Package: terralink (via r-universe)

May 12, 2026

Title Connectivity Corridor Optimization for Raster and Vector Data

Version 1.8.0

Description Standalone R implementation of habitat connectivity corridor optimization for raster and vector workflows. Supports scenario-based planning with budget-constrained optimization, optional impassable areas, packaged parity fixtures, and comparative before-and-after connectivity metrics. The package exposes structural, movement-oriented, and species-oriented strategies in a reproducible workflow aligned with a companion GIS plugin while avoiding a desktop GIS dependency.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/sorus-tools/terralink-r

BugReports https://github.com/sorus-tools/terralink-r/issues

Imports cli, igraph, R6, sf, stars, terra

Suggests gdistance, ggplot2, knitr, lwgeom, raster, rmarkdown, sp, shiny, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Benjamin Bishop [aut, cre], SORUS Consulting LLC [fnd, cph]

Maintainer Benjamin Bishop <benjamin.bishop@sorusconsultingllc.com>

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

Date/Publication 2026-05-12 20:13:51 UTC

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

RemoteRef HEAD

RemoteSha 69ff41b31924448175a7f023bade29265e6a2bd2

Contents

`terralink-package` *terralink: Connectivity Corridor Optimization*

Description

Standalone R implementation of habitat connectivity corridor optimization algorithms for raster and vector workflows, without desktop GIS dependencies.

Details

Main user entry points:

- `terralink_raster()` for raster inputs
- `terralink_vector()` for polygon patch inputs
- `terralink_run()` for a single generic wrapper

Raster inputs are funneled through TerraLink's vector corridor pipeline so the R package matches the current QGIS plugin workflow.

Author(s)

Maintainer: Benjamin Bishop <benjamin.bishop@sorusconsultingllc.com>

Other contributors:

- SORUS Consulting LLC [funder, copyright holder]

See Also

- terralink_raster
- terralink_vector
- terralink_run
- https://github.com/sorus-tools/terralink-r
- Report bugs at https://github.com/sorus-tools/terralink-r/issues

---

`build_contiguous_raster` *Build contiguous network raster (patches + corridors)*

Description

Build contiguous network raster (patches + corridors)

Usage

```
build_contiguous_raster(habitat_mask, corridor_raster, connectivity = 8)
```

Arguments

`habitat_mask`	SpatRaster of habitat.
`corridor_raster`	SpatRaster of corridors.
`connectivity`	Connectivity for patches (4 or 8).

Value

SpatRaster with component sizes.

`build_corridor_raster` *Create corridor raster from selected edges*

Description

Create corridor raster from selected edges

Usage

```
build_corridor_raster(
  labels,
  patch_df,
  corridors,
  min_corridor_width_px = 1,
  assignment_mode = "sum_total_network_area"
)
```

Arguments

`labels`	SpatRaster labels.
`patch_df`	Patch summary data frame.
`corridors`	Data frame with patch1, patch2, and optional line geometry.
`min_corridor_width_px`	Width (pixels) for buffering corridors.
`assignment_mode`	Corridor cell assignment mode.

Value

SpatRaster with corridor cells set by assignment mode.

`build_graph_from_corridors` *Build an igraph graph from patch ids and corridor edges*

Description

Build an igraph graph from patch ids and corridor edges

Usage

```
build_graph_from_corridors(patches, corridors, distance_col = "distance_m")
```

Arguments

patches	Patch ids (vector) or data frame with column id.
corridors	Data frame with patch1, patch2, and optional distance column.
distance_col	Column name to use for edge distance/weight.

Value

An igraph graph.

---

build_patch_candidates

Build candidate edges between patches (centroid distance)

Description

Build candidate edges between patches (centroid distance)

Usage

```
build_patch_candidates(patch_df, max_search_distance, raster_ref)
```

Arguments

patch_df	Patch summary data frame.
max_search_distance	Maximum distance in pixels.
raster_ref	Raster reference for pixel size.

Value

Data frame with patch1, patch2, cost, distance_map, id.

---

build_raster_candidates

Build raster candidates using shortest paths

Description

Build raster candidates using shortest paths

Usage

```
build_raster_candidates(
  labels,
  patch_df,
  passable_mask,
  max_search_distance_px,
  raster_ref,
  min_corridor_width_px = 1,
  pair_index = NULL,
  patch_connectivity = 4,
  habitat_mask = NULL,
  obstacle_mask = NULL
)
```

Arguments

`labels`	SpatRaster of filtered patch labels.
`patch_df`	Patch summary data frame.
`passable_mask`	SpatRaster with 1 for passable cells.
`max_search_distance_px`	Max search distance in pixels.
`raster_ref`	Raster reference for CRS/resolution.
`min_corridor_width_px`	Corridor width in pixels (used for area-based candidate cost).
`pair_index`	Optional two-column matrix of patch index pairs to evaluate.
`patch_connectivity`	Patch connectivity (4 or 8).
`habitat_mask`	SpatRaster of habitat (pre-filter), optional.
`obstacle_mask`	SpatRaster of blocked pixels, optional.

Value

Data frame of candidates with geometry.

`calculate_disturbance_penalty` *Calculate disturbance penalty (diameter normalized)*

Description

Calculate disturbance penalty (diameter normalized)

Usage

```
calculate_disturbance_penalty(graph)
```

Arguments

graph	igraph graph.

Value

Numeric penalty.

---

calculate_failure_probability

Estimate failure probability via edge removal

Description

Estimate failure probability via edge removal

Usage

```
calculate_failure_probability(graph, k_failures = 1, iterations = 100)
```

Arguments

graph	igraph graph.
k_failures	Number of edges removed each trial.
iterations	Number of Monte Carlo iterations.

Value

Failure probability.

---

calculate_movement_entropy

Calculate movement entropy for a graph

Description

Calculate movement entropy for a graph

Usage

```
calculate_movement_entropy(graph, alpha = 0.002)
```

Arguments

graph	igraph graph.
alpha	Dispersal kernel parameter.

Value

Numeric entropy value.

`calculate_topology_penalty` *Calculate topology penalty (cycle count)*

Description

Calculate topology penalty (cycle count)

Usage

```
calculate_topology_penalty(graph)
```

Arguments

`graph`	igraph graph.

Value

Numeric penalty.

`calculate_total_entropy` *Calculate total entropy summary*

Description

Calculate total entropy summary

Usage

```
calculate_total_entropy(graph, lambda_c = 1, lambda_f = 1, lambda_d = 1)
```

Arguments

`graph`	igraph graph.
`lambda_c`	Connectivity penalty multiplier.
`lambda_f`	Topology penalty multiplier.
`lambda_d`	Disturbance penalty multiplier.

Value

Named list of entropy components.

calculate_two_edge_connectivity *Calculate the fraction of node pairs with two edge-disjoint paths*

Description

Calculate the fraction of node pairs with two edge-disjoint paths

Usage

```
calculate_two_edge_connectivity(graph)
```

Arguments

graph	igraph graph.

Value

Numeric ratio.

label_patches *Label contiguous habitat patches*

Description

Label contiguous habitat patches

Usage

```
label_patches(mask, connectivity = 8)
```

Arguments

mask	Logical SpatRaster mask.
connectivity	4 or 8.

Value

SpatRaster of patch labels.

`NetworkOptimizer` *Network optimizer for corridor selection*

Description

Implements a two-phase optimizer: MST backbone, then optional loop additions.

Methods

initialize Create a new optimizer with nodes.

add_candidate Add a candidate edge.

solve Run optimization.

Public fields

`nodes` Node weights.

`edges` Candidate edge list.

`uf` UnionFind instance.

Methods

Public methods:

- `NetworkOptimizer$new()`
- `NetworkOptimizer$add_candidate()`
- `NetworkOptimizer$solve()`
- `NetworkOptimizer$clone()`

Method `new()`:

Usage:

```
NetworkOptimizer$new(nodes)
```

Arguments:

`nodes` Named numeric vector of node weights.

`nodes` Named numeric vector of node weights.

Method `add_candidate()`:

Usage:

```
NetworkOptimizer$add_candidate(u, v, cand_id, cost)
```

Arguments:

`u` Candidate edge start node.

`u` Candidate edge start node.

`v` Candidate edge end node.

`v` Candidate edge end node.

`cand_id` Candidate edge id.

`cand_id` Candidate edge id.
`cost` Candidate edge cost.
`cost` Candidate edge cost.

Method `solve()`**:**

Usage:

```
NetworkOptimizer$solve(budget, loop_fraction = 0.05, max_redundancy = 2)
```

Arguments:

`budget` Numeric budget for corridor costs.
`budget` Numeric budget for corridor costs.
`loop_fraction` Fraction of budget reserved for loops.
`loop_fraction` Fraction of budget reserved for loops.
`max_redundancy` Max redundant edges per component.
`max_redundancy` Max redundant edges per component.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
NetworkOptimizer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`new_candidate` *Create a candidate corridor descriptor*

Description

Create a candidate corridor descriptor

Usage

```
new_candidate(patch_ids, cost, weight = NULL, geometry = NULL)
```

Arguments

`patch_ids`	Integer vector of patch ids.
`cost`	Numeric cost of the corridor.
`weight`	Numeric benefit or ROI.
`geometry`	Optional geometry object.

Value

A candidate list with class 'terralink_candidate'.

new_patch *Create a patch descriptor*

Description

Create a patch descriptor

Usage

```
new_patch(id, weight, geometry = NULL)
```

Arguments

id	Patch identifier.
weight	Numeric patch weight (area, quality).
geometry	Optional geometry object.

Value

A patch list with class 'terralink_patch'.

optimize_largest_network *Optimize for largest connected network (MST backbone + loops)*

Description

Optimize for largest connected network (MST backbone + loops)

Usage

```
optimize_largest_network(
  nodes,
  edges,
  budget,
  loop_fraction = 0.05,
  max_redundancy = 2
)
```

Arguments

nodes	Named numeric vector of patch sizes.
edges	Data frame with u, v, id, cost columns.
budget	Numeric budget for corridor cost.
loop_fraction	Fraction of budget reserved for loops.
max_redundancy	Max redundant edges per component.

Value

List with selected ids and summary.

`optimize_network` *Optimize a network given nodes and candidate edges*

Description

Optimize a network given nodes and candidate edges

Usage

```
optimize_network(
  nodes,
  edges,
  budget,
  loop_fraction = 0.05,
  max_redundancy = 2
)
```

Arguments

`nodes`	Named numeric vector of node weights.
`edges`	Data frame with columns u, v, id, cost.
`budget`	Numeric budget for corridor costs.
`loop_fraction`	Fraction of budget reserved for loops.
`max_redundancy`	Max redundant edges per component.

Value

List with selected edge ids, component sizes/counts, and total cost.

`optimize_strategy` *Choose optimization strategy*

Description

Choose optimization strategy

Usage

```
optimize_strategy(strategy, nodes, edges, candidates, budget, ...)
```

Arguments

strategy	Strategy name.
nodes	Named numeric vector of patch sizes.
edges	Data frame with u, v, id, cost.
candidates	Candidate list (for circuit utility).
budget	Numeric budget.
...	Additional args forwarded to circuit utility selector.

Value

List with selected ids and stats.

---

patch_summary_from_labels

Summarize patch labels into a data frame

Description

Summarize patch labels into a data frame

Usage

```
patch_summary_from_labels(labels)
```

Arguments

labels	SpatRaster of patch labels.

Value

Data frame with patch_id, cell_count, area, x, y.

raster_mask_from_values *Create a logical mask from raster values*

Description

Create a logical mask from raster values

Usage

```
raster_mask_from_values(raster, values)
```

Arguments

raster	SpatRaster.
values	Numeric values to keep.

Value

Logical SpatRaster mask.

run_raster_analysis *Run TerraLink raster workflow*

Description

Run TerraLink raster workflow

Usage

```
run_raster_analysis(
  raster,
  patch_values,
  budget = NULL,
  budget_pixels = NULL,
  strategy = "most_connected_networks",
  min_patch_size = 10,
  min_corridor_width = 3,
  corridor_cell_assignment = "sum_total_network_area",
  max_search_distance = 100,
  obstacle_values = NULL,
  obstacle_ranges = NULL,
  allow_bottlenecks = FALSE,
  patch_connectivity = 4,
  units = "pixels",
```

```
  patch_ranges = NULL,
  allow_large = FALSE,
  max_pair_checks = 2e+06,
  max_candidates = 2e+05,
  verbose = 0,
  progress = FALSE,
  obstacle_strategy = c("error", "straight_line", "disable_obstacles"),
  pc_alpha = NULL,
  pc_cutoff = NULL,
  species_dispersal_distance = NULL,
  species_dispersal_kernel = HABITAT_AVAILABILITY_DEFAULT_KERNEL,
  min_patch_area_for_species = 0,
  patch_area_scaling = HABITAT_AVAILABILITY_DEFAULT_SCALING,
  mobility_detour_cap = 8,
  redundancy_method = "ime",
  keep_candidates = FALSE
)
```

Arguments

raster	SpatRaster or path to raster.
patch_values	Numeric values representing habitat.
budget	Total corridor budget (units defined by units).
budget_pixels	Back-compat alias for budget (pixels).
strategy	Strategy name. Canonical values are "most_connected_networks", "most_connected_networks_2", "largest_single_network", and "landscape_fluidity".
min_patch_size	Minimum patch size (units defined by units).
min_corridor_width	Minimum corridor width (units defined by units).
corridor_cell_assignment	Corridor cell assignment mode.
max_search_distance	Maximum search distance (units defined by units).
obstacle_values	Optional impassable raster values.
obstacle_ranges	Optional list of impassable ranges.
allow_bottlenecks	Whether to allow corridors to squeeze through gaps.
patch_connectivity	Connectivity for patch labeling (4 or 8).
units	Unit system: "pixels", "metric", or "imperial".
patch_ranges	Optional list of value ranges defining habitat.
allow_large	Allow processing very large rasters.

`max_pair_checks`
: Limit for candidate pair checks (prevents O(n^2) blowups).

`max_candidates`
: Limit for candidate corridors.

`verbose`
: Verbosity level (0-2).

`progress`
: Show progress bars.

`obstacle_strategy`
: Behavior when gdistance is unavailable and obstacles are provided.

`pc_alpha`
: Optional dispersal alpha used by Probability of Connectivity metrics.

`pc_cutoff`
: Optional cutoff distance used by Probability of Connectivity metrics.

`species_dispersal_distance`
: Species movement distance used by habitat-availability reporting.

`species_dispersal_kernel`
: Dispersal kernel for habitat availability.

`min_patch_area_for_species`
: Minimum patch area eligible for species metrics.

`patch_area_scaling`
: Patch-area scaling for habitat availability ("sqrt" or "log").

`mobility_detour_cap`
: Cap used by graph-based mobility/fluidity metrics.

`redundancy_method`
: Flow redundancy method ("ime" or "fri").

`keep_candidates`
: Whether to keep candidate list in the output.

Details

Raster inputs are funneled through TerraLink's vector corridor pipeline after habitat and impassable classes are polygonized, matching the current QGIS plugin workflow.

Value

List with patches, corridors, rasters, and summary.

`run_vector_analysis` *Run TerraLink vector workflow*

Description

Run TerraLink vector workflow

Usage

```
run_vector_analysis(
  patches,
  budget,
  strategy = "most_connected_networks",
  min_patch_size = NULL,
  min_corridor_width = 100,
  max_search_distance = 5000,
  obstacle_layers = NULL,
  obstacle_resolution = NULL,
  units = "metric",
  max_pair_checks = 2e+06,
  max_candidates = 2e+05,
  verbose = 0,
  progress = FALSE,
  obstacle_strategy = c("error", "straight_line", "disable_obstacles"),
  return_crs = c("input", "utm"),
  pc_alpha = NULL,
  pc_cutoff = NULL,
  species_dispersal_distance = NULL,
  species_dispersal_kernel = HABITAT_AVAILABILITY_DEFAULT_KERNEL,
  min_patch_area_for_species = 0,
  patch_area_scaling = HABITAT_AVAILABILITY_DEFAULT_SCALING,
  patch_quality_field = NULL,
  mobility_detour_cap = 8,
  redundancy_method = "ime",
  keep_candidates = FALSE
)
```

Arguments

patches	sf polygons (one feature per patch) or file path.
budget	Corridor budget (ha/ac).
strategy	Strategy name. Canonical values are "most_connected_networks", "most_connected_networks_2", "largest_single_network", and "landscape_fluidity".
min_patch_size	Minimum patch size (ha/ac).
min_corridor_width	Minimum corridor width (m/ft).
max_search_distance	Maximum search distance (m/ft).
obstacle_layers	Optional obstacle layers (sf or file paths).
obstacle_resolution	Raster resolution for obstacle routing.
units	"metric" or "imperial".
max_pair_checks	Limit for candidate pair checks.

max_candidates	Limit for candidate corridors.
verbose	Verbosity level (0-2).
progress	Show progress bars.
obstacle_strategy	Behavior when gdistance is unavailable and obstacles are provided.
return_crs	CRS for outputs ("input" or "utm").
pc_alpha	Optional dispersal alpha used by Probability of Connectivity metrics.
pc_cutoff	Optional cutoff distance used by Probability of Connectivity metrics.
species_dispersal_distance	Species movement distance used by habitat-availability reporting.
species_dispersal_kernel	Dispersal kernel for habitat availability.
min_patch_area_for_species	Minimum patch area eligible for species metrics.
patch_area_scaling	Patch-area scaling for habitat availability ("sqrt" or "log").
patch_quality_field	Optional numeric field used to weight patch quality in vector mode.
mobility_detour_cap	Cap used by graph-based mobility/fluidity metrics.
redundancy_method	Flow redundancy method ("ime" or "fri").
keep_candidates	Keep candidate list in output.

Value

List with corridors, networks, and summary.

---

score_edge_for_loops *Score a loop edge for shortcut value*

Description

Score a loop edge for shortcut value

Usage

```
score_edge_for_loops(graph, u, v, weight)
```

Arguments

graph	igraph graph.
u	First node id.
v	Second node id.
weight	Edge cost.

Value

Numeric score.

`select_circuit_utility` *Select corridors for the "Most Connectivity" strategy*

Description

Greedy ROI-based selector with dynamic rescoring, bridge seeding, and optional overlap checks.

Usage

```
select_circuit_utility(
  candidates,
  budget,
  get_patch_ids,
  get_pair_key,
  get_cost,
  get_base_roi,
  get_length,
  get_patch_size,
  overlap_ratio,
  global_overlap_ratio = NULL,
  overlap_obj,
  redundancy_distance_ok = NULL,
  overlap_reject_ratio = 0.3,
  global_overlap_reject_ratio = 0.6,
  max_prior_per_pair = 3,
  diminishing_base = 0.5,
  max_links_per_pair = Inf,
  enable_bridge_pairs = TRUE,
  bridge_max_per_patch = 25,
  distance_guard_for_primary = FALSE,
  global_overlap_for_primary = FALSE,
  parallel_dominance_ratio = 1.35,
  parallel_overlap_penalty_floor = 0.2,
  shortcut_ratio_high = 3,
  shortcut_ratio_mid = 1.5,
  shortcut_ratio_low = 1.5,
  shortcut_mult_high = 0.9,
  shortcut_mult_mid = 0.5,
  shortcut_mult_low = 0.1
)
```

Arguments

candidates	Iterable of candidate objects (data.frame rows or lists).
budget	Total corridor budget.
get_patch_ids	Function that returns patch ids for a candidate.
get_pair_key	Function that returns a sorted pair key for a candidate.
get_cost	Function that returns candidate cost.
get_base_roi	Function that returns candidate base ROI.
get_length	Function that returns candidate length for shortcut scoring.
get_patch_size	Function that returns patch size by id.
overlap_ratio	Function that returns overlap ratio vs prior objects.
global_overlap_ratio	Optional function that returns broader overlap ratio vs globally selected objects.
overlap_obj	Function that returns overlap object representation.
redundancy_distance_ok	Optional callback that can reject near-duplicate redundant corridors.
overlap_reject_ratio	Overlap ratio threshold for heavy redundancy penalty.
global_overlap_reject_ratio	Threshold for rejecting globally parallel candidates.
max_prior_per_pair	Maximum overlap objects retained per patch pair.
diminishing_base	Base for redundancy penalty when no shortcut context is available.
max_links_per_pair	Optional hard limit of selected corridors per patch pair.
enable_bridge_pairs	Whether to pre-seed bridge corridor pairs.
bridge_max_per_patch	Max candidates retained per bridge midpoint patch.
distance_guard_for_primary	Whether to apply distance guard to primary links.
global_overlap_for_primary	Whether to apply global-overlap reject to primary links.
parallel_dominance_ratio	Shortcut dominance threshold for parallel penalties.
parallel_overlap_penalty_floor	Floor multiplier for global-parallel penalties.
shortcut_ratio_high	High shortcut ratio threshold.
shortcut_ratio_mid	Mid shortcut ratio threshold.

shortcut_ratio_low
: Low shortcut ratio threshold.

shortcut_mult_high
: Multiplier when shortcut ratio is high.

shortcut_mult_mid
: Multiplier when shortcut ratio is mid.

shortcut_mult_low
: Multiplier when shortcut ratio is low.

Value

List with picks, selected_ids, and summary stats.

---

terralink_engine *Run TerraLink optimization on abstract nodes and edges*

Description

Run TerraLink optimization on abstract nodes and edges

Usage

```
terralink_engine(
  nodes,
  edges,
  budget,
  loop_fraction = 0.05,
  max_redundancy = 2
)
```

Arguments

nodes	Named numeric vector of patch weights.
edges	Data frame with columns u, v, id, cost.
budget	Numeric budget for corridor costs.
loop_fraction	Fraction of budget available for loops.
max_redundancy	Maximum redundant edges per component.

Value

List with selected edges and component summaries.

terralink_examples *Locate packaged TerraLink example scripts*

Description

Locate packaged TerraLink example scripts

Usage

```
terralink_examples(type = c("all", "raster", "vector"))
```

Arguments

type	Which example scripts to return: "all", "raster", or "vector".

Value

Character vector of absolute file paths.

Examples

```
terralink_examples()
terralink_examples("raster")
```

terralink_raster *Run TerraLink corridor analysis on raster data*

Description

Identifies habitat patches from a classified raster, builds candidate corridors between nearby patches, and selects an optimal corridor network under a budget constraint. Raster inputs are polygonized and routed through TerraLink's vector engine, matching the current QGIS plugin workflow.

Usage

```
terralink_raster(
  raster,
  patch_values = NULL,
  patch_ranges = NULL,
  budget = NULL,
  budget_pixels = NULL,
  strategy = "most_connected_networks",
  min_patch_size = 10,
  min_corridor_width = 3,
  corridor_cell_assignment = "sum_total_network_area",
  max_search_distance = 100,
```

```
  obstacle_values = NULL,
  obstacle_ranges = NULL,
  allow_bottlenecks = FALSE,
  patch_connectivity = 8,
  units = "pixels",
  allow_large = FALSE,
  max_pair_checks = 2e+06,
  max_candidates = 2e+05,
  verbose = 0,
  progress = FALSE,
  obstacle_strategy = c("error", "straight_line", "disable_obstacles"),
  pc_alpha = NULL,
  pc_cutoff = NULL,
  species_dispersal_distance = NULL,
  species_dispersal_kernel = HABITAT_AVAILABILITY_DEFAULT_KERNEL,
  min_patch_area_for_species = 0,
  patch_area_scaling = HABITAT_AVAILABILITY_DEFAULT_SCALING,
  mobility_detour_cap = 8,
  redundancy_method = "ime",
  output_dir = NULL,
  output_prefix = NULL,
  output_paths = NULL,
  write_outputs = FALSE,
  keep_candidates = FALSE
)
```

Arguments

raster	SpatRaster or file path to a single-band raster.
patch_values	Integer vector of cell values that represent habitat (e.g., c(1, 3)). At least one of patch_values or patch_ranges must be provided.
patch_ranges	Optional list of length-2 numeric vectors giving inclusive value ranges that define habitat (e.g., list(c(1, 3))).
budget	Total corridor budget. When units = "pixels" this is the number of corridor cells allowed. When units = "metric" or "imperial", this is the total corridor **area** in hectares or acres. A reasonable starting point is 5–20 percent of total habitat area.
budget_pixels	Back-compat alias for budget in pixel units. Use budget instead.
strategy	Character string selecting the optimization objective. One of "most_connected_networks" (default; Most Connected Networks A, maximizes total structurally connected habitat area), "most_connected_networks_2" (Most Connected Networks B, prioritizes high-value joins between existing components), "largest_single_network" (maximizes the single largest connected component), or "landscape_fluidity" (maximizes ease of movement and route redundancy).
min_patch_size	Numeric. Minimum patch size to include. In pixel units when units = "pixels" (number of cells), in hectares when units = "metric", or acres when units = "imperial". Patches smaller than this are dropped before analysis. Default: 10.

`min_corridor_width`
: Numeric. Minimum corridor width. In pixel units when `units = "pixels"` (cell widths), in meters when `units = "metric"`, or feet when `units = "imperial"`. Controls the buffer applied to corridor center-lines. Default: 3.

`corridor_cell_assignment`
: Character string controlling how corridor cells are valued in the output raster. One of `"sum_total_network_area"` (default; total network area), `"sum_direct_connected_patches"` (area of the two directly linked patches), or `"efficiency"` (cost-efficiency score).

`max_search_distance`
: Numeric. Maximum distance between patch edges to consider a candidate corridor. Same unit system as `min_corridor_width` (pixels / meters / feet). Increase this if few or no corridors are found. Default: 100.

`obstacle_values`
: Optional integer vector of raster cell values that represent impassable barriers (e.g., roads, water bodies).

`obstacle_ranges`
: Optional list of length-2 numeric vectors giving inclusive value ranges for obstacles.

`allow_bottlenecks`
: Logical. If `TRUE`, corridors narrower than `min_corridor_width` are still allowed when no wider path exists. Default: `FALSE`.

`patch_connectivity`
: Integer, 4 or 8. Pixel connectivity rule for grouping habitat cells into patches. 8 (default) includes diagonal neighbors; 4 uses only cardinal neighbors.

`units`
: Character string specifying the unit system: `"pixels"` (raster cell units), `"metric"` (hectares for area, meters for distance), or `"imperial"` (acres for area, feet for distance). Default: `"pixels"`.

`allow_large`
: Logical. Set to `TRUE` to allow processing rasters with more than 10 million cells. Default: `FALSE`.

`max_pair_checks`
: Integer. Upper limit on the number of patch pairs evaluated during candidate generation. Increase for landscapes with many patches; decrease if running out of memory. Default: 2,000,000.

`max_candidates`
: Integer. Upper limit on total candidate corridors retained. Default: 200,000.

`verbose`
: Integer verbosity level: 0 = silent, 1 = progress messages, 2 = detailed diagnostics. Default: 0.

`progress`
: Logical. Show progress bars during long operations. Default: `FALSE`.

`obstacle_strategy`
: Character string controlling behavior when obstacle values are provided but the **gdistance** package is not installed. One of `"error"` (default; stop with an informative error), `"straight_line"` (fall back to straight-line corridors, ignoring obstacles), or `"disable_obstacles"` (silently drop obstacle data).

`pc_alpha`
: Optional dispersal alpha used by Probability of Connectivity metrics.

`pc_cutoff`
: Optional cutoff distance used by Probability of Connectivity metrics.

species_dispersal_distance
: Numeric. Typical movement distance for the focal species, in the same distance units as the analysis (pixels / meters / feet). Used by habitat-availability metrics. If NULL (default), the max_search_distance value is used as a proxy.

species_dispersal_kernel
: Character string. Dispersal probability kernel. Currently only "exponential" is supported (default).

min_patch_area_for_species
: Numeric. Minimum patch area (in analysis area units) for a patch to be included in species-level habitat availability calculations. Default: 0.

patch_area_scaling
: Character string controlling how patch area is transformed before weighting in habitat availability calculations. "sqrt" (default) applies square-root scaling, giving moderate weight to large patches. "log" applies logarithmic scaling (log(1 + area)), reducing the influence of very large patches.

mobility_detour_cap
: Numeric. Maximum detour ratio used by graph-based fluidity metrics. Controls how much longer an indirect route can be relative to the straight-line distance before it is considered non-functional. Default: 8.

redundancy_method
: Character string selecting the flow redundancy calculation method. "ime" (default) uses Inverse Mean Effective-resistance, measuring how many alternative routes exist. "fri" uses the Flow Redundancy Index, an alternative based on current-flow theory.

output_dir
: Optional character path. Directory for writing output files when write_outputs = TRUE.

output_prefix
: Optional character string prepended to output file names.

output_paths
: Optional named list of explicit output file paths, overriding the default naming convention.

write_outputs
: Logical. If TRUE, write output rasters and CSV files to disk. Default: FALSE.

keep_candidates
: Logical. If TRUE, include the full candidate corridor table in the result (useful for debugging). Default: FALSE.

Value

An object of class "terralink_result" (a list) with the following elements:

- corridors: Data frame or sf object of selected corridor geometries with columns patch1, patch2, corridor_area, corridor_length, connected_area, and network_area.
- patches: SpatRaster of labeled patch cells (raster mode).
- patch_table: Data frame of patch attributes (id, area, centroid coordinates).
- networks: sf object of connected network polygons (one feature per component of patches + corridors).
- corridor_raster: SpatRaster where corridor cells are assigned values according to corridor_cell_assignment.
- contiguous_raster: SpatRaster labeling each contiguous patch-corridor network.

- strategy: The strategy key that was used.
- summary: Named list with run overview including budget_total, budget_used, corridors_used, candidate_edges, patches, strategy, units.
- metrics: Named list of PRE/POST landscape metrics. Each metric has a _pre (before corridors) and _post (after corridors) value. Key metrics: total_connected_habitat_area, largest_network_area, habitat_availability, mean_effective_resistance (lower is better), mesh_norm, lcc, pc, flow_redundancy, strategic_mobility, landscape_fluidity, composite_connectivity.
- metrics_report: Character vector with a human-readable PRE/POST metrics table. Print with cat(result$metrics_report, sep = "\n").
- strategy_stats: Named list of strategy-specific optimization statistics (e.g., primary vs. redundant links).
- mode: Character string "raster".
- inputs: Named list echoing key input parameters.
- run_stats: Named list with elapsed_s, candidate_edges, candidate_pairs.
- warnings: Character vector of any warnings raised during the run.
- diagnostics: List of diagnostic messages (e.g., why no corridors were selected).

The object has print(), summary(), and plot() methods.

Parameter guidance

- **budget**: A practical starting point is often around 5–20 percent of total habitat area. Run several budget levels and compare PRE/POST metrics.
- **min_patch_size**: Use this to exclude patches too small to function as habitat in your planning context. For raster mode, 5–20 pixels is a common starting range; for real landscapes, 1–10 ha can be a reasonable first pass.
- **min_corridor_width**: Should reflect the minimum width for species movement. Depending on species and landscape context, 30–100 m is a common starting range for terrestrial mammals and 10–30 m for some birds.
- **max_search_distance**: Should be at or above the maximum distance the focal species can cross non-habitat. 500–5000 m is a common starting range; increase if 0 corridors are generated.
- **species_dispersal_distance**: Set to the focal species' typical natal or daily movement range. This directly affects the habitat-availability metrics.

Examples

```
r <- terra::rast(
  nrows = 6, ncols = 6,
  xmin = 0, xmax = 600,
  ymin = 0, ymax = 600,
  crs = "EPSG:3857"
)
vals <- rep(0, terra::ncell(r))
vals[c(1, 2, 7, 8, 29, 30, 35, 36)] <- 1
```

```
terra::values(r) <- vals

result <- terralink_raster(
  raster = r,
  patch_values = 1,
  budget = 15,
  min_patch_size = 2,
  min_corridor_width = 1,
  max_search_distance = 12,
  units = "pixels"
)
result$summary

# Access PRE/POST metrics
result$metrics$largest_network_area_pre
result$metrics$largest_network_area_post
```

`terralink_run` *Run TerraLink with a single entry point*

Description

Run TerraLink with a single entry point

Usage

```
terralink_run(mode = c("raster", "vector"), input, ...)
```

Arguments

`mode`	"raster" or "vector".
`input`	Raster path/SpatRaster or sf/path.
`...`	Parameters forwarded to terralink_raster or terralink_vector.

Value

Result list.

terralink_sample_data *Locate packaged TerraLink sample data files*

Description

Locate packaged TerraLink sample data files

Usage

```
terralink_sample_data(
 type = c("all", "raster", "vector", "obstacle", "synthetic_raster", "synthetic_vector",
    "synthetic_obstacle")
)
```

Arguments

type	Which sample data path to return: "all", "raster", "vector", "obstacle", "synthetic_raster", "synthetic_vector", or "synthetic_obstacle".

Value

For "all", a named character vector of absolute file paths. For other values, a single absolute file path (character scalar) or character(0) when unavailable.

Examples

```
terralink_sample_data()
terralink_sample_data("raster")
```

terralink_vector *Run TerraLink corridor analysis on vector patches*

Description

Builds candidate corridors between polygon habitat patches and selects an optimal corridor network under a budget constraint. This is the native TerraLink workflow and is usually the better choice when planning inputs are already polygon features.

Usage

```
terralink_vector(
  patches,
  budget,
  strategy = "most_connected_networks",
  min_patch_size = NULL,
  min_corridor_width = 100,
```

```
  max_search_distance = 5000,
  obstacle_layers = NULL,
  obstacle_resolution = NULL,
  units = "metric",
  max_pair_checks = 2e+06,
  max_candidates = 2e+05,
  verbose = 0,
  progress = FALSE,
  obstacle_strategy = c("error", "straight_line", "disable_obstacles"),
  return_crs = c("input", "utm"),
  pc_alpha = NULL,
  pc_cutoff = NULL,
  species_dispersal_distance = NULL,
  species_dispersal_kernel = HABITAT_AVAILABILITY_DEFAULT_KERNEL,
  min_patch_area_for_species = 0,
  patch_area_scaling = HABITAT_AVAILABILITY_DEFAULT_SCALING,
  patch_quality_field = NULL,
  mobility_detour_cap = 8,
  redundancy_method = "ime",
  output_dir = NULL,
  output_prefix = NULL,
  output_paths = NULL,
  write_outputs = FALSE,
  keep_candidates = FALSE
)
```

Arguments

patches	sf object with polygon geometry (one row per patch), or a file path to a GeoPackage / Shapefile. The CRS should be projected (e.g., UTM) so that area and distance calculations are meaningful.
budget	Numeric. Total corridor area budget in hectares (`units = "metric"`) or acres (`units = "imperial"`). A reasonable starting point is 5–20 percent of your total patch area.
strategy	Character string selecting the optimization objective. One of `"most_connected_networks"` (default; Most Connected Networks A, maximizes total structurally connected habitat area), `"most_connected_networks_2"` (Most Connected Networks B, prioritizes high-value joins between existing components), `"largest_single_network"` (maximizes the single largest connected component), or `"landscape_fluidity"` (maximizes ease of movement and route redundancy).
min_patch_size	Numeric. Minimum patch area in hectares (`"metric"`) or acres (`"imperial"`). Patches smaller than this are dropped. Default: `NULL` (no filter).
min_corridor_width	Numeric. Minimum corridor width in meters (`"metric"`) or feet (`"imperial"`). Controls the buffer applied to corridor center-lines. Typical values: 30–100 m for terrestrial mammals. Default: 100.
max_search_distance	Numeric. Maximum edge-to-edge distance (meters or feet) between patches to

consider a candidate corridor. Increase if few or no corridors are generated. Default: 5000.

`obstacle_layers`
: Optional sf object or file path to polygon barriers (roads, water bodies). Requires the **gdistance** package for shortest-path routing around obstacles.

`obstacle_resolution`
: Numeric. Raster cell size (in CRS units) used to rasterize obstacles for shortest-path routing. Smaller values give more accurate routing but increase computation time.

`units`
: Character string: `"metric"` (hectares / meters, default) or `"imperial"` (acres / feet).

`max_pair_checks`
: Integer. Upper limit on patch pairs evaluated. Default: 2,000,000.

`max_candidates`
: Integer. Upper limit on candidate corridors retained. Default: 200,000.

`verbose`
: Integer verbosity level: 0 = silent, 1 = progress, 2 = detailed. Default: 0.

`progress`
: Logical. Show progress bars. Default: `FALSE`.

`obstacle_strategy`
: Character string controlling behavior when obstacles are provided but **gdistance** is not installed. One of `"error"` (default; stop with an error), `"straight_line"` (fall back to straight-line corridors), or `"disable_obstacles"` (silently ignore obstacles).

`return_crs`
: Character string controlling the output CRS. `"input"` (default) returns outputs in the same CRS as the input patches. `"utm"` returns outputs in the UTM zone used internally.

`pc_alpha`
: Optional dispersal alpha used by Probability of Connectivity metrics.

`pc_cutoff`
: Optional cutoff distance used by Probability of Connectivity metrics.

`species_dispersal_distance`
: Numeric. Typical movement distance for the focal species in meters (`"metric"`) or feet (`"imperial"`). Used by habitat-availability metrics. If `NULL` (default), `max_search_distance` is used as a proxy.

`species_dispersal_kernel`
: Character string. Dispersal probability kernel. Currently only `"exponential"` is supported (default).

`min_patch_area_for_species`
: Numeric. Minimum patch area (in analysis area units) for inclusion in species-level metrics. Default: 0.

`patch_area_scaling`
: Character string controlling how patch area is transformed before weighting. `"sqrt"` (default) applies square-root scaling, giving moderate weight to large patches. `"log"` applies logarithmic scaling, reducing the influence of very large patches.

`patch_quality_field`
: Optional character string naming a numeric column in `patches` that provides a 0–1 quality weight per patch (e.g., habitat suitability). Patches with higher quality contribute more to connectivity metrics.

`mobility_detour_cap`
: Numeric. Maximum detour ratio for fluidity metrics. Controls how much longer an indirect route can be before it is considered non-functional. Default: 8.

`redundancy_method`
: Character string selecting the flow redundancy method. `"ime"` (default) uses Inverse Mean Effective-resistance. `"fri"` uses the Flow Redundancy Index.

`output_dir`
: Optional output directory for `write_outputs`.

`output_prefix`
: Optional name prefix for output files.

`output_paths`
: Optional named list of explicit output file paths.

`write_outputs`
: Logical. Write GeoPackage and CSV outputs to disk. Default: `FALSE`.

`keep_candidates`
: Logical. Include full candidate table in result. Default: `FALSE`.

Value

An object of class `"terralink_result"` (a list) with the following elements:

- `corridors`: sf object of selected corridors with columns `patch1`, `patch2` (endpoint patch IDs), `corridor_area` (ha or ac), `corridor_length` (m or ft), `connected_area`, `network_area`, and geometry.
- `patches`: sf object of patches used in the analysis, with area and centroid attributes.
- `networks`: sf object of connected network polygons (one feature per component of patches + corridors).
- `summary`: Named list including `budget_total`, `budget_used`, `corridors_used`, `candidate_edges`, `patches`, `raw_patches`, `filtered_out`, `primary_links`, `redundant_links`, `strategy`, `units`.
- `metrics`: Named list of PRE/POST landscape connectivity metrics. Every metric has a `_pre` and `_post` value. Key metrics: `total_connected_habitat_area`, `largest_network_area`, `habitat_availability`, `mean_effective_resistance` (lower is better), `mesh_norm`, `lcc`, `pc`, `flow_redundancy`, `strategic_mobility`, `landscape_fluidity`, `composite_connectivity`.
- `metrics_report`: Character vector with a human-readable PRE/POST table. Print with `cat(result$metrics_report, sep = "\n")`.
- `strategy_stats`: Named list of strategy-specific statistics.
- `mode`: Character string `"vector"`.
- `inputs`: Named list echoing key input parameters.
- `run_stats`: Named list with `elapsed_s`, `candidate_edges`, `candidate_pairs`.
- `warnings`: Character vector of warnings.
- `diagnostics`: List of diagnostic messages.

The object has `print()`, `summary()`, and `plot()` methods.

Parameter guidance

- **budget**: A practical starting point is often around 5–20 percent of total patch area. Run multiple budgets and compare PRE/POST metrics to find the point of diminishing returns.

- **min_corridor_width**: Depending on species and context, 30–100 m can be a useful starting range for mammals and 10–30 m for some small birds.
- **max_search_distance**: 500–5000 m is a common starting range. Increase if 0 corridors are generated.
- **species_dispersal_distance**: Set to the focal species' typical natal or daily movement range. Directly affects habitat-availability metrics.

Examples

```
p1 <- sf::st_polygon(list(rbind(c(0, 0), c(0, 10), c(10, 10), c(10, 0), c(0, 0))))
p2 <- sf::st_polygon(list(rbind(c(30, 0), c(30, 10), c(40, 10), c(40, 0), c(30, 0))))
patches <- sf::st_sf(id = 1:2, geometry = sf::st_sfc(p1, p2), crs = 32618)

if (identical(Sys.getenv("NOT_CRAN"), "true")) {
  result <- terralink_vector(
    patches = patches,
    budget = 1,
    min_patch_size = 0.001,
    min_corridor_width = 5,
    max_search_distance = 200,
    units = "metric"
  )
  result$summary

  # Access PRE/POST metrics
  result$metrics$largest_network_area_pre
  result$metrics$largest_network_area_post

  # Print the full metrics report
  cat(result$metrics_report, sep = "\n")
}
```

`UnionFind` *Union-Find data structure*

Description

Union-Find data structure

Union-Find data structure

Methods

initialize Create a new UnionFind.

find Find root of a node with path compression.

union Union two nodes; returns TRUE if merged.

get_size Get component size for a node.

get_count Get component count for a node.

Public fields

`parent` Environment mapping nodes to parents.

`size` Environment mapping roots to component sizes.

`count` Environment mapping roots to component counts.

Methods

Public methods:

- `UnionFind$new()`
- `UnionFind$find()`
- `UnionFind$union()`
- `UnionFind$get_size()`
- `UnionFind$get_count()`
- `UnionFind$clone()`

Method `new()`:

Usage:

```
UnionFind$new()
```

Method `find()`:

Usage:

```
UnionFind$find(x)
```

Arguments:

`x` Node id for lookup operations.

`x` Node id for lookup operations.

Method `union()`:

Usage:

```
UnionFind$union(a, b)
```

Arguments:

`a` Node id for union operations.

`a` Node id for union operations.

`b` Node id for union operations.

`b` Node id for union operations.

Method `get_size()`:

Usage:

```
UnionFind$get_size(x)
```

Arguments:

`x` Node id for lookup operations.

`x` Node id for lookup operations.

Method `get_count()`:

Usage:

```
UnionFind$get_count(x)
```

Arguments:

x Node id for lookup operations.

x Node id for lookup operations.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
UnionFind$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`write_terralink_raster_outputs` *Write raster outputs to disk*

Description

Write raster outputs to disk

Usage

```
write_terralink_raster_outputs(
  result,
  output_dir,
  prefix = NULL,
  overwrite = TRUE,
  output_paths = list()
)
```

Arguments

`result`	Result list from terralink_raster.
`output_dir`	Directory to write outputs.
`prefix`	Optional name prefix for outputs.
`overwrite`	Whether to overwrite existing files.
`output_paths`	Named list of explicit file paths to override defaults.

Value

Named list of written file paths.

`write_terralink_vector_outputs` *Write vector outputs to disk*

Description

Write vector outputs to disk

Usage

```
write_terralink_vector_outputs(
  result,
  output_dir,
  prefix = NULL,
  overwrite = TRUE,
  output_paths = list()
)
```

Arguments

`result`	Result list from terralink_vector.
`output_dir`	Directory to write outputs.
`prefix`	Optional name prefix for outputs.
`overwrite`	Whether to overwrite existing files.
`output_paths`	Named list of explicit file paths to override defaults.

Value

Named list of written file paths.

Index