

bioIOT: inverse optimal transport for single-cell state transitions

bioIOT package

bioIOT version 0.2.2

bioIOT solves the semi-relaxed inverse optimal transport (IOT) problem: it finds feature weights θ such that the soft-marginal OT plan induced by the linear cost $C = -\text{einsum}(\phi, \theta)$ reproduces observed state transitions, then turns the plan into transition matrices, flow plots and pseudotime.

1 A reproducible demo in one line

`simulate_iot_states()` generates a synthetic single-cell-like dataset with a known ground truth (states, masses, true plan and transitions, cell-level metadata). The same dataset ships pre-computed as `demo_iot_states`.

```
> library(bioIOT)
> set.seed(1)
> sim <- simulate_iot_states(K = 6, seed = 1)
> names(sim)

[1] "u"           "phi"          "a"            "b"
[5] "P_true"      "T_true"       "theta_true"   "embedding"
[9] "K"           "F_emb"        "cell_embedding" "cell_state"
[13] "cell_time"
```

2 Fit feature weights

`fit_iot()` runs two-stage (l1 selection then debias refit) multi-restart fitting with exact implicit gradients.

```
> fit <- fit_iot(sim$phi, sim$a, sim$b, sim$T_true,
+               n_restart = 2, epochs = 150, seed = 1)
> fit
```

```
bioIOT fit: K=6 states, F=3 features, 1 scenario(s)
  final row-CE loss: 7.8811 (2 restart(s))
  support: 3 / 3 features selected
```

```
> summary(fit)
```

```
bioIOT fit (loss 7.8811)
  feature  theta  theta1 support
1       1  0.9001  0.9346   TRUE
2       2 -0.7001 -0.5584   TRUE
3       3  1.1001  1.0174   TRUE
```

3 Transition matrix and pseudotime

`transition_matrix()` re-solves the plan at the fitted weights; `pseudotime_from_transition()` converts it into random-walk pseudotime (expected hitting times to the root state).

```
> Q <- transition_matrix(fit)
> round(Q, 2)
```

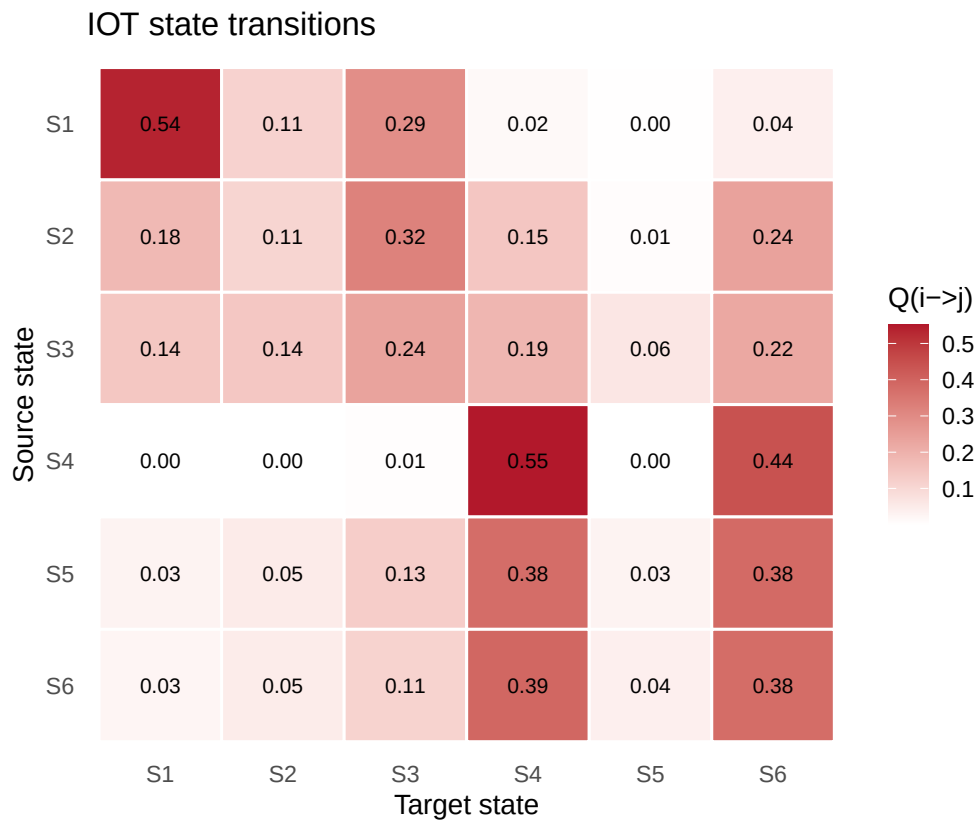
	S1	S2	S3	S4	S5	S6
S1	0.54	0.11	0.29	0.02	0.00	0.04
S2	0.18	0.11	0.32	0.15	0.01	0.24
S3	0.14	0.14	0.24	0.19	0.06	0.22
S4	0.00	0.00	0.01	0.55	0.00	0.44
S5	0.03	0.05	0.13	0.38	0.03	0.38
S6	0.03	0.05	0.11	0.39	0.04	0.38

```
> pseudotime_from_transition(Q, root = "S1")
```

	S1	S2	S3	S4	S5	S6
	0.00000	25.21364	26.61861	34.37971	31.98491	32.22898

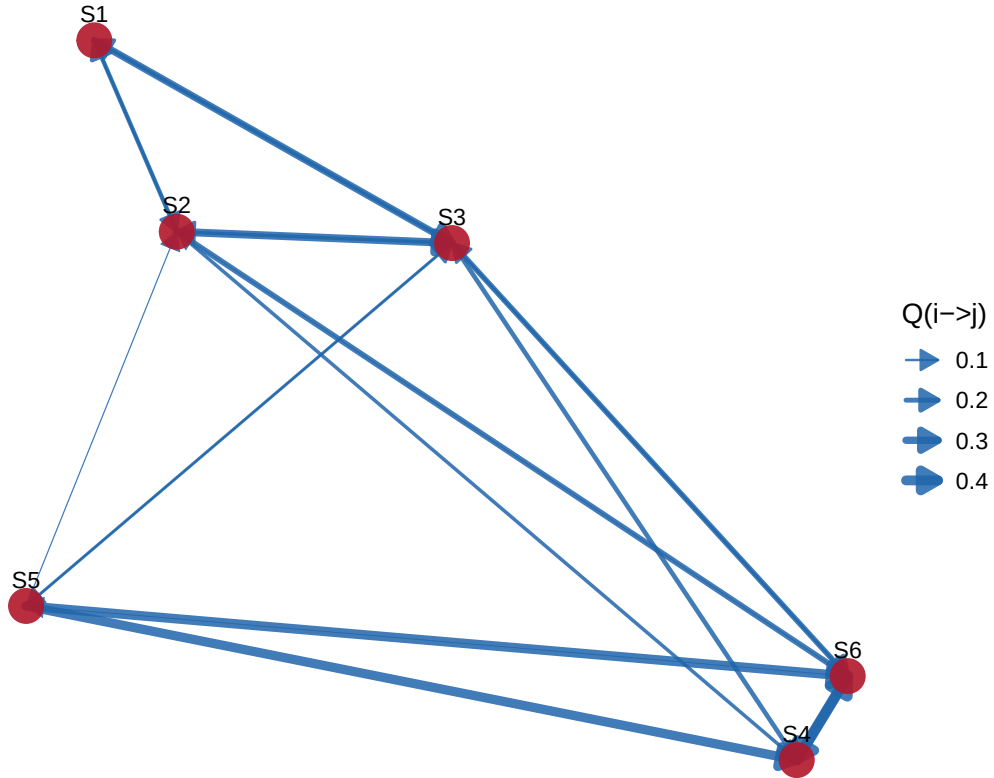
4 Visualisation

```
> plot_transition_heatmap(Q)
```



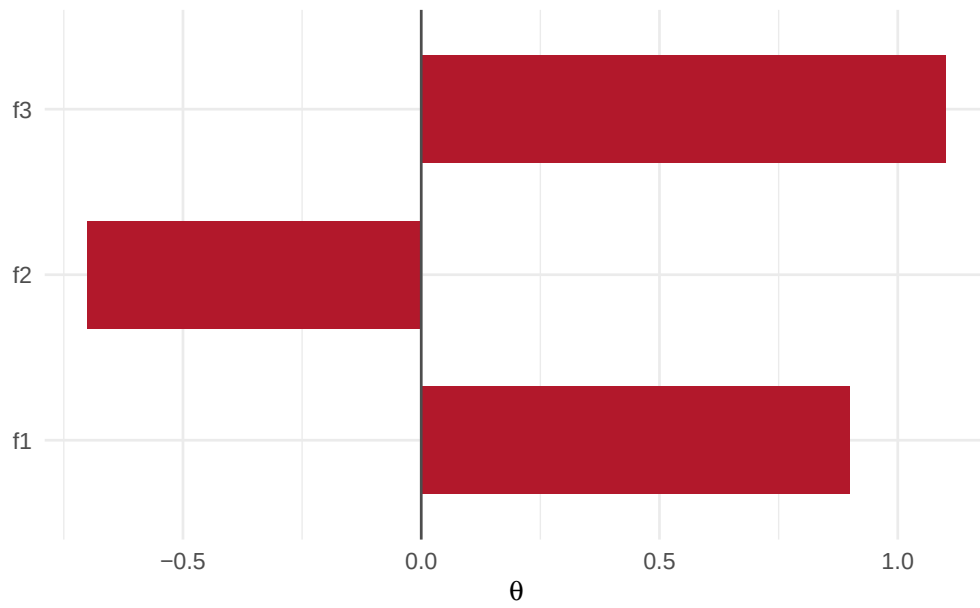
```
> plot_transition_flow(Q, sim$embedding, threshold = 0.04)
```

IOT transition flow



```
> plot_theta(fit)
```

IOT feature weights (red = selected support)



5 Single-cell interface

`runIOT()` works directly on cell-level inputs (base matrix, `SingleCellExperiment` or `Seurat`). Without observed transitions `T_obs` it solves the plan with uniform feature weights; with `T_obs` (e.g. from lineage/clone data) it fits the weights first.

```
> res <- runIOT(sim$cell_embedding, sim$cell_state,
+             from = sim$cell_time == "t0", to = sim$cell_time == "t1",
+             root = "S1")
> round(res$Q[1:3, 1:3], 2)
```

	S1	S2	S3
S1	0.82	0.06	0.11
S2	0.55	0.13	0.27
S3	0.25	0.20	0.27

```
> res$pseudotime
```

	S1	S2	S3	S4	S5	S6
	0.000000	3.169893	5.084061	8.318210	4.904312	7.156977

6 Bulk pathway utilities

The package also ships the paper's bulk-cohort pathway tools: 8-dimensional marker scoring (`score_pathways`), probe-to-gene collapse (`collapse_probes`) and trajectory plots.

```
> 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"
> pw <- score_pathways(expr)
> print(plot_pathway_trend(pw, time = sort(runif(12))))
```

