Package 'bioIOT'

September 24, 2026

Type Package

Title Inverse Optimal Transport for Single-Cell Trajectory Analysis

Version 0.2.2

Description Semi-relaxed inverse optimal transport (IOT) for single-cell state-transition and pseudotime analysis: a self-contained R solver (Anderson-accelerated soft Sinkhorn with exact implicit gradients), feature-weight fitting with a two-stage bias-corrected refit and multi-restart, state transition matrices, random-walk pseudotime, 'ggplot2' visualisation, soft-gated 'Seurat' and 'SingleCellExperiment' interfaces, reproducible simulated demo data, and bulk-cohort pathway scoring utilities.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0.0)

Imports utils, ggplot2

Suggests testthat (>= 3.0.0), knitr, rmarkdown, SingleCellExperiment, Seurat, SummarizedExperiment, S4Vectors

URL https://github.com/XTSgreen/bioIOT-R

BugReports https://github.com/XTSgreen/bioIOT-R/issues

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Han Dong [aut, cre] (Nickname: XTSgreen)

Maintainer Han Dong <dh411424@163.com>

Repository CRAN

Date/Publication 2026-09-24 13:40:24 UTC

Contents

`collapse_probes` *Collapse a probe-level expression matrix to gene level*

Description

For each gene symbol keep the probe with the highest cross-sample mean expression. This is the paper's probe-annotation folding rule.

Usage

```
collapse_probes(expr, probe_id = rownames(expr), gene_id)
```

Arguments

`expr`	Probe x sample numeric matrix.
`probe_id`	Character vector of probe ids, length `nrow(expr)` (defaults to `rownames(expr)`).
`gene_id`	Character vector of gene symbols matching `probe_id`; NA / empty entries are dropped.

Value

Gene x sample numeric matrix (rownames = gene symbols).

Examples

```
expr <- matrix(rnorm(40), nrow = 4)
colnames(expr) <- paste0("S", 1:10)
collapse_probes(expr, c("P1", "P2", "P3", "P4"),
                c("VIM", "VIM", "CDH2", "MYC"))
```

demo_iot_states *Bundled bioIOT demo dataset*

Description

Pre-computed synthetic demo data for quick experimentation with `fit_iot`, `transition_matrix` and `runIOT`. Regenerate with `bioIOT::simulate_iot_states(seed = 1)`.

Format

A list as returned by `simulate_iot_states(seed = 1)` with default arguments: `u`, `phi`, `a`, `b`, `P_true`, `T_true`, `theta_true`, `embedding`, `K`, `F_emb`, `cell_embedding`, `cell_state`, `cell_time`.

Examples

```
data(demo_iot_states)
names(demo_iot_states)
demo_iot_states$K
```

fit_iot *Fit inverse optimal transport feature weights*

Description

Fits feature weights θ so that the soft-marginal OT plan induced by $C = -einsum(\phi, \theta)$ reproduces the observed row-conditional transitions T. Two-stage fitting (l1 selection then Buehlmann-style debias refit) with multi-restart; gradients are exact implicit differentiations of the fixed point (implicit function theorem), which stays numerically stable where unrolled backpropagation diverges.

Usage

```
fit_iot(phi, a, b, T_obs, mu = 0.5, lam = 0.05, eps = 1, epochs = 300,
  lr = 0.01, n_restart = 4, seed = 1, two_stage = TRUE,
  iters = 1000L, damp = 0.5, standardize = TRUE, verbose = FALSE)
## S3 method for class 'bioIOT_fit'
print(x, ...)
## S3 method for class 'bioIOT_fit'
summary(object, ...)
```

Arguments

`phi`	(K, K, F) feature array, or a list of arrays (scenarios).
`a, b`	(K,) masses, or lists (auto-normalized).
`T_obs`	(K, K) observed row-conditional transitions, or a list.
`mu`	Soft-marginal strength (default 0.5, paper working point).

lam	l1 strength in stage 1 (default 0.05).
eps	Entropic regularization (default 1).
epochs	Adam steps per stage (default 300).
lr	Adam learning rate (default 0.01).
n_restart	Random restarts (default 4).
seed	Base seed for restart inits.
two_stage	Debias refit on the selected support (default TRUE).
iters, damp	Forward solver controls.
standardize	Z-score each scenario's features before fitting (default TRUE, paper pipeline).
verbose	Print progress.
x	A bioIOT_fit object.
object	A bioIOT_fit object.
...	Unused.

Value

Object of class bioIOT_fit with components theta (debiased weights), theta1 (stage-1 weights), support (logical mask), loss (final row-CE), restart_losses, scenarios (stored standardized scenarios), meta (standardization transforms), hyper and dimensions K, n_feat.

Examples

```
set.seed(1)
sim <- simulate_iot_states(K = 5, seed = 1)
fit <- fit_iot(sim$phi, sim$a, sim$b, sim$T_true, n_restart = 1, epochs = 100)
fit$theta
summary(fit)
```

---

gsm_id *Extract GSM ids from CEL filenames*

Description

Extract GSM sample ids from CEL filenames, vectorized.

Usage

```
gsm_id(filenames)
```

Arguments

filenames	Character vector of filenames such as "GSM1523727_INT_A.CEL.gz".

Value

Character vector of GSM ids; NA where no match.

Examples

```
gsm_id(c("GSM1523727_INT_A.CEL.gz", "sample_no_id.CEL", NA))
```

---

make_cost *Cost construction, likelihood and feature standardization*

Description

make_cost: linear feature cost $C_{ij} = -\sum_k \phi_{ijk}\theta_k$. row_ce_loss: cross-entropy between observed row-conditional transitions and the model $Q = P/a$; zero-mass source states are masked out. zscore_phi: z-scores each feature across all (i, j) entries and returns the transform for reapplication.

Usage

```
make_cost(phi, theta)
row_ce_loss(T_obs, P, a, eps_t = 1e-12)
zscore_phi(phi)
```

Arguments

phi	(K, K, F) feature array; a (K, K) matrix is promoted to a single feature.
theta	Numeric length F (a scalar is recycled for 2-D phi).
T_obs	(K, K) observed row-conditional transition matrix.
P	(K, K) model plan.
a	(K,) source masses.
eps_t	Log floor.

Value

make_cost: (K, K) cost matrix. row_ce_loss: numeric loss (lower is better). zscore_phi: list with phi_z ((K, K, F) standardized array) and meta (2 x F matrix with rows "mean" and "sd").

Examples

```
phi <- array(rnorm(3 * 3 * 2), c(3, 3, 2))
C <- make_cost(phi, c(1, -1))
out <- zscore_phi(phi)
row_ce_loss(matrix(0.5, 3, 3), matrix(0.25, 3, 3), rep(1, 3) / 3)
```

pathway_markers *IOT pathway marker genes (8 pathways)*

Description

The pathway marker gene library used by the paper's 8-dimensional IOT cost features: EMT, Angiogenesis, Hypoxia, Stemness, Immune_Cytotoxic, CellCycle, TGFb and Chemokine. Genes are HGNC symbols.

Usage

```
pathway_markers
```

Value

A named list of 8 character vectors of HGNC gene symbols.

Examples

```
names(pathway_markers)
length(pathway_markers$EMT)
```

plot_transition_heatmap *bioIOT visualisation*

Description

`plot_transition_heatmap`: transition-matrix heatmap with annotated weights. `plot_transition_flow`: transition arrows on a 2-D state embedding, width proportional to transition mass (CellRank-style). `plot_theta`: barplot of fitted feature weights, selected support highlighted. `plot_pathway_trend`: pathway score trajectories over time (loess).

Usage

```
plot_transition_heatmap(Q, labels = NULL)
plot_transition_flow(Q, embedding, labels = NULL, threshold = 0.05)
plot_theta(fit, labels = NULL)
plot_pathway_trend(pw_scores, time, se = TRUE)
```

Arguments

Q	(K, K) row-stochastic transition matrix.
labels	Optional state / feature labels.
embedding	(K x >= 2) state coordinates (e.g. cluster centroids).
threshold	Minimum transition mass to draw an arrow.
fit	A bioIOT_fit object (or named numeric weights with an optional support attribute).
pw_scores	Sample x pathway score matrix.
time	Numeric time / pseudotime per sample (row).
se	Show the smooth confidence band (default TRUE).

Value

A ggplot object.

Examples

```
Q <- matrix(c(0.7, 0.3, 0, 0.2, 0.6, 0.2, 0.1, 0.1, 0.8), 3, 3, byrow = TRUE)
plot_transition_heatmap(Q)
emb <- matrix(c(0, 0, 1, 1, 2, 0), 3, 2, byrow = TRUE)
plot_transition_flow(Q, emb)

set.seed(1)
sim <- simulate_iot_states(K = 5, seed = 1)
fit <- fit_iot(sim$phi, sim$a, sim$b, sim$T_true, n_restart = 1, epochs = 100)
plot_theta(fit)

expr <- matrix(rnorm(200 * 12), nrow = 200)
rownames(expr) <- paste0("G", 1:200)
colnames(expr) <- paste0("S", 1:12)
rownames(expr)[1] <- "VIM"; rownames(expr)[2] <- "CDH2"; rownames(expr)[3] <- "MKI67"
plot_pathway_trend(score_pathways(expr), time = sort(runif(12)))
```

---

runIOT *Run IOT on a single-cell object*

Description

Aggregates cells into states (clusters), builds state-transition features from the cell embedding, optionally fits IOT weights against observed transitions T_obs, and returns the state transition matrix with optional random-walk pseudotime.

Usage

```
runIOT(object, ...)

## S3 method for class 'matrix'
runIOT(object, state, from, to, root = NULL,
  n_dim = NULL, T_obs = NULL, mu = 0.5, eps = 1, lam = 0.05,
  epochs = 300, lr = 0.01, n_restart = 4, seed = 1,
  two_stage = TRUE, ...)

## S3 method for class 'SingleCellExperiment'
runIOT(object, state_col, time_col,
  from, to, dimred = "PCA", root = NULL, ...)

## S3 method for class 'Seurat'
runIOT(object, group.by = NULL, split.by = NULL,
  from = NULL, to = NULL, reduction = "pca", root = NULL, ...)
```

Arguments

object	Cell embedding matrix (cells x dims) or a SingleCellExperiment / Seurat object.
state	Factor/character of state (cluster) labels, one per cell.
from, to	Logical vectors or indices marking source / target timepoint cells.
root	Optional root state (name or index) for pseudotime.
n_dim	Number of embedding dims used as state features.
T_obs	Optional (K, K) observed row-conditional transition matrix (e.g. from lineage/clone data). When given, feature weights are fitted; otherwise the plan is solved with uniform feature weights.
mu, eps, lam, epochs, lr, n_restart, seed, two_stage	Passed to fit_iot (only used when T_obs is given).
state_col	State label column in colData (or a vector).
time_col	Timepoint column in colData (or a vector).
dimred	Reduced dimension name (default "PCA").
group.by	Seurat ident / metadata column for states.
split.by	Metadata column holding the timepoint values from/to.
reduction	Reduction name (default "pca").
...	Additional arguments.

Value

List with fit (bioIOT_fit or NULL), Q (transition matrix), u (state centroids), a, b, phi, theta and pseudotime (if root given).

Examples

```
set.seed(1)
sim <- simulate_iot_states(K = 5, n_cells = 40, seed = 1)
res <- runIOT(sim$cell_embedding, sim$cell_state,
              from = sim$cell_time == "t0", to = sim$cell_time == "t1",
              root = "S1", n_restart = 1, epochs = 100)
res$Q[1:3, 1:3]
res$pseudotime
```

`score_pathways` *Score bulk-cohort samples on the 8 IOT pathways*

Description

Per-gene z-scores across samples, then the mean of the pathway marker genes per sample. Exactly the scoring used to build the paper's bulk pathway scores from gene x sample expression matrices.

Usage

```
score_pathways(expr_gene, markers = pathway_markers)
```

Arguments

`expr_gene`	Gene x sample numeric matrix (rownames = gene symbols).
`markers`	Named list of character vectors; default `pathway_markers` (8 pathways).

Value

Sample x pathway numeric matrix (rownames = sample ids, colnames = pathway names). Pathways without any marker present in `expr_gene` yield `NA` columns.

Examples

```
set.seed(1)
expr <- matrix(rnorm(200 * 6), nrow = 200)
rownames(expr) <- paste0("G", 1:200)
colnames(expr) <- paste0("S", 1:6)
rownames(expr)[1] <- "VIM"; rownames(expr)[2] <- "CDH2"
pw <- score_pathways(expr)
dim(pw)
```

`simulate_iot_states` *Simulate single-cell state-transition data for bioIOT*

Description

Generates a reproducible synthetic single-cell-like dataset: states with feature centroids, source/target timepoint masses, a true IOT plan and transition matrix, plus cell-level metadata for `runIOT`. The bundled `demo_iot_states` dataset is produced by this function with its default seed.

Usage

```
simulate_iot_states(K = 6, F_emb = 2, n_cells = 50, seed = 1,
  theta_true = c(0.9, -0.7, 1.1), mu = 0.5, eps = 1)
```

Arguments

`K`	Number of states.
`F_emb`	Number of state feature dimensions (>= 2 for plotting).
`n_cells`	Cells per state per timepoint.
`seed`	Random seed.
`theta_true`	True feature weights (length F_emb + 1).
`mu`	Solver hyperparameter used to build the true plan.
`eps`	Solver hyperparameter used to build the true plan.

Value

List with `u` (K x F_emb state centroids), `phi` ((K, K, F_emb + 1) features), `a`, `b` (normalized state masses), `P_true`, `T_true` (row-conditional), `theta_true`, `embedding` (K x 2 state coordinates), `K`, `F_emb`, `cell_embedding` (cells x F_emb), `cell_state` and `cell_time` (cell-level factors).

Examples

```
sim <- simulate_iot_states(K = 5, seed = 1)
names(sim)
round(sim$T_true, 2)
```

`soft_sinkhorn` *Semi-relaxed OT solver and row-conditional transitions*

Description

`soft_sinkhorn` solves the row-marginal-hard / column-marginal-KL-soft problem $\min_P \langle C, P\rangle - \epsilon H(P) + \mu\, KL(col(P)\|b)$ subject to $P1 = a$, via Anderson-accelerated damped fixed-point iteration. `row_conditional` returns $Q(i,\cdot) = P(i,\cdot)/a_i$.

Usage

```
soft_sinkhorn(C, a, b, mu = 0.5, eps = 1, iters = 1000L, damp = 0.5,
  tol = 1e-12)
row_conditional(P, a, eps_t = 1e-300)
```

Arguments

`C`	(K, K) cost matrix.
`a, b`	(K,) source / target masses; any positive scale is accepted (auto-normalized). Zero-mass states are allowed in `a`.
`mu`	Soft-marginal strength; large `mu` approaches hard-marginal OT, `mu = 0` recovers a plain row-softmax.
`eps`	Entropic regularization (> 0).
`iters`	Fixed-point iteration cap.
`damp`	Damping factor of the base iteration.
`tol`	Fixed-point residual tolerance.
`P`	(K, K) transport plan.
`eps_t`	Numerical floor.

Value

`soft_sinkhorn`: (K, K) transport plan with row sums equal to the normalized `a`. `row_conditional`: (K, K) matrix; rows with $a_i = 0$ are all-zero.

Examples

```
K <- 5
set.seed(1)
P <- soft_sinkhorn(matrix(rnorm(25), 5), rep(1, 5), rep(1, 5))
rowSums(P)
row_conditional(P, rep(1, 5) / 5)
```

tar_utils *Inspect a GEO RAW tar without extracting*

Description

Check whether a GEO `*_RAW.tar` contains CEL data files or only platform (bgx.gz) files, without extracting the archive.

Usage

```
has_cel_file(tar_path)
find_platform_file(tar_path)
```

Arguments

`tar_path`	Path to a `*_RAW.tar` archive.

Value

`has_cel_file`: TRUE if the tar contains `*.CEL` (optionally `.gz`) members. `find_platform_file`: the first `*.bgx.gz` platform filename, or `NA_character_`.

Examples

```
d <- file.path(tempdir(), "bioIOT-tar-example")
dir.create(d, showWarnings = FALSE)
cel <- file.path(d, "sample.CEL.gz")
bgx <- file.path(d, "platform.bgx.gz")
writeLines("demo", cel)
writeLines("demo", bgx)
tmp_tar <- tempfile(fileext = ".tar")
utils::tar(tmp_tar, c(cel, bgx), tar = "internal")
has_cel_file(tmp_tar)
find_platform_file(tmp_tar)
unlink(c(tmp_tar, d), recursive = TRUE)
```

transition_matrix *Transition matrix, pseudotime and state features*

Description

`transition_matrix`: re-solves the plan at the fitted θ and returns $Q = P/a$. `pseudotime_from_transition`: expected number of random-walk steps to reach the root state (root absorbing), turning an IOT transition matrix into per-state pseudotime. `build_state_features`: builds the paper's feature library: per-dimension pure-column target-state features plus the state-similarity interaction block $\phi_{ij,F+1} = u_i \cdot u_j$.

Usage

```
transition_matrix(fit, which = 1, phi = NULL, a = NULL, b = NULL)
pseudotime_from_transition(Q, root)
build_state_features(u)
```

Arguments

fit	A bioIOT_fit object.
which	Scenario index (default 1).
phi, a, b	Optional user-supplied scenario overriding the stored one (phi must be standardized the same way as in the fit).
Q	(K, K) row-stochastic transition matrix.
root	Root state: integer index or a name matching rownames of Q.
u	(K, D) matrix of state feature vectors (e.g. cluster centroids in a reduced space).

Value

transition_matrix: (K, K) row-stochastic transition matrix. pseudotime_from_transition: named numeric vector of expected random-walk hitting times (root = 0). build_state_features: (K, K, D + 1) feature array with dimnames list(NULL, NULL, c(colnames(u), "sim")).

Examples

```
set.seed(1)
sim <- simulate_iot_states(K = 5, seed = 1)
fit <- fit_iot(sim$phi, sim$a, sim$b, sim$T_true, n_restart = 1, epochs = 100)
Q <- transition_matrix(fit)
rowSums(Q)
pseudotime_from_transition(Q, root = 1)
u <- matrix(rnorm(6 * 2), 6, 2, dimnames = list(NULL, c("PC1", "PC2")))
dim(build_state_features(u))
```

Index