Package ‘spacemodR’

February 20, 2026

Title Workflow for Environmental Risk Assessment: Habitat, Food Web, Dispersal, Exposure and Risk

Version 0.1.3

Description A set of tools dedicated to modeling food web transfer based on an initial ground raster. It provides a directed acyclic graph structure for a set of rasters representing the flow of elements (e.g., food, energy, contaminants). It also includes tools for working with dispersal algorithms, enabling the combination of flux data with population movement.

License MIT + file LICENSE

Imports ggplot2, httr, sf, terra

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat (>= 3.0.0), JuliaCall

Config/testthat/edition 3

VignetteBuilder knitr

Depends R (>= 4.1.0)

NeedsCompilation no

Author Virgile Baudrot [aut, cre],
Léa Bariod [ctb],
Clémentine Fritsch [ctb],
Renaud Scheifler [ctb]

Maintainer Virgile Baudrot <`virgile.baudrot@qonfluens.com`>

Repository CRAN

Date/Publication 2026-02-20 08:10:03 UTC

Contents

`add_habitat` *Add habitat zones to a Habitat object*

Description

Add habitat zones to a Habitat object

Usage

```
add_habitat(hab, sf_data, weight = 1, ...)
```

Arguments

`hab`	A Habitat object
`sf_data`	An sf object containing geometries to add
`weight`	A weight to apply on the habitat feature. Can then be used as resistance for dispersal functions for instance.
`...`	Additional arguments

Value

A Habitat object with new geometries added as habitat

`add_link` *Add links to a trophic table*

Description

Adds one or several directed links to a trophic_tbl object.

Usage

```
add_link(tbl, from, to, weight = 1)
```

Arguments

`tbl`	A `trophic_tbl` object.
`from`	A single character string indicating the source node.
`to`	A character vector indicating target nodes.
`weight`	A numeric vector of weights associated with each link. If a single value is provided, it is recycled to match the length of `to`.

Details

The function performs several checks:

- `from` must be a scalar character string
- `to` must be a character vector
- Links must be unique
- Self-loops (from == to) are forbidden
- The resulting graph must remain acyclic

Value

A validated `trophic_tbl` object with the new links added.

Examples

```
net <- trophic() |>
  add_link("a", "b", weight = 1)
```

`add_nohabitat` *Add non-habitat zones to a Habitat object*

Description

Add non-habitat zones to a Habitat object

Usage

```
add_nohabitat(hab, sf_data, ...)
```

Arguments

`hab`	A Habitat object
`sf_data`	An sf object containing geometries to add
`...`	Additional arguments

Value

A Habitat object with new geometries added as non-habitat

`compute_dispersal` *Compute dispersal or spread map (Generic Engine)*

Description

Low-level function to apply spatial processing (Convolution or External algorithms) to a raster. It handles NA masking and method dispatch to Julia if necessary.

Usage

```
compute_dispersal(x, method = "convolution", options = list(), mask = NULL)
```

Arguments

`x`	SpatRaster. Source layer to disperse.
`method`	Character. "convolution" or "omniscape" (placeholder).
`options`	List. Parameters (e.g., ‘kernel‘ for convolution).
`mask`	SpatRaster (optional). A mask to apply after dispersion (e.g., maintain original NA structure).

Value

A `SpatRaster` object containing the dispersed values.

`compute_kernel` *Create a 2D Gaussian motion kernel as a SpatRaster*

Description

Create a 2D Gaussian motion kernel as a SpatRaster

Usage

```
compute_kernel(radius, GSD, size_std = 1.5)
```

Arguments

`radius`	Numeric, std of the distribution in meters
`GSD`	Numeric, ground sampling distance in meters per pixel
`size_std`	Numeric, how many std to extend kernel on each side

Value

A matrix defining the kernel

`dispersal` *Disperse a species or variable over the landscape*

Description

Applies a dispersal mechanism to a specific layer of the ‘spacemodel‘ object. This function acts as a wrapper around `compute_dispersal` to handle the ‘spacemodel‘ class structure.

Usage

```
dispersal(
  spacemodel,
  layer = 1,
  method = "convolution",
  method_option = list()
)
```

Arguments

`spacemodel`	A `spacemodel` object.
`layer`	Character or Integer. The name or index of the layer to disperse (e.g., "Fox", 1).
`method`	Character. The dispersal method to use. Options are: • `"convolution"` (default): Uses a moving window (kernel). • `"omniscape"`: Uses Circuit Theory (via Julia and Omniscape.jl).
`method_option`	A list of parameters specific to the chosen method: • For `"convolution"`: must contain `kernel` (a matrix). • For `"omniscape"`: must contain `resistance` (SpatRaster) and `radius` (numeric).

Value

The `spacemodel` object with the specified layer updated with dispersed values.

See Also

`compute_dispersal`

Examples

```
## Not run:
# 1. Convolution example
my_kernel <- matrix(1, nrow=3, ncol=3)
sm_updated <- dispersal(sm, layer = "Predator", method = "convolution",
                        method_option = list(kernel = my_kernel))

# 2. Omniscape example (requires Julia)
sm_updated <- dispersal(sm, layer = "Predator", method = "omniscape",
                        method_option = list(resistance = res_map, radius = 10))

## End(Not run)
```

`flux` *Apply trophic flux from a resource layer*

Description

Computes the contribution of a resource raster to a consumer layer using the normalized weight and a flux function stored in the trophic table.

Usage

```
flux(raster, intakes, from, to)
```

Arguments

`raster`	A `SpatRaster` (or similar) representing exposure from the resource.
`intakes`	A `trophic_tbl` object that must contain a column `normalized_weight` and a column `flux`.
`from`	Character string, name of the source node.
`to`	Character string, name of the target node.

Details

The function extracts the link corresponding to `from -> to` from the trophic table and applies:

1. the associated flux function to the raster values
2. the normalized weight of the link

An error is thrown if no such link exists.

Value

A raster object with transformed values.

`get_departements_for_roi` *Identify codes of departments intersecting a region of interest*

Description

This function takes a region of interest (ROI) as an ‘sf‘ or ‘sfc‘ object and returns the codes of departments intersecting this region.

Usage

```
get_departements_for_roi(roi)
```

Arguments

`roi`	A spatial object of type ‘sf‘ or ‘sfc‘ representing the region of interest. Use only the first geometry (polygon).

Value

A character vector containing the codes of departments intersecting the region of interest.

Examples

```
library(sf)
roi <- sf::st_as_sfc(sf::st_bbox(
  c(xmin = 600000, ymin = 6600000, xmax = 650000, ymax = 6650000),
  crs = 2154)
)
departments <- get_departements_for_roi(roi)
```

get_ocsge_data_fgb *Efficiently retrieve OCS GE data from a remote FlatGeobuf*

Description

This function retrieves OCS GE (Land Cover) data for a specific Region of Interest (ROI) directly from a remote FlatGeobuf (.fgb) file hosted on a server (e.g., S3).

It leverages GDAL's virtual file system ('/vsicurl/') and the spatial indexing capabilities of FlatGeobuf to download only the data chunks intersecting the bounding box of the ROI, making it highly efficient for large datasets.

Usage

```
get_ocsge_data_fgb(roi, fgb_url)
```

Arguments

`roi`	An `sf` object defining the Region Of Interest. It can be in any projection, but will be transformed to EPSG:2154 (Lambert-93) internally.
`fgb_url`	Character string. The public URL to the remote '.fgb' file.

Details

The function performs the following steps:

1. Transforms the `roi` to Lambert-93 (EPSG:2154).
2. calculates the bounding box of the `roi`.
3. Uses `sf::st_read` with a WKT filter to fetch only relevant features from the remote file.
4. Applies a precise geometric intersection ('st_intersection') to clip the data to the exact shape of the `roi`.

Value

An `sf` object containing the OCS GE polygons intersected by the ROI, projected in Lambert-93 (EPSG:2154).

Note

This function requires a working internet connection and GDAL support for the FlatGeobuf driver and network capabilities (vsicurl).

Examples

```
## Not run:
  library(sf)

  # 1. Define a Region of Interest (ROI)
  # Example: A small bounding box in France
  my_roi <- st_as_sf(data.frame(
    lon = c(2.3, 2.4, 2.4, 2.3, 2.3),
    lat = c(48.8, 48.8, 48.9, 48.9, 48.8)
  ), coords = c("lon", "lat"), crs = 4326)

  # 2. URL to the remote FlatGeobuf file
  # (Replace with the actual URL of your OCS GE bucket)
  url_fgb <- "https://example.com/data/ocsge_grand_est.fgb"

  # 3. Fetch data
  ocsge_data <- get_ocsge_data_fgb(roi = my_roi, fgb_url = url_fgb)

  # 4. Check result
  print(ocsge_data)
  plot(st_geometry(ocsge_data))

## End(Not run)
```

`habitat` *Create a Habitat object*

Description

Create a spatial Habitat object based on an optional `sf` data.frame. If no geometry is provided, creates an empty Habitat object. The object has columns:

- `habitat`: logical, TRUE/FALSE
- `weight`: numeric
- `geometry`: sfc geometry

Usage

```
habitat(geometry = NULL, habitat = NULL, weight = NULL)
```

Arguments

`geometry`	An object of class `sf` or `sfc`. Optional, default is empty.
`habitat`	Logical vector indicating habitat presence. Default is logical(0).
`weight`	Numeric vector of weights. Default is numeric(0).

Value

An object of class `habitat`, inheriting from `sf` and `data.frame`.

Examples

```
library(sf)

# Empty habitat
h <- habitat()
h

# Habitat with geometries
geom <- st_sfc(
  st_point(c(0, 0)),
  st_point(c(1, 1)),
  crs = 4326
)

hab <- habitat(
  geometry = geom,
  habitat = c(TRUE, FALSE),
  weight = c(0.8, 0)
)
hab
```

`habitat_raster` *Create a raster from a habitat object*

Description

Create a raster from a habitat object

Usage

```
habitat_raster(ground_raster, habitat)
```

Arguments

`ground_raster`	SpatRaster (reference grid).
`habitat`	habitat object inherited from `sf` data.frame class.

Value

SpatRaster (weight and NA in nogo)

`intake` *Constructor for Intake Parameters*

Description

Creates and configures the trophic flux table with a simplified syntax.

Usage

```
intake(x, ..., default = NULL, normalize = TRUE)
```

Arguments

`x`	A ‘spacemodel‘ object or a ‘trophic_tbl‘.
`...`	Flux definitions. - Key ‘"Target"‘ (e.g., ‘"Fox" = 0.5‘): applies to all links pointing to Fox. - Key ‘"Source -> Target"‘ (e.g., ‘"Soil -> Worm" = 0.8‘): targets a specific link. Values can be numeric (linear coefficient), formulas, or functions.
`default`	The default function for unspecified links (default is identity).
`normalize`	Logical. Whether to normalize diet weights (default TRUE).

Value

A ‘trophic_tbl‘ with a configured ‘flux‘ column.

`is_cyclic` *Test if a directed graph is cyclic*

Description

Implements Kahn’s algorithm to detect cycles in a directed graph.

Usage

```
is_cyclic(df)
```

Arguments

`df`	A data.frame with columns `from` and `to`.

Value

Logical. TRUE if the graph contains at least one cycle.

Examples

```
df <- data.frame(from=c("A","B"), to=c("B","A"))
is_cyclic(df)
```

`load_raster_extdata` *load raster file from internal data*

Description

A raster of the (example).

Usage

```
load_raster_extdata(path_name)
```

Arguments

`path_name`	Path of the raster (.tiff) file to download.

Format

An object ‘SpatRaster‘ (package terra).

Value

A `SpatRaster` object

Source

Internal data package.

See Also

`rast`

`lower_neighbors` *Get resource layers for a given trophic layer*

Description

Returns the upstream neighbors (prey/resources) of a given layer in a trophic graph.

Usage

```
lower_neighbors(trophic_tbl, layer)
```

Arguments

`trophic_tbl`	A `trophic_tbl` object
`layer`	Name of the layer (string) in the spacemodel.

Value

A character vector of names of layers that are resources for `layer`.

`normalize_weights` *Normalize weights of a trophic table*

Description

Adds a new column `normalized_weight` to a `trophic_tbl` object so that, for each target node (`to`), the sum of incoming weights equals 1.

Usage

```
normalize_weights(tbl)
```

Arguments

`tbl`	A `trophic_tbl` object.

Details

For every unique value in the `to` elements of the `link` column, the function divides each corresponding weight by the total weight of all links pointing to that same node.

Nodes with no incoming links are left unchanged.

Value

A `trophic_tbl` object with an additional column `normalized_weight`.

Examples

```
net <- trophic() |>
  add_link("a", "b", weight = 2) |>
  add_link("c", "b", weight = 3)

net_norm <- normalize_weights(net)
```

`ocsge_metaleurop` *SF object defining very simplified OCS-GE soil cover metaleurop*

Description

Simple feature collection with 9 features and 11 fields. Projected CRS: RGF93 v1 / Lambert-93.

Usage

```
data(ocsge_metaleurop)
```

Format

An object of class `sf` (inherits from `data.frame`) with 9 rows and 12 columns.

Examples

```
data(ocsge_metaleurop)
```

`ocsge_species_dict` *Valued weight between OCSGE layer and species*

Description

Valued weight between OCSGE layer and species

Usage

```
data(ocsge_species_dict)
```

Format

An object of class `data.frame` with 13090 rows and 6 columns.

Examples

```
data(ocsge_species_dict)
```

`plot.trophic_tbl` *Plot a trophic table*

Description

Creates a simple graphical representation of a trophic network using ggplot2.

Usage

```
## S3 method for class 'trophic_tbl'
plot(x, shift = TRUE, ...)
```

Arguments

`x`	A `trophic_tbl` object.
`shift`	To shift x_axis between trophic level and avoid the potential overlapping of arrows.
`...`	Additional arguments (not used, for S3 consistency).

Details

Nodes are positioned according to their trophic level:

- The y-axis represents trophic levels
- Nodes of the same level are placed on the same horizontal line
- The x-axis positions are assigned sequentially (0, 1, 2, ...)

Directed links are drawn from lower to higher trophic levels using arrows.

Value

A ggplot object.

Examples

```
net <- trophic() |>
  add_link("a", "b") |>
  add_link("b", "c")

plot(net)
```

raster_stack *Create a RasterStack from a list of rasters and names*

Description

This function creates a 'SpatRaster' stack from a list of rasters and assigns unique names to each layer.

Usage

```
raster_stack(raster_list, names = NULL)
```

Arguments

raster_list	A list of 'SpatRaster' objects or file paths to raster files.
names	A character vector of unique names for each raster layer in the stack.

Details

The function checks that the length of 'raster_list' matches the length of 'names', and that all names are unique. If not, it stops with an error.

Value

A SpatRaster object with named layers.

Examples

```
# Example with terra rasters
library(terra)
r1 <- rast(nrows=10, ncols=10, vals=1:100)
r2 <- rast(nrows=10, ncols=10, vals=101:200)
raster_stack(list(r1, r2), c("layer1", "layer2"))
```

ref_ocsge *Nomenclature of OCS-GE soil cover*

Description

Nomenclature of OCS-GE soil cover

Usage

```
data(ref_ocsge)
```

Format

An object of class `data.frame` with 14 rows and 4 columns.

Examples

```
data(ref_ocsge)
```

`roi_metaleurop` *SF object defining ROI metaleurop*

Description

Simple feature collection with 1 feature and 1 field. Geodetic CRS: WGS 84.

Usage

```
data(roi_metaleurop)
```

Format

An object of class `sf` (inherits from `data.frame`) with 1 rows and 2 columns.

Examples

```
data(roi_metaleurop)
```

`sf_micromammals` *DataBase of collected MicroMammals species*

Description

DataBase of collected MicroMammals species

Usage

```
data(sf_micromammals)
```

Format

An object of class `sf` (inherits from `tbl_df`, `tbl`, `data.frame`) with 1426 rows and 27 columns.

Examples

```
data(sf_micromammals)
```

spacemodel *Create a spacemodel object*

Description

Constructor for the ‘spacemodel‘ class. This function combines spatial data (a raster stack) and ecological data (a trophic table) into a single object used for modelling.

Usage

```
spacemodel(raster_stack, trophic_tbl)
```

Arguments

`raster_stack`	A `SpatRaster` object (multi-layer stack) representing the spatial distribution of the species or groups.
`trophic_tbl`	An object of class `trophic_tbl` containing the ecological parameters and properties of the species.

Details

The function performs several checks to ensure data consistency:

- Verifies that `raster_stack` is a `SpatRaster`.
- Verifies that `trophic_tbl` is a `trophic_tbl` object.
- Ensures the number of raster layers matches the number of levels in the trophic table.
- Ensures that the names of the raster layers match the names in the trophic table.

Value

A `SpatRaster` object with the following additional attributes:

- `trophic_tbl`: The `trophic_tbl` object passed as input.
- `spacemodel`: A logical flag set to `TRUE`, indicating this raster is part of a spacemodel.

See Also

`rast`, `trophic_tbl`

`transfer` *Transfer (food, contaminant) across trophic levels*

Description

Computes the transfer through a trophic network from lower to higher trophic levels using spatial spreading and intake functions.

Usage

```
transfer(
  spacemodel,
  kernels,
  intakes = NULL,
  exposure_weighting = "local",
  verbose = FALSE
)
```

Arguments

`spacemodel`	A named list of spatial layers (e.g. `SpatRaster` objects). Must contain an attribute `trophic_tbl` of class `trophic_tbl`.
`kernels`	A list of kernel parameters for each layer.
`intakes`	A `trophic_tbl` object (or compatible table) containing normalized weights and flux functions for each trophic link.
`exposure_weighting`	Character. Defines how the realized exposure is calculated based on the predator's presence. Options are: • `"local"` (Default): Weight exposure by the local habitat value ('predator_habitat'). Assumes the predator's intake is strictly proportional to the habitat quality/density of the pixel where it resides. • `"diffuse"`: Weight exposure by a dispersed habitat kernel. Represents a "neighborhood" effect where the predator's presence is smoothed over its home range. Useful to avoid edge effects where a predator on a poor pixel surrounded by good habitat would otherwise have 0 exposure. • `"potential"`: No weighting. Returns the pure environmental offer (potential exposure) regardless of the predator's density. Useful for identifying risk hotspots.
`verbose`	Logical. If `TRUE`, prints progress information.

Details

The function processes layers in ascending trophic order, as defined by the `level` attribute of the trophic table.

For each layer:

1. Resources (lower neighbors) are identified.
2. Concentration from each resource is spatially spread using `spread()`.
3. Intake is computed using `intake()`.
4. Contributions from all resources are summed.

The function assumes that intake weights are already normalized so that, for each consumer, the sum of contributions from all resources equals 1.

Value

A named ‘spacemodel‘ object as a list of spatial layers representing values after transfer.

`trophic` *Create a trophic table*

Description

Creates a data.frame of class `trophic_tbl` designed to store trophic links. Can be initialized empty or from an existing data.frame.

Usage

```
trophic(data = NULL, from = NULL, to = NULL, weight = NULL)
```

Arguments

`data`	(Optional) A data.frame containing link information.
`from`	(Optional) Character string. Name of the column in `data` containing source nodes.
`to`	(Optional) Character string. Name of the column in `data` containing target nodes.
`weight`	(Optional) Character string. Name of the column in `data` containing weights. Default is 1 if not specified.

Value

An object of class `trophic_tbl`.

Examples

```
# 1. Empty initialization (pipe style)
net <- trophic() |>
  add_link("sol", "sp1")

# 2. Initialization from data.frame
df_raw <- data.frame(src = c("A", "A"), target = c("B", "C"), w = c(2, 5))
net_from_df <- trophic(df_raw, from = "src", to = "target", weight = "w")
```

`[.trophic_tbl` *Subset method for trophic_tbl*

Description

Ensures that any subsetting or modification preserves the validity of the trophic table.

Usage

```
## S3 method for class 'trophic_tbl'
x[...]
```

Arguments

`x`	A trophic_tbl object.
`...`	Additional arguments passed to the base method.

Value

A validated trophic_tbl object.

Index