## CONTENTS:

1 **Installation** 3

2 **Example - Human Breast Cancer 10x Visium Data** 5
   2.1 Load Data .................................................. 5
   2.2 Process Data .................................................. 6
   2.3 Spatial Domain Identification ................................. 7
   2.4 Domain-specific Spatially Variable Gene Detection .......... 10

3 **Example - Mouse Visual Cortex STARmap Data** 13
   3.1 Load Data .................................................. 13
   3.2 Process Data .................................................. 14
   3.3 Spatial Domain Identification ................................. 15

4 **Environment** 17
iIMPACT is a novel method which integrates image and molecular profiles for spatial transcriptomics analysis. It is a multi-stage statistical method for spatial domain identification and domain-specific spatially variable gene detection. It utilizes an interpretable Bayesian finite mixture model for analyzing the cellular spatial organization and a regression model for domain-specific spatially variable gene analysis. Compared with existing methods for spatial transcriptomics data analysis, iIMPACT fully leverages the biological features from histology image, which causes compromised algorithm accuracy and the interpretability of histological characterization of single cells on domain level. iIMPACT provides a highly accurate and interpretable clustering approach to reveal cellular spatial organization and functional gene landscape from spatial transcriptomics data.
iIMPACT provides a novel method for spatial domain identification in spatially resolved transcriptomics (SRT) data, which integrates image and molecular profiles to improve the domain identification accuracy. It also has the ability to detect domain-specific spatially variable genes via a negative binomial regression model.

To implement iIMPACT function, you need to make sure that the following packages are already installed.

```r
library(ggplot2)
library(Rcpp)
library(RcppArmadillo)
library(RcppDist)
library(DirichletReg)
library(mvtnorm)
library(LaplacesDemon)
library(SingleCellExperiment)
library(scater)
library(scran)
```

Load functions used to run iIMPACT.

```r
source('R/iIMPACT_functions.R')
```
EXAMPLE - HUMAN BREAST CANCER 10X VISIUM DATA

In the following, we choose the human breast cancer FFPE data from 10x Visium platform as an example to display the implementation of iIMPACT on the sequencing-based SRT data.

2.1 Load Data

The current version of iIMPACT requires three input data:

- The gene expression count matrix ‘count’: \( n \) by \( p \) (\( n \) - number of spots; \( p \) - number of genes)
- The location information matrix ‘loc’: \( n \) by 2. It includes the x and y coordinate for each sample point.
- The nuclei identification information matrix ‘cell_info’: \( m \) by 3. It includes the x and y coordinate and the nuclei class for identified cells.

The first two data should be stored in R matrix format. For gene expression count matrix, column names should be gene names.

Files can be downloaded from ‘data’ folder on the Dropbox: https://www.dropbox.com/scl/fo/em51owbpda4id0rnin1x/h?dl=0&rlkey=nk9kc38ghs9w4pqlno7k3e1q

```r
# read data
cell_info <- read.csv('data/human breast cancer FFPE data/10x_breast_cancer_ffpe_cell_info.csv')
spot_loc <- read.csv('data/human breast cancer FFPE data/10x_human_breast_cancer_ffpe_loc.csv', row.names = 1)
count <- read.csv('data/human breast cancer FFPE data/10x_human_breast_cancer_ffpe_count.csv', row.names = 1)

print(dim(count))
## [1] 2518 17943

print(dim(spot_loc))
## [1] 2518 2

print(dim(cell_info))
## [1] 156235 4
```

This human breast cancer FFPE data has 2,518 sample points and 17,943 genes. For the histology image, the HD staining model identified 156,235 nuclei.


## 2.2 Process Data

Before running iIMPACT for spatial domain identification, there are several steps to prepare the data.

### 2.2.1 Generate cell abundance data

We process the nuclei segmentation results from the HD staining model to the cell abundance data \( V \) by counting the cells with different types in each expanded area.

```r

cell_loc <- cell_info[, c('x', 'y')]
cell_type <- cell_info$nucleus_class
V <- get.cell.abundance(cell_loc, cell_type, spot_loc, lattice = 'hexagon')
# [1] "0% has been done"
# [1] "10% has been done"
# [1] "20% has been done"
# [1] "30% has been done"
# [1] "40% has been done"
# [1] "50% has been done"
# [1] "60% has been done"
# [1] "70% has been done"
# [1] "80% has been done"
# [1] "90% has been done"
# [1] "100% has been done"

print(dim(V))
# [1] 2518 7

print(head(V))
# # stroma necrosis lymphocyte blood tumor ductal epithelium macrophage
# [1,] 105 7 28 38 9 3 2
# [2,] 60 0 37 39 0 0 0
# [3,] 81 6 38 20 1 0 0
# [4,] 73 2 21 15 0 0 0
# [5,] 63 1 63 19 0 0 0
# [6,] 86 1 70 19 0 0 0
```

For the histology image, the model identified 156,235 nuclei with 7 cell types.

The obtained cell abundance data \( V \) has dimension 2,518 by 7 (\( n \) by \( q \) matrix, \( q \) is the number of cell types).

### 2.2.2 Generate low-dimensional representation of molecular profiles

Before fitting the finite mixture model, the raw gene expression counts need to be normalized and transformed to the logarithmic scale. We select the normalized expression levels of top 2,000 highly variable genes and do the dimensionality reduction to reduce the dimension using principal component analysis (PCA). Here we set the reduced dimension to be 3.

```r
Y <- process.gene.expression(count, n_PC = 3)

print(dim(Y))
# [1] 2518 3
```

(continues on next page)
print(head(Y))
## PC1   PC2   PC3
## 1  -24.172837 -4.584487 5.5806516
## 2   12.371814 -7.119854 -1.1115033
## 3   -3.332869 8.108653 0.7678203
## 4   18.560136 -5.305400 6.4929754
## 5   19.731972 4.777125 3.9005968
## 6   15.068430 3.544484 -5.7039848

2.2.3 Generate neighborhood information

Instead of spot coordinates, iIMPACT requires the neighbor information of spots. We apply `get.neighbor` function to generate the neighbor information. Sample points for this data are located on a hexagon lattice, so each spots has 6 neighbors.

G <- get.neighbor(spot_loc, 6)

2.3 Spatial Domain Identification

2.3.1 Run Bayesian normal-multinomial mixture model

`run.iIMPACT` function requires the cell abundance data from image profile `V`, molecular profile from SRT data `Y` and neighborhood information `G` as input. We also need to set two parameters: the number of domains (clusters) `n_cluster`, and the scaling parameter to control the contribution of image profile `w`.

After fitting the finite mixture model, a label switching step is necessary. We can specify a cell type as the reference of label switching and pass the corresponding column index in `V` to the function via the `label_switch_refer` parameter. The default index is 1.

# set number of clusters
K <- 5

# set the scaling parameter for image profile
w <- 1/20

# run iIMPACT
result <- run.iIMPACT(V, Y, G, n_cluster = K, w)
## 10% has been done
## 20% has been done
## 30% has been done
## 40% has been done
## 50% has been done
## 60% has been done
## 70% has been done
## 80% has been done
## 90% has been done
## [1] "100% has been done"
2.3.2 Characterize identified spatial domains

After obtaining the posterior samples of Bayesian mixture model via the run.iIMPACT function, we can obtain the spatial domain identification results via the get.spatial.domain function.

```r
spatial_domain <- get.spatial.domain(result)

# plot results
df <- data.frame(x = spot_loc$x, y = spot_loc$y, domain = as.factor(spatial_domain))
ggplot(df, aes(x = x, y = y, color = domain)) +
  geom_point() + scale_color_manual(values=c('1' = '#006400', '2' = '#0000ff', '3' = '#A020F0', '4' = '#ffd800', '5' = '#e41a1a'))
```

Get domain-level cell proportion: each row is the cell-type proportion for the corresponding domain (cluster).

```r
domain_cell_prop <- get.domain.cell.prop(result)
print(domain_cell_prop)
```

```r
## stroma necrosis lymphocyte blood tumor ductal epithelium
## [1,] 0.2407925 0.009875297 0.19702032 0.53671201 0.007796473 0.002970890
## [2,] 0.3798334 0.040353830 0.06321755 0.30010776 0.194550348 0.017041318
## [3,] 0.4022183 0.043747324 0.04294920 0.08177120 0.367408090 0.060929387
## [4,] 0.5122341 0.020637669 0.25527049 0.20140560 0.007971780 0.002090986
## [5,] 0.5443293 0.017708266 0.30624709 0.08886796 0.034487317 0.007642422
## macrophage
## [1,] 0.0048135037
## [2,] 0.0046354940
## [3,] 0.000006316
## [4,] 0.0001363861
## [5,] 0.0007175787
```
Get interactive zones: spots with high uncertainty on domain assignment.

```r
interactive_zone <- get.interactive.zone(result)

df <- data.frame(x = spot_loc$x, y = spot_loc$y, interactive_zone = interactive_zone)
ggplot(df, aes(x = x, y = y, color = as.factor(interactive_zone))) +
  geom_point() + scale_color_manual(values=c('TRUE' = "black", 'FALSE' = "grey"))
```

### 2.3.3 Refine spatial domain results

iIMPACT provides an optional refinement step for the spatial domain identification results. In this step, we need to define a parameter ‘area_unit’ as an unit of small area. For an area with the number of spots is less or equal to the ‘area_unit’, if all neighbors of this area belong to a same cluster, the clustering result of this small area will be relabeled to the same domain of its neighboring area.

```r
spatial_domain_refined <- refine.cluster(G, spatial_domain, area_unit = 3)

# plot results
df <- data.frame(x = spot_loc$x, y = spot_loc$y, domain = spatial_domain_refined)
ggplot(df, aes(x = x, y = y, color = as.factor(domain))) +
  geom_point() + scale_color_manual(values=c('1' = "#006400", '2' = "#0000ff", '3' = "#A020F0", '4' = "#ffd800", '5' = "#e41a1a" ))
```
2.4 Domain-specific Spatially Variable Gene Detection

The second step of iIMPACT is to detect domain-specific SV genes based on the domains identified by the previous step via a negative binomial regression model.

Before fitting the regression model, we need to filter out genes with a high proportion of zero counts. `filter.count` function takes count matrix as input and can output genes (columns) with non-zero entries equal or greater than `min_percentage`.

```r
count_f <- filter.count(count, min_percentage = 0.3)
```

We also need the estimated size factor in the regression model. `get.size.factor` function can estimate size factor through different methods. Here we apply total sum scaling (tss) method by setting the parameter `norm_method` as `tss`.

```r
size_factor <- get.size.factor(count_f, 'tss')
```

In the second stage of iIMPACT, a negative binomial regression model is fitted for a pre-specified spatial domain, and then domain-specific spatially variable genes can be defined via the output p-values. `detect.domainSVG` takes the filtered count matrix, spatial domain assignment results from the previous step, target domain index, and estimated size factor as input, and outputs the estimated coefficients for domain assignment covariate and corresponding p-values for all genes.

```r
# set the domain for domain-specific spatially variable genes
domain_index <- 1

re <- detect.domainSVG(count_f, spatial_domain_refined, domain_index, size_factor)
# [1] "0% has been done"
## [1] "10% has been done"
## [1] "20% has been done"
## [1] "30% has been done"
```

(continues on next page)
(continued from previous page)

## 

```r
# 40% has been done
# 50% has been done
# 60% has been done
# 70% has been done
# 80% has been done
# 90% has been done
# 100% has been done

print(re[1:10, ])

# gene          beta         p_value adjusted_p_value
# 1  NOC2L    -0.1831442  5.306479e-02   8.549866e-02
# 2    HES4    -0.2014012  9.506305e-03   1.870248e-02
# 3   ISG15    -0.3104067  2.107101e-09   1.177261e-08
# 4    AGRN    -0.4901692  1.876475e-23   3.630437e-22
# 5     SDF4    0.1109305  6.393015e-04   1.596662e-03
# 6   B3GALT6   0.1375232  7.804545e-02   1.195188e-01
# 7     UBE2J    -0.1196468  1.065454e-01   1.571002e-01
# 8    ACAP3    -0.2338464  1.438486e-03   3.395794e-03
# 9   INTS11    -0.1668425  2.379041e-03   5.347124e-03
#10     CPTP    -0.2228142  2.216970e-02   3.926804e-02
```

2.4. Domain-specific Spatially Variable Gene Detection
EXAMPLE - MOUSE VISUAL CORTEX STARMAP DATA

This dataset was generated from an imaging-based SRT technology platform with a resolution of single-cell level. Data can be downloaded from ‘data’ folder on the Dropbox: https://www.dropbox.com/scl/fo/em51owbpda4id0rnin1x/h?dl=0&rlkey=nnk9kc38ghs9wdjpqno7k3e1qp

3.1 Load Data

For data generated from imaging-based techniques, the current version of iIMPACT requires two input data:

- The gene expression count matrix ‘count’: \( m \) by \( p \) (\( m \) - number of cells; \( p \) - number of genes)
- The location and cell type information matrix ‘cell_info’: \( m \) by 3. It includes the \( x \) and \( y \) coordinate, and the cell type for each cell.

These two data should be stored in R matrix format. For gene expression count matrix, column names should be gene names.

```r
# read data
starmap_data <- read.csv('data/mouse visual cortex STARmap data/mouse_visual_cortex_STARmap.csv')

# get count and cell_info
count <- starmap_data[, -(1:5)]
cell_info <- starmap_data[, c('x_pixel', 'y_pixel', 'cell_type')] colnames(cell_info)[1:2] <- c('x', 'y')

print(dim(count))
## [1] 1207 1020

print(dim(cell_info))
## [1] 1207 3
```

This mouse visual cortex STARmap data has dimension 1,207 cells and 1,020 genes.
3.2 Process Data

Before running iIMPACT for spatial domain identification, there are several steps to prepare the data. iIMPACT conducts clustering on spot level, so we need to create a grid lattice and assign each cell to the corresponding spot.

3.2.1 Create grid

The first step to handle the imaging-based SRT data is to manually add grids with appropriate size on the whole tissue region.

```r
grid_spot <- create.grid(cell_info, size = 750)
spot_loc <- grid_spot[['spot_loc']]
cell_assignment <- grid_spot[['cell_assignment']]`n
# plot cell and assigned square lattice
plot(cell_info$y, cell_info$x, col = as.factor(cell_info$cell_type), pch = 16, asp = 1, xlab = 'y', ylab = 'x')
points(spot_loc[, 'y'], spot_loc[, 'x'], cex = 2, pch = 16)
```

3.2.2 Generate cell abundance, low-dimensional representation of molecular profiles, and neighborhood information

After creating grids, we obtain cell abundance data $V$ as the counts of cells with different types in each square area. For single-cell level molecular profiles, we normalize, transforme, and reduce the dimension of the gene expression counts following the same steps for data from sequencing-based techniques. Low-dimensional gene expression profile $Y$ is then transformed to the spot level by averaging across all cells in each spot.

```r
# Generate cell abundance and low-dimensional representation of molecular profiles
data_for_iIMPACT <- process.imaging.based.SRT(count, cell_info, cell_assignment, n_PC = ...)
```

(continues on next page)
3.3 Spatial Domain Identification

3.3.1 Run finite mixture model

`run.iIMPACT` function requires the cell abundance data from image profile $V$, molecular profile $Y$ and neighborhood information $G$ as input. We also need to set two parameters: the number of domains (clusters) `n_cluster`, and the scaling parameter to control the contribution of image profile `w` (set as 0.5). After fitting the finite mixture model, a label switching step is necessary.

```r
# set number of clusters
K <- 7

# set the scaling parameter for image profile
w <- 1/2

# run iIMPACT
result <- run.iIMPACT(V, Y, G, n_cluster = K, w)
## 10% has been done
## 20% has been done
## 30% has been done
## 40% has been done
## 50% has been done
## 60% has been done
## 70% has been done
## 80% has been done
## 90% has been done
## [1] "100% has been done"
```

3.3.2 Characterize identified spatial domains

After obtaining the posterior samples of Bayesian mixture model via the `run.iIMPACT` function, we can obtain the spatial domain identification results via the `get.spatial.domain` function. Note that this clustering result is at spot level. To project the results back to single cell level, we need to use the `get.cell.spatial.domain` function.

```r
spatial_domain <- get.spatial.domain(result)
spatial_domain_cell <- get.cell.spatial.domain(spatial_domain, cell_assignment)

# plot results at single cell level
df <- data.frame(x = cell_info$y, y = cell_info$x, domain = as.factor(spatial_domain_cell))
```

(continues on next page)
ggplot(df, aes(x = x, y = y, color = domain)) +
  geom_point()
Our computational environment is below.

```r
sessionInfo()
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252  
## [2] LC_CTYPE=English_United States.1252  
## [3] LC_MONETARY=English_United States.1252  
## [4] LC_NUMERIC=C  
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats4   stats   graphics grDevices utils   datasets methods
## [8] base
##
## other attached packages:
## [1] scran_1.20.1    scater_1.20.1
## [3] scuttle_1.2.1   SingleCellExperiment_1.14.1
## [5] SummarizedExperiment_1.22.0 Biobase_2.52.0
## [7] GenomicRanges_1.44.0  GenomeInfoDb_1.28.4
## [9] IRanges_2.26.0   S4Vectors_0.30.2
## [11] BiocGenerics_0.40.0  MatrixGenerics_1.4.3
## [13] matrixStats_0.61.0   LaplacesDemon_16.1.6
## [15] mvtnorm_1.1-3   DirichletReg_0.7-1
## [17] Formula_1.2-4    RcppDist_0.1.1
## [19] RcppArmadillo_0.11.4.3.1  Rcpp_1.0.10
## [21] ggplot2_3.4.0

## loaded via a namespace (and not attached):
## [1] bitops_1.0-7 tools_4.1.1
## [3] bslib_0.4.0 utf8_1.2.3
## [5] R6_2.5.1 irlba_2.3.5
## [7] viper_0.4.5 DBI_1.1.3
## [9] colorspace_2.1-0 withr_2.5.0
## [11] tidyselect_1.1.2 gridExtra_2.3
```

(continues on next page)
## 

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>compiler_4.1.1</td>
<td>cli_3.4.0</td>
</tr>
<tr>
<td>BiocNeighbors_1.10.0</td>
<td>DelayedArray_0.18.0</td>
</tr>
<tr>
<td>sandwich_3.0-2</td>
<td>labeling_0.4.2</td>
</tr>
<tr>
<td>sass_0.4.2</td>
<td>scales_1.2.1</td>
</tr>
<tr>
<td>stringr_1.4.1</td>
<td>digest_0.6.29</td>
</tr>
<tr>
<td>rmarkdown_2.16</td>
<td>XVector_0.32.0</td>
</tr>
<tr>
<td>pkgconfig_2.0.3</td>
<td>htmltools_0.5.3</td>
</tr>
<tr>
<td>sparseMatrixStats_1.14.2</td>
<td>highr_0.9</td>
</tr>
<tr>
<td>limma_3.48.3</td>
<td>fastmap_1.1.0</td>
</tr>
<tr>
<td>rlang_1.0.6</td>
<td>rstudioapi_0.14</td>
</tr>
<tr>
<td>DelayedMatrixStats_1.14.3</td>
<td>farver_2.1.1</td>
</tr>
<tr>
<td>jquerylib_0.1.4</td>
<td>generics_0.1.3</td>
</tr>
<tr>
<td>zoo_1.8-9</td>
<td>jsonlite_1.8.0</td>
</tr>
<tr>
<td>BiocParallel_1.26.2</td>
<td>dplyr_1.0.10</td>
</tr>
<tr>
<td>RCurl_1.98-1.5</td>
<td>magrittr_2.0.2</td>
</tr>
<tr>
<td>BiocSingular_1.8.1</td>
<td>GenomeInfoDbData_1.2.6</td>
</tr>
<tr>
<td>Matrix_1.5-1</td>
<td>ggbeeswarm_0.6.0</td>
</tr>
<tr>
<td>munsell_0.5.0</td>
<td>fansi_1.0.4</td>
</tr>
<tr>
<td>viridis_0.6.2</td>
<td>lifecycle_1.0.3</td>
</tr>
<tr>
<td>yaml_2.3.5</td>
<td>stringi_1.7.8</td>
</tr>
<tr>
<td>edgeR_3.34.1</td>
<td>zlibbioc_1.38.0</td>
</tr>
<tr>
<td>grid_4.1.1</td>
<td>dqrng_0.3.0</td>
</tr>
<tr>
<td>parallel_4.1.1</td>
<td>lattice_0.20-44</td>
</tr>
<tr>
<td>beachmat_2.8.1</td>
<td>locfit_1.5-9.4</td>
</tr>
<tr>
<td>metapod_1.0.0</td>
<td>knitr_1.40</td>
</tr>
<tr>
<td>pillar_1.8.1</td>
<td>igraph_1.2.11</td>
</tr>
<tr>
<td>ScaledMatrix_1.0.0</td>
<td>glue_1.6.1</td>
</tr>
<tr>
<td>evaluate_0.16</td>
<td>vctrs_0.5.2</td>
</tr>
<tr>
<td>miscTools_0.6-26</td>
<td>gtable_0.3.1</td>
</tr>
<tr>
<td>purrr_0.3.4</td>
<td>assertthat_0.2.1</td>
</tr>
<tr>
<td>cachem_1.0.6</td>
<td>xfun_0.33</td>
</tr>
<tr>
<td>rsvid_1.0.5</td>
<td>viridisLite_0.4.1</td>
</tr>
<tr>
<td>tibble_3.1.8</td>
<td>beeswarm_0.4.0</td>
</tr>
<tr>
<td>cluster_2.1.2</td>
<td>maxLik_1.5-2</td>
</tr>
<tr>
<td>statmod_1.4.36</td>
<td>bluster_1.2.1</td>
</tr>
</tbody>
</table>