

Package: smotifrs (via r-universe)

October 19, 2026

Title Rust backend for smotif motif enumeration

Version 0.2.0

Description Drop-in Rust acceleration for the smotif R package:
parallel enumeration of size-2 / size-3 / size-4 multi-cellular
motifs in spatial omics graphs, plus direct parquet I/O for
million-cell datasets. Distributed as pre-compiled binaries via
R-universe.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/rextendr/version 0.5.0

SystemRequirements Cargo (Rust's package manager), rustc >= 1.65.0, xz

Depends R (>= 4.2)

Imports data.table

Suggests arrow, FNN, GiottoDisk, igraph, ggplot2, knitr, nanoarrow,
rmarkdown, smotif, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs xz-utils libclang-dev

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

Date/Publication 2026-10-19 20:49:57 UTC

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

RemoteRef HEAD

RemoteSha aac514826cd8015864af3bce664d67ae914df4de

Contents

edge_store_nodes	2
find_motifs_from_parquet	2
find_motifs_rs	3
motif_enrichment_edge_store	4
motif_enrichment_rs	5
motif_enrichment_stream	7
motif_instances_rs	8
Index	10

edge_store_nodes	<i>Node ids and integer codes from a GiottoDisk edge store sidecar</i>
------------------	--

Description

The order returned here is the order cell_type must be supplied in to `motif_enrichment_edge_store()`.

Usage

`edge_store_nodes(nodes_path)`

Arguments

`nodes_path` path to the store’s nodes/ parquet.

Value

a `data.table` with `node_id` and `int_id`.

find_motifs_from_parquet	<i>Find motifs by reading nodes/edges parquet directly in Rust.</i>
--------------------------	---

Description

Skips the R-side `SpatialGraph` round-trip: opens both parquet files inside Rust via `arrow-rs`, builds a CSR adjacency, and returns the same `Motifs` object as `find_motifs_rs()`. Intended for million-cell datasets where holding the graph in R memory would be wasteful.

Usage

```
find_motifs_from_parquet(
  nodes_path,
  edges_path,
  size = 3L,
  colored = TRUE,
  anchored_on = NULL,
  max_instances = NULL
)
```

Arguments

nodes_path, edges_path	file paths to the parquet outputs of <code>smotif::write_spatial_graph()</code> .
size	integer 2, 3, or 4.
colored	logical; if TRUE, distinguish motifs by color tuple.
anchored_on	optional character vector of cell ids to restrict enumeration to motifs touching at least one anchor.
max_instances	optional integer cap on enumerated instances.

Value

a `smotif::Motifs` S3 object.

find_motifs_rs	<i>Find motifs in a SpatialGraph using the Rust backend.</i>
----------------	--

Description

Drop-in replacement for `smotif::find_motifs()` reached via `smotif::find_motifs(..., backend = "rust")`. Uses a CSR-backed Rust kernel with sorted-merge triangle enumeration and rayon-friendly loops over vertices. Output is bit-for-bit compatible with the `igraph` backend so downstream `Motifs`-consumers (`null_models`, `genes`, `compare`) work unchanged.

Usage

```
find_motifs_rs(
  sg,
  size = 3L,
  colored = TRUE,
  anchored_on = NULL,
  max_instances = NULL
)
```

Arguments

sg	a smotif::SpatialGraph object.
size	integer 2, 3, or 4.
colored	logical; if TRUE, distinguish motifs by color tuple.
anchored_on	optional character vector of cell ids to restrict enumeration to motifs touching at least one anchor.
max_instances	optional integer cap on enumerated instances.

Value

a smotif::Motifs S3 object.

motif_enrichment_edge_store
Motif enrichment straight from a GiottoDisk edge store

Description

Reads a parquetEdgeStore in Rust and runs the enrichment without materializing the graph in R.

Usage

```
motif_enrichment_edge_store(
  nodes_path,
  edges_path,
  cell_type,
  size = 3L,
  n_perm = 1000L,
  seed = 1L,
  null = c("label", "conditional"),
  cond_temp = 1
)
```

Arguments

nodes_path, edges_path	paths to the store's nodes/ and edges/ parquet files.
cell_type	factor or character vector of labels, one per node, in sidecar int_id order.
size	motif size: 2, 3 or 4.
n_perm, seed	permutation count and seed.
null	"label" or "conditional".
cond_temp	Metropolis temperature for the conditional null.

Details

GiottoDisk interns node ids at write time, so the edge parquet already holds integer endpoints. Reading those columns straight through skips the string-to-index hashing that `find_motifs_from_parquet()` has to do for smotif's own on-disk format – there is no hash map and no string per cell.

Cell type labels are not in the edge store by design (the node sidecar carries ids only), so they are supplied here, aligned to the sidecar's `int_id` order. `edge_store_nodes()` returns that order.

Value

a `data.table` in the same shape as `motif_enrichment_rs()`.

See Also

`edge_store_nodes()`

<code>motif_enrichment_rs</code>	<i>Motif enrichment in one pass</i>
----------------------------------	-------------------------------------

Description

Enumerates every connected induced subgraph of the given size, then tests each colored motif class against a label-permutation null. Enumeration happens once and each permutation is a relabel-and-count over the fixed instance set, so per-instance data never reaches R and the cost of the null is integer arithmetic rather than string construction.

Usage

```
motif_enrichment_rs(
  from,
  to,
  cell_type,
  n_nodes = NULL,
  size = 3L,
  n_perm = 1000L,
  seed = 1L,
  null = c("label", "stratified", "conditional"),
  cond_temp = 1,
  strata = NULL,
  anchored_on = NULL
)
```

Arguments

<code>from, to</code>	integer vectors of 1-based node indices, one pair per edge. Direction is ignored and duplicate or reversed edges are collapsed.
<code>cell_type</code>	factor or character vector of cell type labels, one per node, in node-index order.

n_nodes	number of nodes. Defaults to length(cell_type).
size	motif size: 2, 3 or 4.
n_perm	number of label permutations.
seed	integer seed. Results depend only on (seed, draw index), never on thread scheduling.
null	which null to test against. • "label" shuffles labels over all nodes. Answers "is this motif more common than the tissue's cell type composition implies". • "stratified" shuffles only within strata, preserving each stratum's composition exactly. • "conditional" holds the observed pairwise edge-type composition and randomizes everything else, so a size-3 or size-4 motif that is still enriched is enriched <i>beyond</i> what its constituent pairs already explain. Under "label" any motif built from an attracting pair looks enriched, and the higher-order signal cannot be separated from an echo of the pairwise one. The conditional null conditions by tolerance, not exactly: a Metropolis chain over label swaps holds the pairwise table close rather than pinning it. Three attributes report what the chain actually did – cond_dev (the table deviation, as a fraction of all edges), cond_accept (acceptance rate) and cond_moved (mean fraction of labels displaced from the observed assignment). Report them rather than assuming the constraint held. Watch cond_moved in particular. A chain held too cold barely moves, so every draw is essentially the observed data and <i>everything</i> comes back non-significant – a failure that looks like a clean result. Below 5% displacement this warns.
cond_temp	Metropolis temperature for null = "conditional". Lower holds the pairwise table tighter but mixes more slowly.
strata	optional factor or integer vector, one per node. Required for null = "stratified". When given, labels are only exchanged between nodes sharing a stratum, which preserves each stratum's cell type composition exactly.
anchored_on	optional integer vector of 1-based node indices. Enumeration is restricted to subgraphs containing at least one of them.

Details

The graph is supplied as plain vectors rather than as any package's graph class, so callers outside **smotif** can use it directly. Cell types are passed as a factor (or coerced to one), which fixes the level ordering on the R side – the backend never applies its own collation.

Value

a data.table, one row per motif class, with columns motif_id, topology, size, color_tuple (a list column of labels in canonical orbit order), observed, expected, sd_null, z, fold, p_enrich, p_deplete and p_adj. n_instances is attached as an attribute.

Positions within color_tuple are structural roles, not an arbitrary sort, so A-B-B and B-A-A are different motifs. The first position is the distinguished vertex wherever a topology has one: the

centre of a open wedge or a claw, and the degree-3 vertex of a paw. For path the tuple runs end to end, for cycle it runs around the ring, and for closed (triangle), diamond and K4 the remaining positions are interchangeable within their degree class.

Examples

```
# a 6-cycle alternating between two cell types
from <- c(1L, 2L, 3L, 4L, 5L, 6L)
to <- c(2L, 3L, 4L, 5L, 6L, 1L)
ct <- rep(c("A", "B"), 3)
motif_enrichment_rs(from, to, ct, size = 3L, n_perm = 99L)
```

`motif_enrichment_stream`

Motif enrichment over an Arrow stream of edges

Description

Runs the enrichment against edges pulled from an Arrow stream rather than read from a file path, so the producer decides what the edge set *is*.

Usage

```
motif_enrichment_stream(
  edges,
  int_ids,
  cell_type,
  size = 3L,
  n_perm = 1000L,
  seed = 1L,
  null = c("label", "conditional"),
  cond_temp = 1
)
```

Arguments

<code>edges</code>	anything <code>nanoarrow::as_nanoarrow_array_stream()</code> accepts – an <code>arrow::RecordBatchReader</code> , <code>Table</code> , or <code>Dataset</code> query. Batches must carry integer <code>from_id</code> and <code>to_id</code> columns.
<code>int_ids</code>	integer ids keying the label lookup, in the same order as <code>cell_type</code> , without duplicates. Every edge endpoint must appear; ids the stream never uses are ignored.
<code>cell_type</code>	factor or character vector of labels, one per entry of <code>int_ids</code> .
<code>size</code>	motif size: 2, 3 or 4.
<code>n_perm, seed</code>	permutation count and seed.
<code>null</code>	"label" or "conditional".
<code>cond_temp</code>	Metropolis temperature for the conditional null.

Details

`motif_enrichment_edge_store()` opens a store's parquet files directly, which means it sees them as they sit on disk. That is wrong whenever the caller's view of the store differs from its files: a `parquetEdgeStore` carrying a pending subset, a dataset written across several files, or a query over something that is not parquet at all. A stream moves that decision to the producer – `GiottoDisk::storeRead(x, output = "arrowstream")` applies a store's pending ops before yielding any batch – and leaves this package knowing only that batches arrive with `from_id` and `to_id` columns.

The network's nodes are the stream's endpoints. Labels travel beside the stream as a lookup keyed by `int_ids`; mapping cell IDs to those integers (a store's node sidecar, say) is the caller's business, and nothing here reads it. The lookup may cover more ids than the stream uses – a node with no edges is not in the network and does not enter the null – but every endpoint needs a label.

Ownership of the stream moves to Rust, which releases it when the last batch has been pulled. Pass a fresh stream per call.

Value

a `data.table` in the same shape as `motif_enrichment_rs()`.

See Also

`motif_enrichment_edge_store()` for the path-based form.

<code>motif_instances_rs</code>	<i>Instances of selected motif classes</i>
---------------------------------	--

Description

Returns the cells making up each occurrence of the named motif classes.

Usage

```
motif_instances_rs(
  from,
  to,
  cell_type,
  motif_ids,
  n_nodes = NULL,
  size = 3L,
  max_per_class = 5000
)
```

Arguments

<code>from</code> , <code>to</code>	integer vectors of 1-based node indices, one pair per edge. Direction is ignored and duplicate or reversed edges are collapsed.
<code>cell_type</code>	factor or character vector of cell type labels, one per node, in node-index order.

motif_ids	character vector of motif_id values from <code>motif_enrichment_rs()</code> .
n_nodes	number of nodes. Defaults to <code>length(cell_type)</code> .
size	motif size: 2, 3 or 4.
max_per_class	cap on instances returned per class. Inf for no cap.

Details

Deliberately narrow. `motif_enrichment_rs()` never returns per-instance data because at size 4 on a real dataset that is tens of millions of rows; this returns instances for *named classes only*, so the output is bounded by those classes' observed counts. Pick the motifs worth looking at first, then ask for their instances.

Value

a `data.table` with `motif_id`, `instance`, `slot` (the structural position within the motif, 1-based) and `node` (the 1-based node index), in long form – k rows per instance.

Examples

```
from <- c(1L, 2L, 3L, 4L, 5L, 6L)
to <- c(2L, 3L, 4L, 5L, 6L, 1L)
ct <- rep(c("A", "B"), 3)
e <- motif_enrichment_rs(from, to, ct, size = 3L, n_perm = 19L)
motif_instances_rs(from, to, ct, size = 3L, motif_ids = e$motif_id[1])
```

Index

edge_store_nodes, [2](#)
edge_store_nodes(), [5](#)

find_motifs_from_parquet, [2](#)
find_motifs_from_parquet(), [5](#)
find_motifs_rs, [3](#)
find_motifs_rs(), [2](#)

motif_enrichment_edge_store, [4](#)
motif_enrichment_edge_store(), [2](#), [8](#)
motif_enrichment_rs, [5](#)
motif_enrichment_rs(), [5](#), [8](#), [9](#)
motif_enrichment_stream, [7](#)
motif_instances_rs, [8](#)

nanoarrow::as_nanoarrow_array_stream(),
[7](#)