

Package: spatialDWLS (via r-universe)

October 16, 2026

Title Cell Type Deconvolution of Spatial Transcriptomics with Dampened Weighted Least Squares

Version 0.1.1

Description Estimates the cell type composition of each spot of a spatial transcriptomics experiment from a single-cell reference, following spatialDWLS (Dong and Yuan 2021, <[doi:10.1186/s13059-021-02362-7](https://doi.org/10.1186/s13059-021-02362-7)>), which builds on dampened weighted least squares (DWLS; Tsoucas et al. 2019, <[doi:10.1038/s41467-019-10802-z](https://doi.org/10.1038/s41467-019-10802-z)>). Spots are solved in two rounds: candidate cell types are chosen per cluster by enrichment, then each spot is re-solved against the types present in it. Works on dense or sparse matrices, in parallel through 'BiocParallel', and over windows of spots for expression stores that do not fit in memory.

License MIT + file LICENSE

Encoding UTF-8

URL <https://drieslab.github.io/spatialDWLS/>,
<https://github.com/drieslab/spatialDWLS>

BugReports <https://github.com/drieslab/spatialDWLS/issues>

Depends R (>= 4.1.0)

Imports Matrix, MatrixGenerics, methods, quadprog, stats

Suggests BiocParallel, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.1

Repository <https://drieslab.r-universe.dev>

Date/Publication 2026-10-16 14:36:46 UTC

RemoteUrl <https://github.com/drieslab/spatialDWLS>

RemoteRef HEAD

RemoteSha 45d201da6af14d23ffa718858a223c731563a2d5

Contents

dwls	2
dwls_signature	3
dwls_stream	4
Index	6

dwls	<i>Deconvolve spots into cell type proportions</i>
------	--

Description

Runs both rounds of spatialDWLS on an expression matrix. Each spot’s result is a composition: the proportions of a solved spot sum to one.

Usage

```
dwls(  
  expr,  
  sign_matrix,  
  clusters,  
  n_cell = 50,  
  cutoff = 2,  
  logbase = 2,  
  na_as_zero = FALSE,  
  BPPARAM = NULL,  
  chunk_size = 256L  
)
```

Arguments

expr	log-scaled, normalised expression: features (rows) by spots (columns), as a base matrix or a Matrix (sparse is kept sparse). Spot IDs are taken from the column names.
sign_matrix	mean expression per cell type: signature features (rows) by cell type (columns), on the reverse-logged scale, e.g. from <code>dwls_signature()</code> . Not a binary marker matrix.
clusters	cluster of every spot. Either named by spot ID (matched against <code>colnames(expr)</code> ; extra entries, such as the clusters of a whole dataset when <code>expr</code> is a subset, are ignored) or unnamed in the column order of <code>expr</code> . Spots are grouped by cluster to pick candidate cell types in round 1. Every spot needs a non-NA cluster.
n_cell	numeric. expected number of cells per spot. In round 2 a cell type is kept for a spot only when its round-1 proportion is at least $1 / n_cell$.
cutoff	numeric. PAGE enrichment z-score a cell type must reach in at least one spot of a cluster to be a candidate for that cluster. The two highest-scoring types are always candidates.

logbase	numeric. log base of expr.
na_as_zero	logical. spots that cannot be solved (no signal on the signature features) are NA by default; TRUE returns them as 0.
BPPARAM	a BiocParallel::BiocParallelParam for the per-cluster plans and per-spot solves, or NULL (default) to run serially.
chunk_size	integer. spots per parallel task.

Value

a numeric matrix, cell types (rows) by spots (columns).

See Also

[dwls_stream\(\)](#) for expression read in windows.

Examples

```
set.seed(1)
genes <- paste0("g", 1:30)
sig <- matrix(1, 30, 3, dimnames = list(genes, c("A", "B", "C")))
sig[1:10, "A"] <- sig[11:20, "B"] <- sig[21:30, "C"] <- 40

# 20 spots mixing A and B, 20 spots of C alone
mix <- cbind(rbind(a <- runif(20, 0.3, 0.7), 1 - a, 0),
             matrix(c(0, 0, 1), 3, 20))
counts <- matrix(rpois(30 * 40, sig %*% mix * 5), 30,
                 dimnames = list(genes, paste0("s", 1:40)))
clusters <- rep(c("mixed", "pure"), each = 20)

prop <- dwls(log2(counts + 1), sig, clusters, n_cell = 5)
round(prop[, c(1:3, 21:23)], 2)
```

dwls_signature	<i>Build a DWLS signature matrix</i>
----------------	--------------------------------------

Description

Mean expression per cell type over the signature features: the reference [dwls\(\)](#) expects.

Usage

```
dwls_signature(expr, sign_gene, cell_type)
```

Arguments

expr	single-cell expression, features (rows) by cells (columns). Pass the scale the spots will be solved on, i.e. normalised and <i>not</i> logged (dwls() reverse-logs the spot expression before solving).
sign_gene	character. signature features, e.g. cell type markers. Features missing from expr are dropped.
cell_type	cell type of each column of expr.

Value

a numeric matrix, signature features (rows) by cell type (columns, in order of first appearance in cell_type).

Examples

```
set.seed(1)
x <- matrix(rpois(200, 3), 20, 10,
            dimnames = list(paste0("g", 1:20), paste0("c", 1:10)))
dwls_signature(x, sign_gene = paste0("g", 1:5),
              cell_type = rep(c("A", "B"), each = 5))
```

dwls_stream

*Deconvolve spots read in windows***Description**

The spatialDWLS driver for expression backends that read spots in windows, such as an on-disk store. `dwls()` calls it with a single window.

The reader is called once per window in each of four passes: cluster and feature means, PAGE z-scores, the cluster round of solves and the spot round. Only the signature features are ever requested, and no more than one window is held at a time (apart from the cell types x spots result).

Usage

```
dwls_stream(
  read_window,
  n_windows,
  feat_ids,
  clusters,
  sign_matrix,
  n_cell = 50,
  cutoff = 2,
  logbase = 2,
  na_as_zero = FALSE,
  BPPARAM = NULL,
  chunk_size = 256L
)
```

Arguments

<code>read_window</code>	function(<i>i</i> , <i>features</i>) returning the log-scaled expression of window <i>i</i> (1.. <i>n_windows</i>) for exactly <i>features</i> , in that order, as a <i>features</i> x <i>spots</i> matrix or <i>Matrix</i> with spot IDs as column names. It must return the same spots for the same <i>i</i> on every pass, in any order.
<code>n_windows</code>	integer. number of windows.
<code>feat_ids</code>	character. every feature the backend holds; the signature is restricted to these.
<code>clusters</code>	cluster of every spot, named by spot ID. Extra entries are ignored.
<code>sign_matrix</code>	mean expression per cell type: signature features (rows) by cell type (columns), on the reverse-logged scale, e.g. from <code>dwls_signature()</code> . Not a binary marker matrix.
<code>n_cell</code>	numeric. expected number of cells per spot. In round 2 a cell type is kept for a spot only when its round-1 proportion is at least 1 / <i>n_cell</i> .
<code>cutoff</code>	numeric. PAGE enrichment z-score a cell type must reach in at least one spot of a cluster to be a candidate for that cluster. The two highest-scoring types are always candidates.
<code>logbase</code>	numeric. log base of <i>expr</i> .
<code>na_as_zero</code>	logical. spots that cannot be solved (no signal on the signature features) are NA by default; TRUE returns them as 0.
<code>BPPARAM</code>	a <code>BiocParallel::BiocParallelParam</code> for the per-cluster plans and per-spot solves, or NULL (default) to run serially.
<code>chunk_size</code>	integer. spots per parallel task.

Value

a numeric matrix, cell types (rows) by spots (columns), spots in window order.

Examples

```
set.seed(1)
genes <- paste0("g", 1:30)
sig <- matrix(1, 30, 3, dimnames = list(genes, c("A", "B", "C")))
sig[1:10, "A"] <- sig[11:20, "B"] <- sig[21:30, "C"] <- 40
mix <- cbind(rbind(a <- runif(20, 0.3, 0.7), 1 - a, 0),
             matrix(c(0, 0, 1), 3, 20))
expr <- log2(matrix(rpois(30 * 40, sig %*% mix * 5), 30,
                  dimnames = list(genes, paste0("s", 1:40))) + 1)
clusters <- setNames(rep(c("mixed", "pure"), each = 20), colnames(expr))

# read the spots ten at a time
windows <- split(colnames(expr), rep(1:4, each = 10))
prop <- dwls_stream(
  read_window = function(i, features) expr[features, windows[[i]]],
  n_windows = length(windows), feat_ids = rownames(expr),
  clusters = clusters, sign_matrix = sig, n_cell = 5
)
all.equal(prop, dwls(expr, sig, clusters, n_cell = 5))
```

Index

`dwls`, [2](#)
`dwls()`, [3](#), [4](#)
`dwls_signature`, [3](#)
`dwls_signature()`, [2](#), [5](#)
`dwls_stream`, [4](#)
`dwls_stream()`, [3](#)