Package: dicepro (via r-universe)

October 24, 2026

Title Semi-Supervised Deconvolution of Bulk RNA-Seq Data with Hyperparameter Optimization

Version 1.0.2

Description Performs semi-supervised deconvolution of bulk RNA sequencing (RNA-seq) data. Known cell-type proportions are estimated using supervised methods -- 'CIBERSORTx' (CSx), 'CIBERSORT' (CS), 'FARDEEP' (Fast And Robust DEconvolution of Expression Profiles), and 'DCQ' (Digital Cell Quantifier) -- while unknown components are inferred using non-negative matrix factorization ('NMF') with limited-memory Broyden-Fletcher-Goldfarb-Shanno with bounds ('L-BFGS-B') optimization. Hyperparameters are selected automatically using a Pareto-frontier-based approach with knee-point detection, allowing application when the reference signature matrix is incomplete. More details about 'DICEpro' can be found in Ba et al. (2026) <doi:10.64898/2026.06.17.732876>.

Depends R (>= 4.1.0)

Encoding UTF-8

License MIT + file LICENSE

LazyData true

LazyDataCompression xz

Config/testthat/edition 3

VignetteBuilder knitr

LinkingTo Rcpp (>= 1.0.12), RcppEigen

Imports Rcpp (>= 1.0.12), lbfgsb3c, KraljicMatrix, ggplot2, patchwork, FARDEEP, ComICS, lme4, e1071, preprocessCore, parallel

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

URL https://github.com/kalidouBA/dicepro

BugReports https://github.com/kalidouBA/dicepro/issues

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

NeedsCompilation yes

Author Kalidou BA [aut, cre], Boris P. Hejblum [aut]

Maintainer Kalidou BA <sistm.soft.maintain@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-06-30 12:20:14 UTC

RemoteUrl https://github.com/cran/dicepro

RemoteRef HEAD

RemoteSha 98e5ba8f5f2b183b41c6e8ac541ab2773c0f5733

Contents

dicepro-package . . . 3
best_hyperParams . . . 3
BlueCode . . . 4
CellMixtures . . . 6
contains_nan_or_inf . . . 8
create_gamma_lambda_plot . . . 9
dicepro . . . 10
full_metrics . . . 13
generate_ref_matrix . . . 14
generateProp . . . 15
heatmap_abundances . . . 17
makeTable1Tool . . . 18
metric_plot . . . 18
nmf_lbfgsb . . . 20
nmf_lbfgsb_hyperOpt . . . 21
nrmse . . . 22
objective_opt . . . 23
plot.dicepro . . . 23
plot_hyperopt . . . 24
research_hyperOpt . . . 26
row_norm_pos . . . 27
run_CSx . . . 27
run_experiment . . . 28
running_method . . . 29
samplewise_metrics . . . 31
simulation . . . 32
simulation_bluecode . . . 34

Index **36**

`dicepro-package` *dicepro: Semi-Supervised Deconvolution of Bulk RNA-Seq Data with Hyperparameter Optimization*

Description

Performs semi-supervised deconvolution of bulk RNA sequencing (RNA-seq) data. Known cell-type proportions are estimated using supervised methods – 'CIBERSORTx' (CSx), 'CIBERSORT' (CS), 'FARDEEP' (Fast And Robust DEconvolution of Expression Profiles), and 'DCQ' (Digital Cell Quantifier) – while unknown components are inferred using non-negative matrix factorization ('NMF') with limited-memory Broyden-Fletcher-Goldfarb-Shanno with bounds ('L-BFGS-B') optimization. Hyperparameters are selected automatically using a Pareto-frontier-based approach with knee-point detection, allowing application when the reference signature matrix is incomplete. More details about 'DICEpro' can be found in Ba et al. (2026) doi:10.64898/2026.06.17.732876.

Author(s)

Maintainer: Kalidou BA <sistm.soft.maintain@gmail.com>

Authors:

- Boris P. Hejblum <boris.hejblum@u-bordeaux.fr>

See Also

Useful links:

- https://github.com/kalidouBA/dicepro
- Report bugs at https://github.com/kalidouBA/dicepro/issues

`best_hyperParams` *Select optimal hyper-parameters using a Pareto frontier*

Description

Identifies the best (λ, γ) pair from a set of optimization trials by:

1. Filtering trials where $|1 - \text{constraint}|$ exceeds `constraint_threshold`.
2. Computing `abs_constraint = |1 - constraint|`.
3. Extracting the Pareto frontier (minimize `frobNorm` AND `abs_constraint`) via `KraljicMatrix::get_frontier` with `quadrant = "bottom.left"`.
4. Selecting the knee point via maximum perpendicular distance to the utopia-nadir line in normalized objective space.

Usage

```
best_hyperParams(trials_df, W, H, constraint_threshold = 0.1)
```

Arguments

`trials_df`	data.frame of optimization trials as returned by `research_hyperOpt` (`$trials`). Must contain at least: `lambda_`, `gamma`, `frobNorm`, `constraint`.
`W`	List of W matrices, one per trial.
`H`	List of H matrices, one per trial.
`constraint_threshold`	Numeric scalar. Maximum allowed $\|1 - \text{constraint}\|$; trials above this threshold are discarded (default `0.1`).

Value

Named list, or `invisible(NULL)` when no valid configuration survives filtering:

hyperparameters List with `lambda` and `gamma`.

metrics List with `frobNorm`, `abs_constraint`, and `constraint` for the selected trial.

trials Filtered trials data.frame (constraint-passing rows only).

W The W matrix of the selected trial.

H The H matrix of the selected trial.

plot A `ggplot2` Pareto frontier figure.

`BlueCode` *BlueCode Reference Signature Matrix*

Description

Gene expression reference matrix for bulk RNA-seq deconvolution, derived from sorted cell populations spanning Immune, Stromal, Endothelial, epithelial, and muscle compartments. BlueCode serves as the default reference for dicepro and directly informs the hierarchical Dirichlet simulation framework.

Usage

```
data(BlueCode)
```

Format

A numeric matrix of dimensions $G \times 34$, where G denotes the number of genes after quality filtering. Rows are labeled by HGNC gene symbol; columns are labeled by cell-type name. All values are non-negative read counts prior to normalization; apply `.normalize_zscore_per_gene` or TPM scaling before use in downstream analyses.

Details

Construction:

BlueCode was built from publicly available sorted bulk RNA-seq profiles downloaded from the ENCODE and GEO repositories. For each cell population, replicate profiles were averaged after quantile normalization. Genes with zero variance across all 34 profiles, or with missing values in more than 10 \ log-transformed so that it can be used directly with both Gaussian and Poisson deconvolution models.

Compartment structure:

The 34 cell types are partitioned into five major tissue compartments. The column ordering in the matrix follows this partition exactly (columns 1–9, 10–17, 18–20, 21–25, 26–34).

Immune compartment – columns 1–9:

1. `B.cell.naive`
2. `B.cell.memory`
3. `T.Cell.CD4`
4. `T.cell.CD4.memory`
5. `T.cell.CD8.memory`
6. `NK.cell.CD56`
7. `Monocyte.CD14`
8. `Macrophage`
9. `Mature.neutrophil`

Stromal compartment – columns 10–17:

1. `Fibroblast.cardiac.ventricle`
2. `Fibroblast.of.arm`
3. `Fibroblast.of.lung`
4. `Fibroblast.of.the.aortic.adventitia`
5. `Fibroblast.or.papilla.dermal.cell`
6. `MSC.like.pluripotent.cell`
7. `Chondrocyte.articular`
8. `Osteoblast`

Endothelial compartment – columns 18–20:

1. `Endothelial.large.blood.vessel`
2. `Endothelial.microvascular.mammary.or.endometrial`
3. `Endothelial.microvascular.non.reproductive`

Epithelial compartment – columns 21–25:

1. `Epithelial.cell.mammary`
2. `Epithelial.renal.cortical`
3. `Epithelial.respiratory`
4. `Hair.follicular.keratinocyte`
5. `Melanocyte.of.skin`

Muscle compartment – columns 26–34:

1. `Smooth.muscle.cell.aortic`

2. Smooth.muscle.cell.bronchial
3. Smooth.muscle.cell.coronary.artery
4. Smooth.muscle.cell.pulmonary.artery
5. Smooth.muscle.cell.of.bladder
6. Smooth.muscle.cell.of.trachea
7. Smooth.muscle.cell.uterine
8. Myocyte.regular.cardiac
9. Myometrial.cell

Link to the simulation framework:

This compartment structure directly informs the hierarchical Dirichlet model in `generateProp` with `scenario = "hierarchical"`. The Dirichlet concentration parameters at each level ($\alpha_{\text{Immune}} = 6$, $\alpha_{\text{Stromal}} = 2$, $\alpha_{\text{Endothelial}} = \alpha_{\text{Epithelial}} = 1.5$, $\alpha_{\text{Muscle}} = 1$) reflect the expected relative abundance of each compartment in mixed tissue samples, so that simulated proportions are biologically realistic with respect to BlueCode cell-type coverage.

Recommended usage:

```
data(BlueCode)

# Quick inspection
dim(BlueCode)          # G x 34
colnames(BlueCode)     # 34 cell-type labels

# Use directly in dicepro
out <- dicepro(
  reference    = BlueCode,
  bulk         = my_bulk_matrix,
  methodDeconv = "FARDEEP"
)
```

Source

BlueCode reference matrix constructed for the dicepro benchmarking framework from ENCODE (https://www.encodeproject.org) and GEO (https://www.ncbi.nlm.nih.gov/geo/) sorted RNA-seq profiles. See the accompanying manuscript for full details on data sources, replicate averaging, quality filtering, and normalization.

`CellMixtures` *CellMixtures Bulk RNA-seq Dataset*

Description

Experimentally constructed bulk RNA-seq mixtures of sorted cell populations, designed for benchmarking deconvolution algorithms. Each sample is a known mixture of cell types drawn from the BlueCode reference panel, making ground-truth proportions available for quantitative evaluation.

Usage

```
data(CellMixtures)
```

Format

A numeric matrix of dimensions $G \times 12$, where G denotes the number of genes (~31 400 after quality filtering). Rows are labeled by HGNC gene symbol; columns are labeled `A` through `L`, corresponding to 12 experimentally mixed samples. Values are raw read counts prior to normalization.

Details

Experimental design:

The 12 mixture samples (A–L) were prepared by combining sorted cell populations at known proportions across the five tissue compartments represented in `BlueCode`. Each sample targets a distinct mixture composition, providing a controlled benchmark spanning a wide range of cell-type abundance profiles – from immune-dominated samples to Stromal- or muscle-enriched configurations.

Gene coverage:

CellMixtures covers approximately 31 400 gene symbols, a super-set of those present in BlueCode. dicepro automatically intersects the two gene sets before deconvolution; the effective number of informative genes is therefore determined by the BlueCode reference (see `BlueCode` for details on its gene filtering criteria). A gene-overlap check prior to running `dicepro` is recommended:

```
data(BlueCode)
data(CellMixtures)
n_common <- length(intersect(rownames(BlueCode), rownames(CellMixtures)))
cat(sprintf("Common genes: %d / %d reference genes\n",
            n_common, nrow(BlueCode)))
```

normalization:

Raw counts should be normalized before deconvolution. `dicepro` applies `.normalize_zscore_per_gene` internally; no preprocessing is required when using the main function. For exploratory analyses, log-CPM or TPM normalization is recommended:

```
# log2-CPM normalization
cpm <- sweep(CellMixtures, 2, colSums(CellMixtures) / 1e6, FUN = "/")
log2_cpm <- log2(cpm + 1)
```

Recommended usage:

```
data(BlueCode)
data(CellMixtures)

out <- dicepro(
  reference             = BlueCode,
  bulk                  = CellMixtures,
  methodDeconv          = "FARDEEP",
  bulkName              = "CellMixtures",
```

```
  refName               = "BlueCode",
  hp_max_evals          = 100,
  hspaceTechniqueChoose = "all"
)

# Inspect estimated proportions
head(out$H)
```

Relationship to BlueCode:

CellMixtures and BlueCode are designed as a paired benchmark: the reference matrix is derived from the same cell populations used to construct the mixtures, ensuring that all cell types present in the bulk are represented in the reference. This controlled setting enables direct evaluation of the reconstruction accuracy of dicepro.

Source

CellMixtures was constructed for the DICEPro benchmarking framework. Sorted cell populations were obtained from ENCODE (https://www.encodeproject.org) and mixed *in silico* at predefined proportions from experimentally validated RNA-seq profiles. See the accompanying manuscript for full details on mixture design, library preparation, and sequencing.

---

contains_nan_or_inf *Check if a value contains NaN or Inf*

Description

Check if a value contains NaN or Inf

Usage

```
contains_nan_or_inf(value)
```

Arguments

value	Numeric, vector, matrix, or data frame to check.

Value

Logical. TRUE if NaN or Inf is present, FALSE otherwise.

`create_gamma_lambda_plot` *Visualize the gamma–lambda hyper-parameter search space*

Description

Generates a ggplot2 scatter plot of γ vs λ sampled from the search space defined by `hspaceTechniqueChoose`, with feasibility bounds overlaid for `"restrictionEspace"`.

Usage

```
create_gamma_lambda_plot(
  hspaceTechniqueChoose = c("all", "restrictionEspace"),
  n_samples = 200,
  seed = NULL
)
```

Arguments

`hspaceTechniqueChoose`
: Character scalar.

  `"all"` Full independent log-uniform grid for `lambda_`, `gamma`, `p_prime`. No constraint lines.

  `"restrictionEspace"` Restricted space with `lambda_ = gamma * lambda_factor`, `lambda_factor` $\in [2, 100]$. Lower ($\lambda = 2\gamma$) and upper ($\lambda = 100\gamma$) bounds are drawn.

`n_samples`
: Positive integer. Configurations to draw (default `200`).

`seed`
: Integer. Random seed used for full pipeline reproducibility. Defaults to `NULL`.

Value

A `ggplot2` figure object.

Examples

```
create_gamma_lambda_plot(hspaceTechniqueChoose = "all")
create_gamma_lambda_plot(hspaceTechniqueChoose = "restrictionEspace")
```

`dicepro` *Semi-supervised bulk RNA-seq deconvolution with hyper-parameter optimization*

Description

Combines supervised estimation of known cell types with unsupervised discovery of latent components, with automatic Pareto-frontier-based hyper-parameter optimization.

Usage

```
dicepro(
  reference,
  bulk,
  methodDeconv = "CS",
  cibersortx_email = NULL,
  cibersortx_token = NULL,
  cibersort_perm = 0,
  cibersort_QN = TRUE,
  W_prime = 0,
  bulkName = "Bulk",
  refName = "Reference",
  hp_max_evals = 100,
  N_unknownCT = 1L,
  algo_select = "random",
  output_path = NULL,
  hspaceTechniqueChoose = "gamma_dominant",
  gamma_ratio_min = 10,
  constraint_threshold = 0.1,
  out_Decon = NULL,
  normalize = TRUE,
  seed = NULL
)
```

Arguments

`reference`	Numeric matrix (genes * cell types).
`bulk`	Numeric matrix (genes * samples).
`methodDeconv`	Character. One of `"CS"`, `"CSx"`, `"DCQ"`, `"FARDEEP"`.
`cibersortx_email`	Character. CIBERSORTx email (required for `"CSx"`).
`cibersortx_token`	Character. CIBERSORTx token (required for `"CSx"`).
`cibersort_perm`	Non-negative integer. Number of permutations for p-value estimation in the built-in `"CIBERSORT"` method. Set to `0` (default) to skip p-value computation. Ignored for all other methods.

`cibersort_QN` Logical. Apply quantile normalisation to the bulk mixture in the built-in `"CIBERSORT"` method (default `TRUE`). Ignored for all other methods.

`W_prime` Initial unknown-signature matrix or `0`.

`bulkName` Character scalar. Label for the bulk data-set.

`refName` Character scalar. Label for the reference.

`hp_max_evals` Positive integer. Number of hyper-parameter trials.

`N_unknownCT` Positive integer. Number of unknown cell types.

`algo_select` Character. One of `"random"`, `"tpe"`, `"atpe"`, `"anneal"`.

`output_path` Character scalar. Root output directory. Defaults to tempdir(); pass an explicit directory to persist results across sessions.

`hspaceTechniqueChoose`

Character. One of `"gamma_dominant"` or `"all"`, or `"restrictionEspace"`. **The most efficient and strongly recommended strategy is** `"gamma_dominant"`**, as it provides faster convergence and more precise results.**

`"gamma_dominant"` **Recommended.** Same unconstrained grid as `"all"`, but candidates where `gamma <= gamma_ratio_min * lambda_` are rejected, ensuring `gamma >> lambda_`. This reduces the search space, leading to faster optimization and more stable solutions.

```
\item{\code{"all"}}{Full independent log-uniform grid for
  \code{lambda_}, \code{gamma}, \code{p_prime}.}

\item{\code{"restrictionEspace"}}{Restricted space where
  \code{lambda_ = gamma * lambda_factor},
  \code{lambda_factor} \eqn{\in [2, 100]}.}
```

`gamma_ratio_min`

Positive numeric. Minimum ratio γ/λ enforced when `hspaceTechniqueChoose = "gamma_dominant"` (default `10`). Ignored for other strategies.

`constraint_threshold`

Positive numeric. Maximum allowed deviation from the constraint (|1 - constraint|) when filtering hyper-parameter trials before Pareto selection (default `0.1`).

`0.05` Very strict (risk of no valid solution).

`0.1` Good trade-off between feasibility and robustness.

`0.2 - 0.3` More permissive, useful for broader exploration.

`out_Decon` Optional pre-computed deconvolution matrix (samples * cell types). When provided, the supervised deconvolution step is skipped entirely.

`normalize` Logical. Apply z-score normalisation per gene (default `TRUE`).

`seed` Integer. Random seed used for full pipeline reproducibility. Defaults to `NULL`. If set, ensures deterministic behaviour of the hyper-parameter optimisation and downstream stochastic components. Set to `NULL` if you explicitly want random (non-reproducible) runs.

Details

When `out_Decon` is provided, the supervised step is skipped. Gene matrices are optionally z-score normalized per gene, and only intersecting genes are retained before optimization.

The built-in `"CIBERSORT"` method closely follows Newman et al. (2015): nu-SVR with three candidate nu values (0.25, 0.50, 0.75), optional quantile normalisation, and optional permutation-based p-values. It requires packages **e1071**, **parallel**, and **preprocessCore**. No external account or Docker installation is needed, unlike `"CSx"`.

Value

An object of class `"dicepro"` (a named list) containing:

- `hyper-parameters` – selected λ and γ
- `metrics` – loss and constraint of the best trial
- `trials` – all evaluated configurations
- `W` – estimated unknown signature matrix
- `H` – estimated proportion matrix
- `plot` – Pareto frontier ggplot2 figure
- `plot_hyperopt` – hyper-parameter space ggplot2 figure

Returns `invisible(NULL)` with a warning when no valid configuration is found.

References

Newman AM et al. (2015). Robust enumeration of cell subsets from tissue expression profiles. *Nature Methods*, 12(5), 453-457. doi:10.1038/nmeth.3337

See Also

`running_method`, `run_experiment`, `best_hyperParams`

Examples

```
sim_data <- simulation(
  loi        = "gauss",
  scenario   = "hierarchical",
  nSample    = 5,
  nGenes     = 50,
  nCellsType = 5,
  sigma_bio  = 0.07,
  sigma_tech = 0.07,
  seed       = 2101
)

out <- dicepro(
  reference             = as.matrix(sim_data$W)[, -c(1)],
  bulk                  = as.matrix(sim_data$B),
  methodDeconv          = "FARDEEP",
  W_prime               = 0,
```

```
  bulkName              = "SimBulk",
  refName               = "SimRef",
  hp_max_evals          = 5,
  algo_select           = "random",
  output_path           = tempdir(),
  hspaceTechniqueChoose = "all",
  normalize             = FALSE
)
```

`full_metrics` *Full agreement metrics via mixed-effects modeling*

Description

Fits a two-way mixed-effects model (populations and subjects as random effects) and derives ICC(3,1), Concordance correlation coefficient, and relative RMSE, complemented by sample-wise correlation and RMSE from `samplewise_metrics`.

Usage

```
full_metrics(x, y)
```

Arguments

`x`	Numeric matrix of observed values (rows = subjects, columns = variables / populations).
`y`	Numeric matrix of predicted values; same dimensions as `x`.

Value

Named list:

Correlation_mean Mean sample-wise Pearson correlation.

RMSE_mean Mean sample-wise RMSE.

ICC3_adapted ICC(3,1) from the mixed model (`NA` for constant data).

CCC_adapted Concordance correlation coefficient (`NA` for constant data).

rRMSE_adapted Relative RMSE $\sqrt{\sigma_\varepsilon^2 / V_T}$ (`NA` for constant data).

`generate_ref_matrix` *Generate a Reference Signature Matrix*

Description

Constructs a synthetic gene-by-cell-type reference matrix using either Poisson-based (via Gaussian copula) or log-normal gene expression models, with optional block sparsity and TPM normalization.

Usage

```
generate_ref_matrix(
  loi = "rpois",
  tpm = FALSE,
  bloc = FALSE,
  nGenesByCellType = 50,
  nCell = 500,
  nCellsType = 10,
  nGenes = 500,
  lambda_vec = NULL,
  corr = NULL,
  sparse = FALSE,
  prob_sparse = NULL
)
```

Arguments

`loi`	Character. Expression model: `"rpois"` for Poisson-based simulation via Gaussian copula or any other value for log-normal simulation. Default: `"rpois"`.
`tpm`	Logical. If `TRUE`, columns are TPM-normalized. Default: `FALSE`.
`bloc`	Logical. If `TRUE`, introduces block sparsity so that each cell type expresses only a subset of genes. Default: `FALSE`.
`nGenesByCellType`	Integer. Number of genes per cell-type block (used when `bloc = TRUE`).
`nCell`	Integer. Total number of cells (unused directly; retained for API compatibility).
`nCellsType`	Integer. Number of cell types. Default: `10`.
`nGenes`	Integer. Number of genes. Default: `500`.
`lambda_vec`	Numeric vector of length `nCellsType`. Per cell-type Poisson lambda parameters. If `NULL`, drawn uniformly in `[20, 60]`. Only used when `loi = "rpois"`.
`corr`	Numeric matrix. Inter-cell-type correlation matrix. If `NULL`, generated via `.rcorrmatrix`.
`sparse`	Logical. If `TRUE`, introduces random sparsity by zeroing entries with probability `1 - prob_sparse`. Default: `FALSE`.
`prob_sparse`	Numeric. Probability of a non-zero entry when `sparse = TRUE`. Default: `NULL`.

Details

For `loi = "rpois"`, correlated Poisson counts are generated via a Gaussian copula: multivariate normal draws with covariance `corr` are mapped to uniform marginals via `pnorm`, then to Poisson quantiles via `qpois`. This preserves the inter-cell-type correlation structure without requiring **SimMultiCorrData**.

For any other value of `loi`, a log-normal model is used: multivariate normal draws are exponentiated after a location shift of 6.

Value

A numeric matrix of dimensions `nGenes` x `nCellsType`. Row names are `Gene_1`, ..., `Gene_nGenes`; column names are `CellType_1`, ..., `CellType_nCellsType`.

Examples

```
set.seed(2101)
ref_pois  <- generate_ref_matrix(loi = "rpois", nGenes = 50, nCellsType = 5)
ref_gauss <- generate_ref_matrix(loi = "gauss", nGenes = 50, nCellsType = 5)
dim(ref_pois)   # 50 x 5
dim(ref_gauss)  # 50 x 5
```

`generateProp` *Generate Cell-Type Proportion Matrix*

Description

Simulates cell-type proportion matrices for bulk RNA-seq deconvolution benchmarking. Four scenarios are supported: equal proportions across cell types, uniform random sampling, hierarchical Dirichlet sampling reflecting realistic tissue compartment organization, or a BlueCode-specific hierarchical model that preserves real cell-type names.

Usage

```
generateProp(
  n_cell_types = NULL,
  nSample,
  nCell = 500,
  scenario = NULL,
 alpha_groups = c(Immune = 4, Stromal = 2.5, Endothelial = 1.8, Epithelial = 1.8, Muscle
    = 1.5),
 alpha_subtypes = list(Immune = 8, Stromal = 8, Endothelial = 8, Epithelial = 8, Muscle
    = 8),
  seed = NULL
)
```

Arguments

n_cell_types	Integer. Number of distinct cell types. Ignored when scenario = "bluecode" (fixed at 34 by the BlueCode reference).
nSample	Integer. Number of samples to generate.
nCell	Integer. Total number of cells per sample (used for rounding precision in the uniform scenario).
scenario	Character. Proportion generation strategy. One of: "even" for near-equal proportions, "uniform" for random uniform sampling, "hierarchical" for two-level Dirichlet sampling with generic CellType_k names, or "bluecode" for two-level Dirichlet sampling using the real cell-type names and compartment structure of the BlueCode reference. Any other value defaults to a flat Dirichlet draw.
alpha_groups	Named numeric vector of length 5. Dirichlet concentration parameters for the five tissue compartments (Immune, Stromal, Endothelial, Epithelial, Muscle). Only used when scenario = "bluecode". Default: c(Immune=4, Stromal=2.5, Endothelial=1.8, Epithelial=1.8, Muscle=1.5).
alpha_subtypes	Named list with one scalar per compartment. Controls the concentration of sub-type Dirichlet draws within each compartment. Higher values produce more even sub-type distributions. Only used when scenario = "bluecode". Default: all compartments set to 8.
seed	Integer. Random seed. Only used when scenario = "bluecode". Default: NULL.

Value

A numeric matrix of dimensions nSample x n_cell_types where each row sums to 1. For all scenarios except "bluecode", row names are Sample_1, ..., Sample_nSample and column names are CellType_1, ..., CellType_n_cell_types. For "bluecode", row names are sample_1, ..., sample_nSample and column names are the real BlueCode cell-type names ordered by compartment.

Examples

```
set.seed(2101)

# Equal proportions
prop_even <- generateProp(nSample = 20, n_cell_types = 10, scenario = "even")

# Generic hierarchical Dirichlet
prop_hier <- generateProp(nSample = 20, n_cell_types = 34,
  scenario = "hierarchical")
all(abs(rowSums(prop_hier) - 1) < 1e-8)  # TRUE

# BlueCode hierarchical Dirichlet (real cell-type names)
prop_bc <- generateProp(nSample = 20, scenario = "bluecode")
colnames(prop_bc)  # real BlueCode cell-type names
```

heatmap_abundances *Heatmap of cell-type abundances*

Description

Generates a raster heatmap showing the estimated abundance of each cell type across NMF iterations (or samples). Color encodes abundance using the viridis scale (reversed: dark = high abundance).

Usage

```
heatmap_abundances(res2plot, base_size = 9, title = NULL, midpoint = NULL)
```

Arguments

res2plot	A data.frame with: • one column named Iterate (iteration or sample index, numeric or integer); • one column per cell type containing non-negative numeric abundance values.
base_size	Numeric. Base font size in points (default 9).
title	Character. Plot title (default NULL, no title).
midpoint	Numeric or NULL. When not NULL, a diverging viridis scale is used with this value as the midpoint, useful for highlighting deviations from a reference abundance. Default NULL (sequential scale).

Value

A ggplot object. It can be printed or saved using ggplot2::ggsave().

Examples

```
df <- data.frame(
  Iterate   = 1:10,
  CellTypeA = runif(10),
  CellTypeB = runif(10)
)
heatmap_abundances(df)
heatmap_abundances(df, title = "Estimated abundances", midpoint = 0.5)
```

makeTable1Tool *Build a performance metrics table for composition matrices*

Description

Aligns predicted and observed matrices to their common populations, row-normalizes both, then computes global and sample-wise agreement metrics.

Usage

```
makeTable1Tool(pred_mat, obs_mat)
```

Arguments

pred_mat	Numeric matrix of predicted compositions (rows = samples, columns = populations).
obs_mat	Numeric matrix of observed compositions; must share at least one column name with pred_mat.

Details

Populations absent from pred_mat but present in obs_mat (and vice versa) are silently dropped after the intersection step. Both matrices are row-normalized via row_norm_pos before any metric is computed.

Value

A list with one element:

Perf A one-row data.frame containing: Correlation, RMSE, Correlation_mean, RMSE_mean, NRMSE, ICC3, CCC. All fields are NA when computation is not possible.

See Also

full_metrics, row_norm_pos

metric_plot *Performance metric line plot across iterations*

Description

Plots the evolution of one or more scalar performance metrics (e.g., NRMSE, R^2, loss) over NMF iterations. Useful for monitoring convergence and diagnosing early stopping.

Usage

```
metric_plot(
  perf2plot,
  ylab = "Error between folds",
  title = NULL,
  base_size = 9,
  add_smooth = FALSE,
  show_points = FALSE
)
```

Arguments

perf2plot	A data.frame with at least two columns: • Iterate – iteration index (numeric or integer). • metric – scalar performance value at each iteration. Optionally a group column (character or factor) to draw multiple colored lines, one per group.
ylab	Character. Y-axis label (default "Error between folds").
title	Character. Plot title (default NULL).
base_size	Numeric. Base font size in points (default 9).
add_smooth	Logical. When TRUE, overlays a LOESS smoothing line. Useful for noisy convergence curves (default FALSE).
show_points	Logical. When TRUE, adds individual data points on top of the line (default FALSE).

Value

A ggplot object.

Examples

```
df <- data.frame(
  Iterate = 1:50,
  metric  = cumsum(runif(50, -0.01, 0.1))
)
metric_plot(df)
metric_plot(df, ylab = "NRMSE", add_smooth = TRUE)

# Multi-group
df2 <- data.frame(
  Iterate = rep(1:20, 2),
  metric  = c(cumsum(runif(20, 0, 0.1)), cumsum(runif(20, 0, 0.05))),
  group   = rep(c("Fold 1", "Fold 2"), each = 20)
)
metric_plot(df2)
```

`nmf_lbfgsb` *NMF with L-BFGS-B optimization*

Description

Performs non-negative matrix factorization (NMF) using L-BFGS-B optimization. The objective combines a Gaussian log-likelihood with an augmented Lagrangian penalty that enforces the sum-to-one constraint on cell-type proportions.

Usage

```
nmf_lbfgsb(
  r_dataset,
  W_prime = 0,
  p_prime = 0,
  lambda_ = 10,
  gamma_par = 100,
  N_unknownCT = 1L,
  con = list(maxit = 3000)
)
```

Arguments

`r_dataset`	Named list with three elements: **B** Numeric matrix (N_gene × N_sample). Bulk expression to factorize. **P_cb** Numeric matrix (N_sample × N_cells_Type). Known cell-type proportion estimates from a supervised deconvolution step. **W_cb** Numeric matrix (N_gene × N_cells_Type). Reference signature matrix for known cell types.
`W_prime`	Numeric scalar or matrix. Initial value(s) for the unknown cell-type column(s) in W. Default `0`.
`p_prime`	Numeric scalar or matrix. Initial value(s) for the unknown cell-type column(s) in H. Default `0`.
`lambda_`	Numeric scalar. Augmented Lagrangian multiplier initialization (default `10`).
`gamma_par`	Numeric scalar. Penalty coefficient for the sum-to-one constraint (default `100`). Reduce if the optimizer returns infinite values.
`N_unknownCT`	Positive integer. Number of latent cell types to estimate (default `1`).
`con`	Named list of control parameters forwarded to `lbfgsb3c::lbfgsb3c`. Default: `list(maxit = 3000)`.

Value

A named list on success:

W Optimized signature matrix (N_gene × N_cells_Type).

frob_W Frobenius norm of `W`.

var_W Variance of the unknown column of `W`.

H Optimized proportion data.frame (N_sample × N_cells_Type + 1), with a leading `Mixture` column.

frob_H Frobenius norm of `H`.

var_H Variance of the unknown column of `H`.

loss Gaussian log-likelihood (objective terms 1 + 2).

frobNorm Objective term 1 (data-fit component).

constNorm Objective term 2 (log-normalization component).

c1 Penalty term 1 (linear Lagrangian).

c2 Penalty term 2 (quadratic penalty).

objectiveValue Total objective value (`loss`).

penalty Total penalty (`c1` + `c2`).

cvrge Convergence code from `lbfgsb3c`.

constraint Absolute constraint violation: $|1 - \sum h_i|$.

Returns `NULL` when the optimizer fails or the solution is degenerate.

`nmf_lbfgsb_hyperOpt` *NMF L-BFGS-B wrapper for hyper-parameter optimization*

Description

Thin wrapper around `nmf_lbfgsb` that normalizes the `p_prime` argument and cleans the returned matrices.

Usage

```
nmf_lbfgsb_hyperOpt(
  dataset,
  W_prime = NULL,
  p_prime = NULL,
  lambda_ = 10,
  gamma_par = 100
)
```

Arguments

`dataset`	List containing matrices `B`, `W`, and `P`.
`W_prime`	Optional numeric matrix. Initial W'.
`p_prime`	Optional numeric matrix or scalar. Initial P'. If `NULL`, defaults to a matrix of `0.1`.
`lambda_`	Numeric. Regularization parameter λ.
`gamma_par`	Numeric. Regularization parameter γ.

Value

List output from nmf_lbfgsb with H and W cleaned by .clean_nmf_matrix.

---

nrmse *Normalized Root Mean Square Error (NRMSE)*

Description

Computes RMSE between back-transformed pred and obs vectors, then normalizes by a summary statistic of the observations.

Usage

```
nrmse(
  pred,
  obs,
  method = "sd",
  transformation = "none",
  trans_function = "none"
)
```

Arguments

pred	Numeric vector of predicted values (on the transformed scale).
obs	Numeric vector of observed values (same scale as pred; same length).
method	Normalization denominator: "sd" (default), "mean", "maxmin", or "iq" (interquartile range).
transformation	Back-transformation applied before computing RMSE. One of "none" (default), "sqrt", "4thrt", "log", "log10", "log2", "log1p", "arcsine", or "other".
trans_function	R expression string used when transformation = "other" (the variable inside the expression must be named x).

Value

Non-negative numeric scalar, or NA_real_ (with a message) when NRMSE is undefined (fewer than 2 complete observations, or zero denominator).

Examples

```
set.seed(1)
obs  <- rnorm(100)
pred <- obs + rnorm(100, sd = 0.1)
nrmse(pred, obs)
nrmse(pred, obs, method = "maxmin")
```

objective_opt *Objective function for hyper-parameter optimization*

Description

Runs one NMF trial for a given (λ, γ, p') configuration and returns a structured result list, or NULL when the trial produces invalid values.

Usage

```
objective_opt(
  dataset,
  config = list(),
  lambda_ = NULL,
  gamma_factor = NULL,
  gamma = NULL,
  p_prime = NULL,
  W_prime = 0
)
```

Arguments

dataset	List with matrices B, W, and P.
config	List. Configuration object (needs $exp for the output directory).
lambda_	Numeric. Regularization parameter λ.
gamma_factor	Numeric or NULL. When not NULL, γ is derived as lambda_ * gamma_factor.
gamma	Numeric. γ used directly when gamma_factor is NULL.
p_prime	Numeric matrix (nrow = n_samples, ncol = n_unknown_ct).
W_prime	Numeric matrix or scalar. Initial W'.

Value

A named list with elements loss, constraint, status, current_params, W, H, cvrge; or NULL when the trial is invalid.

plot.dicepro *Plot cell abundance heatmap and error plot*

Description

Combines a cell-abundance heatmap with a fold-error plot using **patchwork**. Requires at least two unique iterations in x$Matrix_prediction.

Usage

```
## S3 method for class 'dicepro'
plot(x, ...)
```

Arguments

x	A dicepro object as returned by best_hyperParams.
...	Additional arguments (currently unused; reserved for future use).

Value

A **patchwork** figure, or invisible(NULL) with a warning when only one iteration is present.

See Also

heatmap_abundances, metric_plot

---

plot_hyperopt *Plot hyperparameter optimization report*

Description

Generic function for plotting the hyperparameter search report stored in a dicepro object. Dispatches to plot_hyperopt.dicepro.

Builds a scatter-matrix of all evaluated (λ, γ, p') configurations, color-coded by loss value, with violin/bar marginals for the top 5\

Usage

```
plot_hyperopt(x, ...)

## S3 method for class 'dicepro'
plot_hyperopt(
  x,
  params,
  metric = "loss",
  loss_metric = "loss",
  loss_behaviour = "min",
  not_log = NULL,
  categorical = NULL,
  max_deviation = NULL,
  title = NULL,
  ...
)
```

Arguments

x	A dicepro object. Trials are read from x$trials.
...	Currently unused. Reserved for future extensions.
params	Character vector of hyperparameter column names to display (e.g. c("lambda_", "gamma", "p_prime")).
metric	Character scalar. Column used for point size (default "loss").
loss_metric	Character scalar. Column used as the loss axis (default "loss").
loss_behaviour	Character scalar. Direction of the loss: "min" (default) or "max".
not_log	Character vector of parameter names that should not be log-scaled on their axis (default NULL).
categorical	Character vector of categorical parameter names; these are displayed as bar charts in the marginal row (default NULL).
max_deviation	Numeric scalar. Trials with $\|loss - mean(loss)\|$ above this threshold are excluded as outliers (default NULL, no exclusion).
title	Character scalar. Optional title displayed above the combined figure (default NULL).

Value

Whatever the dispatched method returns (a patchwork figure for dicepro objects).

A patchwork object which can be printed, saved with ggplot2::ggsave(), or embedded in R Markdown documents.

See Also

plot_hyperopt.dicepro

Examples

```
sim_data <- simulation(
  loi        = "gauss",
  scenario   = "hierarchical",
  nSample    = 5,
  nGenes     = 150,
  nCellsType = 10,
  sigma_bio  = 0.07,
  sigma_tech = 0.07,
  seed       = 2101
)

out <- dicepro(
  reference            = as.matrix(sim_data$W)[, -c(1, 5, 10)],
  bulk                 = as.matrix(sim_data$B),
  methodDeconv         = "FARDEEP",
  W_prime              = 0,
  bulkName             = "SimBulk",
  refName              = "SimRef",
```

```
  hp_max_evals          = 10,
  algo_select           = "random",
  output_path           = tempdir(),
  hspaceTechniqueChoose = "all",
  normalize             = FALSE
)
plot_hyperopt(out, params = c("lambda_", "gamma", "p_prime"))
```

`research_hyperOpt` *Hyper-parameter optimization loop for dicepro*

Description

Runs `hp_max_evals` NMF trials by sampling from `hp_space`, using either random or TPE sampling, and collects results.

Usage

```
research_hyperOpt(
  objective_opt,
  dataset,
  config,
  hp_space = NULL,
  W_prime = NULL,
  seed = NULL
)
```

Arguments

`objective_opt`	Function passed verbatim to `objective_wrapper()`.
`dataset`	List with `$B`, `$W`, `$P`.
`config`	List. Configuration object; must contain `exp`, `hp_max_evals`, `hp_method`. Optionally contains `gamma_ratio_min` (positive numeric) to activate rejection sampling (mode `"gamma_dominant"`).
`hp_space`	prebuilt named list of parsed hyper-parameter specs. If `NULL`, built from `config$hp_space`.
`W_prime`	Optional initial W matrix.
`seed`	Integer. Random seed used for full pipeline reproducibility.

Value

A named list:

trials data.frame of all successful trial results.

W List of W matrices, one per successful trial.

H List of H matrices, one per successful trial.

row_norm_pos *Row-normalize a matrix with non-negative clamping*

Description

Clamps negative values to 0, then divides each row by its sum so that rows sum to 1. Rows whose sum equals 0 become all-NaN (degenerate samples are flagged rather than silently zeroed).

Usage

```
row_norm_pos(mat)
```

Arguments

mat	Numeric matrix.

Value

Numeric matrix of the same dimensions as mat, with each row summing to 1 (or all-NaN when the row sum is 0).

Examples

```
m <- matrix(c(-1, 2, 3, 0, 0, 0, 1, 1, 1), nrow = 3, byrow = TRUE)
row_norm_pos(m)
```

run_CSx *Run CIBERSORTx Deconvolution Method*

Description

This function runs the CIBERSORTx deconvolution method using Docker with the provided email and token. The method is used to estimate cell type proportions in bulk RNA-seq data.

Usage

```
run_CSx(
  bulk,
  reference,
  cibersortx_email,
  cibersortx_token,
  work_dir = tempdir()
)
```

Arguments

`bulk`	A matrix of bulk RNA-seq data.
`reference`	A matrix of reference cell type gene expression profiles.
`cibersortx_email`	The CIBERSORTx account email.
`cibersortx_token`	The CIBERSORTx account token.
`work_dir`	Character. Path to the working directory where CIBERSORTx output files are written. Defaults to `tempdir()`.

Details

The function performs the following steps:

1. Runs CIBERSORTx with the provided email and token.
2. Reads the output results from CIBERSORTx.
3. Extracts cell type proportions and returns them as a matrix.

Value

A matrix containing the estimated cell type proportions.

`run_experiment` *Run a dicepro hyperparameter optimization experiment*

Description

Builds the hyperparameter search space from `hspaceTechniqueChoose` and runs `research_hyperOpt()`, returning the collected trials.

Usage

```
run_experiment(
  dataset,
  W_prime = 0,
  bulkName,
  refName,
  hp_max_evals,
  algo_select,
  output_base_dir = tempdir(),
  hspaceTechniqueChoose,
  gamma_ratio_min = 10,
  seed
)
```

Arguments

dataset	List containing at least $B, $W, and $P matrices.
W_prime	Numeric matrix (or 0). Passed verbatim to research_hyperOpt() as the initial W for NMF.
bulkName	Character scalar. Identifier for the bulk dataset (used in output path construction).
refName	Character scalar. Identifier for the reference dataset (used in output path construction).
hp_max_evals	Positive integer. Number of hyperparameter trials to run.
algo_select	Character scalar. Sampling algorithm passed to the configuration (e.g. "random").
output_base_dir	Character scalar. Root directory for all outputs (default ".").
hspaceTechniqueChoose	Character scalar. Search-space strategy: "all" Full independent log-uniform grid for lambda_, gamma, p_prime. "restrictionEspace" Restricted space where gamma is the base variable and lambda_ is derived as lambda_ = gamma * lambda_factor, with lambda_factor in (2, 100). "gamma_dominant" Same unconstrained space as "all", but any candidate such that gamma <= gamma_ratio_min * lambda_ is rejected. This enforces gamma >> lambda_ without constraining bounds.
gamma_ratio_min	Positive numeric. Minimum ratio gamma / lambda_ enforced when hspaceTechniqueChoose = "gamma_dominant" (default 10). Ignored for the other strategies.
seed	Integer. Random seed used for full pipeline reproducibility.

Value

The list returned by research_hyperOpt: trials, W, and H.

---

running_method *Run cell-type deconvolution*

Description

Performs cell-type deconvolution on bulk RNA-seq data using one of four supported methods. All methods receive gene-intersected, numeric matrices and return a proportion matrix normalised to sum to 1 per sample (columns).

Usage

```
running_method(
  bulk,
  reference,
  methodDeconv = "CS",
  cibersortx_email = NULL,
  cibersortx_token = NULL,
  cibersort_perm = 0,
  cibersort_QN = TRUE
)
```

Arguments

`bulk`	Numeric matrix (genes x samples). Bulk RNA-seq expression data. Row names must be gene symbols.
`reference`	Numeric matrix (genes x cell types). Reference cell-type signature profiles. Row names must be gene symbols.
`methodDeconv`	Character scalar. Deconvolution method: `"CSx"` CIBERSORTx - requires Docker and credentials. `"CS"` Built-in CIBERSORT nu-SVR implementation. Requires **e1071**, **parallel**, and **preprocessCore**. No external account needed. `"DCQ"` Requires **ComICS**. `"FARDEEP"` Requires **FARDEEP**.
`cibersortx_email`	Character. CIBERSORTx account email (required when `methodDeconv = "CSx"`).
`cibersortx_token`	Character. CIBERSORTx account token (required when `methodDeconv = "CSx"`).
`cibersort_perm`	Non-negative integer. Number of permutations for CIBERSORT p-values (default `0` = no p-values). Only used when `methodDeconv = "CS"`.
`cibersort_QN`	Logical. Apply quantile normalisation in CIBERSORT (default `TRUE`). Only used when `methodDeconv = "CS"`.

Details

Gene intersection is performed automatically: only genes present in both `bulk` and `reference` are used. Both matrices are coerced to numeric before processing.

For `"CS"`, the built-in implementation closely follows the original Newman et al. (2015) algorithm: nu-SVR with three candidate nu values (0.25, 0.50, 0.75), optional quantile normalisation, and optional permutation-based p-values. Packages **e1071**, **parallel**, and **preprocessCore** must be installed.

Value

Numeric matrix (cell types x samples) of estimated cell-type proportions. Each column sums to 1.

References

Newman AM et al. (2015). Robust enumeration of cell subsets from tissue expression profiles. *Nature Methods*, 12(5), 453-457. doi:10.1038/nmeth.3337

See Also

run_CSx

`samplewise_metrics` *Sample-wise Pearson correlation and RMSE*

Description

For each row (sample) of two matrices, computes the Pearson correlation and RMSE between predicted and observed values, then returns their cross-sample means.

Usage

```
samplewise_metrics(obs_mat, pred_mat)
```

Arguments

`obs_mat`	Numeric matrix of observations (rows = samples, columns = variables).
`pred_mat`	Numeric matrix of predictions; must have the same dimensions as `obs_mat`.

Details

Rows where either `obs_mat` or `pred_mat` are constant (zero standard deviation) yield `NA` for correlation but still contribute to the RMSE mean.

Value

Named list with two elements:

Correlation_mean Mean Pearson correlation across samples (ignoring `NA` rows).

RMSE_mean Mean RMSE across samples.

See Also

full_metrics

simulation *Simulate Bulk RNA-seq Data with Biological and Technical Noise*

Description

Generates synthetic bulk RNA-seq expression matrices by linearly mixing reference cell-type profiles according to simulated proportions, with optional biological and technical noise. The noise model follows the Gaussian likelihood framework underlying the dicepro objective function: biological noise is multiplicative, whereas technical noise is additive. Consequently, normalization to a Gaussian scale prior to deconvolution is required.

Usage

```
simulation(
  W = NULL,
  prop = NULL,
  nSample = 50,
  nCell = 500,
  nCellsType = 50,
  nGenes = 500,
  lambda_vec = NULL,
  corr = NULL,
  scenario = NULL,
  loi = "rpois",
  tpm = FALSE,
  bloc = FALSE,
  nGenesByCellType = 50,
  sparse = FALSE,
  prob_sparse = 0.5,
  sigma_bio = 0.07,
  sigma_tech = 0.07,
  seed = NULL
)
```

Arguments

W	Numeric matrix (genes x cell types) or NULL. If NULL, a reference matrix is generated internally via generate_ref_matrix.
prop	Numeric matrix (samples x cell types) or NULL. If NULL, proportions are generated via generateProp.
nSample	Integer. Number of samples. Default: 50.
nCell	Integer. Total cells per sample. Default: 500.
nCellsType	Integer. Number of cell types. Default: 50.
nGenes	Integer. Number of genes. Default: 500.
lambda_vec	Numeric vector of length nCellsType. Per cell-type Poisson lambda parameters. Passed to generate_ref_matrix.

corr	Numeric matrix. Inter-cell-type correlation. If NULL, generated internally.
scenario	Character. Proportion scenario passed to generateProp. Use "hierarchical" for realistic tissue composition (recommended).
loi	Character. Expression law for reference generation. "rpois" for Poisson via Gaussian copula, any other value for log-normal. Default: "rpois".
tpm	Logical. TPM normalization of reference. Default: FALSE.
bloc	Logical. Block sparsity in reference. Default: FALSE.
nGenesByCellType	Integer. Genes per block (when bloc = TRUE). Default: 50.
sparse	Logical. Random sparsity in reference. Default: FALSE.
prob_sparse	Numeric. Sparsity probability. Default: 0.5.
sigma_bio	Numeric. Standard deviation of multiplicative biological noise. Default: 0.07.
sigma_tech	Numeric. Standard deviation of additive technical noise (proportional to expression level). Default: 0.07.
seed	Integer. Random seed. Default: NULL.

Details

The bulk expression matrix is generated as:

$$\mathbf{B} = \mathbf{W} \cdot \mathbf{P}^\top$$

Noise is then applied as:

$$B^*_{gn} = B_{gn}(1 + \varepsilon^{\text{bio}}_{gn}) + \varepsilon^{\text{tech}}_{gn} \cdot B_{gn}$$

where $\varepsilon^{\text{bio}} \sim \mathcal{N}(0, \sigma^2_{\text{bio}})$ and $\varepsilon^{\text{tech}} \sim \mathcal{N}(0, \sigma^2_{\text{tech}})$. Negative values are clipped to zero.

Value

A named list with three elements:

p data.frame. True cell-type proportions (samples x cell types).

W data.frame. Reference signature matrix (genes x cell types).

B data.frame. Noisy bulk expression matrix (genes x samples).

Examples

```
set.seed(2026)
sim <- simulation(
  scenario   = "hierarchical",
  nSample    = 20,
  nGenes     = 100,
  nCellsType = 10,
  sigma_bio  = 0.07,
  sigma_tech = 0.07,
  seed       = 2101
)
dim(sim$p)  # 20 x 10
dim(sim$B)  # 100 x 20
```

`simulation_bluecode` *Simulate Bulk RNA-seq Data Using the BlueCode Reference Matrix*

Description

Generates synthetic bulk RNA-seq expression matrices by mixing the bundled BlueCode reference signature matrix with hierarchically simulated cell-type proportions, then applying biological and technical noise.

Usage

```
simulation_bluecode(
  nSample = 50,
 alpha_groups = c(Immune = 4, Stromal = 2.5, Endothelial = 1.8, Epithelial = 1.8, Muscle
    = 1.5),
 alpha_subtypes = list(Immune = 8, Stromal = 8, Endothelial = 8, Epithelial = 8, Muscle
    = 8),
  sigma_bio = 0.15,
  sigma_tech = 0.02,
  seed = NULL
)
```

Arguments

`nSample`	Integer. Number of samples to simulate. Default: `50`.
`alpha_groups`	Named numeric vector of length 5. Dirichlet concentration parameters for the five tissue compartments (Immune, Stromal, Endothelial, Epithelial, Muscle). Larger values produce more even compartment proportions. Default: `c(Immune=4, Stromal=2.5, Endothelial=1.8, Epithelial=1.8, Muscle=1.5)`.
`alpha_subtypes`	Named list with one numeric scalar per compartment. Controls the concentration of sub-type Dirichlet draws within each compartment. Higher values produce more even sub-type distributions. Default: all compartments set to `8`.
`sigma_bio`	Numeric. Standard deviation of multiplicative biological noise (log-normal). Default: `0.15`.
`sigma_tech`	Numeric. Standard deviation of additive technical noise, expressed as a fraction of the global expression standard deviation. Default: `0.02`.
`seed`	Integer. Random seed for full reproducibility. Proportion sampling uses `seed`. Default: `NULL`.

Details

The proportion model follows a two-level Dirichlet hierarchy that mirrors the biological organization of human tissues:

1. **Compartment level** – five major tissue compartments (Immune, Stromal, Endothelial, Epithelial, Muscle) are drawn from Dirichlet(α_{groups}).

2. **Cell-type level** – within each compartment, sub-type proportions are drawn from a symmetric Dirichlet controlled by `alpha_subtypes` and scaled by the compartment proportion.

This two-level structure induces realistic positive within-compartment and negative between-compartment correlations between cell types.

The bulk matrix is computed as $B = W \cdot P^\top$, then noise is applied as:

$$B^*_{gn} = B_{gn} \cdot \exp(\varepsilon^{\text{bio}}_{gn}) + \varepsilon^{\text{tech}}_{gn}$$

where $\varepsilon^{\text{bio}} \sim \mathcal{N}(0, \sigma^2_{\text{bio}})$ and $\varepsilon^{\text{tech}} \sim \mathcal{N}(0,\ \sigma^2_{\text{tech}} \cdot \text{sd}(B))$. Negative values are clipped to zero.

The function validates at runtime that the cell-type names in `.bluecode_cell_groups` exactly match `colnames(BlueCode)`, ensuring consistency between the proportion model and the reference matrix.

Value

A named list with three elements:

`p` data.frame (samples x 34 cell types). True cell-type proportions with real BlueCode cell-type names. Each row sums to 1.

`W` data.frame (genes x 34 cell types). BlueCode reference signature matrix, column-ordered to match `p`.

`B` data.frame (genes x samples). Noisy simulated bulk expression matrix.

Examples

```
sim <- simulation_bluecode(
  nSample    = 30,
  sigma_bio  = 0.15,
  sigma_tech = 0.02
)
dim(sim$p)
dim(sim$W)
dim(sim$B)
all(abs(rowSums(sim$p) - 1) < 1e-8)
```

Index

∗ **data**
BlueCode, 4
CellMixtures, 6

best_hyperParams, 3, *12*
BlueCode, 4

CellMixtures, 6
contains_nan_or_inf, 8
create_gamma_lambda_plot, 9

dicepro, 10
dicepro-package, 3

full_metrics, 13, *18*, *31*

generate_ref_matrix, 14, *32*
generateProp, 15, *32*, *33*

heatmap_abundances, 17

makeTable1Tool, 18
metric_plot, 18

nmf_lbfgsb, 20, *21*, *22*
nmf_lbfgsb_hyperOpt, 21
nrmse, 22

objective_opt, 23

plot.dicepro, 23
plot_hyperopt, 24

research_hyperOpt, *4*, 26, *29*
row_norm_pos, *18*, 27
run_CSx, 27, *31*
run_experiment, *12*, 28
running_method, *12*, 29

samplewise_metrics, *13*, 31
simulation, 32
simulation_bluecode, 34