

Package: smotif (via r-universe)

October 15, 2026

Type Package

Title Spatial Multi-Cellular Motifs

Version 0.2.0

Description Discovery and statistical evaluation of recurrent multi-cellular motifs (size 2, 3, and 4) in spatial omics data. Provides a lightweight 'SpatialGraph' representation backed by parquet, enumeration over igraph, spatially-aware null models, motif-to-gene association, and cross-region or cross-sample comparison. Designed to be framework-agnostic so that the core can be folded into Giotto Suite later.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.2)

Imports arrow, data.table, igraph, Matrix

Suggests FNN, Giotto, smotifrs (>= 0.2.0), GiottoClass, GiottoVisuals, dbscan, deldir, ggplot2, gridExtra, httr2, jsonlite, knitr, motifr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake libglpk-dev libxml2-dev libssl-dev

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

Date/Publication 2026-10-15 21:20:48 UTC

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

RemoteRef HEAD

RemoteSha 5cae17932f26332bb84a598cd0360b8ebecbfd1f

Contents

as_igraph	2
associate_genes	3
build_spatial_graph	5
compare_motifs	6
find_motifs	7
motif_enrichment	8
motif_enrichment_store	10
motif_enrichment_stream	11
motif_instances	12
Motifs	12
plot.Motifs	13
read_spatial_graph	14
SpatialGraph	14
store_node_order	15
subset_region	16
test_motif_enrichment	17
write_spatial_graph	18
Index	19

as_igraph	<i>Return (and cache) the igraph view of a SpatialGraph</i>
-----------	---

Description

Builds an undirected igraph graph with one vertex per node and one edge per row in sg\$edges. Vertex attributes mirror sg\$nodes columns; edge attributes mirror non-endpoint columns of sg\$edges. The graph is memoized inside sg\$.cache\$igraph and reused on subsequent calls — the cache is keyed on the row-counts of nodes/edges so it auto-invalidates across subsets.

Usage

as_igraph(sg)

Arguments

sg a SpatialGraph.

Value

an igraph::igraph object.

Examples

```
sg <- SpatialGraph(
  data.frame(cell_id = c("a","b"), x = 0:1, y = 0,
    cell_type = "T", sample_id = "s1",
    stringsAsFactors = FALSE),
  data.frame(source = "a", target = "b", sample_id = "s1",
    stringsAsFactors = FALSE)
)
g <- as_igraph(sg)
igraph::vcount(g)
```

associate_genes

Associate genes (or gene-set scores) with motifs or motif orbits

Description

For each motif (or motif $\times$ orbit), test whether genes are differentially expressed between cells participating in that motif/orbit ("in") and the rest of the dataset ("out"). Two methods are supported:

Usage

```
associate_genes(
  motifs,
  expression,
  sg,
  by = c("motif", "orbit"),
  method = c("wilcoxon", "module_score"),
  modules = NULL,
  min_cells = 10L,
  min_in_expressed = 3L,
  pooling = c("pooled", "per_sample")
)
```

Arguments

motifs	a Motifs object from <code>find_motifs()</code> .
expression	matrix or <code>Matrix::dgCMatrix</code> , genes $\times$ cells, with row names (genes) and column names (cell ids matching sg).
sg	the <code>SpatialGraph</code> motifs was computed from.
by	"motif" (one test per motif) or "orbit" (one test per motif $\times$ orbit_id combination).
method	"wilcoxon" or "module_score".
modules	named list of gene sets; required when method = "module_score".
min_cells	motifs/orbits with fewer than this many unique cells in the "in" set are skipped (per-sample under pooling = "per_sample").

`min_in_expressed` for method = "wilcoxon", only test genes expressed in at least this many "in" cells.

`pooling` "pooled" (default) or "per_sample". See "Details".

Details

- "wilcoxon": rank-sum test per gene with tie correction. The full per-cell rank matrix is computed once and reused across motifs. Genes are filtered by `min_in_expressed` (number of "in" cells with non-zero expression).
- "module_score": per-cell scores are computed as the mean of row-standardized (z-scored) expression over each gene set, then compared between motif and background using the same rank-sum framework.

Two pooling modes are supported:

- `pooling = "pooled"` (default): all cells across all samples contribute to a single in/out test per motif. Fastest; appropriate for single-sample data or when sample-level heterogeneity is not of interest.
- `pooling = "per_sample"`: each sample is tested independently and per-sample statistics are combined via a Stouffer-weighted Z meta-analysis (weights = $\sqrt{n_{in}}$ per sample). The output reports `n_samples_tested`, `mean_log2FC` and `sd_log2FC` so between-sample heterogeneity is visible: a gene with a large `z_combined` and a tiny `sd_log2FC` reproduces across samples; a gene with a large `z_combined` and large `sd_log2FC` is being driven by one sample.

Value

a `data.table`. Columns under `pooling = "pooled"`: `motif_id`, `orbit_id`, `gene`, `n_in`, `n_out`, `n_in_expressed`, `mean_log2FC`. Under `pooling = "per_sample"`: `motif_id`, `orbit_id`, `gene`, `n_samples_tested`, `n_in_total`, `n_out_total`, `mean_log2FC`, `sd_log2FC`.

Examples

```
set.seed(1)
coords <- cbind(x = runif(60), y = runif(60))
types <- sample(c("T", "B"), 60, TRUE)
sg <- build_spatial_graph(coords, types, "s1", method = "knn", k = 4L)
m <- find_motifs(sg, size = 3L)
expr <- Matrix::Matrix(
  matrix(stats::rpois(60 * 50, 1), nrow = 50),
  sparse = TRUE
)
rownames(expr) <- paste0("g", 1:50)
colnames(expr) <- sg$nodes$cell_id
associate_genes(m, expr, sg, min_cells = 8L)[1:5]
```

build_spatial_graph	<i>Build a SpatialGraph from coordinates</i>
---------------------	--

Description

Build a SpatialGraph over a set of cells given their spatial coordinates, cell types, and sample id, using one of three connectivity rules:

- "knn" (default): each cell connects to its k nearest neighbors. Edges are deduplicated and stored undirected, so the resulting degree can exceed k (a cell may be in another's k-NN without the converse).
- "delaunay": edges from the 2D Delaunay triangulation (xy plane).
- "radius": edges between every pair within radius of each other.

Usage

```
build_spatial_graph(
  coords,
  cell_types,
  sample_id,
  region_id = NULL,
  method = c("knn", "delaunay", "radius"),
  k = 6L,
  radius = NULL,
  max_distance = NULL,
  cell_id = NULL
)
```

Arguments

coords	numeric matrix or data.frame with columns x, y and optionally z (z is preserved on nodes but ignored by all connectivity rules in v0.1).
cell_types	character vector aligned with rows of coords.
sample_id	character; recycled to length nrow(coords) if scalar.
region_id	optional character vector aligned with rows of coords.
method	one of "knn", "delaunay", "radius".
k	integer; neighbors when method = "knn".
radius	numeric; cutoff when method = "radius".
max_distance	optional numeric cap applied after edge construction.
cell_id	optional character vector of cell ids; if missing, ids cell_<i>i</i> are generated.

Details

Implementations rely on lightweight optional dependencies (FNN, deldir, dbscan); install only the one(s) you need. An optional max_distance cap is applied after edge construction.

Value

a SpatialGraph.

Examples

```
set.seed(1)
coords <- cbind(x = runif(20), y = runif(20))
sg <- build_spatial_graph(
  coords, cell_types = sample(c("T", "B"), 20, TRUE),
  sample_id = "s1", method = "knn", k = 3
)
nrow(sg$edges)
```

compare_motifs	<i>Compare motif distributions across regions or samples</i>
----------------	--

Description

Pools per-group motif counts into a wide table and tests, for each motif, whether its count distribution across groups departs from the overall distribution of motif instances across groups (chi-square test for independence).

Usage

```
compare_motifs(
  motifs_list,
  group = NULL,
  design = NULL,
  method = c("wilcoxon", "negbin")
)
```

Arguments

motifs_list	named list of Motifs objects, one per group (or per replicate if group is supplied).
group	optional character vector of length <code>length(motifs_list)</code> giving the group label for each element. Defaults to <code>names(motifs_list)</code> .
design	reserved for v0.2 covariate adjustment; ignored in v0.1.
method	"wilcoxon" or "negbin"; both currently fall through to chi-square. Kept on the signature so call sites are stable.

Details

If multiple `Motifs` objects are assigned to the same group via the `group` argument, their counts are summed within the group — useful for replicates. Motifs absent from a group contribute zero to that column.

v0.1 uses a chi-square statistic for any number of groups. The `method` argument is reserved for future per-replicate tests ("wilcoxon" rank-sum, "negbin" count regression); both currently fall through to chi-square with a warning. `design` is reserved for covariate adjustment in v0.2.

Value

a `data.table` with one row per motif present in at least one group: `motif_id`, `size`, `canonical_iso_class`, `color_tupl`.

Examples

```
set.seed(1)
coords <- cbind(x = runif(40), y = runif(40))
types <- sample(c("T","B"), 40, TRUE)
sg <- build_spatial_graph(coords, types, "s1", method = "knn", k = 4L,
                          region_id = sample(c("a","b"), 40, TRUE))
sa <- subset_region(sg, region_id == "a")
sb <- subset_region(sg, region_id == "b")
ma <- find_motifs(sa, size = 3L)
mb <- find_motifs(sb, size = 3L)
compare_motifs(list(a = ma, b = mb))[1:3]
```

find_motifs

Find motifs in a SpatialGraph

Description

Enumerate all motifs of a given size in `sg`, returning a `Motifs` S3 object. v0.1 supports sizes 2 and 3 (size 4 lands in step 8). All connected sub-graphs of that size are enumerated; the resulting catalog distinguishes topology classes (e.g., closed triangle vs open wedge for size 3) via the `canonical_iso_class` column.

Usage

```
find_motifs(
  sg,
  size = 3L,
  backend = "igraph",
  colored = TRUE,
  anchored_on = NULL,
  max_instances = NULL
)
```

Arguments

<code>sg</code>	a <code>SpatialGraph</code> .
<code>size</code>	integer; one of 2, 3 (4 in v0.2).
<code>backend</code>	"igraph" (only available in v0.1) or "rust" (planned).
<code>colored</code>	logical; if TRUE, distinguish motifs by ordered color tuples in addition to topology.
<code>anchored_on</code>	optional character vector of <code>cell_ids</code> . When set, only motifs containing at least one anchored cell are returned — intended to keep enumeration tractable on large graphs (size ≥ 4).
<code>max_instances</code>	optional integer cap on enumerated instances per motif. If exceeded, a deterministic random sample of size <code>max_instances</code> is kept and a warning is emitted.

Details

When `colored = TRUE` (default), motifs are further keyed by the sorted tuple of cell-type colors per orbit, so e.g. a T-T-B triangle is a different motif from a T-B-B triangle. The canonical labeling sorts vertices by (degree-in-motif, color), which is exact for sizes 2 and 3 combined with topology class.

Value

a `Motifs` object — a list with `catalog`, `incidence`, `meta`.

Examples

```
set.seed(1)
coords <- cbind(x = runif(20), y = runif(20))
sg <- build_spatial_graph(
  coords, cell_types = sample(c("T", "B"), 20, TRUE),
  sample_id = "s1", method = "knn", k = 3
)
m <- find_motifs(sg, size = 2)
m$catalog
```

motif_enrichment

Motif enrichment on a spatial graph

Description

Enumerates every connected induced subgraph of the requested size and tests each colored motif class against a permutation null. Enumeration happens once; each permutation is a relabel-and-count over the fixed instance set.

Usage

```

motif_enrichment(
  sg,
  size = 3L,
  n_perm = 1000L,
  seed = 1L,
  null = c("label", "stratified", "conditional"),
  strata_column = NULL,
  anchored_on = NULL,
  backend = c("auto", "rust", "r"),
  ...
)

```

Arguments

<code>sg</code>	a SpatialGraph() .
<code>size</code>	motif size: 2, 3 or 4.
<code>n_perm</code>	number of null draws.
<code>seed</code>	integer seed.
<code>null</code>	which null to test against. "label" shuffles labels over all nodes; "stratified" shuffles within <code>strata_column</code> ; "conditional" holds the observed pairwise edge-type composition fixed, so a size-3 or size-4 motif that is still enriched is enriched <i>beyond</i> what its constituent pairs explain. See smotifrs::motif_enrichment_rs() for the sampler and its diagnostics.
<code>strata_column</code>	name of a column on <code>sg\$nodes</code> to stratify by. Defaults to <code>region_id</code> when that exists and <code>null = "stratified"</code> .
<code>anchored_on</code>	optional character vector of <code>cell_ids</code> ; enumeration is restricted to subgraphs touching at least one of them.
<code>backend</code>	"auto", "rust" or "r".
<code>...</code>	passed to the backend (e.g. <code>cond_temp</code>).

Value

a `data.table`, one row per motif class: `motif_id`, `topology`, `size`, `color_tuple`, `observed`, `expected`, `sd_null`, `z`, `fold`, `p_enrich`, `p_deplete`, `p_adj`.

Examples

```

set.seed(1)
coords <- cbind(x = runif(60), y = runif(60))
sg <- build_spatial_graph(coords,
  cell_types = sample(c("T", "B"), 60, TRUE),
  sample_id = "s1", method = "knn", k = 4
)
motif_enrichment(sg, size = 3L, n_perm = 99L)

```

motif_enrichment_store

Motif enrichment straight from a disk-backed edge store

Description

Runs enrichment against a GiottoDisk parquetEdgeStore without building the graph in R. Useful when the network is large enough that materializing an igraph is itself the expensive step.

Usage

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

Arguments

`nodes_path`, `edges_path` paths to the store's nodes/ and edges/ parquet files.

`cell_type` labels, one per node, in sidecar `int_id` order.

`size`, `n_perm`, `seed`, `null`, ... passed through to the backend.

Details

Node ids are already integers in that format, so no string hashing happens on either side. Cell type labels are supplied aligned to the store's sidecar order, which `smotifrs::edge_store_nodes()` returns.

Value

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

motif_enrichment_stream

Motif enrichment over a stream of edges

Description

Runs enrichment against edges pulled from an Arrow stream, so whatever produced the stream decides the edge set – a pending subset on a disk-backed store is honoured without the store’s files being read here. The network’s nodes are the stream’s endpoints.

Usage

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

Arguments

edges	anything <code>nanoarrow::as_nanoarrow_array_stream()</code> accepts, with integer <code>from_id</code> and <code>to_id</code> columns. Read once.
int_ids	integer node ids keying the label lookup, without duplicates.
cell_type	labels, one per entry of <code>int_ids</code> .
size, n_perm, seed, null, ...	passed through to the backend.

Details

Labels travel beside the stream as a lookup keyed by integer node id. Mapping cell IDs to those integers is the caller’s business; the lookup may cover nodes the stream never reaches, but every endpoint needs a label.

Value

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

motif_instances	<i>Cells making up occurrences of selected motif classes</i>
-----------------	--

Description

Pick the motifs worth looking at from `motif_enrichment()` first, then ask where they are. Instances are returned for named classes only, so the output is bounded by those classes' counts rather than by the whole enumeration – at size 4 the full instance set is tens of millions of rows on a real dataset, which is why it is never returned by default.

Usage

```
motif_instances(sg, motif_ids, size = 3L, max_per_class = 5000)
```

Arguments

<code>sg</code>	a <code>SpatialGraph()</code> .
<code>motif_ids</code>	character vector of <code>motif_id</code> values.
<code>size</code>	motif size the ids came from.
<code>max_per_class</code>	cap on instances per class; Inf for no cap.

Value

a `data.table` with `motif_id`, `instance`, `slot` (structural position within the motif) and `cell_id`, in long form.

Examples

```
set.seed(1)
sg <- build_spatial_graph(cbind(x = runif(50), y = runif(50)),
  cell_types = sample(c("T", "B"), 50, TRUE),
  sample_id = "s1", method = "knn", k = 4
)
e <- motif_enrichment(sg, size = 3L, n_perm = 49L)
motif_instances(sg, e$motif_id[1], size = 3L)
```

Motifs	<i>Construct a Motifs object</i>
--------	----------------------------------

Description

Internal-style constructor used by `find_motifs()` to assemble its return value. Kept exported because tests and downstream code may build `Motifs` objects from custom enumerators.

Usage

```
Motifs(catalog, incidence, instance_meta = NULL, meta = list())
```

```
## S3 method for class 'Motifs'
print(x, ...)
```

Arguments

catalog	a data.table with motif_id, size, canonical_iso_class, color_tuple, count.
incidence	a data.table with motif_id, instance_id, cell_id, orbit_id.
instance_meta	optional named list of per-topology-class wide instance tables, used by test_motif_enrichment() for fast permutation recounts. Expected names: edge, triangle, wedge. Each table holds the cell ids of an instance in canonical role order — e.g. wedge has instance_id, end1, end2, center. May be NULL for Motifs objects produced by hand.
meta	a named list (size, backend, colored, sg_hash).
x	a Motifs object.
...	ignored.

Value

a Motifs S3 object.

plot.Motifs	<i>Bar chart of the most common motifs</i>
-------------	--

Description

Returns a ggplot2 object (requires the optional ggplot2 Suggests). Bars are sorted by count, descending. For colored motifs the labels include the color tuple, which can get long — a default truncation keeps display readable.

Usage

```
## S3 method for class 'Motifs'
plot(x, top = 20L, max_label_chars = 60L, ...)
```

Arguments

x	a Motifs object.
top	integer; number of top motifs to plot.
max_label_chars	integer; truncate motif_id labels longer than this with an ellipsis.
...	ignored.

Value

a ggplot object.

read_spatial_graph	<i>Read a SpatialGraph from parquet</i>
--------------------	---

Description

Reads nodes.parquet and edges.parquet, validates the schemas, and returns a SpatialGraph. If a sidecar <nodes_path>.meta.json exists, it is loaded into sg\$meta; otherwise meta records the source paths so a round-trip is reproducible.

Usage

```
read_spatial_graph(nodes_path, edges_path)
```

Arguments

nodes_path	path to a nodes.parquet file.
edges_path	path to an edges.parquet file.

Value

a SpatialGraph.

Examples

```
## Not run:
sg <- read_spatial_graph("nodes.parquet", "edges.parquet")

## End(Not run)
```

SpatialGraph	<i>Construct a SpatialGraph</i>
--------------	---------------------------------

Description

Validates schemas, normalizes column types, ensures undirected edges are canonical (source < target), and returns an S3 object of class SpatialGraph. The igraph view is built lazily by as_igraph().

Usage

```
SpatialGraph(nodes, edges, meta = list())

## S3 method for class 'SpatialGraph'
print(x, ...)
```

Arguments

nodes a data.frame / data.table of cells (see "Details").
 edges a data.frame / data.table of edges (see "Details").
 meta named list of graph-construction metadata.
 x a SpatialGraph.
 ... ignored.

Value

a SpatialGraph.
 x invisibly (for print).

Examples

```

nodes <- data.frame(
  cell_id = c("a", "b", "c", "d"),
  x       = c(0, 1, 2, 3),
  y       = c(0, 0, 1, 1),
  cell_type = c("T", "B", "T", "B"),
  sample_id = "s1",
  stringsAsFactors = FALSE
)
edges <- data.frame(
  source = c("a", "b", "c"), target = c("b", "c", "d"),
  distance = c(1, sqrt(2), 1), sample_id = "s1",
  stringsAsFactors = FALSE
)
sg <- SpatialGraph(nodes, edges, meta = list(build_method = "manual"))
  
```

store_node_order	<i>Node ids of a disk-backed edge store, in sidecar order</i>
------------------	---

Description

The order `motif_enrichment_store()` expects `cell_type` in.

Usage

```
store_node_order(nodes_path)
```

Arguments

nodes_path path to the store's nodes/ parquet.

Value

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

subset_region	<i>Subset a SpatialGraph by a node-column expression</i>
---------------	--

Description

These wrap a non-standard-evaluation filter on `sg$nodes`. After filtering nodes, edges with at least one endpoint outside the surviving set are dropped. The igraph cache is invalidated.

Usage

```
subset_region(sg, expr)
```

```
subset_sample(sg, expr)
```

Arguments

<code>sg</code>	a <code>SpatialGraph</code> .
<code>expr</code>	an unquoted expression evaluated against <code>sg\$nodes</code> columns (e.g. <code>region_id == "niche_1", sample_id %in% c("a", "b")</code>).

Value

a new `SpatialGraph`.

Examples

```
nodes <- data.frame(
  cell_id = letters[1:5], x = 0:4, y = 0,
  cell_type = "T", sample_id = "s1",
  region_id = c("r1", "r1", "r2", "r2", "r2"),
  stringsAsFactors = FALSE
)
edges <- data.frame(
  source = c("a", "b", "c", "d"), target = c("b", "c", "d", "e"),
  sample_id = "s1", stringsAsFactors = FALSE
)
sg <- SpatialGraph(nodes, edges)
sg1 <- subset_region(sg, region_id == "r1")
nrow(sg1$nodes) # 2
```

test_motif_enrichment *Test motif enrichment under a spatially-aware null*

Description

Compares observed per-motif counts against a permutation null distribution and returns a `data.table` of test statistics. Three null models are supported (see [null_models](#)):

Usage

```
test_motif_enrichment(
  motifs,
  sg,
  null = c("label_permutation", "region_stratified", "geometric_jitter"),
  n_perm = 1000L,
  seed = 1L,
  parallel = FALSE,
  null_args = list()
)
```

Arguments

<code>motifs</code>	a <code>Motifs</code> object from find_motifs() .
<code>sg</code>	the <code>SpatialGraph</code> <code>motifs</code> was computed from.
<code>null</code>	one of "label_permutation", "region_stratified", "geometric_jitter".
<code>n_perm</code>	integer; number of permutations.
<code>seed</code>	integer; RNG seed for reproducibility.
<code>parallel</code>	logical; future hook — v0.1 ignores and runs serially.
<code>null_args</code>	optional list of arguments specific to the chosen null. Currently used only by "geometric_jitter" (<code>sd</code>).

Details

- "label_permutation": shuffle `cell_type` across all nodes; graph topology stays fixed.
- "region_stratified": shuffle `cell_type` within each `region_id`; preserves per-region cell-type composition. Requires `sg$nodes$region_id`.
- "geometric_jitter": perturb coordinates by Gaussian noise of standard deviation `null_args$sd`, rebuild the graph using the original `build_spatial_graph` parameters, recount. Requires `sg$meta$build_method` to be one of "knn", "delaunay", "radius".

Empirical p-values use the unbiased "+1" estimator $p = (1 + \sum I[\text{null}_i \geq \text{observed}]) / (1 + n_{\text{perm}})$ so the result is in $(0, 1]$ for finite `n_perm`. The fold change is `observed / max(mean_null, .Machine$double.eps)`. Adjustment for multiple testing is BH on `p_emp`.

Value

a data.table with columns motif_id, observed, mean_null, sd_null, z, p_emp, p_adj_BH, fold.

Examples

```
set.seed(1)
coords <- cbind(x = runif(40), y = runif(40))
sg <- build_spatial_graph(
  coords, cell_types = sample(c("T","B"), 40, TRUE),
  sample_id = "s1", method = "knn", k = 4
)
m <- find_motifs(sg, size = 3L)
test_motif_enrichment(m, sg, null = "label_permutation",
  n_perm = 50L, seed = 1L)
```

write_spatial_graph	<i>Write a SpatialGraph to parquet</i>
---------------------	--

Description

Writes sg\$nodes and sg\$edges as separate parquet files. If jsonlite is available, also writes <nodes_path>.meta.json so that meta survives a round trip.

Usage

```
write_spatial_graph(sg, nodes_path, edges_path)
```

Arguments

sg	a SpatialGraph.
nodes_path	output path for nodes.parquet.
edges_path	output path for edges.parquet.

Value

sg, invisibly.

Examples

```
## Not run:
write_spatial_graph(sg, "nodes.parquet", "edges.parquet")

## End(Not run)
```

Index

as_igraph, [2](#)
associate_genes, [3](#)

build_spatial_graph, [5](#)

compare_motifs, [6](#)

find_motifs, [7](#)
find_motifs(), [3](#), [17](#)

motif_enrichment, [8](#)
motif_enrichment(), [10–12](#)
motif_enrichment_store, [10](#)
motif_enrichment_store(), [15](#)
motif_enrichment_stream, [11](#)
motif_instances, [12](#)
Motifs, [12](#)

nanoarrow::as_nanoarrow_array_stream(),
 [11](#)
null_models, [17](#)

plot.Motifs, [13](#)
print.Motifs(Motifs), [12](#)
print.SpatialGraph(SpatialGraph), [14](#)

read_spatial_graph, [14](#)

smotifrs::edge_store_nodes(), [10](#)
smotifrs::motif_enrichment_rs(), [9](#)
SpatialGraph, [14](#)
SpatialGraph(), [9](#), [12](#)
store_node_order, [15](#)
subset_region, [16](#)
subset_sample(subset_region), [16](#)

test_motif_enrichment, [17](#)
test_motif_enrichment(), [13](#)

write_spatial_graph, [18](#)