .	20
2.6.2.12	Step3 拟时分析基础上的差异分析和基因表达模块 . . . . .	22
2.6.2.13	(进阶) 根据拟时分析结果重新划分细胞群体 . . . . .	24
2.6.2.14	以下开始细胞通讯的示例分析 . . . . .	25
2.6.2.15	As-job-cellchat 对选定的细胞进行细胞通讯分析 . . . . .	25
2.6.2.16	Step1 构建通讯网络以及可视化 . . . . .	26
2.6.2.17	Step2 进一步分析通路通讯、受体配体通讯和可视化 . . . . .	29
2.6.2.18	(进阶) . . . . .	30
2.6.3	完整示例代码 . . . . .	30
2.7	技巧 . . . . .	30

<b>Reference</b>	<b>30</b>
------------------	-----------

## List of Figures

1	Workflow frame overview. 左图展示的是 Step 系列所有对象的框架结构和运行路线。右图展示的是，每一个圆球都代表一个方法或数据库或分析平台形成的数据对象，也就是左图中的‘job’，而它们之间的线，代表所有对象之间的转化或映射关系（仅目前；还在不断拓展）；具体而言，我们可以简单的通过‘map’或‘asjob’这类的方法，将一个数据对象转化或映射到另一个对象，实现跨越多种方法或体系的联并分析。 . . . . .	1
2	Quality Control . . . . .	10
3	Ranking of principle components . . . . .	12
4	UMAP Clustering . . . . .	13
5	SCSA Cell type annotation . . . . .	16
6	Trajectories . . . . .	19
7	Principal points . . . . .	20
8	Deomo subset of cells visualization . . . . .	21
9	SUB pseudotime . . . . .	22
10	SUB gene module heatmap . . . . .	23
11	Sub the regroup by hclust . . . . .	24
12	The cell mapped from monocle . . . . .	25
13	Overall communication count . . . . .	27
14	Communication bubble . . . . .	30

## List of Tables

1	All Markers . . . . .	14
2	SUB graph test results . . . . .	23

3	Lp net . . . . .	28
4	Pathway net . . . . .	28

## 1 Step 系列共性特征

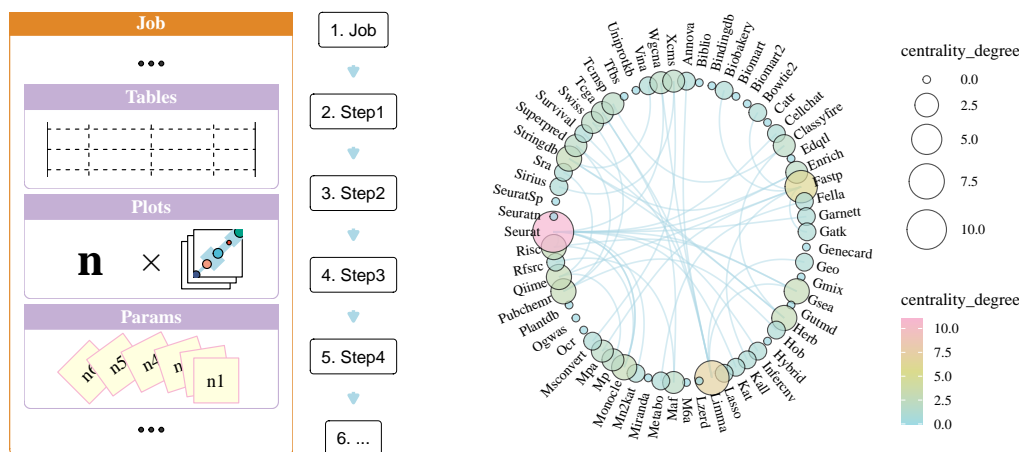


Figure 1: Workflow frame overview. 左图展示的是 Step 系列所有对象的框架结构和运行路线。右图展示的是，每一个圆球都代表一个方法或数据库或分析平台形成的数据对象，也就是左图中的 ‘job’，而它们之间的线，代表所有对象之间的转化或映射关系（仅目前；还在不断拓展）；具体而言，我们可以简单的通过 ‘map’ 或 ‘asjob’ 这类的方法，将一个数据对象转化或映射到另一个对象，实现跨越多种方法或体系的联并分析。

## 1.1 理念

生物信息分析工具种类繁多，开发者出于各种原因，导致工具的适用性千差万别。学习和应用这些工具的成本可能是高昂的（甚至它们可能有一些不为人知的漏洞），想要把它们串连在一起分析，更是要付出一些分析之外的代价：像开发者一样调试这些程序。

Step 系列的分析方法为每一个制定好的方法、思路或工具（一般是领域中的权威、经典或翘楚）设立统一使用的标准，统一的数据存储，统一的应用流程，大幅度降低了学习成本、使用成本、应用成本。通过避免大量“分析之外”的繁琐工作，达到提高分析效率的目的。

Step 系列分析方法的一些基本特性:

- 统一的分析平台。Step 背后涵盖的工具可能涉及：R 包、Python 包、Java 包、Linux 命令行工具等。但最终用于分析的始终是 R 语言。Step 系列的方法通过在 R 中调用各种分析工具，实现不同分析平台之间的工具调用和数据对接 (降低了学习和应用成本)。
- 统一的数据存储单位。Step 系列所有的分析流程 (workflow) 都以一个对象 (Object) 存储。这避免了如果一个流程中涉及纷繁复杂的分析方法，人工存储数据 (中间数据、最终输出的图片、表格) 出错的可能 (提高效率)。必要时，通过统一的方法提取这些数据 (就像在图书馆的某一层的某一排的书架的某一个柜子取出一本书)。
- 统一的方法名称。Step 系列的所有 Workflow 的方法名称都是统一的 (step1、step2、step3..... map、vis 等)，但不会因为一同调用而出错。这是为了减免分析者的负担而设计的 (如果一次完整分析涉及十几种工具，每个工具又有十几种方法名称，那对分析者的记忆量和细心度的考验是惊人的) (提高效率)。
- 规范的分析流程。大多数的分析工具本身具备多种方法以适用灵活分析，但也提高了学习、应用成本。

Step 系列的各个 Workflow 的流程都是单向的，对分析方法或思路的组合应用大都是建立在官方指南或教程的基础上，又或者是泛用性 (降低了学习成本)。如果有不同思路，可以通过关键参数调整方法，或者视情况搭建一个额外的 Workflow。因此，每个 Workflow 是以分析思路为分门别类的，而不是工具本身。

- 提供权威、经典、泛用的分析方法。Workflow 创建前，会广泛查阅时下文献，对工具择优而选、择新而用 (比如，更趋向于选择更新的来自于 Nature Biotechnology 的分析方法)。
- 提供泛用的组合思路。通过组合各种数据库、分析工具，应对千奇百怪的分析需求。同时，一些适当的组合，会发挥超出单一工具的价值 (Fig. ?? 的右图提供了现如今存在的许多组合思路；同时，每个 Workflow 内部又存在一些思路组合)。
- 不断开发进化。每一个完成的 Workflow 不是固定不变的，在面临新的分析环境、新的分析需求，更多的功能会被加入到“step”方法中，让每一次的进步都直接应用于今后所有的同类分析。另外，一些大大小小的创新 (比如，更严谨或更酷炫的绘图) 也会不断追加到方法中。
- 效率至上。所有分析方法以输入命令形式实现，允许批量处理。
- 附带实用工具。分析流程相对固定，但通过提供一些工具 (比如用于额外的绘图)，可以将分析更加灵活。

## 1.2 局限性

- 适配性。由于多数生信工具都基于 Linux，此外 Step 系列在编写时，也只在 Linux 上调试过；所以，这些工具将只适用于 Linux 系统。

## 1.3 泛用方法

R input

```
sr <- job_seurat("./data")
sr <- step1(sr)
sr <- step2(sr)
sr <- step3(sr)
sr <- step4(sr)
sr <- step5(sr)
sr <- step6(sr)

mn <- asjob_monocle(sr)
mn <- step1(mn)
mn <- step2(mn)
mn <- step3(mn)
```

### 1.3.1 提取方法

### 1.3.2 存储方法

R input

```
autosv(sr@plots$step1)
```

### 1.3.3 空间管理

某些 R 包生成的数据或加载的数据可能极其占用运行内存 (RAM)。可以用 `space()` 函数查看当前内存占用：

R input

```
space()
```

### 1.3.4 查看 step 方法的默认参数

step 方法的参数力求精简，一般只保留关键的参数用以控制分析。但这些参数可能会在将来被拓展 (添加额外的参数) 以适应新的分析需求。

可以通过类似以下方式查看默认参数：

R input

```
## 示例: seurat 工作流的 step1 的默认参数
not(.job_seurat())
step1
```

```
## job_seurat:
```

```
##      x
```

```
##
```

```
## -- Methods parameters -----
```

R input

```
## 示例: monocle 工作流 step2 的默认参数
not(.job_monocle())
step2
```

```
## job_monocle:
```

```
##      x, roots
```

```
##
```

```
## -- Methods parameters -----
```

### 1.3.5 额外的信息

工作流创建时参考的信息源参考文献信息

R input

```
# 创建一个空的 Seurat 工作流对象 'sr_1'  
sr_1 <- .job_seurat()  
# 查看方法说明  
sr_1@method
```

```
## [1] "The R package `Seurat` used for scRNA-seq processing; `SCSA` (python) used for cell type annota
```

R input

```
# 查看官方网站或信息源网站  
sr_1@info
```

```
## [1] "Tutorial: https://satijalab.org/seurat/articles/pbmc3k\_tutorial.html"
```

## 1.4 关于安装配置

所有 Step 系列的方法的安装配置会尽可能详细的罗列，但由于笔者（开发者）仅在自身的 Linux (Ubuntu 发行版) 系统做过调试，并不确定在其它的机器上会遇到哪些安装的特殊问题。如有疑问，或安装上的困难，请联系：Huang Lichuang (huanglichuang@wie-biotech.com)。

## 1.5 关于本文档

# 2 Step 系列：scRNA-seq 基本分析

## 2.1 摘要

### 2.1.1 目的

一般化 scRNA-seq 的分析流程，从基本的数据处理，到细胞注释，再到拟时分析、通讯分析等。数据处理的中心在于‘Seurat’，以它为衔接点，从各个分析工具中将数据转换来去，延续彼此的分析。

### 2.1.2 解决的问题

不同的 R 包或其他工具之间的数据衔接。

## 2.2 适配性

大多数涉及的程序都是 R；但是，我设计的 SCSA 的程序接口可能得在 Linux 下才能成功运行。

## 2.3 方法

以下是我在这个工作流中涉及的方法和程序：



Mainly used method:

- R package **CellChat** used for cell communication analysis<sup>1</sup>.
- GEO <https://www.ncbi.nlm.nih.gov/geo/> used for expression dataset aquisition.
- R package **Monocle3** used for cell pseudotime analysis<sup>2,3</sup>.
- The R package **Seurat** used for scRNA-seq processing; **SCSA** (python) used for cell type annotation<sup>4-6</sup>.
- Other R packages (eg., **dplyr** and **ggplot2**) used for statistic analysis or data visualization.

## 2.4 安装 (首次使用)

### 2.4.1 安装依赖

#### 2.4.1.1 一些额外可能需要的系统依赖工具

如果你是 Ubuntu 发行版, 据我的经验, 安装 **devtools**, **BiocManager** 等工具之前, 估计需要先安装以下:

Bash input

```
## Libraries for installing 'usethis' and 'devtools'.
sudo apt install -y libssl-dev libcurl4-openssl-dev libblas-dev
sudo apt install -y liblapack-dev libgfortran-11-dev gfortran libharfbuzz-dev libfribidi-dev
## Libraries for installing 'BiocManager' and its some packages.
sudo apt install -y libnetcdf-dev libopenbabel-dev libeigen3-dev
## Libraries For installing other graphic packages.
sudo apt install -y libfontconfig1-dev librsvg2-dev libmagick++-dev
```

#### 2.4.1.2 安装 Seurat v5

R input

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
if (!requireNamespace("remotes", quietly = TRUE))
  install.packages("remotes")
install.packages(c("sva"))
BiocManager::install(c('SparseArray', 'fastDummies', 'RcppHNSW', 'RSpectra'))
remotes::install_github("satijalab/seurat", "seurat5")
```

#### 2.4.1.3 安装 seurat-wrappers

R input

```
remotes::install_github('satijalab/seurat-wrappers')
```

#### 2.4.1.4 安装 monocle3

R input

```
BiocManager::install(c('BiocGenerics', 'DelayedArray', 'DelayedMatrixStats',  
  'limma', 'lme4', 'S4Vectors', 'SingleCellExperiment',  
  'SummarizedExperiment', 'batchelor', 'HDF5Array',  
  'terra', 'ggrastr', 'rsample'))  
remotes::install_github('cole-trapnell-lab/monocle3')
```

#### 2.4.1.5 安装 CellChat

R input

```
BiocManager::install(c('NMF', 'circlize', 'ComplexHeatmap', 'BiocNeighbors'))  
remotes::install_github("sqjin/CellChat")
```

#### 2.4.1.6 安装 SCSA

以下代码是在 bash 中运行的。确保你的 git 和 pip3 可用。

Bash input

```
git clone https://github.com/bioinfo-ibms-pumc/SCSA.git ~/SCSA  
pip3 install numpy scipy openpyxl  
# pandas 版本太高会报错  
pip3 install pandas==1.5.3
```

#### 2.4.1.7 其它程序

以下可能是其它需要安装的程序：

R input

```
install.packages("reticulate")  
reticulate::install_miniconda()  
reticulate::py_install(packages = 'umap-learn')
```

你可能还想要安装：

R input

```
BiocManager::install(c("SingleR"))  
BiocManager::install(c("cellidex"))
```

## 2.4.2 安装主体

Bash input

```
git clone https://github.com/shaman-yellow/utils.tool.git ~/utils.tool
```

`utils.tool` 是标准的 R 包结构形式，这意味着，即使你不用 `git` 获取它，单纯用：

- `remotes::install_github("shaman-yellow/utils.tool")`

也能成功获取并直接安装完成。但是，这包里面大多数的方法都没有导出 (export) 到用户层次 (这是因为，这个包的改动情形太多了，我一直在创建新的方法或者调整旧的方法)，你即使 `library(utils.tool)` 加载了它，也会出现使用不了许多方法的情况。万无一失的做法是：

R input

```
if (!requireNamespace("devtools", quietly = TRUE))  
  install.packages("devtools")  
# 前提当然是，你已经 git 获取了这个包了  
devtools::load_all("~/utils.tool")
```

## 2.5 使用说明

## 2.6 示例分析

### 2.6.1 数据准备 (从 GEO 的单细胞数据库开始分析)

10X Genomics (其他格式也行，但我遇到过的几乎都是 10X 的，所以其他格式的没机会调试) 的文件。这里为了方便起见，我从 GEO 下载一批数据实时以代码示例。

#### 2.6.1.1 快速获取示例数据

运行以下代码获取数据：

R input

```
geo <- job_geo("GSE171306")  
geo <- step1(geo)  
geo <- step2(geo)  
untar("./GSE171306/GSE171306_RAW.tar", exdir = "./GSE171306")  
prepare_10x("./GSE171306/", "ccRCC1", single = F)  
# 创建 job_seurat 对象：  
sr <- job_seurat("./GSE171306/GSM5222644_ccRCC1_barcode")
```

### 2.6.1.2 一些补充说明 (上述快速获取数据方式的实用价值)

`job_geo` 是另外一个可以用于高效获取、整理 GEO 数据集的 step 系列工作流 (以后介绍)，简而言之，它的作用在于帮我们极速整理好数据还有元数据 (整合大量数据集的时候很有用!)。例如，我们可以通过以下方式查看这批 GEO 数据的样品信息和数据前处理：

R input

```
geo$guess
```

```
## # A tibble: 2 x 3
##   rownames      title                                     tissue.ch1
##   <chr>         <chr>                                     <chr>
## 1 GSM5222644 ccRCC1, Right clear cell renal cell carcinoma Homogenate Right clear cell renal cell carcinoma
## 2 GSM5222645 ccRCC2, Left clear cell renal cell carcinoma Homogenate Left clear cell renal cell carcinoma
```

R input

```
geo$prods
```

```
## Preliminary sequencing results (bcl files) were converted
## to FASTQ files with Cell Ranger V3.1
##
## R1 end, at the beginning of 16bp is CellBarcode sequence,
## then 10bp is UMI sequence, R2 end, we can truncate 151bp to
## 98bp
##
## The CellRanger (10X Genomics) secondary analysis pipeline
## was used to generate a digital gene expression matrix
##
## Normalization and additional analysis by Seurat R package
##
## Genome_build: GRCh38 for human data
##
## Supplementary_files_format_and_content: CellRanger output
## files (the barcode, gene, expression matrix file of each
## sample)
```

确认解压得到了些什么文件：

R input

```
list.files("./GSE171306/")
```

```
## [1] "GSE171306_RAW.tar" "GSM5222644_ccRCC1_barcode" "GSM5222645_ccRCC2_barcode"
## [4] "GSM5222645_ccRCC2_features.tsv.gz" "GSM5222645_ccRCC2_matrix.mtx.gz"
```

GEO 中的单细胞数据 (Supplementary file) 的存储形式不统一 (很随意)，一一整理起来用于输入很繁琐。`prepare_10x` 是我写的一个将同一样本三个文件 (barcodes.tsv, matrix.mtx, features.tsv) 存放于一个目录中的高效办法。

R input

```
# 如果有三个文件:
prepare_10x("./GSE171306/", "ccRCC1", single = F)
# 如果只有一个 Matrix 文件:
prepare_10x("./GSE171306/", "ccRCC1", single = T)
```

## 2.6.2 分析流程

### 2.6.2.1 Job-seurat 从 Seurat 开始单细胞数据分析

在 2.6.1.1 中, 已经运行过:

R input

```
sr <- job_seurat("./GSE171306/GSM5222644_ccRCC1_barcodes")
```

如你不想用 Step 系列的风格来分析, 更想用 Seurat 的原生代码, 那么你可以:

R input

```
seurat <- object(sr)
# 这样, `seurat` 就是你需要的数据对象。
# 你可以参考: <https://satijalab.org/seurat/articles/pbmc3k\_tutorial.html>
# 不借助 Step 的默认设定, 而开展自主分析。
```

### 2.6.2.2 Step1 数据质控

只需要运行:

R input

```
# 这一步不需要输入参数
sr <- step1(sr)
```

R input

```
# 获取运行的结果
sr@plots$step1$p.qc
```

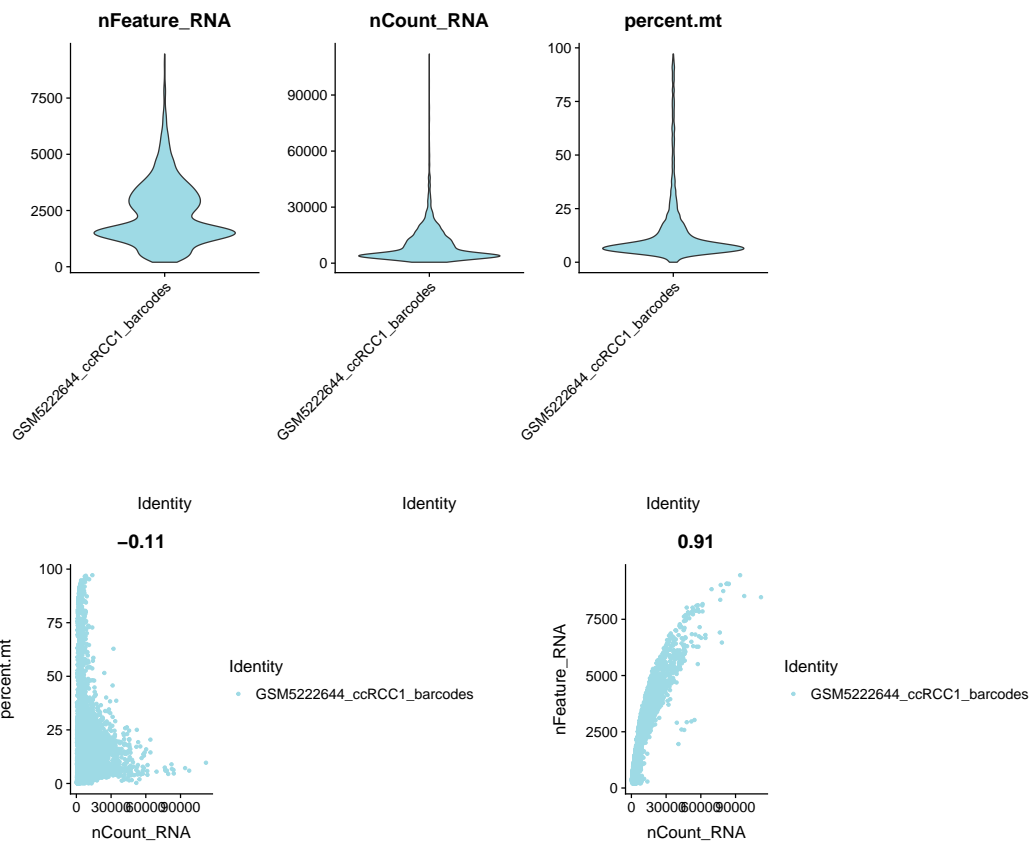


Figure 2: Quality Control

Fig. 2, 你可能会觉得 x 轴坐标太长了, 如果在此前 (创建对象的时候) 输入类似 `job_seurat("./GSE171306/", project = "Demo data")`, 就可以避免。

### 2.6.2.3 Step2 根据上一步 QC 作图过滤数据并标准化

这里的参数相当重要。

R input

```
sr <- step2(sr, 0, 7500, 35)
```

可以确认默认的参数:

R input

```
not(sr)
step2
```

## job\_seurat:

```
## x, min.features, max.features, max.percent.mt = 5, nfeatures = 2000, use = "nFeature_RNA"
##
```

```
## -- Methods parameters -----
```

输入的三个参数对应以下：

- min.features
- max.features
- max.percent.mt

官网对 `percent.mt` 做了一些解释：

- The number of unique genes detected in each cell.
  - Low-quality cells or empty droplets will often have very few genes
  - Cell doublets or multiplets may exhibit an aberrantly high gene count
- Similarly, the total number of molecules detected within a cell (correlates strongly with unique genes)
  - The percentage of reads that map to the mitochondrial genome
  - Low-quality / dying cells often exhibit extensive mitochondrial contamination
- We calculate mitochondrial QC metrics with the `PercentageFeatureSet()` function, which calculates the percentage of counts originating from a set of features
  - We use the set of all genes starting with MT- as a set of mitochondrial genes

总而言之，这一步需要根据 Fig. 2 中的小提琴图选择合适的参数。

最后，这一步可以得到 4 个 Figure：

R input

```
sr@plots$step2$p.pca_pcComponents  
sr@plots$step2$p.pca_1v2  
sr@plots$step2$p.pca_heatmap  
sr@plots$step2$p.pca_rank
```

这里不展开。其中，下一步的主要依据的是 `sr@plots$step2$p.pca_rank`，即 Fig. 3

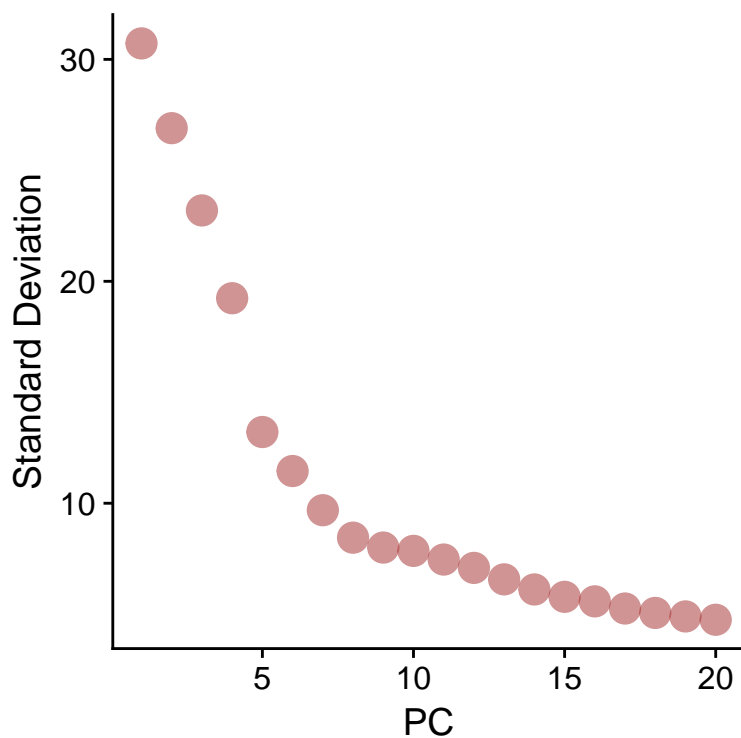


Figure 3: Ranking of principle components

#### 2.6.2.4 Step3 完成降维、聚类和 Marker 筛选

R input

```
sr <- step3(sr, 1:15, 1.2)
```

可以确认默认的参数：

R input

```
step3
```

```
## job_seurat:
```

```
##     x, dims, resolution, force = F
```

```
##
```

```
## -- Methods parameters -----
```

参数 `dim` 需要根据 Fig. ?? 判定。`resolution` 需要根据细胞数判定。这两个参数会传递到：

- `Seurat::FindNeighbors`
- `Seurat::FindClusters`

官方有一段解释：



The `FindClusters()` function implements this procedure, and contains a resolution parameter that sets the ‘granularity’ of the downstream clustering, with increased values leading to a greater number of clusters. We find that setting this parameter between 0.4-1.2 typically returns good results for single-cell datasets of around 3K cells. Optimal resolution often increases for larger datasets.

运行后将可以得到：

R input

```
sr@plots$step3$p.umap
```

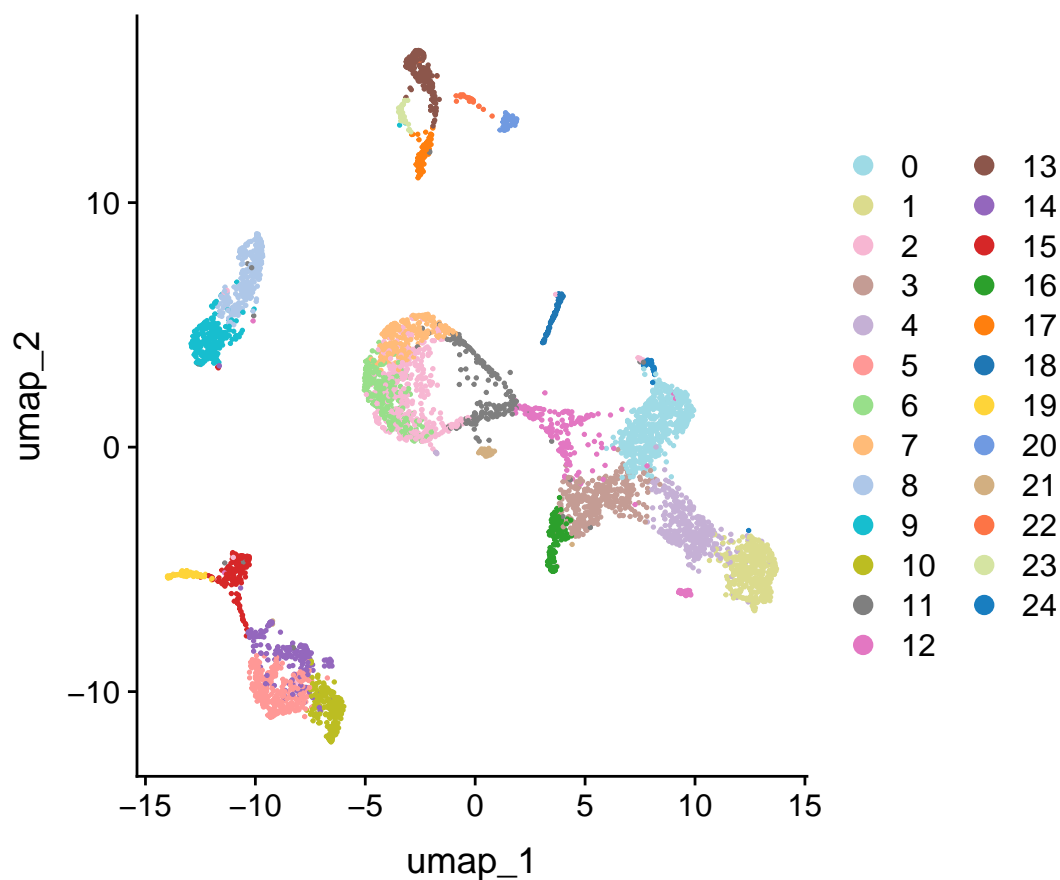


Figure 4: UMAP Clustering

#### 2.6.2.5 Step4 使用 SingleR 注释细胞类

(我现在都不用 SingleR 注释了，因为我发现它时常把其他组织的细胞注释到目标组织上，比如，注释肾脏组织注释得到肝脏细胞；默认使用的参考数据是：`celldex::HumanPrimaryCellAtlasData`，同样的，可以输入 `step4` 查看默认参数。)

如果要运行这一步（挺耗时的，并确保你安装了 SingleR 和 celldex，详情见 2.4.1.7）：

R input

```
sr <- step4(sr, "SingleR")
```

跳过这一步：

R input

```
sr <- step4(sr, "")
```

如果你运行了这一步，你可以得到（我没有运行，所以不展示了）：

R input

```
sr@plots$step4$p.score_SingleR  
sr@plots$step4$p.map_SingleR  
sr@tables$step4$anno_SingleR
```

#### 2.6.2.6 Step5 计算各细胞类的 Marker 基因（差异分析）

唯一需要输入的参数是线程数（不知道是否是 Seurat 程序有问题，我好像基本都是单线程的，即使设置了这个参数）：

R input

```
## 设置成 NULL，单线程  
sr <- step5(sr, 5)
```

这样，你就能得到：

R input

```
sr@tables$step5$all_markers  
sr@tables$step5$all_markers_no_filter
```

Table 1: All Markers

rownames	p_val	avg_lo...	pct.1	pct.2	p_val_adj	cluster	gene
GZMK	0	4.2402...	0.982	0.112	0	0	GZMK
CD27	0	3.2768...	0.747	0.094	0	0	CD27
CD8B	0	3.1693...	0.747	0.099	0	0	CD8B
DUSP4	0	3.0453...	0.774	0.144	0	0	DUSP4
CD8A	0	2.7603...	0.816	0.158	0	0	CD8A
GZMA	1.1171...	2.2365...	0.989	0.344	2.1412...	0	GZMA
CCL5	6.7488...	2.2809...	0.996	0.393	1.2935...	0	CCL5
CD3D	1.8166...	1.9909...	0.96	0.3	3.4820...	0	CD3D
TRAC	4.5242...	2.0246...	0.918	0.258	8.6716...	0	TRAC

rownames	p_val	avg_lo...	pct.1	pct.2	p_val_adj	cluster	gene
APOBEC3G	7.8727...	2.3392...	0.904	0.345	1.5089...	0	APOBEC3G
TRBC2	1.3387...	2.1611...	0.886	0.305	2.5659...	0	TRBC2
ITM2A	6.6648...	2.3921...	0.844	0.307	1.2774...	0	ITM2A
RPL28	4.5358...	1.0709...	1	1	8.6938...	0	RPL28
CST7	9.2732...	1.6915...	0.965	0.343	1.7773...	0	CST7
CD3E	3.8141...	1.8107...	0.914	0.294	7.3104...	0	CD3E
...	...	...	...	...	...	...	...

### 2.6.2.7 Step6 使用 SCSA 注释细胞类

SCSA 是 Python 编写的命令行工具。step6 已经调用以及数据的转换集成了，只需要运行如下，就能简便地得到结果：

R input

```
# 这个样本的组织来源是 Kidney
sr <- step6(sr, "Kidney")
```

注：如果你的 SCSA 存放在其它位置，那么你可能需要额外输入 cmd 和 db 参数，例如：

R input

```
# 以下只是示例，不需要运行
sr <- step6(sr, "Kidney", cmd = "python3 /<your_path>/SCSA/SCSA.py", db = "/<your_path>/SCSA/whole_v")
```

现在你可以获取经 SCSA 注释过的 UMAP 图了。

R input

```
sr@plots$step6$p.map_scsa
```

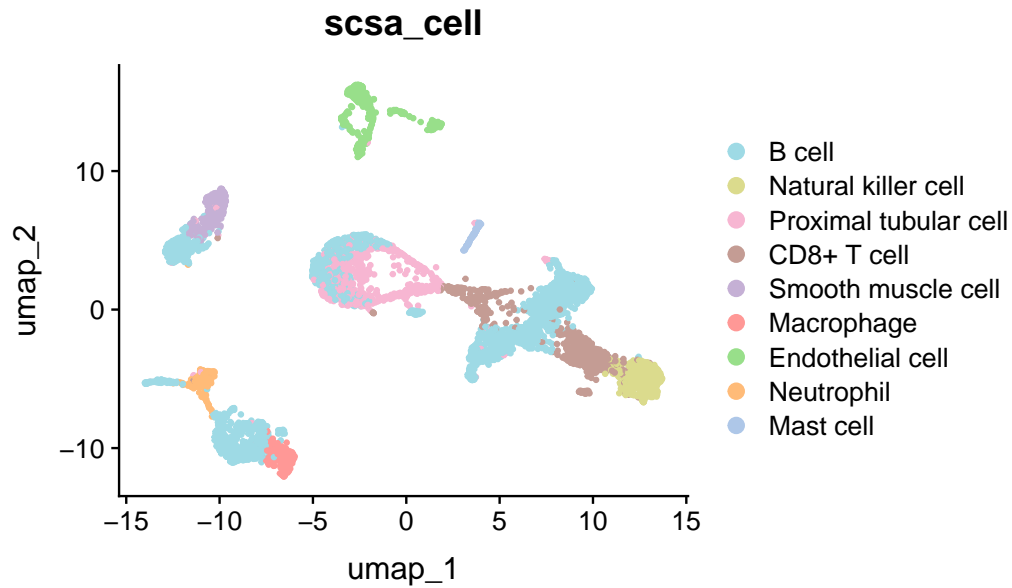


Figure 5: SCSA Cell type annotation

你还可以查看 SCSA 的注释表格。

```
R input
sr@tables$step6$scsa_res
```

这里，你也可以选择使用 `vis`，根据其它注释来绘制 UMAP 图。

```
R input
# 绘制 SCSA 注释结果，同 `sr@plots$step6$p.map_scsa` 的图
vis(sr, "scsa_cell")
# 同 `sr@plots$step3$p.umap`
vis(sr, "seurat_clusters")
```

#### 2.6.2.8 从以下开始拟时分析的示例

#### 2.6.2.9 As-job-monocle 对选定的细胞进行拟时分析

将 `seurat` 对象转化为 `monocle` (version 3) 对象，并继承聚类的结果（而不是重新开始分析，毕竟 `monocle` 有自己的分析体系）。

这里提供了两种选择（第二种是第一种封装，形式更固定，但也更简洁，所以更推荐）

R input

```
# getsub 会将参数传递到: SeuratObject::subset.Seurat
# 这里匹配了两类细胞: B 细胞和近端小管细胞
sr.sub <- getsub(sr, cells = grep("B cell|Proximal", sr@object@meta.data[[ "scsa_cell" ]]))
# 将 step 设置成 2, 这样就能重新进行 step3 计算了
sr.sub@step <- 2L
# 重新进行 step3 是因为, 我们选择了一个小类的细胞群体, 为了拟时分析能够
# 更加细致的划分这个群体的
sr_sub <- step3(x, dims = 1:15, resolution = 1.2)
# 转化到 monocle 时, 以重新聚类
mn_sub <- asjob_monocle(sr_sub, "seurat_clusters")
# `asjob_monocle` 不像表面看起来那么简单, 我参考了:
# <http://htmlpreview.github.io/?https://github.com/satijalab/seurat-wrappers/blob/master/docs/monoc
# <https://github.com/cole-trapnell-lab/monocle3/issues/438>
# 中的介绍和一些后续分析的 Bug 的解决办法。
# 你可以通过输入 `selectMethod("asjob_monocle", "job_seurat")` 查看细节
```

选择想要研究的细胞群, 重新聚类成更小的群体 (可能是亚型, 或者细胞不同阶段), 然后传递到 Monocle 中拟时分析, 这是一种实用且能泛用的策略, 因此我将它封装成了更加简洁的形式, 以便随时调用:

R input

```
# 这和上面的代码块的效果是一样的
mn_sub <- do_monocle(sr, "B cell|Proximal")
```

你可以直接输入 do\_monocle 来查看它的默认参数。

R input

```
do_monocle
```

```
## job_seurat, character:
##      x, ref, dims = 1:15, resolution = 1.2, group.by = x@params$group.by
## job_seurat, job_kat:
##      x, ref, dims = 1:15, resolution = 1.2
##
## -- Methods parameters -----
```

如果你对‘面向对象编程’和‘参数化多态’不熟悉, 你可能会感到惊讶, 因为它列出了两个参数列表。这是因为 Step 系列大部分用的都是方法 (Method), 面向对象设计的。这里, 第一个参数列表是我们实际使用的; 而第二个, 只有当我们输入的参数 ref (也就是第二个参数) 是 job\_kat 对象时, 它才会被触发, 并且在随后会调用一个截然不同的函数来后续处理。如果你感兴趣, 可以:

R input

```
# 这会展示我们这里实际调用的函数
selectMethod("do_monocle", c("job_seurat", "character"))

# 这会调用截然不同不同的函数
selectMethod("do_monocle", c("job_seurat", "job_kat"))
```

`job_kat` 是 `copyKAT` R 包的 `step` 封装，是用来专门鉴定癌细胞的。上面第二种设计是，对鉴定完毕的癌细胞进行拟时分析，也相当实用。我们之后介绍。

言归正状，`mn_sub` 是我们取得的 `monocle` 对象的 `Step` 形式的封装。如果你对 `monocle` 的代码体系更加熟悉，那么你可以直接提取 `mn_sub` 中存储的 `monocle` 对象：

R input

```
object(mn_sub)
```

#### 2.6.2.10 Step1 构建拟时轨迹

R input

```
mn_sub <- step1(mn_sub)
```

现在，你可以得到：

R input

```
mn_sub@plots$step1$p.traj$seurat_clusters
mn_sub@plots$step1$p.prin
```

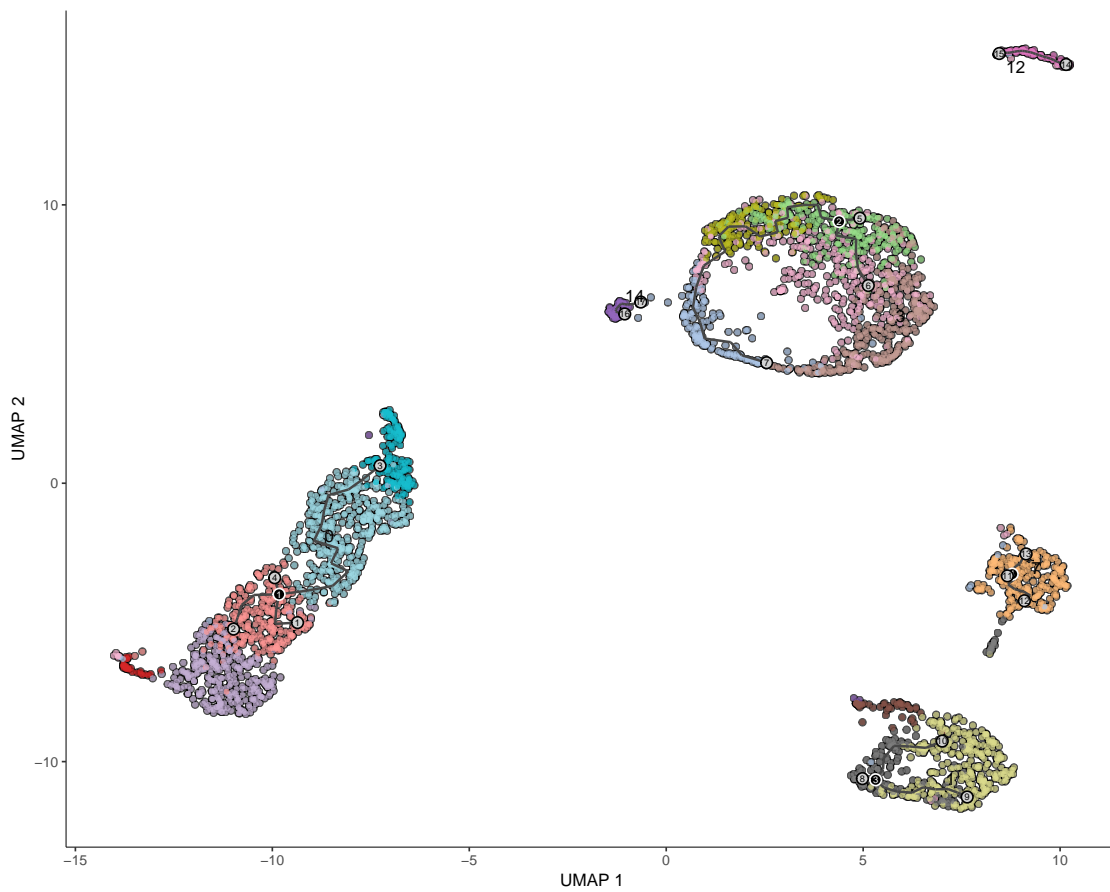


Figure 6: Trajectories

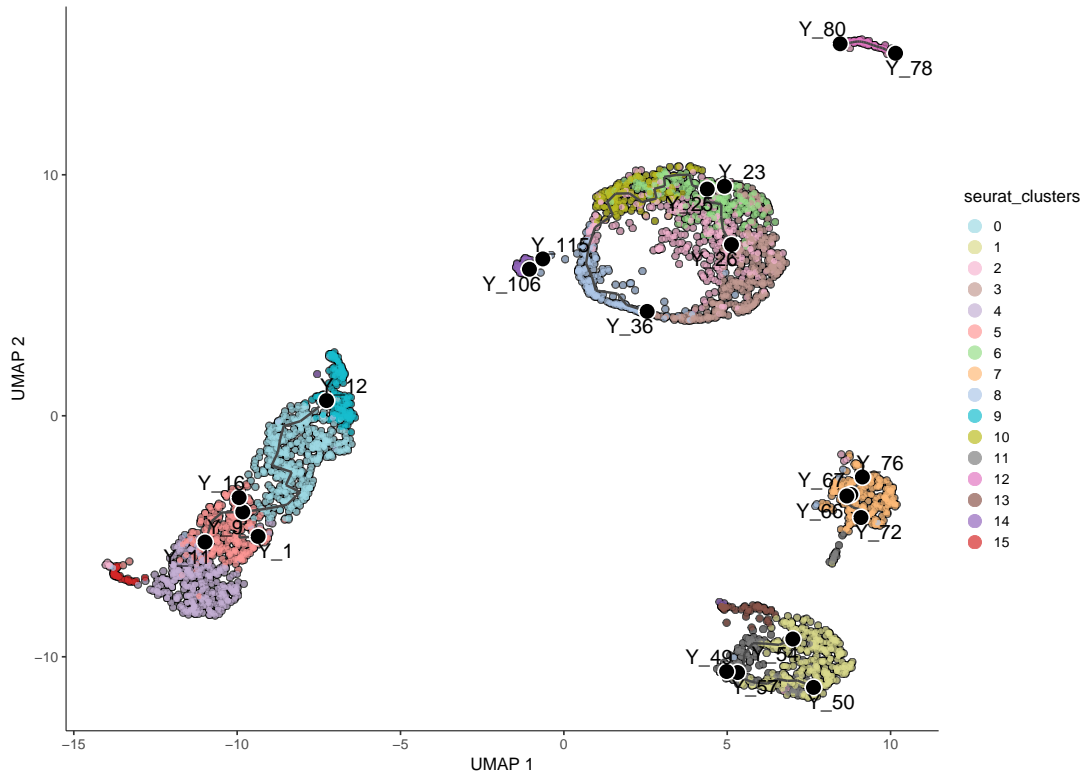


Figure 7: Principal points

Fig. 7 更有实用价值，因为它可以帮助我们选择拟时起点。

#### 2.6.2.11 Step2 选择拟时起点

在 Fig. 7 的帮助下，我们为每一个聚类团选择一个起点。为了便于演示，这里的起点是随意选择的：

R input

```
mn_sub <- step2(mn_sub, c("Y_12", "Y_50", "Y_72", "Y_36", "Y_78"))
```

实际分析中，可以根据细胞种类来选择起点，例如，在癌细胞数据集的分析中，选择非癌细胞作为拟时起点。在 Fig. 7 中没有显示细胞类型，这里，我们可以借助转化成 monocle 之前的 seurat 对象的 job 来检视：

R input

```
# 如果你是用 `do_monocle` 转化的，那么就能这样提取到它：
mn_sub$sr_sub
# 上述写法等同于：`mn_sub@params$sr_sub`
# 可以用 `vis` 来可视化它
p.sr_sub <- vis(mn_sub$sr_sub, "scsa_cell")
p.sr_sub
```



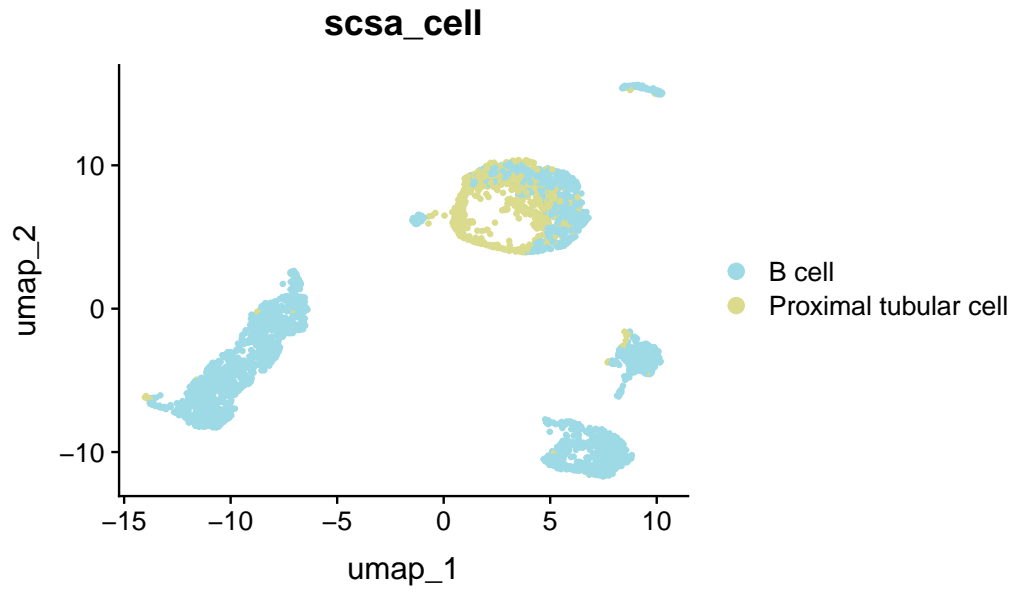


Figure 8: Deomo subset of cells visualization

在更复杂的分析中，或许得借助更多的因素来判定拟时起点。例如样本的来源，细胞位置（空间转录组），或者 Marker 表达量的高低。这里不再赘述。

言归正状，运行完 `step2` 后，将得到：

R input

```
mn_sub@plots$step2$p.pseu
```

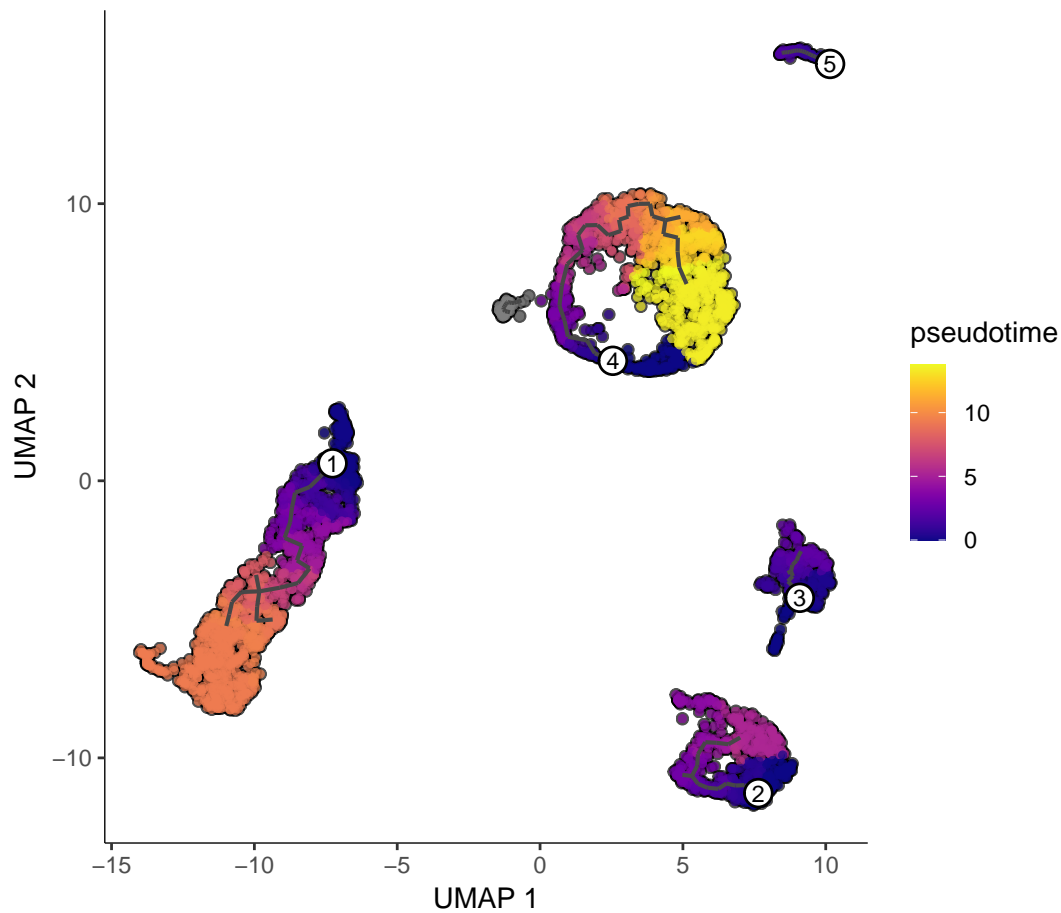


Figure 9: SUB pseudotime

#### 2.6.2.12 Step3 拟时分析基础上的差异分析和基因表达模块

R input

```
mn_sub <- step3(mn_sub)
```

这会得到：

R input

```
mn_sub@plots$step3$gene_module_heatdata$graph_test.sig
mn_sub@tables$step3$graph_test
mn_sub@tables$step3$gene_module$graph_test.sig
```

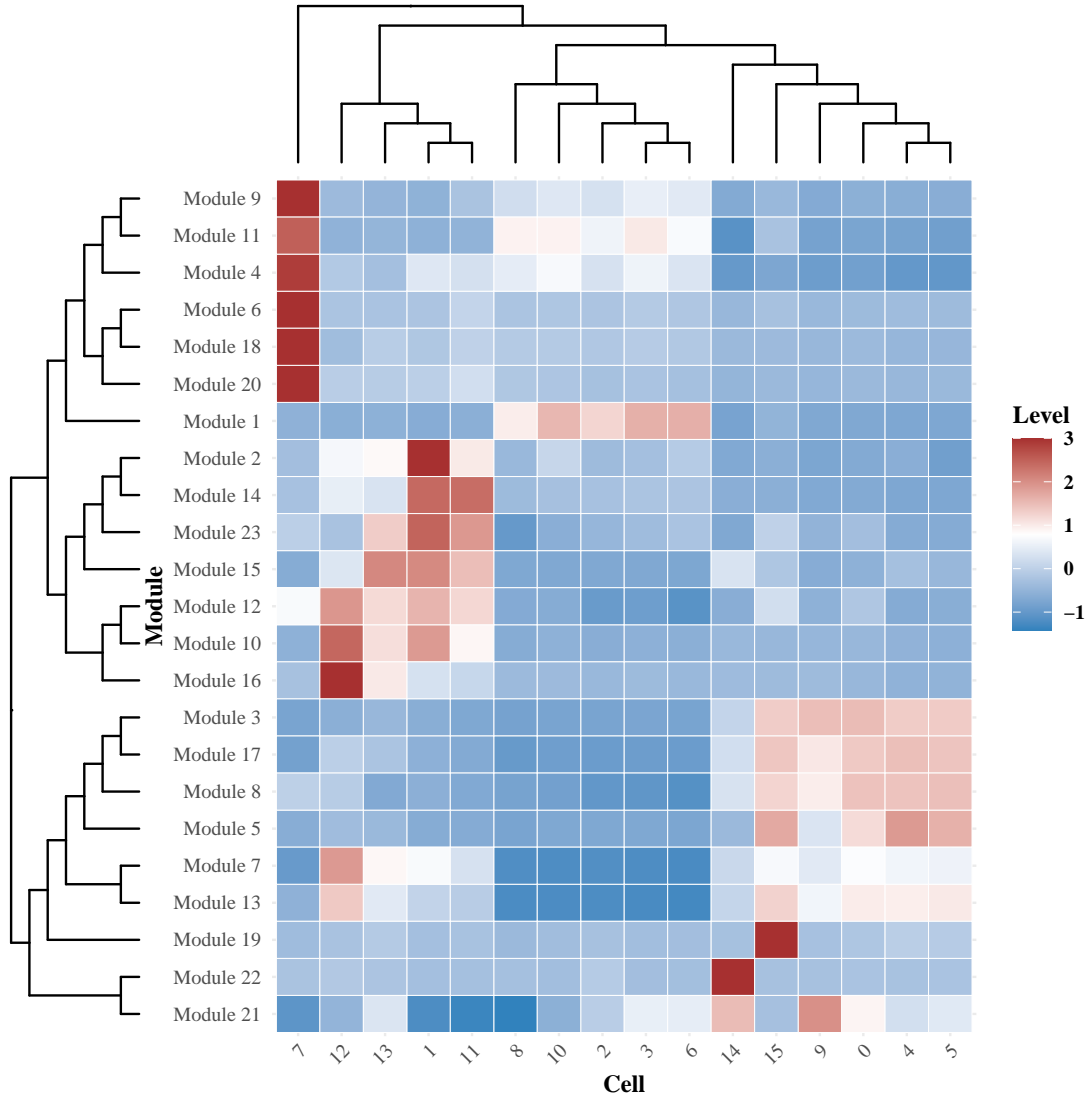


Figure 10: SUB gene module heatmap

Table 2: SUB graph test results

rownames	status	p_value	morans_tes...	morans_I	q_value
AL627309.1	OK	0.71713641...	-0.5743556...	-0.0028796...	0.74052249...
AL627309.3	OK	0.56865824...	-0.1729591...	-0.0007942...	0.64807206...
AL669831.2	OK	0.56600105...	-0.1662021...	-0.0009540...	0.64710506...
AL669831.5	OK	0.00012749...	3.65720355...	0.01709405...	0.00022429...
FAM87B	OK	0.66025318...	-0.4131542...	-0.0021470...	0.69573809...
LINC00115	OK	0.84067430...	-0.9972330...	-0.0050549...	0.84805084...
FAM41C	OK	0.26836391...	0.61776869...	0.00261844...	0.34797022...
AL645608.1	OK	0.55654189...	-0.1422073...	-0.0008524...	0.64609095...
SAMD11	OK	2.26008037...	4.58590871...	0.02126656...	4.33505225...

rownames	status	p_value	morans_tes...	morans_I	q_value
NOC2L	OK	0.01774149...	2.10280310...	0.00974723...	0.02671787...
KLHL17	OK	0.02881144...	1.89855590...	0.00866824...	0.04248129...
PLEKHN1	OK	0.00017895...	3.56931611...	0.01662486...	0.00031221...
PERM1	OK	0.00020134...	3.53831261...	0.01622553...	0.00035024...
AL645608.8	OK	0	42.5620756...	0.20314781...	0
HES4	OK	0	114.647755...	0.54828135...	0
...	...	...	...	...	...

### 2.6.2.13 (进阶) 根据拟时分析结果重新划分细胞群体

方法 `asjob_seurat` 较为复杂，这里试着解释：

- 首先，它无疑会将数据对象 `job_monocle` 转换回 `job_seurat`。
- 在转换过程中，会根据拟时分析结果以及你的参数，重新划归细胞聚类。
- 重新聚类主要取决于 `mn@plots$step3$gene_module_heatdata$graph_test.sig` (即，Fig. 10) 热图上方聚类树。

例如，下述代码，我根据 Fig. 10，将细胞重分为 4 个聚类：

R input

```
sr_sub_regroup <- asjob_seurat(mn_sub, 4, rename = "Test")
p.sr_sub_regroup <- vis(sr_sub_regroup, "regroup.hclust")
```

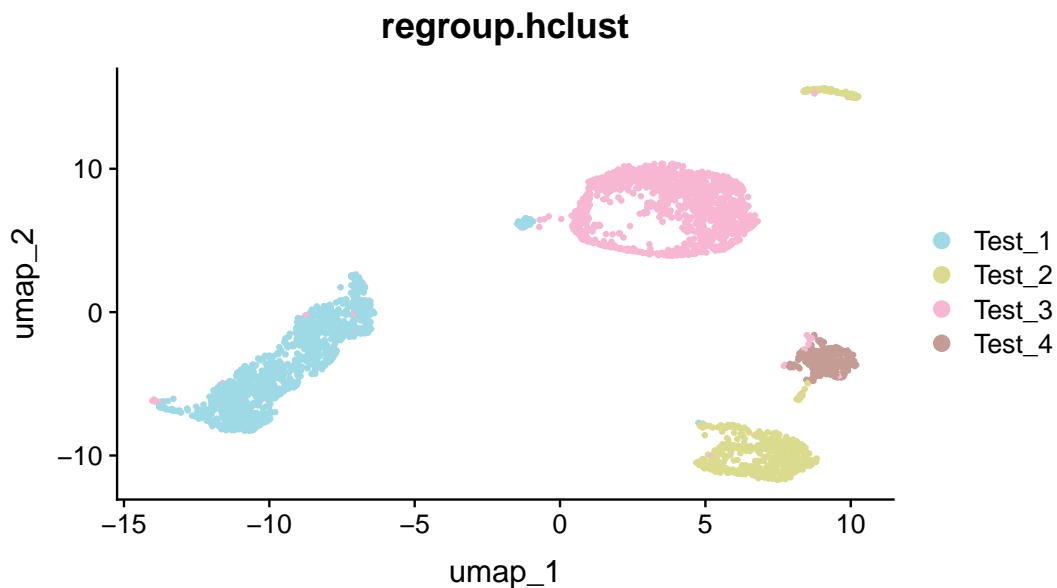


Figure 11: Sub the regroup by hclust

这种分析方式在肿瘤细胞或其它细胞的亚型分析上有一定参考价值，且可以泛用。

更进一步的是，我们可以将重新聚类完的 `sr_sub_regroup` 映射回到最初的 `job_seurat` 对象中，也就是 `sr` 对象：

R input

```
# 这行代码会将 `sr` 对象中的 `scsa_cell` 注释，根据 `sr_sub_regroup`  
# 中的 `regroup.hclust` 注释重新改写，然后命名为 `cell_mapped`  
# 你也能通过额外加入参数 `name = 'cell_mapped'`，如此，改成其它名称。  
sr <- map(sr, sr_sub_regroup, "scsa_cell", "regroup.hclust")
```

可以确认我们得到了什么：

R input

```
# 其实，`vis` 方法可以通过加入参数 `palette` 自定义颜色。  
# 它将参数传递到 `Seurat::DimPlot` 中。  
p.sr_mapped <- vis(sr, "cell_mapped")  
p.sr_mapped
```

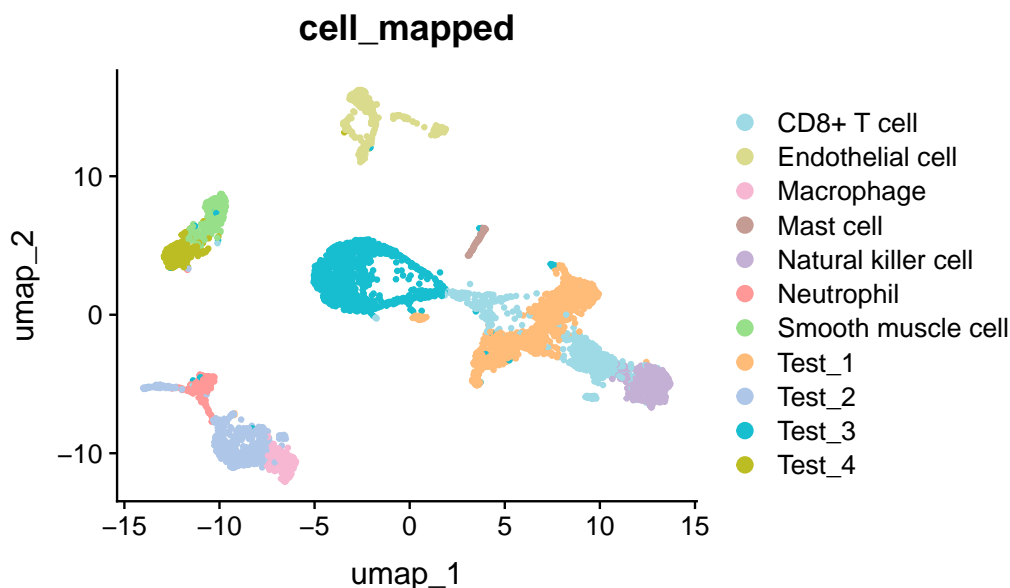


Figure 12: The cell mapped from monocle

#### 2.6.2.14 以下开始细胞通讯的示例分析

#### 2.6.2.15 As-job-cellchat 对选定的细胞进行细胞通讯分析

为了展示整个分析的延续性，我们继续从 `sr` (`job_seurat` 对象) 往下分析，也就是 Fig. 12 所示的细胞的来源数据。注意，细胞通讯也是相对耗时的，太多的细胞数会非常占用内存。这里，我们取一部分的子集来演示 (实际分析，我们也可以取子集，因为我们其实可能并不需要对所有的细胞都通讯分析)：

R input

```
sr_cc_sub <- getsub(sr,  
  cells = grep("Macro|Test", sr@object@meta.data[[ "cell_mapped" ]])  
)
```

现在，我们可以把它转化为 job\_cellchat 对象了。

R input

```
cc <- asjob_cellchat(sr_cc_sub, "cell_mapped")
```

同样的，你也可以通过提取 cc@object 或者 object(cc) 以 CellChat 原来的代码开始分析。

#### 2.6.2.16 Step1 构建通讯网络以及可视化

R input

```
# 这可能需要运行较长时间  
cc <- step1(cc)
```

请注意，细胞通讯计算需要选定合适的参考数据集，我这里的方法设计默认是人类的，所以不需要指定任何参数。如果你是小鼠的数据集 (CellChat 好像只支持人类和小鼠的)，那么需要再输入 db 和 ppi 参数。例如：

R input

```
# 以下示例不需要运行  
cc <- step1(cc, db = CellChat::CellChatDB.mouse, ppi = CellChat::PPI.mouse)
```

另外需要注意的是，CellChat 的 UMAP 聚类是以 Python 的包实现的。CellChat 内部调用 Python 包的形式并不太高明，可能会和你的设定发生冲突。因为我出现过这样的情况，所以我默认指定的是 python = "/usr/bin/python3"。你可以指定成你的 Python 所在路径，或者设置成 NULL。一旦指定，以下会被执行：

- base::Sys.setenv(RETICULATE\_PYTHON = python)
- reticulate::py\_config()

请确保你安装了 reticulate (见 2.4.1.7)。

运行完成后，你将可以得到 (Figure)：

R input

```
# 这是 CellChat 所用参考数据集的展示
cc@plots$step1$p.showdb
# 通讯的 'Count' 统计
cc@plots$step1$p.aggre_count
# 通讯的 'Weight' 统计
cc@plots$step1$p.aggre_weight
cc@plots$step1$p.commSep
```

这里只展示了 'Count' 统计

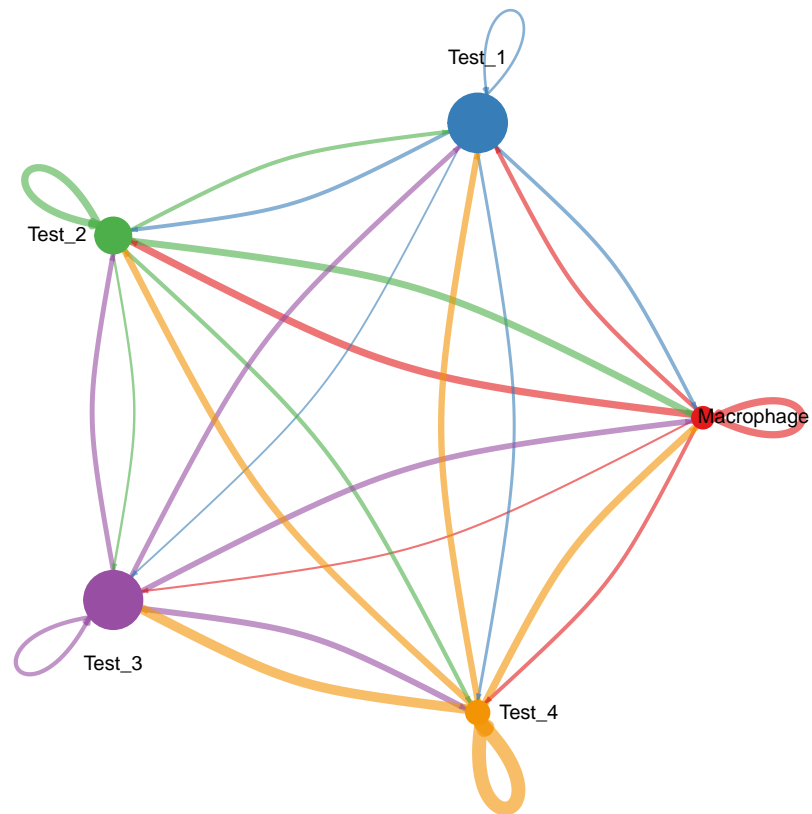


Figure 13: Overall communication count

其实，比起 Fig. 13，下一步的热图更适合展示整体通讯，因为包含更多的信息。

你还能得到 Tables:

R input

# 以下两个表格其实是通讯分析的主要内容

```
cc@tables$step1$lp_net
```

```
cc@tables$step1$pathway_net
```

Table 3: Lp net

source	target	ligand	receptor	prob	pval	intera.....7	intera.....8	pathwa...	annota...
Macrop...	Macrop...	TGFB1	TGFbR1_R2	0.0048...	0	TGFB1_...	TGFB1 ...	TGFb	Secret...
Test_1	Macrop...	TGFB1	TGFbR1_R2	0.0048...	0	TGFB1_...	TGFB1 ...	TGFb	Secret...
Test_2	Macrop...	TGFB1	TGFbR1_R2	0.0061...	0	TGFB1_...	TGFB1 ...	TGFb	Secret...
Test_4	Macrop...	TGFB1	TGFbR1_R2	0.0012...	0	TGFB1_...	TGFB1 ...	TGFb	Secret...
Macrop...	Test_2	TGFB1	TGFbR1_R2	0.0048...	0	TGFB1_...	TGFB1 ...	TGFb	Secret...
Test_1	Test_2	TGFB1	TGFbR1_R2	0.0048...	0	TGFB1_...	TGFB1 ...	TGFb	Secret...
Test_2	Test_2	TGFB1	TGFbR1_R2	0.0061...	0	TGFB1_...	TGFB1 ...	TGFb	Secret...
Test_4	Test_2	TGFB1	TGFbR1_R2	0.0012...	0	TGFB1_...	TGFB1 ...	TGFb	Secret...
Test_3	Macrop...	GDF15	TGFBR2	0.0053...	0	GDF15_...	GDF15 ...	GDF	Secret...
Test_3	Test_2	GDF15	TGFBR2	0.0053...	0	GDF15_...	GDF15 ...	GDF	Secret...
Test_3	Test_4	GDF15	TGFBR2	0.0053...	0	GDF15_...	GDF15 ...	GDF	Secret...
Test_3	Test_3	TGFA	EGFR	0.0040...	0	TGFA_EGFR	TGFA -...	EGF	Secret...
Test_1	Test_3	AREG	EGFR	0.0063...	0	AREG_EGFR	AREG -...	EGF	Secret...
Test_2	Test_3	AREG	EGFR	0.0040...	0	AREG_EGFR	AREG -...	EGF	Secret...
Macrop...	Test_3	HBEGF	EGFR	0.0159...	0	HBEGF_...	HBEGF ...	EGF	Secret...
...	...	...	...	...	...	...	...	...	...

Table 4: Pathway net

source	target	pathway_name	prob	pval
Macrophage	Macrophage	ANNEXIN	0.0142752959860733	0
Macrophage	Macrophage	CADM	0.00134474271905693	0
Macrophage	Macrophage	CCL	0.197165266984973	0
Macrophage	Macrophage	CD45	0.0511750497900894	0
Macrophage	Macrophage	CD99	0.00134474271905693	0
Macrophage	Macrophage	COMPLEMENT	0.179867247416985	0
Macrophage	Macrophage	CXCL	0.0315784487714539	0
Macrophage	Macrophage	GALECTIN	0.0510613013929917	0
Macrophage	Macrophage	GAS	0.0371625768894343	0
Macrophage	Macrophage	ICAM	0.133417588216607	0
Macrophage	Macrophage	ITGB2	0.115608598396661	0



source	target	pathway_name	prob	pval
Macrophage	Macrophage	MHC-I	0.00134474271905693	0
Macrophage	Macrophage	MHC-II	1.12708370370619	0
Macrophage	Macrophage	PECAM1	0.0119738702725686	0
Macrophage	Macrophage	SEMA4	0.00734078826527416	0
...	...	...	...	...

### 2.6.2.17 Step2 进一步分析通路通讯、受体配体通讯和可视化

R input

```
# 默认的，如果你不指定 'pathway' 参数，它会绘制所有的 'pathway' 通讯
# 如果通路很多，会比较耗时且占用内存（全部存储在 `cc` 中）
# 更建议根据 'cc@tables$step1$lp_net' 或 'cc@tables$step1$pathway_net'
# 筛选后再运行
# 这里数据集不算大，直接运行了
cc <- step2(cc)
```

R input

```
# `cc@plots$step2$cell_comm_heatmap` 是一个 'list'
# 存储了大量其它的 figure，以下示例提取 'ALL'
cc@plots$step2$cell_comm_heatmap$ALL
cc@plots$step2$lr_comm_bubble
# `cc@plots$step2` 还有更多的 figure，不一一展示了
```

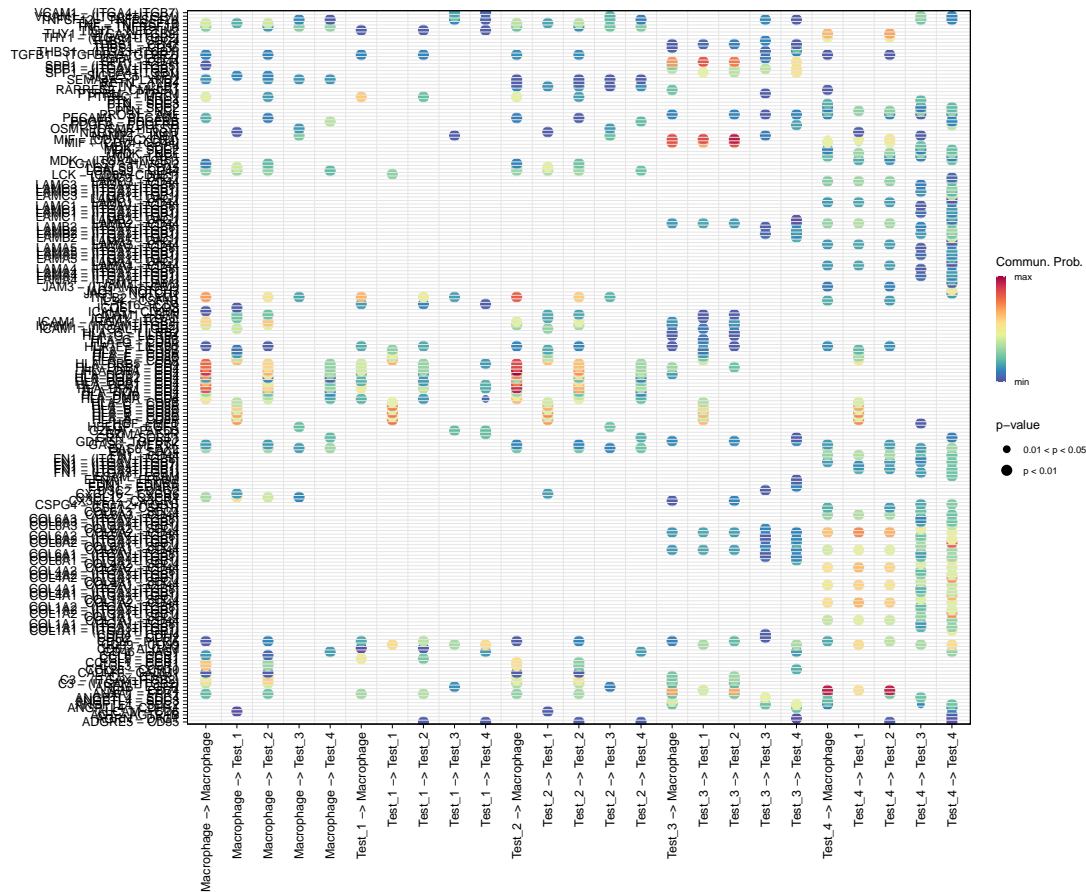


Figure 14: Communication bubble

## 2.6.2.18 (进阶)

## 2.6.3 完整示例代码

## 2.7 技巧

## Reference

1. Jin, S. *et al.* Inference and analysis of cell-cell communication using cellchat. *Nature Communications* **12**, (2021).
2. Qiu, X. *et al.* Reversed graph embedding resolves complex single-cell trajectories. *Nature Methods* **14**, (2017).
3. Trapnell, C. *et al.* The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature Biotechnology* **32**, (2014).
4. Hao, Y. *et al.* Integrated analysis of multimodal single-cell data. *Cell* **184**, (2021).
5. Stuart, T. *et al.* Comprehensive integration of single-cell data. *Cell* **177**, (2019).
6. Cao, Y., Wang, X. & Peng, G. SCSA: A cell type annotation tool for single-cell rna-seq data. *Frontiers*

*in genetics* **11**, (2020).