DeMixNB: Deconvolution for Sparse Count Data

Last updated: 2025-07-08

1 Introduction

Cancer complexity arises from intratumor heterogeneity and tumor cell plasticity, which create substantial variations in transcriptional profiles and affect tumor transcript proportions in mixed samples. While tumor cells exhibit greater transcriptional activity than normal cells, current deconvolution methods struggle with low-abundance RNA species (e.g., microRNAs) and sparse data from technologies like spatially resolved transcriptomics (SRT).

DeMixNB is a novel method integrated within the DeMixT2.0 framework, building upon the original DeMixT approach to address these limitations. DeMixNB uses a two-tier approach: first estimating non-tumor component parameters from reference samples, then applying an Iterated Conditional Modes (ICM) algorithm to jointly estimate tumor-specific expression parameters and transcript proportions. The method requires no external single-cell reference data or predefined marker genes, enabling accurate tumor transcript deconvolution in sparse count data.

2 Feature Description

Building on these foundational concepts, DeMixNB provides a robust transcriptomic deconvolution framework with several key enhancements. It has demonstrated high accuracy and efficiency in comparisons with other methods on both benchmarking and real-world datasets.

The features of DeMixNB include:

Incorporate Negative Binomial Distribution: Sparse count data such as microRNA and SRT are often overdispersed, and the Negative Binomial (NB) distribution naturally accommodates this, providing more accurate modeling than Poisson or normal distributions.

Joint Estimation: DeMixNB jointly estimates component proportions and expression profiles for each individual sample. By requiring reference samples for the non-tumor component, it bypasses the need for pre-selected marker genes.

Efficient Estimation: DeMixNB adopts an Iterated Conditional Modes (ICM) approach to guarantee rapid convergence to a local maximum, ensuring computational efficiency.

Parallel Computing: OpenMP enables parallel computing on a single computer by taking advantage of multiple cores. The ICM framework is well-suited for parallelization, which helps reduce the computational time required for its iterative optimization steps.

Multi-seed for Result Selection: A multi-seed system runs the deconvolution multiple times from different starting points, allowing users to evaluate result stability and select the optimal outcome based on convergence criteria.

3 Functions

The following table shows the extra functions included in DeMixNB.

Function	Description
`DeMixNB`	Performs deconvolution of tumor samples with two components using a multi-seed approach for robustness.
`DeMixNB_single_seed`	Estimates the proportions and expression profiles of mixed samples in a single run.

4 Methods

4.1 Model

We model the transcript counts for tumor and non-tumor cells. Let Y_{ig} denote the observed mixed expression count at sample i for gene g. The deconvolution model assumes:

$$Y_{ig} = T_{ig} + N_{ig},$$

where T_{ig} and N_{ig} are the unobserved transcript counts from the tumor and non-tumor components, respectively. We assume these counts follow Negative Binomial distributions:

$$T_{ig} N_{ig} \sim \mathrm{NB}(\pi_i \mu_{tg}, \phi_{tg}), \sim \mathrm{NB}((1-\pi_i)\mu_{ng}, \phi_{ng}),$$

where π_i is the tumor transcript proportion for sample i, μ_{tg} and μ_{ng} are the mean expression levels for the tumor and non-tumor components, and ϕ_{tg} and ϕ_{ng} are the corresponding dispersion parameters. The dispersion parameters are assumed to be constant across samples for each gene. The expected expression of the mixture is:

$$E(Y_{ig}) = \pi_i \mu_{tg} + (1-\pi_i)\mu_{ng}.$$

The model requires a set of non-tumor reference samples (where $\pi_i = 0$) to first estimate the non-tumor parameters (μ_{ng} and ϕ_{ng}), enabling a semi-reference-based deconvolution.

4.2 The DeMixNB Algorithm for Deconvolution

The DeMixNB algorithm estimates the parameters π_i, μ_{tg}, ϕ_{tg}, μ_{ng}, and ϕ_{ng} by maximizing the Negative Binomial likelihood. The inputs are a matrix Y of mixed expression data and a matrix Y_{ref} of non-tumor reference data. The outputs are the estimated tumor transcript proportions and component-specific expression parameters.

The algorithm proceeds as follows:

1. **Estimate Non-Tumor Parameters**: For each gene `g` from 1 to `G`:
   - Given the reference data `Y_ref`, estimate the non-tumor parameters `μ_ng` and `ϕ_ng` using Maximum Likelihood Estimation (MLE).
   - If MLE fails to converge, use the Method of Moments (MOM) as a fallback.
2. **Initialize Parameters**:
   - For each mixed sample `i` from 1 to `S`, initialize the tumor transcript proportion `π_i` with a random value, e.g., from `Unif(0.3, 0.7)`.
   - For each gene `g`, obtain initial estimates for the tumor parameters `μ_tg` and `ϕ_tg` using MOM on the mixed samples `Y`, treating them as pure tumor for initialization.
3. **Iterative Optimization (ICM)**: Repeat until convergence or max iterations (`max_iter`) is reached:

- **Update Tumor Parameters**: For each gene `g`, update the tumor parameters `μ_tg` and `ϕ_tg` by maximizing the likelihood function conditional on the current estimates of proportions (`π`) and the fixed non-tumor parameters.
- **Update Proportions**: For each sample `i`, update the proportion `π_i` by maximizing the likelihood function conditional on the newly updated tumor parameters and the fixed non-tumor parameters.
- Increment iteration counter `k`.

The likelihood for a single observation $Y_{i,g}$ is the convolution of the two Negative Binomial distributions:

$$L(Y_{i,g}) = \sum_{z=0}^{Y_{i,g}} f_{T,ig}(z) \cdot f_{N,ig}(Y_{i,g} - z),$$

where $f_{T,ig}(z)$ and $f_{N,ig}(z)$ are the probability mass functions for the tumor and non-tumor components, respectively:

$$f_{T,ig}(z) f_{N,ig}(z) = \frac{\Gamma(z+\phi_{tg})}{\Gamma(\phi_{tg})z!}(1-p_{T,ig})^z p_{\phi_{tg}T,ig}, = \frac{\Gamma(z+\phi_{ng})}{\Gamma(\phi_{ng})z!}(1-p_{N,ig})^z p_{\phi_{ng}N,ig},$$

with

$$p_{T,ig} p_{N,ig} = \phi_{tg}\pi_i\mu_{tg} + \phi_{tg}, = \phi_{ng}(1-\pi_i)\mu_{ng} + \phi_{ng}.$$

5 Real data Example

5.1 Spatially Resolved Transcriptomics

Here we demonstrate a workflow for `DeMixNB` using spatial transcriptomic data.

```
# ------------------------------------
# Step 0: Install and load packages
# ------------------------------------
# Install packages
install.packages('Seurat' ,'hdf5r', 'Rfast2')

if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("glmGamPoi")

# Load library
library(Seurat)
library(DeMixT)
library(dplyr)
library(ggplot2)

# ------------------------------------
# Step 1: Load and Preprocess Data
# ------------------------------------
# Load the LUSC spatial transcriptomics dataset (Seurat object)

data_dir <-
"/rsrch6/home/bcb/wwang7/Team/Spatial_TmS/DeMixNB_tutorial_data/lusc_public_data"

spatial_img <- Read10X_Image(image.dir = file.path(data_dir, "spatial/"),
                             image.name = "tissue_lowres_image.png")
lusc_seurat <- Load10X_Spatial(data.dir = file.path(data_dir,
                                                    "filtered_feature_bc_matrix/"),
                               filename = "filtered_feature_bc_matrix.h5",
                               assay = "Spatial",slice = "Lung_P14",
                               filter.matrix = TRUE,
                               to.upper = FALSE,
                               image = spatial_img)

# Display initial dataset dimensions
message("Before quality control, the dataset contains ",
        ncol(lusc_seurat), " spots and ",
        nrow(lusc_seurat), " genes.")
```

Before quality control, the dataset contains 3858 spots and 18085 genes.

```
# ------------------------------------
# Step 2: Filter Genes
# ------------------------------------
# Remove unwanted genes (e.g., mitochondrial, pseudogenes, etc.)
# Regular expressions identify genes to exclude based on their names
unwanted_genes <- grepl("^(MT-|MIR-|LINC|A[A-Z]|B[A-Z]|
                        C[A-Z]|Rik|Gm|Vmn|Olfr|Trav)",
                        rownames(lusc_seurat))
lusc_seurat <- lusc_seurat[!unwanted_genes, ]

# Filter genes with low expression (detected in fewer than 10 spots)
count_data <- GetAssayData(lusc_seurat, assay = "Spatial",layer = "counts")
gene_detection_counts <- rowSums(count_data > 0)
genes_to_keep <- names(gene_detection_counts[gene_detection_counts >= 10])

# ------------------------------------
# Step 3: Load and Process Cell Metadata
# ------------------------------------
# Load cell metadata from CSV file
cell_metadata <- read.csv(file.path(data_dir,

"CytAssist_FFPE_Lung_Squamous_Cell_Carcinoma_pyramidLZW.tiff.csv"))

# Filter valid barcodes and classify cells as tumor or non-tumor
cell_metadata <- cell_metadata %>%
  filter(!is.na(Barcode) & Barcode != "" & Barcode %in% colnames(count_data)) %>%
  mutate(class = ifelse(class2 == "t", "tumor", "non_tumor"))

# Summarize spot-level cell information (e.g., coordinates, cell type proportions)
barcode_summary <- cell_metadata %>%
  group_by(Barcode) %>%
  summarise(
    x_coor = mean(x_fullres),
    y_coor = mean(y_fullres),
    tumor_proportion = sum(class == "tumor") / n(),
    stromal_proportion = sum(class2 == "f") / n(),
    immune_proportion = sum(class2 == "l") / n(),
    non_tumor_proportion = 1 - tumor_proportion,
    number_of_cells = n()
  )

# ------------------------------------
# Step 4: Quality Control (QC) on Spots
# ------------------------------------
# Subset Seurat object to keep spots with sufficient counts and features
lusc_seurat <- subset(
  x = lusc_seurat,
  subset = (nCount_Spatial >= 600) & (nFeature_Spatial >= 300),
  features = genes_to_keep
)

# Update barcode summary to match filtered Seurat object
barcode_summary <- cell_metadata %>%
  filter(Barcode %in% colnames(lusc_seurat)) %>%
  group_by(Barcode) %>%
  summarise(
    x_coor = mean(x_fullres),
    y_coor = mean(y_fullres),
    tumor_proportion = sum(class == "tumor") / n(),
    stromal_proportion = sum(class2 == "f") / n(),
    immune_proportion = sum(class2 == "l") / n(),
    non_tumor_proportion = 1 - tumor_proportion,
    tumor_cell = sum(class == "tumor"),
    number_of_cells = n()
  )

# Filter spots with at least 5 cells and tumor proportion between 0.05 and 0.95
barcode_summary_filtered <- barcode_summary %>%
  filter(number_of_cells >= 5 & (tumor_proportion == 0 |
```

```
                                    (tumor_proportion >= 0.05
                                     & tumor_proportion <= 0.95)))

# Display post-QC dataset dimensions
count_matrix <- GetAssayData(lusc_seurat, assay = "Spatial", layer = "counts")
message("After quality control, the dataset contains ",
        nrow(barcode_summary_filtered), " spots and ",
        nrow(count_matrix), " genes.")
```

After quality control, the dataset contains 3088 spots and 14546 genes.

```
# ------------------------------------
# Step 5: Classify Spots
# ------------------------------------
# Identify mixed (tumor + non-tumor) and reference (non-tumor) spots
barcode_mixed <- barcode_summary_filtered %>%
  filter(tumor_proportion != 0) %>%
  pull(Barcode)
barcode_reference <- barcode_summary_filtered %>%
  filter(tumor_proportion == 0) %>%
  pull(Barcode)

# Create data frames for mixed and reference spots
cell_proportions <- barcode_summary_filtered %>%
  select(Barcode, tumor_proportion, stromal_proportion,
         immune_proportion, non_tumor_proportion, number_of_cells)

mixed_spots <- cell_proportions %>%
  filter(Barcode %in% barcode_mixed) %>%
  mutate(Type = "Candidate mixed spot")
reference_spots <- cell_proportions %>%
  filter(Barcode %in% barcode_reference) %>%
  mutate(Type = "Candidate reference spot")

# Combine spot data
combined_spots <- rbind(mixed_spots, reference_spots)

# ------------------------------------
# Step 6: Subset Seurat Object
# ------------------------------------
# Create Seurat objects for reference and mixed spots
reference_seurat <- lusc_seurat[, colnames(lusc_seurat) %in% barcode_reference]
mixed_seurat <- lusc_seurat[, colnames(lusc_seurat) %in% barcode_mixed]
combined_seurat <- lusc_seurat[, colnames(lusc_seurat) %in% combined_spots$Barcode]

# Add sample type metadata
combined_seurat$sample_type <- ifelse(
  colnames(combined_seurat) %in% barcode_reference, "Normal",
  ifelse(colnames(combined_seurat) %in% barcode_mixed, "Tumor Mixed", NA)
)

# ------------------------------------
# Step 7: Visualize Spot Classifications
# ------------------------------------
# Plot spatial distribution of normal and tumor-mixed spots
spatial_plot <- SpatialDimPlot(
  combined_seurat,
  group.by = "sample_type",
  cols = c("Normal" = "blue", "Tumor Mixed" = "red"),
  pt.size.factor = 1.5,
  interactive = FALSE,
  crop = FALSE
)

# ------------------------------------
# Step 8: Prepare Data for DeMixNB
# ------------------------------------
# Extract count matrices for tumor and normal spots
genes_shared <- intersect(rownames(reference_seurat), rownames(mixed_seurat))
tumor_counts <- GetAssayData(mixed_seurat, layer = "counts")[genes_shared, ]
normal_counts <- GetAssayData(reference_seurat, layer = "counts")[genes_shared, ]
```

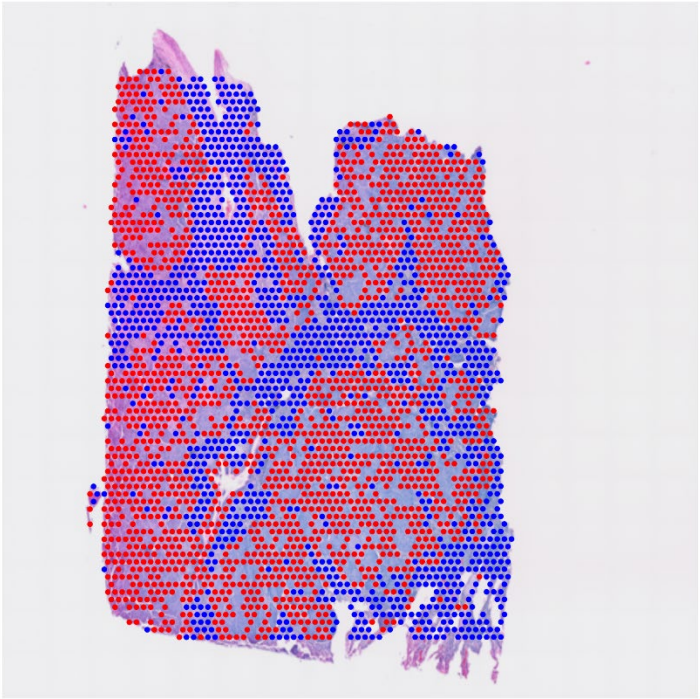
```r
# ------------------------------------
# Step 9: SCTransform and Dimensionality Reduction
# ------------------------------------
# Normalize and transform data using SCTransform
combined_seurat <- SCTransform(combined_seurat, assay = "Spatial", verbose = FALSE)

# Run PCA, clustering, and UMAP
combined_seurat <- RunPCA(combined_seurat, assay = "SCT", verbose = FALSE)
combined_seurat <- FindNeighbors(combined_seurat, reduction = "pca", dims = 1:50)
combined_seurat <- FindClusters(combined_seurat, verbose = FALSE, resolution = 0.8)
combined_seurat <- RunUMAP(combined_seurat, reduction = "pca", dims = 1:50)

# ------------------------------------
# Step 10: Identify Spatially Variable Genes
# ------------------------------------
# Select top 1000 spatially variable genes using Moran's I
# This step will take a while to process
# User can choose number of genes as needed
combined_seurat <- FindSpatiallyVariableFeatures(
  combined_seurat,
  assay = "SCT",
  features = VariableFeatures(combined_seurat)[1:1000],
  selection.method = "moransi"
)

# Extract Moran's I results and sort by observed Moran's I
moransi_results <- combined_seurat@assays$SCT@meta.features %>%
  na.exclude() %>%
  arrange(desc(MoransI_observed))
top_genes <- rownames(moransi_results)[1:1000]


# ------------------------------------
# Step 11: Run DeMixNB
# ------------------------------------
# Define gene list sizes and prepare gene lists
# For illustration purpose, this tutorial only uses gene size of 1000.

gene_list_sizes <- c(1000)

# Initialize deconvolution results matrix
deconvolution_results <- matrix(nrow = length(barcode_mixed), ncol = 0)

for (i in 1:length(gene_list_sizes)) {
  gl <- top_genes[1:gene_list_sizes[i]]
  filtered_mat <- count_matrix[gl, ]  # Subset count matrix for selected genes
  data.N1 <- as.matrix(filtered_mat[, barcode_reference])  # Normal spot counts
  data.Y <- as.matrix(filtered_mat[, barcode_mixed])
  matrix_data <- as.matrix(filtered_mat[, barcode_mixed])
  data.Y <- SummarizedExperiment(assays = list(counts = matrix_data))
  matrix_data <- as.matrix(filtered_mat[, barcode_reference])
  data.N1 <- SummarizedExperiment(assays = list(counts = matrix_data))
  test_simulation <- DeMixNB(data.Y = data.Y, data.N1 = data.N1, niter = 30, tol = 1e-
3)

  ngene.selected <- gene_list_sizes[i]
  name <- paste0("DemixNB_PUSC_hard_threshold_0.05_0.95_v2_", ngene.selected,
".RData")
  save(test_simulation, file = paste0('./Deconvolution_result/', name))

  # Get pi results and add gene count prefix to column names
  pi_results <- test_simulation$pi_t_summary
  colnames(pi_results) <- paste0(ngene.selected, "_", colnames(pi_results))
  deconvolution_results <- cbind(deconvolution_results, pi_results)
}


# ------------------------------------
# Step 12: Visualize DeMixNB Results
# ------------------------------------
# Subset Seurat object for mixed spots and add tumor proportion estimates
```

```
consistent_rows_1000 <- deconvolution_results$`1000_consistency` == "consistent"

# Select the 500 gene results specifically
tumor_proportions <- data.frame(
  sample.id = barcode_mixed[consistent_rows_1000],
  PiT = deconvolution_results$`1000_pi_run1`[consistent_rows_1000]
)

mixed_seurat <- subset(lusc_seurat, cells = tumor_proportions$sample.id)
matched_indices <- match(Cells(mixed_seurat), tumor_proportions$sample.id)
tumor_proportions <- tumor_proportions[matched_indices, ]
mixed_seurat$PiT <- tumor_proportions$PiT

# Plot tumor proportion estimates spatially
tumor_proportion_plot <- SpatialFeaturePlot(
  mixed_seurat,
  features = "PiT",
  pt.size.factor = 1.5,
  crop = FALSE
) + theme(legend.position = "top")

# Save or display plots
ggsave("./spatial_plot.png", spatial_plot)
ggsave("./tumor_proportion_plot.png", tumor_proportion_plot)
```

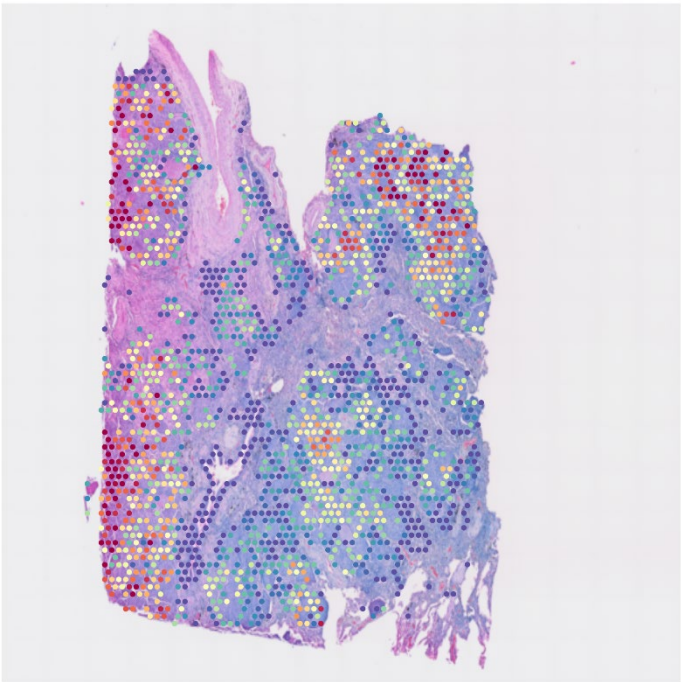
5.1.1 Results

When top 1000 genes selected, the resulted plots are shown below:

Spatial_plot.png



Tumor_proportion_plot.png



6 Session Information

This vignette was compiled using the following R and package versions:

```
sessionInfo()
## R version 4.5.1 (2025-06-13)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.7.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;  LAPACK version 3.12.1
##
## 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/Chicago
## tzcode source: internal
##
## attached base packages:
```

```
## [1] stats     graphics  grDevices utils     datasets  methods
base
##
## other attached packages:
## [1] BiocStyle_2.36.0
##
## loaded via a namespace (and not attached):
##  [1] digest_0.6.37       R6_2.6.1            bookdown_0.43
##  [4] fastmap_1.2.0       xfun_0.52           cachem_1.1.0
##  [7] knitr_1.50          htmltools_0.5.8.1   rmarkdown_2.29
## [10] lifecycle_1.0.4     cli_3.6.5           sass_0.4.10
## [13] jquerylib_0.1.4     compiler_4.5.1      rstudioapi_0.17.1
## [16] tools_4.5.1         evaluate_1.0.3      bslib_0.9.0
## [19] yaml_2.3.10         BiocManager_1.30.26 jsonlite_2.0.0
## [22] rlang_1.1.6
```