# Lab 3: Analysis of Large Single-Cell and Spatial Omics Datasets in R/Bioconductor

Omics Data Science

Himel Mallick

## Contents



## Single-cell Differential Expression Using Tweedieverse

```
library(ggplot2)
library(Tweedieverse)  # devtools::install_github('himelmallick/Tweedieverse')
library(scran)  # Bioconductor package
library(SC2P)  # devtools::install_github('haowulab/SC2P')
library(boost)  # devtools::install_github('estfernan/boost')
library(SPARK)  # devtools::install_github('xzhoulab/SPARK')
library(spatialDE)  # Bioconductor package
library(Seurat)  # CRAN package
library(utils)
library(stats)
library(grDevices)
library(mgcv)
theme_set(theme_bw())
```

### Introduction

In this part, we will highlight an example workflow for performing differential expression using `Tweedieverse` on a single-cell RNA-seq (scRNA-seq) dataset.

## Input

`Tweedieverse` requires two tab-delimited input files, one for expression counts (`input_features`) and one for metadata or covariates (`input_metadata`). The rows of the `input_features` correspond to genes and the columns correspond to cells (and vice versa). Similarly, the columns of the `input_metadata` correspond to cell-specific covariates (e.g., cell types) and the rows correspond to cells.

The `input_features` file can contain cells not included in the `input_metadata` file (along with the reverse case) although it is expected that they have matching cells. For both cases, cells not included in either of the files will be removed from the analysis. Also cells do not need to be in the same order in the two files as `Tweedieverse` will automatically detect the right order based on the cell names (assuming common cell IDs across files).

For the purpose of this vignette, it is assumed that these two input data have already been quality-controlled with necessary preprocessing steps, e.g.„ as described in the Seurat tutorial. For demonstration purposes, we will be using a cleaned version of the Brain dataset (available from the R package SC2P). The original dataset is available from the Gene Expression Omnibus database under accession number GSE67835.

```
# Load data
data(brain_scRNAseq)
colnames(Y) <- rownames(design)

# Prepare data for Tweedieverse
input_features <- as.data.frame(t(Y))
input_metadata <- design
cellIDs <- paste("Cell", 1:nrow(input_features), sep = "")
rownames(input_metadata) <- rownames(input_features) <- cellIDs  # Arbitrary Names

# Take a quick look at the data
head(input_features[1:5, 1:5])
#>       A2M  A2ML1  A2MP1  AAAS  AACS
#> Cell1   0      0      0     0     0
#> Cell2   0      0      0     0     0
#> Cell3  34     25      0   111     0
#> Cell4   0      0      0     0     0
#> Cell5   0   1290      0     0     0
head(input_metadata[1:5, 1:5])
#>               tissue   celltype                      age                    chip
#> Cell1 tissue: cortex astrocytes age: postnatal 54 years c1 chip id: 17720782217
#> Cell2 tissue: cortex astrocytes age: postnatal 37 years c1 chip id: 17720782236
#> Cell3 tissue: cortex astrocytes age: postnatal 37 years c1 chip id: 17720782236
#> Cell4 tissue: cortex astrocytes age: postnatal 37 years c1 chip id: 17720782237
#> Cell5 tissue: cortex astrocytes age: postnatal 37 years c1 chip id: 17720782237
#>       experiment
#> Cell1      AB_S8
#> Cell2     AB_S11
#> Cell3     AB_S11
#> Cell4     AB_S11
#> Cell5     AB_S11

# Calculate sparsity (percentage of zeroes)
round(sum(input_features == 0)/(nrow(input_features) * ncol(input_features)) *
    100, 1)
#> [1] 64.9
```

For brevity, we will consider top 100 most variable genes.

```r
sds <- apply(input_features, 2, function(x) {
    sd(x, na.rm = TRUE)
})
top_n_IDs <- order(sds, decreasing = TRUE)[1:100]
input_features <- input_features[, top_n_IDs]
dim(input_features)
#> [1] 100 100
```

## Normalization

For normalization, Tweedieverse expects a variable named `scale_factor` in the `input_metadata` file which is included as an offset in the base model when prompted (i.e. `adjust_offset = TRUE`). If not found in metadata, Tweedieverse, by default, includes the library size as an offset in the model (unless `adjust_offset = FALSE`).

```r
# SCRAN normalization
sce <- t(input_features)
scale_factor <- scran::calculateSumFactors(sce)
input_metadata$scale_factor <- scale_factor
```

## Differential Expression

```r
if (!dir.exists('demo_output')) dir.create('demo_output')
Brain <- Tweedieverse(
  input_features,
  input_metadata,
  output = 'demo_output/Brain', # Assuming demo_output exists
  fixed_effects = c('celltype'),
  base_model = 'CPLM',
  standardize = FALSE,
  adjust_offset = TRUE)
#> [1] "Creating output folder"
#> [1] "Creating output figures folder"
#> 2024-04-06 11:28:19.774074 INFO::Writing function arguments to log file
#> 2024-04-06 11:28:19.782175 INFO::Determining format of input files
#> 2024-04-06 11:28:19.782389 INFO::Input format is data samples as rows and metadata samples as rows
#> 2024-04-06 11:28:19.783241 INFO::Factor detected for categorial metadata 'celltype'. Provide a refer
#> 2024-04-06 11:28:19.783439 INFO::Filter data based on min abundance and min prevalence
#> 2024-04-06 11:28:19.783638 INFO::Total samples in data: 100
#> 2024-04-06 11:28:19.783822 INFO::Min samples required with min abundance for a feature not to be fil
#> 2024-04-06 11:28:19.784498 INFO::Total filtered features with prevalence-abundance filtering: 0
#> 2024-04-06 11:28:19.784738 INFO::Filtered feature names:
#> 2024-04-06 11:28:19.785706 INFO::Total filtered features with variance filtering: 0
#> 2024-04-06 11:28:19.785922 INFO::Filtered feature names:
#> 2024-04-06 11:28:19.787738 INFO::Excluded metadata with entropy less or equal to 0: adult
#> 2024-04-06 11:28:19.788021 INFO::Formula for fixed effects: expr ~  celltype
#> 2024-04-06 11:28:19.788318 INFO::Bypass z-score application to metadata
#> 2024-04-06 11:28:19.788586 INFO::Running selected analysis method: CPLM
#> 2024-04-06 11:28:19.789652 INFO::Fitting model to feature number 1, no_feature
#> 2024-04-06 11:28:19.970243 INFO::Fitting model to feature number 2, alignment_not_unique
#> 2024-04-06 11:28:20.648791 INFO::Fitting model to feature number 3, PLP1
#> 2024-04-06 11:28:20.678655 INFO::Fitting model to feature number 4, SLC1A2
#> 2024-04-06 11:28:20.711257 INFO::Fitting model to feature number 5, MALAT1
```

```
#> 2024-04-06 11:28:20.760286 INFO::Fitting model to feature number 6, SPARCL1
#> 2024-04-06 11:28:20.788237 INFO::Fitting model to feature number 7, ambiguous
#> 2024-04-06 11:28:20.82653 INFO::Fitting model to feature number 8, CLU
#> 2024-04-06 11:28:20.854269 INFO::Fitting model to feature number 9, PIP4K2A
#> 2024-04-06 11:28:20.880325 INFO::Fitting model to feature number 10, SAT1
#> 2024-04-06 11:28:20.904661 INFO::Fitting model to feature number 11, AQP4
#> 2024-04-06 11:28:20.927881 INFO::Fitting model to feature number 12, DPYSL2
#> 2024-04-06 11:28:20.950473 INFO::Fitting model to feature number 13, GLUL
#> 2024-04-06 11:28:20.973978 INFO::Fitting model to feature number 14, CNP
#> 2024-04-06 11:28:21.004539 INFO::Fitting model to feature number 15, SLC1A3
#> 2024-04-06 11:28:21.047998 INFO::Fitting model to feature number 16, MBP
#> 2024-04-06 11:28:21.074089 INFO::Fitting model to feature number 17, CLDND1
#> 2024-04-06 11:28:21.102083 INFO::Fitting model to feature number 18, HSPA5
#> 2024-04-06 11:28:21.132634 INFO::Fitting model to feature number 19, FOS
#> 2024-04-06 11:28:21.162319 INFO::Fitting model to feature number 20, GPM6A
#> 2024-04-06 11:28:21.186447 INFO::Fitting model to feature number 21, TMEM144
#> 2024-04-06 11:28:21.212911 INFO::Fitting model to feature number 22, DST
#> 2024-04-06 11:28:21.25599 INFO::Fitting model to feature number 23, ALDOC
#> 2024-04-06 11:28:21.283847 INFO::Fitting model to feature number 24, ARRDC3
#> 2024-04-06 11:28:21.317089 INFO::Fitting model to feature number 25, PCDH9
#> 2024-04-06 11:28:21.336457 INFO::Fitting model to feature number 26, DDAH1
#> 2024-04-06 11:28:21.362059 INFO::Fitting model to feature number 27, ATP1A2
#> 2024-04-06 11:28:21.385158 INFO::Fitting model to feature number 28, PON2
#> 2024-04-06 11:28:21.411776 INFO::Fitting model to feature number 29, GPR98
#> 2024-04-06 11:28:21.444703 INFO::Fitting model to feature number 30, ERMN
#> 2024-04-06 11:28:21.493373 INFO::Fitting model to feature number 31, HSP90AA1
#> 2024-04-06 11:28:21.515456 INFO::Fitting model to feature number 32, CPE
#> 2024-04-06 11:28:21.540016 INFO::Fitting model to feature number 33, MAP1B
#> 2024-04-06 11:28:21.567339 INFO::Fitting model to feature number 34, TF
#> 2024-04-06 11:28:21.59492 INFO::Fitting model to feature number 35, CALM2
#> 2024-04-06 11:28:21.619113 INFO::Fitting model to feature number 36, NTRK2
#> 2024-04-06 11:28:21.64376 INFO::Fitting model to feature number 37, PPP1CB
#> 2024-04-06 11:28:21.690319 INFO::Fitting model to feature number 38, MACF1
#> 2024-04-06 11:28:21.71485 INFO::Fitting model to feature number 39, ATP1B2
#> 2024-04-06 11:28:21.741998 INFO::Fitting model to feature number 40, SPP1
#> 2024-04-06 11:28:21.770856 INFO::Fitting model to feature number 41, SRSF6
#> 2024-04-06 11:28:21.795731 INFO::Fitting model to feature number 42, EGR1
#> 2024-04-06 11:28:21.823489 INFO::Fitting model to feature number 43, SCD
#> 2024-04-06 11:28:21.846223 INFO::Fitting model to feature number 44, LAMP2
#> 2024-04-06 11:28:21.870965 INFO::Fitting model to feature number 45, GJA1
#> 2024-04-06 11:28:21.894369 INFO::Fitting model to feature number 46, LIMCH1
#> 2024-04-06 11:28:21.946001 INFO::Fitting model to feature number 47, AXL
#> 2024-04-06 11:28:21.976597 INFO::Fitting model to feature number 48, AGXT2L1
#> 2024-04-06 11:28:21.996983 INFO::Fitting model to feature number 49, BMPR1B
#> 2024-04-06 11:28:22.024093 INFO::Fitting model to feature number 50, SLC24A2
#> 2024-04-06 11:28:22.054393 INFO::Fitting model to feature number 51, HSPA8
#> 2024-04-06 11:28:22.082024 INFO::Fitting model to feature number 52, APC
#> 2024-04-06 11:28:22.10829 INFO::Fitting model to feature number 53, DDX5
#> 2024-04-06 11:28:22.160328 INFO::Fitting model to feature number 54, GPM6B
#> 2024-04-06 11:28:22.177544 INFO::Fitting model to feature number 55, ENO1
#> 2024-04-06 11:28:22.199066 INFO::Fitting model to feature number 56, H3F3B
#> 2024-04-06 11:28:22.218402 INFO::Fitting model to feature number 57, TUBA1A
#> 2024-04-06 11:28:22.243684 INFO::Fitting model to feature number 58, AASS
```

```
#> 2024-04-06 11:28:22.269008 INFO::Fitting model to feature number 59, F3
#> 2024-04-06 11:28:22.291007 INFO::Fitting model to feature number 60, STMN1
#> 2024-04-06 11:28:22.318477 INFO::Fitting model to feature number 61, C6orf62
#> 2024-04-06 11:28:22.346657 INFO::Fitting model to feature number 62, CRYAB
#> 2024-04-06 11:28:22.367818 INFO::Fitting model to feature number 63, SRSF3
#> 2024-04-06 11:28:22.408398 INFO::Fitting model to feature number 64, CKB
#> 2024-04-06 11:28:22.43476 INFO::Fitting model to feature number 65, TSPAN7
#> 2024-04-06 11:28:22.464763 INFO::Fitting model to feature number 66, MARCKS
#> 2024-04-06 11:28:22.491762 INFO::Fitting model to feature number 67, SLCO1C1
#> 2024-04-06 11:28:22.518116 INFO::Fitting model to feature number 68, RFTN2
#> 2024-04-06 11:28:22.549357 INFO::Fitting model to feature number 69, OMG
#> 2024-04-06 11:28:22.572616 INFO::Fitting model to feature number 70, SYNE1
#> 2024-04-06 11:28:22.59856 INFO::Fitting model to feature number 71, ENPP2
#> 2024-04-06 11:28:22.655728 INFO::Fitting model to feature number 72, C10orf46
#> 2024-04-06 11:28:22.68494 INFO::Fitting model to feature number 73, TBL1XR1
#> 2024-04-06 11:28:22.705219 INFO::Fitting model to feature number 74, STON2
#> 2024-04-06 11:28:22.73346 INFO::Fitting model to feature number 75, QKI
#> 2024-04-06 11:28:22.752807 INFO::Fitting model to feature number 76, SLC12A2
#> 2024-04-06 11:28:22.782704 INFO::Fitting model to feature number 77, LSAMP
#> 2024-04-06 11:28:22.811332 INFO::Fitting model to feature number 78, NFIA
#> 2024-04-06 11:28:22.83473 INFO::Fitting model to feature number 79, NFIB
#> 2024-04-06 11:28:22.883249 INFO::Fitting model to feature number 80, PTPRZ1
#> 2024-04-06 11:28:22.906862 INFO::Fitting model to feature number 81, DTNA
#> 2024-04-06 11:28:22.93106 INFO::Fitting model to feature number 82, GAPDH
#> 2024-04-06 11:28:22.958321 INFO::Fitting model to feature number 83, ITGA6
#> 2024-04-06 11:28:23.072898 INFO::Fitting model to feature number 84, GLUD1
#> 2024-04-06 11:28:23.100284 INFO::Fitting model to feature number 85, RBBP6
#> 2024-04-06 11:28:23.132957 INFO::Fitting model to feature number 86, CLDN11
#> 2024-04-06 11:28:23.163754 INFO::Fitting model to feature number 87, DNM3
#> 2024-04-06 11:28:23.196897 INFO::Fitting model to feature number 88, ACTB
#> 2024-04-06 11:28:23.220065 INFO::Fitting model to feature number 89, SEPT7
#> 2024-04-06 11:28:23.240986 INFO::Fitting model to feature number 90, CLK1
#> 2024-04-06 11:28:23.271473 INFO::Fitting model to feature number 91, PDK4
#> 2024-04-06 11:28:23.325097 INFO::Fitting model to feature number 92, SLAIN1
#> 2024-04-06 11:28:23.349251 INFO::Fitting model to feature number 93, ABCA1
#> 2024-04-06 11:28:23.37611 INFO::Fitting model to feature number 94, RYR3
#> 2024-04-06 11:28:23.407071 INFO::Fitting model to feature number 95, EZR
#> 2024-04-06 11:28:23.426373 INFO::Fitting model to feature number 96, MGEA5
#> 2024-04-06 11:28:23.458055 INFO::Fitting model to feature number 97, TMEM33
#> 2024-04-06 11:28:23.487077 INFO::Fitting model to feature number 98, HSPH1
#> 2024-04-06 11:28:23.54583 INFO::Fitting model to feature number 99, TIMP3
#> 2024-04-06 11:28:23.574373 INFO::Fitting model to feature number 100, IDI1
#> 2024-04-06 11:28:23.612707 INFO::Counting prevalence for each feature
#> 2024-04-06 11:28:23.61546 INFO::Writing all results to file (ordered by increasing q-values): demo_o
#> 2024-04-06 11:28:23.617355 INFO::Writing the significant results (those which are less than or equal
#> 2024-04-06 11:28:23.618129 INFO::Writing Tweedie inxed plot to file: demo_output/Brain
#> 2024-04-06 11:28:23.618485 INFO::Plotting tweedie.index  colored by metadata
#> 2024-04-06 11:28:23.618707 INFO::Plotting tweedie.index
```

## Output

After computation, `Brain` is a data frame containing coefficient estimates, p-values, and q-values (multiplicity-adjusted p-values) along with other parameter estimates from the fitted per-feature models. By default,

p-values are adjusted with the Benjamini-Hochberg method.

```
# Table with DE analysis results
head(Brain)
#>    feature metadata            value      coef    stderr         pval
#> 1 SPARCL1  celltype oligodendrocytes -4.372554 0.2662480 6.780362e-30
#> 2 PIP4K2A  celltype oligodendrocytes  3.258072 0.2231474 2.501369e-26
#> 3    RYR3  celltype oligodendrocytes -7.315941 0.5571044 2.501425e-23
#> 4   GPR98  celltype oligodendrocytes -6.325254 0.4976605 1.902643e-22
#> 5  SLC1A2  celltype oligodendrocytes -4.608657 0.3644178 2.583854e-22
#> 6 SLC12A2  celltype oligodendrocytes  2.962680 0.2476006 7.101488e-21
#>           qval base.model tweedie.index   N N.not.zero percent.zero
#> 1 6.780362e-28       CPLM         1.643 100         87           13
#> 2 1.250684e-24       CPLM         1.671 100         85           15
#> 3 8.338083e-22       CPLM         1.626 100         69           31
#> 4 4.756608e-21       CPLM         1.577 100         72           28
#> 5 5.167707e-21       CPLM         1.595 100         80           20
#> 6 1.183581e-19       CPLM         1.685 100         85           15

# How many significant at 5% FDR
nrow(Brain[Brain$qval < 0.05, ])  # 85
#> [1] 86
```

Using the adjusted p-values, there are 85 genes significant at 5% FDR.

## Spatial Differential Expression Using boost

In this part, we will use the `boost` R package to conduct spatial variable gene (SVG) detection analysis using a variety of methods.

## Load Data

The current version of boost requires two input data:

- The gene expression count matrix $Y : n \times p$ (`n` = number of spots. `p` = number of genes)
- The location information matrix $T : n \times 2$. It includes the `x` and `y` coordinate for each sample point.

Both data should be stored in R matrix format. For gene expression count matrix $Y$, column names should be gene names.

In the following, we conduct the analysis on Mouse Olfactory Bulb data (replicate 11) as an example to show the functions in boost package.

```
# Download data from STAr website
url <- "https://lce.biohpc.swmed.edu/star/download/st/mouse_olfactory_bulb_st/processed_data/mouse_olfa
download.file(url, destfile = "mob.zip", mode = "wb")
unzip("mob.zip")

# Read data
count <- read.csv("mouse_olfactory_bulb_replicate_11.count.csv")
count <- as.matrix(count)
sample_info <- read.csv("mouse_olfactory_bulb_replicate_11.loc.csv")
sample_info <- as.matrix(sample_info[, 1:2])
gene_name <- read.csv("mouse_olfactory_bulb_replicate_11.gene_name.csv")

# Set column names of count as gene names
```

```
colnames(count) <- gene_name$x
print(dim(count))
#> [1]   262 16218
print(dim(sample_info))
#> [1] 262   2
```

The Mouse Olfactory Bulb data (replicate 11) has 262 sample points and 16218 genes. The `sample_info` table records the x and y coordinates of all sample points.

```
print(count[1:10, 1:10])
#>       Nrf1 Zbtb5 Ccnl1 Lrrfip1 Bbs1 Lix1 Whrn Ate1 Ubac1 Rab34
#>  [1,]    1     1     1       2    1    2    1    1     2     1
#>  [2,]    0     0     3       2    2    7    0    2     3     2
#>  [3,]    0     1     1       0    0    0    1    0     0     0
#>  [4,]    1     0     1       0    4    6    1    4     3     1
#>  [5,]    0     0     0       3    0    2    0    2     4     0
#>  [6,]    0     0     0       5    0    1    1    0     2     0
#>  [7,]    0     0     2       1    3    4    0    3     2     0
#>  [8,]    1     1     3       0    1    3    0    4     0     0
#>  [9,]    0     0     2       0    0    4    0    1     0     2
#> [10,]    0     0     2       3    0    2    0    1     0     0
print(head(sample_info))
#>           X      Y
#> [1,] 16.920  9.015
#> [2,] 16.945 11.075
#> [3,] 16.970 10.118
#> [4,] 16.939 12.132
#> [5,] 16.949 13.055
#> [6,] 16.942 15.088
```

## Data Pre-processing

Before SV gene detection, there are several steps to pre-process the data.

### Filtering and Quality Control

Spots with fewer than ten total counts across all genes are excluded. In addition, genes with more than 90% zero read counts across all spots are dropped.

```
# filter data
filter.st <- function(count, sample_info, min_total = 10, min_percentage = 0.1) {
    gene_num <- ncol(count)
    sample_num <- nrow(count)
    if (sum(rowSums(count) < min_total) == 0) {
        if (sum(colSums(count == 0) > (1 - min_percentage) *
            sample_num) > 0) {
            sample_f <- sample_info
            count_f <- count[, -(which(colSums(count == 0) >
                (1 - min_percentage) * sample_num))]
        } else {
            sample_f <- sample_info
            count_f <- count
        }
    } else {
        if (sum(colSums(count[-which(rowSums(count) < min_total),
```

```
            ] == 0) > (1 - min_percentage) * sample_num) > 0) {
            sample_f <- sample_info[-which(rowSums(count) < min_total),
                ]
            count_f <- count[-which(rowSums(count) < min_total),
                -(which(colSums(count[-which(rowSums(count) <
                  min_total), ] == 0) > (1 - min_percentage) *
                  sample_num))]
        } else {
            sample_f <- sample_info[-which(rowSums(count) < min_total),
                ]
            count_f <- count[-which(rowSums(count) < min_total),
                ]
        }
    }
    return(list(sample_f, count_f))
}


filter_result <- filter.st(count, sample_info)
count <- filter_result[[2]]
sample_info <- filter_result[[1]]

print(paste0("Dimension of count matrix: ", dim(count)[1], ", ",
    dim(count)[2]))
#> [1] "Dimension of count matrix: 260, 11360"
print(paste0("Dimension of location information matrix: ", dim(sample_info)[1],
    ", ", dim(sample_info)[2]))
#> [1] "Dimension of location information matrix: 260, 2"
```

## Size Factor Estimation (Normalization)

Size factor is the input for some SV gene detection approaches. In `boost` package, we use `get.size.factor` function to estimate the size factor. The inputs of this function are count matrix and one of the size factor estimation methods discussed in the class (`'TSS'`, `'Q75'`, `'RLE'`, and `'TMM'`). Output is a vector representing the estimated size factor for spots. In this example, we choose TSS (total sum scaling) as the estimation method.

```
size_factor <- get.size.factor(count, estimation.method = "TSS")
size_factor <- size_factor/mean(size_factor)
```

## SV Gene Detection

The `boost` R package includes the functions of five SV gene identification methods: `BOOST-GP`, `BOOST-MI`, `SPARK`, `BinSpect`, and `SpatialDE`. The expressions (counts or normalized levels) for one gene, and other information, are the inputs for these functions. We choose the gene `Doc2g` in Mouse Olfactory Bulb data (replicate 11) and show the SV gene detection process on it.

```
abs_expr <- count[, "Doc2g"]
```

### BinSpect

Before applying `BinSpect` (Binary Spatial Extraction of genes from Giotto), several data pre-processing steps are required. First, we need to binarize the normalized expression level using the `binarize.st` function.

```
normalized_count <- normalize.st(count, scaling.method = "TSS")
binary_expr <- binarize.st(normalized_count, "Doc2g", cluster.method = "GMC")
```

Instead of sample_info matrix, `BinSpect` method requires the neighbor information of sample points. We apply `get.neighbors` function to generate the neighbor information. Sample points for this data are located on a square lattice, so each spots has at most 4 neighbors. `BinSpect` requires the binarized expression levels for one gene and the neighbor information as inputs.

```
neighbor_info <- get.neighbors(sample_info, n.neighbors = 4,
    method = "distance")
result_binspect <- binSpect(binary_expr, neighbor_info, do.fisher.test = FALSE,
    gene.name = "Doc2g")
print(result_binspect)
#>
#> Call:
#> binSpect(bin.expr = binary_expr, neighbor.info = neighbor_info,
#>     do.fisher.test = FALSE, gene.name = "Doc2g")
#>
#> Model: BinSpect
#>
#> Contingency Table for Classified Edges:
#>     0   1
#> 0 298 144
#> 1 144 356
#>
#> Odds ratio in favor of a spatially-variable pattern: 5.12
#> p-value in favor of a spatially-variable pattern: <0.001
```

The outputs include the contingency table summarized from the neighbor pairing binarized expresssion. P-value is the criteria for determining whether a gene is an SV gene or not. If the output p-value is less than 0.05, we infer that it is an SV gene. In this example, p-value is less than 0.001, indicating strong evidence that gene `Doc2g` is an SV gene.

### SPARK

SPARK (Spatial PAttern Recognition via Kernels) requires the expression counts for one gene (one column of the original count matrix :$Y$), the location information $T$, and the estimated size factor as inputs.

```
result_SPARK <- SPARK(abs_expr, sample_info, size.factor = size_factor,
    gene.name = "Doc2g")
print(result_SPARK)
#>
#> Call:
#> SPARK(abs.expr = abs_expr, spots = sample_info, size.factor = size_factor,
#>     gene.name = "Doc2g")
#>
#> Model: SPARK
#>
#> Summary:
#>     GSP1   COS1    GSP2 COS2 GSP3 COS3 GSP4    COS4 GSP5    COS5
#>  2.6e-09 0.0045 6.1e-05    1 0.12 0.24  0.6 3.9e-11 0.73 2.8e-11
#>
#> p-value in favor of a spatially-variable pattern: <0.001
```

The outputs include the p-values under different kernel function settings. Combined p-value is the criteria for

determining whether a gene is an SV gene or not. If the output combined p-value is less than 0.05, we infer that it is an SV gene. In this example, the combined p-value is less than 0.001, indicating strong evidence that gene `Doc2g` is an SV gene.

### SpatialDE

`SpatialDE` method assumes the expression data are normally distributed. So instead of TSS normalization method, we need to use log-VST choice to normalize the data. This normalization method includes stabilizing the variance of counts data and regressing out the effect of library size. *NOTE: SpatialDE has a Python dependency!!!*

```
normalized_count_log_vst <- normalize.st(count, scaling.method = "log-VST")
norm_expr <- normalized_count_log_vst[, "Doc2g"]
result_spatialde <- SpatialDE(norm_expr, sample_info, gene.name = "Doc2g")
print(result_spatialde)
#>
#> Call:
#> SpatialDE(norm.expr = norm_expr, spots = sample_info, gene.name = "Doc2g")
#>
#> Model: SpatialDE
#>
#> Summary:
#>       g   n  FSV   l BIC
#>  Doc2g 260 0.57 1.1 609
#>
#> p-value in favor of a spatially-variable pattern: <0.001
```

The outputs include the fraction of expression variance (FSV), characteristic length scale in the kernel function, and Bayesian information criterion (BIC). P-value is the criteria for determining whether a gene is an SV gene or not. If the output p-value is less than 0.05, we infer that it is an SV gene. In this example, the p-value is less than 0.001, indicating strong evidence that gene `Doc2g` is an SV gene.

### Spatial Tweedie GAM

```
df <- cbind.data.frame(abs_expr, sample_info)
fit <- gam(abs_expr ~ s(X, Y), family = "tw", data = df)
summary(fit)
#>
#> Family: Tweedie(p=1.99)
#> Link function: log
#>
#> Formula:
#> abs_expr ~ s(X, Y)
#>
#> Parametric coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  3.12175    0.03747   83.32   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>          edf Ref.df     F  p-value
#> s(X,Y) 18.09  23.06 2.922 2.21e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#>
#> R-sq.(adj) =  0.107   Deviance explained = 21.9%
#> -REML = 1040.8  Scale est. = 0.37654   n = 260
coefficients(fit)
#> (Intercept)    s(X,Y).1    s(X,Y).2    s(X,Y).3    s(X,Y).4    s(X,Y).5
#>  3.12175337 -0.14501100 -0.65976309 -0.17950849 -0.06917796 -0.14935092
#>    s(X,Y).6    s(X,Y).7    s(X,Y).8    s(X,Y).9   s(X,Y).10   s(X,Y).11
#> -0.90988606 -0.20890708 -0.69682632 -0.70558478  0.26239238  0.19215739
#>   s(X,Y).12   s(X,Y).13   s(X,Y).14   s(X,Y).15   s(X,Y).16   s(X,Y).17
#>  0.14800460 -0.27830713 -0.59634830 -0.15560036  0.72815008 -0.70262690
#>   s(X,Y).18   s(X,Y).19   s(X,Y).20   s(X,Y).21   s(X,Y).22   s(X,Y).23
#> -0.56239398  0.42632223 -0.49242512  0.61525757  0.07053301  0.25170253
#>   s(X,Y).24   s(X,Y).25   s(X,Y).26   s(X,Y).27   s(X,Y).28   s(X,Y).29
#>  0.40304539  0.13286860  0.46204433  3.54900908 -0.20296714  0.39491965
summary(fit)$s.table
#>             edf  Ref.df        F      p-value
#> s(X,Y) 18.08912 23.0555 2.922016 2.210167e-05
```

Based on the results of the spatial Tweedie GAM model, the gene `Doc2g` is a spatially variable gene.

## Homework 3

- As we discussed in the class, one of the most popular package for single-cell data science is the `Seurat` package which is an R package from CRAN. Re-run the analysis of this lab using the PBMC data described in the `Seurat` single-cell workflow. In order to conduct DE analysis, consider the CD4 and CD8 cell types (see the https://github.com/himelmallick/TweedieLabs/blob/main/R/Tweedieverse_scRNASeq_UMI.html for inspiration). Note that, in order to get the cell types, you need to run all the steps in the Seurat tutorial
- Compare the results with both `edgeR` and `DESeq2` on the same data. Which genes are found DE in all analyses? Which method finds the most DE genes? Explore the similarities and differences graphically.
- The R package `boost` is great inasmuch as it is one of the most comprehensive R packages that includes many SVG detection methods. However, one serious shortcoming of this package is that most of the functions work on one gene at a time. In order to run on the full dataset, we need to write wrappers for these individual methods.

Repeat the spatial analysis by writing a `Spatial Tweedieverse` wrapper which takes as input a (normalized) count matrix of gene expression and a 2D matrix of spatial coordinates. Follow the preprocessing steps above and use the `gam()` function from the `mgcv` R package as the base function. Based on this analysis, benchmark various SVG detection methods in terms of the number of detected SVGs as follows:

Use the spatial dataset in the Seurat vignette [https://satijalab.org/seurat/articles/spatial_vignette] and apply the following five **distinct** methods to either the full dataset or a subset of randomly selected genes in case computation is an issue:

- Spatial Tweedieverse (Wrapper must be written)
- nnSVG (Bioconductor package)
- SPARK-X (GitHub package)
- Any of the methods implemented in the `boost` package (Wrapper must be written)
- Any of the methods implemented in the `Seurat` package (using the `FindSpatiallyVariableFeatures` function)

## Session information

```
sessionInfo()
#> R version 4.3.2 (2023-10-31)
#> Platform: aarch64-apple-darwin20 (64-bit)
#> Running under: macOS Sonoma 14.4.1
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib;  LAPACK v
#>
#> locale:
#> [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
#>
#> time zone: America/New_York
#> tzcode source: internal
#>
#> attached base packages:
#> [1] stats4    stats     graphics  grDevices utils     datasets  methods
#> [8] base
#>
#> other attached packages:
#>  [1] mgcv_1.9-1                  nlme_3.1-164
#>  [3] Seurat_5.0.3                SeuratObject_5.0.1
#>  [5] sp_2.1-3                    spatialDE_1.8.1
#>  [7] SPARK_1.1.1                 boost_0.2.0
#>  [9] SC2P_1.0.2                  limma_3.58.1
#> [11] scran_1.30.2                scuttle_1.12.0
#> [13] SingleCellExperiment_1.24.0 SummarizedExperiment_1.32.0
#> [15] Biobase_2.62.0              GenomicRanges_1.54.1
#> [17] GenomeInfoDb_1.38.8         IRanges_2.36.0
#> [19] S4Vectors_0.40.2            BiocGenerics_0.48.1
#> [21] MatrixGenerics_1.14.0       matrixStats_1.2.0
#> [23] Tweedieverse_0.0.1          ggplot2_3.5.0
#> [25] knitr_1.45
#>
#> loaded via a namespace (and not attached):
#>   [1] spatstat.sparse_3.0-3     bitops_1.0-7
#>   [3] httr_1.4.7                RColorBrewer_1.1-3
#>   [5] doParallel_1.0.17         tools_4.3.2
#>   [7] sctransform_0.4.1         backports_1.4.1
#>   [9] utf8_1.2.4                R6_2.5.1
#>  [11] lazyeval_0.2.2            uwot_0.1.16
#>  [13] withr_3.0.0               gridExtra_2.3
#>  [15] progressr_0.14.0          cli_3.6.2
#>  [17] formatR_1.14              logging_0.10-108
#>  [19] spatstat.explore_3.2-7    fastDummies_1.7.3
#>  [21] biglm_0.9-2.1             labeling_0.4.3
#>  [23] spatstat.data_3.0-4       ggridges_0.5.6
#>  [25] pbapply_1.7-2             foreign_0.8-86
#>  [27] dbscan_1.1-12             parallelly_1.37.1
#>  [29] rstudioapi_0.16.0         generics_0.1.3
#>  [31] gtools_3.9.5              ica_1.0-3
#>  [33] spatstat.random_3.2-3     dplyr_1.1.4
```

```
#>  [35] Matrix_1.6-5               fansi_1.0.6
#>  [37] abind_1.4-5                lifecycle_1.0.4
#>  [39] yaml_2.3.8                 edgeR_4.0.16
#>  [41] CompQuadForm_1.4.3         gplots_3.1.3.1
#>  [43] SparseArray_1.2.4          Rtsne_0.17
#>  [45] grid_4.3.2                 promises_1.2.1
#>  [47] dqrng_0.3.2                crayon_1.5.2
#>  [49] dir.expiry_1.10.0          miniUI_0.1.1.1
#>  [51] lattice_0.22-6             beachmat_2.18.1
#>  [53] cowplot_1.1.3              magick_2.8.3
#>  [55] pillar_1.9.0               metapod_1.10.1
#>  [57] rjson_0.2.21               future.apply_1.11.2
#>  [59] codetools_0.2-20           leiden_0.4.3.1
#>  [61] glue_1.7.0                 data.table_1.15.4
#>  [63] vctrs_0.6.5                png_0.1-8
#>  [65] spam_2.10-0                gtable_0.3.4
#>  [67] xfun_0.43                  S4Arrays_1.2.1
#>  [69] mime_0.12                  pracma_2.4.4
#>  [71] coda_0.19-4.1              survival_3.5-8
#>  [73] iterators_1.0.14           statmod_1.5.0
#>  [75] bluster_1.12.0             cplm_0.7-12
#>  [77] fitdistrplus_1.1-11        ROCR_1.0-11
#>  [79] filelock_1.0.3             RcppAnnoy_0.0.22
#>  [81] irlba_2.3.5.1              KernSmooth_2.23-22
#>  [83] rpart_4.1.23               matlab_1.0.4
#>  [85] DBI_1.2.2                  colorspace_2.1-0
#>  [87] Hmisc_5.1-2                nnet_7.3-19
#>  [89] tidyselect_1.2.1           compiler_4.3.2
#>  [91] htmlTable_2.4.2            BiocNeighbors_1.20.2
#>  [93] basilisk.utils_1.14.1      DelayedArray_0.28.0
#>  [95] plotly_4.10.4              checkmate_2.3.1
#>  [97] scales_1.3.0               caTools_1.18.2
#>  [99] lmtest_0.9-40              stringr_1.5.1
#> [101] SpatialExperiment_1.12.0   digest_0.6.35
#> [103] goftest_1.2-3              minqa_1.2.6
#> [105] spatstat.utils_3.0-4       rmarkdown_2.26
#> [107] basilisk_1.14.3            XVector_0.42.0
#> [109] htmltools_0.5.8.1          pkgconfig_2.0.3
#> [111] base64enc_0.1-3            sparseMatrixStats_1.14.0
#> [113] fastmap_1.1.1              rlang_1.1.3
#> [115] htmlwidgets_1.6.4          shiny_1.8.1.1
#> [117] DelayedMatrixStats_1.24.0  farver_2.1.1
#> [119] zoo_1.8-12                 jsonlite_1.8.8
#> [121] BiocParallel_1.36.0        mclust_6.1
#> [123] BiocSingular_1.18.0        RCurl_1.98-1.14
#> [125] magrittr_2.0.3             Formula_1.2-5
#> [127] GenomeInfoDbData_1.2.11    dotCall64_1.1-1
#> [129] patchwork_1.2.0            munsell_0.5.1
#> [131] Rcpp_1.0.12                reticulate_1.35.0
#> [133] stringi_1.8.3              pROC_1.18.5
#> [135] zlibbioc_1.48.2            MASS_7.3-60.0.1
#> [137] plyr_1.8.9                 parallel_4.3.2
#> [139] listenv_0.9.1              ggrepel_0.9.5
```

```
#> [141] deldir_2.0-4              splines_4.3.2
#> [143] tensor_1.5                locfit_1.5-9.9
#> [145] igraph_2.0.3              spatstat.geom_3.2-9
#> [147] RcppHNSW_0.6.0            reshape2_1.4.4
#> [149] ScaledMatrix_1.10.0       evaluate_0.23
#> [151] TopKLists_1.0.8           foreach_1.5.2
#> [153] httpuv_1.6.15             RANN_2.6.1
#> [155] tidyr_1.3.1               purrr_1.0.2
#> [157] polyclip_1.10-6           future_1.33.2
#> [159] scattermore_1.2           rsvd_1.0.5
#> [161] tweedie_2.3.5             xtable_1.8-4
#> [163] RSpectra_0.16-1           later_1.3.2
#> [165] viridisLite_0.4.2         tibble_3.2.1
#> [167] cluster_2.1.6             globals_0.16.3
```