JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 00, Number 00, 2025

Pp. 000–000
DOI: 10.1177/15578666251383562

ClusterDE: A Statistical Software Package for Removing Double-Dipping Bias in Post-Clustering Differential Expression Analysis

CHRISTY LEE,[1] DONGYUAN SONG,[2] SIQI CHEN,[1] and JINGYI JESSICA LI[1,*]

ABSTRACT

Typical pipelines for single-cell and spatial transcriptomics involve clustering cells or spatial spots, followed by post-clustering differential expression (DE) analysis to identify marker genes for annotating clusters as cell types or spatial domains. However, using the same data for both clustering and DE analysis—a problem known as double-dipping—can lead to spurious detection of DE genes. In particular, over-clustering can produce artificial clusters that are incorrectly interpreted as distinct cell types or spatial domains. To address this issue, the ClusterDE R package implements a statistical method using a synthetic null dataset, which consists of a single homogeneous cell population or spatial domain but is constructed to match the real dataset in terms of gene means, variances, and gene-gene rank correlations. By serving as a parallel negative control, the synthetic null data allow users to identify and remove false-positive DE genes arising from double-dipping. This article introduces the ClusterDE R package and provides practical guidance on installation and usage for more reliable marker gene detection following clustering.

Keywords: clustering, differential expression, single-cell RNA-seq, spatial transcriptomics.

1. INTRODUCTION

Single-cell RNA-seq (scRNA-seq) and spatial transcriptomics (ST) provide transcriptomic profiles of individual cells and spatial locations, respectively. A common analytical goal is to cluster cells or spatial spots and then perform post-clustering differential expression (DE) analysis to identify cluster marker genes for annotating clusters as cell types or spatial domains. However, using the same data for both clustering and DE analysis can introduce bias: gene expression differences may arise between clusters that result from over-clustering, leading to false-positive DE genes and misannotation of spurious clusters. ClusterDE (Song et al., 2024, 2025) provides a statistical solution to this problem by generating synthetic null data—either a single homogeneous population of synthetic cells or a smooth spatial domain without sharp gene expression changes—designed to preserve the gene means, variances, and gene–gene rank correlations observed in the real data. The synthetic null data serves as a parallel negative control to help identify and remove spurious DE genes caused by over-clustering. We refer to the real data as the target data and the synthetic null data as the null data.

[1]Department of Statistics and Data Science, University of California, Los Angeles, California, USA.
[2]Department of Genetics and Genome Sciences, University of Connecticut School of Medicine, Connecticut, USA.
*Present Address: Biostatistics Program, Public Health Sciences Division, Fred Hutchinson Cancer Center, Seattle, WA 98109, USA.

This article provides a brief guide to using the `ClusterDE` R package. For troubleshooting, feedback, or additional documentation and tutorials, please visit the GitHub page: https://songdongyuan1994.github.io/ClusterDE/index.html.

2. TECHNICAL INSIGHTS: SYNTHETIC NULL DATA GENERATION AND CONTRASTIVE USE

2.1. Synthetic null data generation

`ClusterDE` generates an *in silico* negative control ("synthetic null data") representing a single homogeneous population (for scRNA-seq) or a single smooth spatial domain (for ST). For scRNA-seq, the null simulator generates the same number of synthetic cells as in the target data, while preserving per-gene means and variances and approximating gene–gene rank correlations (Kendall's τ); by default, counts for each gene are drawn from a negative binomial distribution. For ST, the null model additionally imposes spatial smoothness via a Gaussian process layer, so that genes may vary smoothly over space but do not exhibit sharp boundaries between putative domains.

2.2. Parallel analysis with the same pipeline

To ensure a fair contrast, the synthetic null data are processed with the same preprocessing, dimensionality reduction, clustering, and DE-testing pipeline (e.g., Seurat or BayesSpace + Seurat) as the target data. For clustering algorithms that control the number of clusters indirectly (e.g., Louvain or Leiden clustering), the resolution parameter is tuned on the null so that exactly two clusters are obtained, mirroring the two clusters under examination in the target data. This parallel analysis enables ClusterDE to evaluate whether the two putative clusters in the target data represent reliable standalone clusters rather than artifacts of over-clustering.

2.3. Contrast score and error control

For each gene, `ClusterDE` computes a target DE score and a null DE score, defined by default as the negative logarithm of the DE *p* value from the user's chosen test (filters such as minimum expression fraction and log-fold-change thresholds are disabled so every gene receives a score). `ClusterDE` then constructs a contrast score for each gene by subtracting the null DE score from the target DE score; reliable markers are those whose contrast scores exceed a data-adaptive cutoff selected by Clipper (Ge et al., 2021) to control the false discovery rate (FDR) at a user-specified level (e.g., 0.05).

2.4. Assumptions, tuning, and practical tips

The scRNA-seq null model assumes a single true cell type with unimodal gene expression, whereas the ST null model assumes a single spatial domain with smooth spatial variation. In practice, users are advised to (1) apply the same feature-selection and normalization steps to both the target and synthetic null data, (2) disable DE prefilters so that every gene receives both a target and a null DE score, and (3) record key simulator settings (e.g., distribution family, smoothing basis/knots, random seeds, number of cores) to ensure reproducibility. Sensitivity analyses across different clustering resolutions on the synthetic null data can further help confirm that conclusions are not driven by a specific tuning choice.

3. INSTALLATION

To install the `ClusterDE` R package from GitHub, users must first install the `devtools` R package. In addition, the `scDesign3` package—used for simulating single-cell and spatial transcriptomic data—must also be installed.

```
if (!require("devtools", quietly = TRUE))
    install.packages("devtools")
devtools::install_github("SONGDONGYUAN1994/scDesign3")
devtools::install_github("SONGDONGYUAN1994/ClusterDE")
library(ClusterDE)
```

4. APPLICATION TO SINGLE-CELL RNA-SEQ DATA

For this example, we use the pbmc3k dataset from the SeuratData package (Satija Lab, 2020). Users are required to have the Seurat R package installed beforehand.

```
library(Seurat)
devtools::install_github("satijalab/seurat-data")
library(SeuratData)
InstallData("pbmc3k")
data("pbmc3k")
```

We begin by preprocessing and clustering the cells. While this tutorial follows the standard Seurat pipeline, users may modify these steps to suit their preferred analysis workflow.

```
pbmc3k <- UpdateSeuratObject(pbmc3k)
pbmc3k <- NormalizeData(pbmc3k)
pbmc3k <- FindVariableFeatures(pbmc3k)
pbmc3k <- ScaleData(pbmc3k)
pbmc3k <- RunPCA(pbmc3k)
pbmc3k <- FindNeighbors(pbmc3k)
pbmc3k <- FindClusters(pbmc3k)
```

ClusterDE analyzes one pair of clusters at a time. In cases with multiple clusters, we recommend applying ClusterDE along a hierarchy of clusters to identify marker genes that distinguish each branch. Users may use any hierarchical clustering method to organize the clusters into a tree structure. In this tutorial, we demonstrate the approach using clusters 4 and 5, which are identified as leaf nodes by Seurat's BuildClusterTree function. No DE genes are expected in this case, as both clusters correspond to CD14+ monocytes based on the provided cell-type annotations—although in practice, this prior knowledge would typically be unavailable.

For a given pair of clusters, we obtain *p* values using Seurat's standard DE approach. To ensure that every gene receives a *p* value, we disable common filters such as the minimum percentage of cells in which a gene is expressed and the minimum log-fold change a gene exhibits between the two clusters.

```
Idents(pbmc3k) <- "seurat_clusters"
pbmc3k_subset <- subset(pbmc3k, idents = c(4, 5))
target_data_markers <- FindMarkers(pbmc3k_subset, ident.1 = 4, min.pct = 0,
    logfc.threshold = 0, min.cells.feature = 1, min.cells.group = 1)
```

Thus far, we have followed the standard pipeline, resulting in *p* values for genes that may be affected by double-dipping. If we interpret these *p* values at face value, we risk identifying false-positive DE genes. We now apply ClusterDE to help remove these spurious discoveries.

We begin by generating synthetic null data. The null model assumes a single cell type, representing the null hypothesis that only one true cell type is present and that any observation of more than one cluster is the result of over-clustering. The input to constructNull() is a gene-by-cell count matrix. For UMI-based scRNA-seq data, we use the negative binomial distribution (family = "nb", the default setting), although users may choose other distributions as appropriate for their data.

```
count_matrix <-LayerData(pbmc3k_subset, assay = "RNA", layer = "counts")
set.seed(1)
null_data <-constructNull(count_matrix, nCores = 3, fastVersion = T)
```

We now process the null data using the same pipeline applied to the original data. If any modifications were made to the original pipeline, the same adjustments should be applied here. However, it is important to ensure that the null data are ultimately partitioned into exactly two clusters. For clustering algorithms such as Louvain or Leiden, which rely on a resolution parameter rather than a fixed number of clusters, achieving this may require tuning the resolution parameter.

```
null_data <- CreateSeuratObject(null_data)
null_data <- NormalizeData(null_data)
null_data <- FindVariableFeatures(null_data)
null_data <- ScaleData(null_data)
null_data <- RunPCA(null_data)
null_data <- FindNeighbors(null_data)
res <- 0.3
null_data <- FindClusters(null_data, resolution = res)
num_clusters <- length(unique(null_data$seurat_clusters))
while(num_clusters!=2){
    res <- ifelse(num_clusters > 2, res-0.05, res+0.05)
    print(res)
    null_data <- FindClusters(null_data, resolution = res)

    num_clusters <- length(unique(null_data$seurat_clusters))
    print(num_clusters)
}
## res = 0.35
```

Using the null data, we perform a DE test for each gene. As with the real data, we disable common filters—such as the minimum percentage of cells in which a gene is expressed and the minimum log-fold change—to ensure that every gene receives a p value.

```
null_data_markers <- FindMarkers(null_data, ident.1 = 0, min.pct = 0,
    logfc.threshold = 0, min.cells.feature = 1, min.cells.group = 1)
```

Finally, we compare the two sets of p values—one from the target data and one from the null data—yielding two p values per gene. `ClusterDE` computes a contrast score for each gene, defined as the difference between the negative log p value from the target DE test and the negative log p value from the null DE test. For true DE genes, we expect the contrast score to be large and positive.

```
target_pval <- target_data_markers$p_val
names(target_pval) <- rownames(target_data_markers)

null_pval <- null_data_markers$p_val
names(null_pval) <- rownames(null_data_markers)

res <- callDE(target_pval, null_pval, FDR = 0.05)
## no DE genes
```

5. APPLICATION TO SPATIAL TRANSCRIPTOMICS DATA

For this section, we use spatial transcriptomics data from the Human Dorsolateral Prefrontal Cortex, obtained from the Lieber Institute for Brain Development (LIBD) project (Pardo et al., 2022). We focus on slice 151673 and compare the Layer 6 (L6) and white matter (WM) spatial domains. Similar to scRNA-seq data, `ClusterDE` is designed to test a pair of spatial clusters as candidate domains. In settings with multiple clusters, we recommend testing pairs of adjacent clusters to assess whether they form distinct boundaries characterized by sharp expression changes in at least some genes.

Users are required to have the `Seurat` and `SingleCellExperiment` R packages installed beforehand.

```
if (!requireNamespace("BiocManager", quietly = TRUE)) {
  install.packages("BiocManager")
}
BiocManager::install("spatialLIBD")
BiocManager::install("BayesSpace")
```

```
library(spatialLIBD)
library(SingleCellExperiment)
library(BayesSpace)
library(Seurat)
library(ClusterDE)
spe <- fetch_data("spe")
sub_slice <- spe[, spe$sample_id=="151673"]
index <- c(which(sub_slice$spatialLIBD=="L6"), which(sub_slice$spatialLIBD=='WM'))
sub_slice <- sub_slice[, index]

## For faster estimation, retain genes expressed in at least 20% of spatial spots
data <- sub_slice@assays@data$counts
zero_expre_rate <- apply(data, 1, function(x){
   zero_true <- x==0
   zero_num <- length(which(zero_true==TRUE))/dim(data)[2]
   return(zero_num)
})
sub_slice <- sub_slice[which(zero_expre_rate < 0.8),]
```

For the standard double-dipping approach, we follow the `BayesSpace` pipeline to preprocess and cluster spatial spots. Users may modify these steps to suit their preferred analysis workflow.

```
##BayesSpace for preprocessing and clustering
sce <- SingleCellExperiment(list(counts = sub_slice@assays@data$counts))
sce$array_row <- sub_slice@int_colData@listData$spatialCoords[, 1]
sce$array_col <- sub_slice@int_colData@listData$spatialCoords[, 2]
sce$cell_type <- sub_slice@colData$spatialLIBD

sce <- spatialPreprocess(sce, platform = "ST", n.PCs = 7, log.normalize = TRUE)
sce <- spatialCluster(sce, q = 2, platform = "ST", d = 7, init.method = "mclust",
   model = "t", gamma = 2, nrep = 1000, burn.in = 100, save.chain = TRUE)
```

We apply the Wilcoxon rank-sum test in `Seurat` to obtain a p value for each gene between two adjacent spatial clusters.

```
count_matrix <- sce@assays@data$counts
target_data <- CreateSeuratObject(count_matrix)
target_data <- NormalizeData(target_data)
target_data <- ScaleData(target_data, features = rownames(target_data))
target_data$cluster <- sce$spatial.cluster
Idents(target_data) <- "cluster"
target_data_markers <- FindMarkers(target_data, ident.1 = 1, min.pct = 0,
   logfc.threshold = 0, min.cells.feature = 1, min.cells.group = 1)
```

Up to this point, we have followed a standard `BayesSpace` + `Seurat` pipeline. We now apply `ClusterDE` to filter out spurious DE genes resulting from double-dipping. We begin by generating synthetic null data, which assumes a single spatial domain and mimics the gene expression patterns observed in the real data, but in a smooth, non-discrete fashion without sharp transitions.

```
location_matrix <- data.frame(X = sce$array_row, Y = sce$array_col)
null_data <- constructNull(count_matrix, family = "nb", extraInfo = location_matrix,
   formula = "s(X, Y, bs = 'gp', k = 4)", nCores = 3)
null_sce <- SingleCellExperiment(list(counts = null_data))
null_sce$array_row <- sce$array_row
null_sce$array_col <- sce$array_col
```

We then process the null data using the same pipeline as the real data, ensuring that clustering yields exactly two clusters to enable a fair comparison with the two clusters identified in the real data.

```
## BayesSpace on the null data
null_sce <- spatialPreprocess(null_sce, platform = "ST", n.PCs = 7,
    log.normalize = TRUE)
null_sce <- spatialCluster(null_sce, q = 2, platform = "ST", d = 7,
    init.method = "mclust", model = "t", gamma = 2, nrep = 1000, burn.in = 100,
    save.chain = TRUE)

## Seurat on the null data
count_matrix <- null_sce@assays@data$counts
null_data <- CreateSeuratObject(count_matrix)
null_data <- NormalizeData(null_data)
null_data <- ScaleData(null_data, features = rownames(null_data))
null_data$cluster <- null_sce$spatial.cluster
Idents(null_data) <- "cluster"
null_data_markers <- FindMarkers(null_data, ident.1 = 1, min.pct = 0,
    logfc.threshold = 0, min.cells.feature = 1, min.cells.group = 1)
```

Finally, we compare the two sets of p values—one from the target data and one from the null data—yielding two p values per gene. `ClusterDE` computes a contrast score for each gene, defined as the difference between the negative log p value from the target DE test and the negative log p value from the null DE test. For true DE genes, we expect the contrast score to be large and positive.

```
target_pval <-target_data_markers$p_val
names(target_pval) <-rownames(target_data_markers)

null_pval <-null_data_markers$p_val
names(null_pval) <-rownames(null_data_markers)

res <-callDE(target_pval, null_pval, FDR = 0.05, nlogTrans = T)
```

6. CONCLUSION

We have demonstrated the application of the `ClusterDE` R package to scRNA-seq and ST data using popular analysis pipelines. However, we emphasize that the utility of `ClusterDE` is not limited to these specific pipelines or data types. The core contribution of `ClusterDE` lies in its simulation-based framework, which generates a single hypothetical cell type or spatial domain to serve as a contrast for identifying spurious DE genes. This approach is broadly applicable to other workflows and data modalities. We hope users find this tutorial helpful for conducting more reliable and interpretable analyses.

SOFTWARE AVAILABILITY

The `ClusterDE` R package is under the MIT License and available at https://github.com/SONGDONGYUAN1994/ClusterDE.

ACKNOWLEDGMENTS

The authors thank the members of the Junction of Statistics and Biology lab (https://jsb-lab.org/) for their helpful feedback and comments. The full article has been posted as a preprint on bioRxiv (Song et al., 2024), and a peer-reviewed version of this work also appeared in the proceedings of the Research in Computational Molecular Biology (RECOMB) conference (Song et al., 2025).

AUTHORS' CONTRIBUTIONS

All the authors have read and approved the final article. Author contributions are described using the CRediT taxonomy: C.L. contributed to writing—original draft and formal analysis; S.C. contributed to formal analysis; D.S. contributed to conceptualization, software, and formal analysis; and J.J.L. contributed to conceptualization, supervision, and writing—review and editing.

AUTHOR DISCLOSURE STATEMENT

The authors declare that they have no competing interests.

FUNDING INFORMATION

This work was supported by the National Science Foundation (DBI-1846216 and DMS-2113754), the National Institutes of Health/NIGMS (R35GM140888), and the Chan Zuckerberg Initiative Single-Cell Biology Data Insights Grant (Silicon Valley Community Foundation Award 2022-249355) (to J.J.L.).

REFERENCES

Ge X, Chen YE, Song D, et al. 'Clipper: P-value-free fdr control on high-throughput data from two conditions'. Genome Biol 2021;22(1):288.

Pardo B, Spangler A, Weber LM, et al. 'spatiallibd: An r/bioconductor package to visualize spatially-resolved transcriptomics data'. BMC Genomics 2022;23(1):434; doi: 10.1186/s12864-022-08601-w

Satija Lab. (2020). pbmc3k.SeuratData: 3k PBMCs from 10X Genomics. R package version 3.1.4. Available from: https://satijalab.org/seurat

Song D, Chen S, Lee C, et al. 'Synthetic control removes spurious discoveries from double dipping in single-cell and spatial transcriptomics data analyses'. bioRxiv 2024:2023.07.21.550107. Available from: https://www.biorxiv.org/content/early/2024/12/30/2023.07.21.550107

Song D, Chen S, Lee C, et al. (2025). Synthetic control removes spurious discoveries from double dipping in single-cell and spatial transcriptomics data analyses. In: Research in Computational Molecular Biology. (Sankararaman S, ed.) Springer Nature: Switzerland, Cham, pp. 400–404.

Address correspondence to:
Mr. Jingyi Jessica Li
Department of Statistics and Data Science
University of California
Los Angeles, CA 90095-1554
USA

E-mail: lijy03@fredhutch.org