Package: spacemodR (via r-universe)

October 13, 2026

Title Spatial Environmental Risk Assessment Through Food Webs

Version 0.3.5

Description A workflow for spatially explicit environmental risk assessment of contaminated sites: habitat, food web, dispersal, exposure and risk. Starting from a raster of soil contamination, the package builds habitat layers for a set of species, organises them in a directed acyclic graph representing a food web, and propagates the flow of elements (e.g., food, energy, contaminants) along its links. It also provides soil interpolation by kriging, dispersal by kernel convolution or by circuit theory (through the optional 'Julia' package 'Omniscape'), land cover retrieval for France ('OCS-GE') and Europe ('EUNIS'), and Bayesian calibration of transfer models with 'Stan'.

License MIT + file LICENSE

URL https://github.com/Qonfluens/spacemodR, https://qonfluens.github.io/spacemodR/

BugReports https://github.com/Qonfluens/spacemodR/issues

Imports dplyr, ggplot2, httr, methods, patchwork, Rcpp (>= 0.12.0), rlang, rstan (>= 2.18.1), rstantools (>= 2.4.0), sf, terra, tools

Encoding UTF-8

LazyData true

Suggests archive, knitr, rmarkdown, testthat (>= 3.0.0), JuliaCall, tidyterra, scales, purrr, quarto, tidyr, leaflet, gstat

Config/testthat/edition 3

VignetteBuilder knitr

Depends R (>= 4.1.0)

Biarch true

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements GNU make, Julia (optional, for dispersal with 'Omniscape')

Config/roxygen2/version 8.0.0

NeedsCompilation yes

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

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

Config/pak/sysreqs make

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

Date/Publication 2026-10-12 19:10:08 UTC

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

RemoteRef HEAD

RemoteSha 8039ec1584c28cde2a8c4fc541c9458492ac28f6

Contents

spacemodR-package . . . 3
[.trophic_tbl . . . 4
add_flux . . . 4
add_habitat . . . 5
add_link . . . 6
add_nohabitat . . . 7
apply_flux . . . 8
bappet_cd . . . 8
calibrate_direct . . . 9
calibrate_simple . . . 9
calibrate_trophic . . . 10
compute_dispersal . . . 10
compute_kernel . . . 11
cv_soil . . . 11
DBFunc_BirdFuncDat . . . 13
DBFunc_MamFuncDat . . . 14
dispersal . . . 16
earthworm_cd . . . 17
eunis_species_dict . . . 17
fit_soil_variogram . . . 19
flux . . . 20
FmrBT . . . 20
get_departements_for_roi . . . 21
get_eunis_data . . . 22
get_gbif_data . . . 23
get_ocsge . . . 25
get_ocsge_data_fgb . . . 26
get_ocsge_data_ign . . . 28

habitat . . . 29
habitat_raster . . . 31
is_cyclic . . . 32
join_ocsge_species . . . 33
kernel_smooth_soil . . . 34
krige_soil . . . 35
load_raster_extdata . . . 37
lower_neighbors . . . 38
normalize_weights . . . 38
ocsge_metaleurop . . . 39
ocsge_species_dict . . . 40
ocsge_us_metaleurop . . . 42
plot.habitat . . . 42
plot.soil_vgm . . . 43
plot.trophic_tbl . . . 44
plot_species_habitat . . . 45
raster_stack . . . 46
ref_eunis . . . 46
ref_ocsge . . . 47
ref_ocsge_eunis . . . 48
roi_metaleurop . . . 49
sf_micromammals . . . 50
soil_grid . . . 50
soil_metaleurop . . . 51
soil_variogram . . . 52
soil_vgm . . . 53
source_covariates . . . 53
spacemodel . . . 54
SSD_ecoSSL_all . . . 55
summary.soil_cv . . . 56
transfer . . . 57
trophic . . . 58

Index **59**

`spacemodR-package` *The 'spacemodR' package.*

Description

Workflow for spatially explicit environmental risk assessment: habitat, food web, dispersal, exposure and risk.

References

Stan Development Team (2020). RStan: the R interface to Stan. R package version 2.21.2. https://mc-stan.org

[.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.

add_flux *Add a Flux Rule*

Description

Adds a specific flux rule to a trophic link or target in a trophic table.

Usage

```
add_flux(tbl, from = NULL, to, value)
```

Arguments

tbl	A ‘trophic_tbl‘ object with an initialized ‘flux‘ column (output of ‘flux()‘).
from	Character. The source species. If ‘NULL‘ (default), the rule applies to all links pointing to ‘to‘.
to	Character. The target species.
value	Numeric, formula, or function. The flux definition.

Value

The updated ‘trophic_tbl‘.

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, priority = NULL, touches = TRUE, ...)
```

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.
priority	Stacking order of the added layer when the habitat is rasterized with habitat_raster: where layers overlap, the one with the highest priority gives its value to the pixel. A single number for the whole layer, or one value per feature. By default (NULL) the layer is placed on top of everything already in hab (highest priority so far + 1), so that layers stack in the order they are added.
touches	Logical, a single value for the whole layer or one value per feature. If TRUE (default), the features fill every pixel they touch when rasterized with habitat_raster; if FALSE, a polygon only fills the pixels whose centre it covers, and a line only the pixels of a thin line. See Details.
...	Additional arguments

Details

touches controls which pixels a feature occupies, not how hard it is to cross: the permeability of a structure is carried by its weight (or by add_nohabitat for an absolute barrier).

- touches = TRUE keeps thin structures (hedgerows, rivers, roads, whether drawn as lines or as narrow polygons) continuous: every pixel they cross is filled, so they cannot vanish between pixel centres nor be crossed diagonally between two pixels. The price is a footprint larger than the feature: it grows by up to one pixel in width.
- touches = FALSE keeps areas unbiased, which is preferable for a layer made of contiguous polygons covering the whole landscape (a land cover map): with touches = TRUE, each polygon would spread on the pixels of its neighbours and the border pixels would go to the feature on top of the stack.

A common set-up is therefore a land cover layer added with touches = FALSE and a low priority, with the thin structures that must stay continuous added on top with touches = TRUE.

Value

A Habitat object with new geometries added as habitat

Examples

```
library(sf)
square <- function(a, b) {
  st_sf(geometry = st_sfc(st_polygon(list(
    cbind(c(a, b, b, a, a), c(a, a, b, b, a))))))
}
hedge <- st_sf(geometry = st_sfc(st_linestring(cbind(c(0, 10), c(0, 10)))))

# the hedge is declared first but kept on top of the field with `priority`
hab <- habitat() |>
  add_habitat(hedge, weight = 1, priority = 10) |>
  add_habitat(square(2, 8), weight = 0.2, priority = 1)
hab
```

---

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, priority = NULL, touches = TRUE, ...)
```

Arguments

`hab`	A Habitat object
`sf_data`	An sf object containing geometries to add
`priority`	Stacking order of the added layer when the habitat is rasterized with `habitat_raster`: the non-habitat zone only excludes the pixels where no layer of higher priority lies on top of it. Same default as in `add_habitat`: on top of everything already in `hab`.
`touches`	Logical, a single value for the whole layer or one value per feature. If `TRUE` (default), every pixel touched by the features is excluded, so that the barrier is watertight: it cannot vanish between pixel centres nor be crossed diagonally between two pixels. If `FALSE`, a polygon only excludes the pixels whose centre it covers and a line only the pixels of a thin line. See `add_habitat` for details.
`...`	Additional arguments

Value

A Habitat object with new geometries added as non-habitat

apply_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

```
apply_flux(raster, fluxes, from, to)
```

Arguments

raster	A SpatRaster (or similar) representing exposure from the resource.
fluxes	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.

bappet_cd *Data concentration Cd Soil - Vegetation extracted from BAPPET*

Description

Data concentration Cd Soil - Vegetation extracted from BAPPET

Usage

```
data(bappet_cd)
```

Format

An object of class `data.frame` with 3040 rows and 80 columns.

Examples

```
data(bappet_cd)
```

---

calibrate_direct *Calibrate direct transfer model*

Description

Fits a Bayesian direct transfer model using Stan. The model handles continuous and categorical explanatory variables to predict a response variable.

Usage

```
calibrate_direct(data, ...)
```

Arguments

`data`	A named list containing the data formatted for the Stan model. Required elements are: `N`, `K`, `M`, `y`, `x`, `x_cat`, and `x_sim`.
`...`	Additional arguments passed to `sampling` (e.g., `iter`, `chains`, `cores`, `control`).

Value

A `stanfit` object containing the posterior samples.

---

calibrate_simple *Calibrate simple transfer model*

Description

Fits a Bayesian simple transfer model using Stan. The model handles continuous and categorical explanatory variables to predict a response variable.

Usage

```
calibrate_simple(data, ...)
```

Arguments

`data`	A named list containing the data formatted for the Stan model. Required elements are: `N`, `M`, `y`, `x`, and `x_sim`.
`...`	Additional arguments passed to `sampling` (e.g., `iter`, `chains`, `cores`, `control`).

Value

A `stanfit` object containing the posterior samples.

`calibrate_trophic` *Calibrate trophic transfer model*

Description

Fits a Bayesian trophic transfer model using Stan. This model calibrates relationships across multiple trophic compartments.

Usage

```
calibrate_trophic(data, ...)
```

Arguments

`data`	A named list containing the data formatted for the Stan model. Required elements are: `Np`, `Nw`, `Nh`, `Ni`, `M`, `yp`, `xp`, `yw`, `xw`, `yh`, `xh`, `yi`, `xi`, and `x_sim`.
`...`	Additional arguments passed to `sampling` (e.g., `iter`, `chains`, `cores`, `control`).

Value

A `stanfit` object containing the posterior samples.

`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

`cv_soil` *Cross-validate a soil interpolation method*

Description

Estimates the prediction error of an interpolation method by k-fold cross-validation: the points are split in ‘nfold‘ groups; each group is in turn removed, the model is re-fitted (including the variogram) on the remaining points and used to predict the removed ones.

With ‘block_size‘, folds are made of square spatial blocks instead of random points (*spatial cross-validation*). This is more demanding: it mimics the prediction of areas located far from any sample, where interpolators that simply copy neighbours are penalised.

Usage

```
cv_soil(
  points,
  value,
  method = c("kriging", "kernel"),
  covariates = NULL,
  model = "Exp",
  log10 = TRUE,
  nfold = 10,
  block_size = NULL,
  seed = 1,
  radius = 200,
  size_std = 3,
  kappa = 0.5,
  cutoff = NULL,
  width = NULL
)
```

Arguments

points	An ‘sf‘ object of POINT geometries in a projected CRS (metres).
value	Character. Name of the concentration column (e.g. ‘"cd"‘).
method	Character. ‘"kriging"‘ ([krige_soil()]) or ‘"kernel"‘ ([kernel_smooth_soil()]).
covariates	Optional ‘SpatRaster‘ of drift covariates (must cover the points and the output grid, same CRS).
model	Character vector of candidate variogram models (see [fit_soil_variogram()]); the best fitting one is used. Ignored when ‘vgm‘ is given.
log10	Logical. Krige ‘log10(value)‘ (default ‘TRUE‘).
nfold	Integer. Number of folds (default 10).
block_size	Numeric. Side (m) of the spatial blocks; ‘NULL‘ (default) for random folds.
seed	Integer. Random seed for the fold assignment.
radius, size_std	Kernel standard deviation (m) and window half-size (in number of ‘radius‘), for ‘method = "kernel"‘ (see [kernel_smooth_soil()]). For speed, the cross-validation evaluates the Gaussian kernel exactly at the left-out points instead of on a raster.
kappa	Matérn smoothness, passed to [fit_soil_variogram()].
cutoff, width	Passed to [soil_variogram()].

Value

A ‘data.frame‘ of class ‘soil_cv‘ with the observed value (‘obs‘), the prediction (‘pred‘) and the ‘fold‘ of each point (on the log10 scale if ‘log10 = TRUE‘). Use [summary()] on it to get RMSE, MAE, bias and R2.

See Also

[krige_soil()], [kernel_smooth_soil()]

Examples

```
data(soil_metaleurop)
cv <- cv_soil(soil_metaleurop, "cd", model = "Exp")
summary(cv)
```

`DBFunc_BirdFuncDat` *Bird Functional Traits Dataset (EltonTraits 1.0)*

Description

Dietary category percentages, foraging strategies, and functional traits for birds.

Usage

```
DBFunc_BirdFuncDat
```

Format

A data frame containing species dietary breakdowns (summing to 100

SpecID Unique species identifier.

PassNonPass Indicates whether the bird is a Passerine or Non-Passerine.

IOCOrder IOC taxonomic order.

BLFamilyLatin BirdLife International family Latin name.

BLFamilyEnglish BirdLife International family English name.

BLFamSequID BirdLife International family sequence ID.

Taxo Taxonomic grouping code.

Scientific Scientific name of the bird species.

English Common English name of the bird species.

Diet.Inv Percentage of invertebrates in the diet.

Diet.Vend Percentage of vertebrate endotherms consumed.

Diet.Vect Percentage of vertebrate ectotherms consumed.

Diet.Vfish Percentage of fish consumed.

Diet.Vunk Percentage of unknown/unclassified vertebrates in the diet.

Diet.Scav Percentage of scavenging activity.

Diet.Fruit Percentage of fruits consumed.

Diet.Nect Percentage of nectar/pollen consumed.

Diet.Seed Percentage of seeds/nuts consumed.

Diet.PlantO Percentage of other plant tissues consumed.

Diet.5Cat Simplified 5-category diet classification.

Diet.Source Source of the dietary information.

Diet.Certainty Certainty score for the diet data.

Diet.EnteredBy Identifier of the person who entered the diet data.

ForStrat.watbelowsurf Percentage of foraging time spent underwater.

ForStrat.wataroundsurf Percentage of foraging time spent at the water surface.

ForStrat.ground Percentage of foraging time spent on the ground.

ForStrat.understory Percentage of foraging time spent in the understory.

ForStrat.midhigh Percentage of foraging time spent in the mid-high canopy.

ForStrat.canopy Percentage of foraging time spent in the upper canopy.

ForStrat.aerial Percentage of foraging time spent in aerial foraging.

PelagicSpecialist Indicator of pelagic specialization.

ForStrat.Source Source of the foraging strategy data.

ForStrat.SpecLevel Specificity level of the foraging strategy record.

ForStrat.EnteredBy Identifier of the person who entered the foraging data.

Nocturnal Indicator of nocturnal activity.

BodyMass.Value Body mass value in grams.

BodyMass.Source Source of the body mass information.

BodyMass.SpecLevel Specificity level of the body mass record.

BodyMass.Comment Notes on body mass data.

Record.Comment General comments about the record.

Source

EltonTraits 1.0 database.

`DBFunc_MamFuncDat` *Mammal Functional Traits Dataset (EltonTraits 1.0)*

Description

Dietary category percentages, foraging strategies, and functional traits for mammals.

Usage

```
DBFunc_MamFuncDat
```

Format

A data frame containing species dietary breakdowns (summing to 100

MSW3_ID Mammal Species of the World (3rd ed.) identifier.

Scientific Scientific name of the mammal species.

MSWFamilyLatin Mammal Species of the World (3rd ed.) family name.

Diet.Inv Percentage of invertebrates in the diet.

Diet.Vend Percentage of vertebrate endotherms consumed (birds and mammals).

Diet.Vect Percentage of vertebrate ectotherms consumed (reptiles and amphibians).

Diet.Vfish Percentage of fish consumed.

Diet.Vunk Percentage of unknown/unclassified vertebrates in the diet.

Diet.Scav Percentage of scavenging activity.

Diet.Fruit Percentage of fruits consumed.

Diet.Nect Percentage of nectar/pollen consumed.

Diet.Seed Percentage of seeds/nuts consumed.

Diet.PlantO Percentage of other plant tissues consumed (leaves, stems, roots).

Diet.Source Source of the dietary information.

Diet.Certainty Certainty score for the diet data.

ForStrat.Value Primary foraging stratum/habitat.

ForStrat.Certainty Certainty score for the foraging stratum.

ForStrat.Comment Notes on foraging strategy.

Activity.Nocturnal Binary flag for nocturnal activity.

Activity.Crepuscular Binary flag for crepuscular activity.

Activity.Diurnal Binary flag for diurnal activity.

Activity.Source Source of the activity information.

Activity.Certainty Certainty score for the activity pattern.

BodyMass.Value Body mass value in grams.

BodyMass.Source Source of the body mass information.

BodyMass.SpecLevel Specificity level of the body mass record.

Source

EltonTraits 1.0 database.

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)
```

`earthworm_cd` *Data concentration Cd Soil - Earthworm*

Description

Data concentration Cd Soil - Earthworm

Usage

```
data(earthworm_cd)
```

Format

An object of class `data.frame` with 91 rows and 4 columns.

Examples

```
data(earthworm_cd)
```

`eunis_species_dict` *Species habitat dictionary for EUNIS Level 2 land cover classes*

Description

Habitat suitability, breeding, foraging and movement scores of 935 European terrestrial vertebrates (amphibians, reptiles, birds, mammals) for each of the 45 EUNIS Level 2 classes of the EEA Ecosystem Type Map v3.1 (terrestrial part), as returned by `get_eunis_data` with `level = "L2"`. It is the EUNIS counterpart of `ocsge_species_dict`, with an improved schema: scores follow an anchored rubric, breeding is separated from foraging, and resistance is derived from permeability on the 1-100 log scale expected by connectivity models such as Omniscape.

Usage

```
data(eunis_species_dict)
```

Format

A data frame with 42,075 rows (935 species x 45 EUNIS classes) and 13 columns:

code_eunis Character. EUNIS Level 2 code (e.g. `"G1"`); matches `ref_eunis$code` for `level == "L2"`.

species_name Character. Scientific name, genus and species separated by `"_"`.

sppID Character. Species identifier from the EEA source table; the first letter gives the group (A, R, B, M).

taxon_group Character. `"amphibian"`, `"reptile"`, `"bird"` or `"mammal"`.

suitability Integer (0-10). Overall habitat suitability: 0 unused or avoided, 2 marginal, 5 secondary, 8 main, 10 core habitat.

weight_breeding Integer (0-10). Value of the class as a breeding, nesting, denning, roosting or larval site.

weight_foraging Integer (0-10). Value of the class for feeding.

permeability Integer (0-10). Ease of movement through the class: 10 moves freely, 7 readily, 4 reluctantly, 1 strong barrier, 0 impassable. Accounts for flight, swimming or fossorial habits.

resistance Numeric (1-100). Derived from permeability as `10^(2 * (1 - permeability / 10))`: 10 gives 1, 5 gives 10, 0 gives 100.

habitat_breadth Integer. Species-level: number of EUNIS classes with `suitability >= 5`.

specialization Numeric (0-1). Species-level: 1 minus the normalised Shannon evenness of the suitability profile (0 generalist, 1 specialist).

confidence Integer (1-3). Species-level: 1 vague or missing source description, 2 medium, 3 detailed.

in_france Integer (0/1). Species-level: 1 if the species occurs natively and regularly in metropolitan France (expert knowledge).

Details

Scores were assigned by a large language model (Claude Opus 5.5) reading the EEA habitat and avoided-habitat descriptions of each species (`raw_data/spp_EEA_habitats_cleaned.csv`) against short ecological descriptions of each EUNIS class. They are expert-judgement scores and have not been validated against occurrence data. Four species had an empty description and were scored from general knowledge (`confidence = 1`). The scoring files, the comparison with `ocsge_species_dict` and the build scripts are in `raw_data/opus_scoring/` of the source repository.

See Also

`get_eunis_data`, `ref_eunis`, `ocsge_species_dict`

Examples

```
data(eunis_species_dict)
subset(eunis_species_dict, species_name == "Apodemus_sylvaticus")
```

`fit_soil_variogram` *Fit a variogram model to an empirical soil variogram*

Description

Fits one or several theoretical variogram models to an empirical semi-variogram (from [soil_variogram()]) by weighted least squares, using the weights N_j/h_j^2 (number of pairs over squared distance), which give more importance to the short distances that matter most for kriging (same default as ‘gstat::fit.variogram(fit.method = 7)‘).

Available models (the ‘range‘ parameter has the same meaning as in gstat):

- ‘"Exp"‘: exponential, $\gamma(h) = c_0 + c(1 - e^{-h/a})$;
- ‘"Sph"‘: spherical, reaches the sill exactly at $h = a$;
- ‘"Gau"‘: Gaussian, very smooth at the origin;
- ‘"Mat"‘: Matérn with smoothness ‘kappa‘ (‘kappa = 0.5‘ is ‘"Exp"‘).

Usage

```
fit_soil_variogram(variogram, model = "Exp", kappa = 0.5)
```

Arguments

`variogram`	A ‘soil_variogram‘ object.
`model`	Character vector of model(s) to fit. When several are given, all are fitted and the one with the lowest weighted sum of squared errors is returned (the full comparison table is kept in ‘$comparison‘).
`kappa`	Numeric. Smoothness of the Matérn model (ignored otherwise).

Value

An object of class ‘soil_vgm‘: a list with ‘model‘, ‘nugget‘, ‘psill‘, ‘range‘, ‘kappa‘, ‘sse‘ (weighted sum of squared errors), ‘empirical‘ (the input variogram) and ‘comparison‘ (a table of all fitted models).

See Also

[soil_variogram()], [krige_soil()]

Examples

```
data(soil_metaleurop)
v <- soil_variogram(soil_metaleurop, "cd")
m <- fit_soil_variogram(v, model = c("Exp", "Sph", "Gau"))
m
m$comparison
```

flux *Constructor for Flux Parameters*

Description

Initializes the trophic flux table for a spacemodel.

Usage

```
flux(x, default = 1, normalize = TRUE)
```

Arguments

x	A ‘spacemodel‘ object or a ‘trophic_tbl‘.
default	The default function or coefficient for unspecified links (default is 1, meaning full transfer: f(x) = x).
normalize	Logical. Whether to normalize diet weights (default TRUE).

Value

A ‘trophic_tbl‘ with an initialized ‘flux‘ column.

FmrBT *Field Metabolic Rates Database (FmrBT)*

Description

A comprehensive database containing Field Metabolic Rates (FMR), body mass, and ambient temperature across more than 700 species.

Usage

```
FmrBT
```

Format

A data frame with variables detailing energetic properties and taxonomy:

Kingdom Taxonomic kingdom.

Phylum Taxonomic phylum.

Class Taxonomic class.

Order Taxonomic order.

Family Taxonomic family.

Genus Taxonomic genus.

SpeciesVerbatim Original species name as recorded in the source.

SpeciesAcceptedName Standardized accepted species name.

FMR_kJ_d Field metabolic rate per individual in kJ/day.

Mass_g Body mass of the individual in grams.

Temp_C The ambient temperature recorded during the study in Celsius.

Endotherm Indicator of thermal status (1/TRUE for endotherm, 0/FALSE for ectotherm).

Reference Source literature reference.

Comment Additional notes from the dataset creators.

Outlier Flag indicating if the record is considered a statistical outlier.

Source

De Castro et al. (2025) FmrBT database.

`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_eunis_data` *Efficiently retrieve EUNIS land cover data from the EEA Ecosystem Type Map*

Description

This function retrieves EUNIS (European Nature Information System) land cover data for a Region of Interest (ROI), as an alternative to the French OCS-GE nomenclature used by `get_ocsge_data_fgb`. It queries the European Environment Agency (EEA) "Ecosystem Type Map v3.1, terrestrial part" ArcGIS MapServer, a 100 m resolution, EEA-39-wide classified raster available at EUNIS Level 1 (9 broad classes) or Level 2 (46 detailed classes).

Usage

```
get_eunis_data(
  roi,
  level = c("L1", "L2"),
  resolution = 100,
  service_url = NULL
)
```

Arguments

`roi`	An `sf` (or `sfc`) object defining the Region Of Interest. It can be in any projection, but will be transformed to EPSG:3035 (ETRS89-LAEA Europe) internally, the native grid of the source service.
`level`	Character. `"L1"` (9 broad classes, default) or `"L2"` (46 detailed classes). See Details for a caveat on `"L2"`.
`resolution`	Numeric. Requested pixel size in meters. Default is 100, the native resolution of the source raster; values below 100 only upsample without adding real detail. Use a coarser value for large ROIs to stay under the service's image-size limit (see Details).
`service_url`	Character. The ArcGIS MapServer base URL. Exposed mainly for testing against a different endpoint.

Details

Unlike the OCS-GE FlatGeobuf source, this EUNIS layer has no raw-value raster endpoint (no ArcGIS ImageServer is published for it) – only a symbolized MapServer `export`. This function requests that export image at the native resolution with nearest-neighbor resampling (so pixel colors are not blended at class boundaries), then decodes each pixel's exact RGB color back into an EUNIS code using the fixed legend palette shipped as `ref_eunis`.

Limitations:

- The service caps a single export at 4096 x 4096 pixels. At the default 100 m resolution that is about 410 x 410 km; larger ROIs must use a coarser `resolution` or be split and fetched in tiles.

- At `level = "L2"`, two pairs of classes are rendered with the identical color by the source service and cannot be told apart from the image alone: `D5/F6` and `D6/F7`. Affected pixels are returned with a combined code (e.g. `"D5/F6"`) rather than silently assigned to one of the two. `level = "L1"` has no such collision.
- Areas outside the mapped terrestrial extent (e.g. open sea) are detected from the export's alpha channel and returned as `NA`, not guessed from color.

Value

A categorical `SpatRaster` in EPSG:3035, with cell values pointing to a factor level table (`levels`) giving the EUNIS `code` for that level (see `ref_eunis` for the matching labels and colors). Cells outside the mapped terrestrial extent (e.g. sea) are `NA`.

Note

This function requires a working internet connection.

See Also

`ref_eunis` for the EUNIS code/label/color lookup table, and `get_ocsge_data_fgb` for the equivalent OCS-GE fetcher.

Examples

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

  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) |> st_combine() |> st_cast("POLYGON")

  eunis_data <- get_eunis_data(roi = my_roi, level = "L1")
  terra::plot(eunis_data)

## End(Not run)
```

`get_gbif_data` *Retrieve species occurrences from GBIF*

Description

This function retrieves the occurrence records of a species from the Global Biodiversity Information Facility (GBIF), with the public occurrence search API. No account and no API key are needed.

Only records with coordinates and without known geospatial issue are returned.

Usage

```
get_gbif_data(species, roi = NULL, limit = 1000, ..., api_url = NULL)
```

Arguments

species	Character. Scientific name of the species, for instance "Apodemus sylvaticus". Underscores are read as spaces, so the names of ocsge_species_dict and eunis_species_dict can be used directly.
roi	An sf (or sfc) object defining the Region Of Interest, in any projection. The search is made on its bounding box, then only the records inside roi are kept. With NULL (default), the search is made on the whole world.
limit	Integer. Maximum number of records to download. Default is 1000. The API does not return more than 100,000 records for one search (see Details).
...	Other filters of the GBIF occurrence search API, as named arguments. For instance year = "2000,2024", coordinateUncertaintyInMeters = "0,100", basisOfRecord = "HUMAN_OBSERVATION" or country = "FR". A range is written "min,max".
api_url	Character. Base URL of the GBIF API. Exposed mainly for testing against a different endpoint.

Details

The name is first matched against the GBIF taxonomy (species/match), then the records are downloaded by pages of 300 (occurrence/search).

Limitations:

- The search API stops at 100,000 records. For a larger request, split the region of interest, or the years with year.
- The search API gives no DOI. For a publication, GBIF asks to cite a download: see https://www.gbif.org/citation-guidelines and the function occ_download() of the 'rgbif' package.
- GBIF records are opportunistic observations. They show where the species was seen, not where it is absent, and the position can be uncertain by several kilometers.

Value

An sf object of points in EPSG:4326, one row for each record, with the columns:

key GBIF identifier of the record (https://www.gbif.org/occurrence/<key>).

species Scientific name of the species.

coordinateUncertaintyInMeters Uncertainty of the position, in meters. Often NA.

eventDate, year Date and year of the observation.

basisOfRecord Type of record, for instance "HUMAN_OBSERVATION" or "PRESERVED_SPECIMEN".

datasetKey GBIF identifier of the dataset (https://www.gbif.org/dataset/<datasetKey>).

license License of the record.

The total number of records matching the search in GBIF is stored in the attribute "gbif_count".

Note

This function requires a working internet connection.

See Also

`get_eunis_data` and `get_ocsge` for the land cover of the same region of interest.

Examples

```
## Not run:
  data("roi_metaleurop")
  occ <- get_gbif_data("Apodemus sylvaticus", roi = roi_metaleurop)
  plot(sf::st_geometry(occ))

  # recent records with a position known within 100 m
  occ <- get_gbif_data("Apodemus sylvaticus", roi = roi_metaleurop,
                       year = "2000,2024",
                       coordinateUncertaintyInMeters = "0,100")

## End(Not run)
```

---

get_ocsge *Retrieve OCS GE data for a region of interest*

Description

Generic entry point to get OCS GE (Occupation du Sol à Grande Echelle) polygons, with their cover (`code_cs`) and use (`code_us`) codes, for a Region of Interest (ROI). The `source` argument selects where the data come from:

`"ign"` the official per-department deliveries of the IGN Géoplateforme, see `get_ocsge_data_ign`. Nothing to host, but each department is downloaded in full the first time.

`"fgb"` a remote FlatGeobuf file that you host, see `get_ocsge_data_fgb`. Only the features of the ROI are downloaded.

Usage

```
get_ocsge(roi, source = c("ign", "fgb"), ...)
```

Arguments

`roi`	An `sf` (or `sfc`) object defining the Region Of Interest.
`source`	Character. `"ign"` (default) or `"fgb"`.
`...`	Arguments passed to the function of the selected source: `year`, `cache_dir` and `quiet` for `"ign"`, `fgb_url` for `"fgb"`.

Value

An sf object containing the OCS GE polygons clipped to the ROI.

See Also

join_ocsge_species to score the polygons for a species, and get_eunis_data for the EUNIS land cover.

Examples

```
## Not run:
  data(roi_metaleurop)

  # from the IGN Géoplateforme (latest vintage of each department)
  ocsge_data <- get_ocsge(roi_metaleurop)
  ocsge_data <- get_ocsge(roi_metaleurop, source = "ign", year = 2021)

  # from a FlatGeobuf file
  ocsge_data <- get_ocsge(roi_metaleurop, source = "fgb",
                          fgb_url = "https://example.com/data/ocsge_france.fgb")

## End(Not run)
```

---

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).

See Also

get_ocsge, get_ocsge_data_ign

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)
```

get_ocsge_data_ign *Retrieve OCS GE data from the IGN Géoplateforme*

Description

This function retrieves OCS GE (Land Cover) data for a specific Region of Interest (ROI) from the download service of the IGN Géoplateforme (https://data.geopf.fr/telechargement/resource/OCSGE), which needs no API key.

IGN delivers the OCS GE as one compressed GeoPackage per department and vintage (15 to 270 MB each). The departments intersecting the ROI are downloaded in full, extracted and kept in `cache_dir`, so that later calls on the same departments read the local file only.

Usage

```
get_ocsge_data_ign(
  roi,
  year = NULL,
  cache_dir = tools::R_user_dir("spacemodR", "cache"),
  quiet = FALSE
)
```

Arguments

`roi`	An `sf` (or `sfc`) object defining the Region Of Interest. It can be in any projection.
`year`	Integer. Vintage (*millésime*) to retrieve. Default `NULL` takes the latest vintage of each department. Vintages differ between departments; the error raised for a missing one lists those available.
`cache_dir`	Character. Directory where the extracted GeoPackages are kept. Defaults to the user cache directory of the package. Each department takes several hundred MB once extracted; delete the files to free the space.
`quiet`	Logical. If `FALSE` (default), reports the datasets used and the download progress.

Details

The function performs the following steps:

1. Finds the departments intersecting the `roi` with `get_departements_for_roi`.
2. Lists the OCS GE deliveries of these departments (version 2, GeoPackage) and selects the vintage.
3. Downloads the archives missing from `cache_dir`, checks their MD5 sum and extracts the `OCCUPATION_SOL` layer.
4. Reads the features in the bounding box of the `roi` and clips them to its exact shape.

Corrective patches published separately by IGN (`PATCHCORRECTIF`) and the layers of differences between two vintages are not used.

Value

An sf object containing the OCS GE polygons intersected by the ROI, with the columns of the IGN delivery (notably code_cs, code_us and millesime), in the projection of the delivery: Lambert-93 (EPSG:2154) in mainland France and Corsica, the local UTM projection in overseas departments.

Note

This function requires a working internet connection and a way to extract 7z archives: either the **archive** package, or the 7z command line tool (7-Zip, p7zip).

See Also

get_ocsge, get_ocsge_data_fgb

Examples

```
## Not run:
  data(roi_metaleurop)

  # latest vintage
  ocsge_data <- get_ocsge_data_ign(roi_metaleurop)

  # a given vintage
  ocsge_2021 <- get_ocsge_data_ign(roi_metaleurop, year = 2021)

  habitat_vole <- join_ocsge_species(ocsge_2021, "Microtus_arvalis")

## 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
- priority: numeric, stacking order used by habitat_raster
- touches: logical, pixels filled by the feature in habitat_raster
- geometry: sfc geometry

Usage

```
habitat(
  geometry = NULL,
  habitat = NULL,
  weight = NULL,
  priority = NULL,
  touches = 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).
priority	Numeric vector giving the stacking order of the features when they are rasterized with habitat_raster: where features overlap, the one with the highest priority gives its value to the pixel. Default is 1 for every feature, in which case the order of the features decides (last on top).
touches	Logical vector. If TRUE (default), a feature fills every pixel it touches when rasterized with habitat_raster. If FALSE, a polygon only fills the pixels whose centre it covers and a line only the pixels of a thin line. See add_habitat for guidance.

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 *Rasterize a Habitat Object*

Description

Converts a spatial habitat object into a terra::SpatRaster. The resulting raster stores the weight values of the habitat geometries. Areas explicitly defined as non-habitat (habitat = FALSE) are assigned NA.

Usage

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

Arguments

ground_raster	A terra::SpatRaster object used as a reference template. It provides the extent, resolution, and coordinate reference system (CRS).
habitat	A habitat object (inheriting from sf).

Details

The function maps the continuous or categorical weights of the habitat geometries onto a raster grid.

Background handling: By default, pixels not covered by any geometry are assigned a weight of 0 and are considered passable (habitat = TRUE).

Exclusion zones: Geometries added via add_nohabitat() act as absolute barriers. Any pixel intersecting these geometries will be assigned NA, overriding any underlying weight.

Stacking order: Where several features overlap, the pixel takes the value of the feature with the highest priority (see add_habitat); features are stacked from the lowest to the highest priority and the top one is read. With equal priorities, the order of the features in the habitat object decides (last on top). As add_habitat() and add_nohabitat() give by default an increasing priority to each new layer, layers stack in the order they were added unless priority is set explicitly. The same rule applies to exclusion zones: a non-habitat feature excludes a pixel only if no feature of higher priority covers it.

Pixels filled by a feature: The touches value of each feature (see add_habitat) sets which pixels it occupies. With touches = TRUE (the default of add_habitat() and add_nohabitat()) a line or a polygon fills every pixel it touches. A thin structure such as a hedgerow, a river or a road is then a chain of pixels sharing an edge: a corridor stays connected for 4-neighbour movements and a barrier cannot be crossed diagonally. With touches = FALSE, a polygon fills only the pixels whose centre it covers and a line only the pixels of a thin line, which keeps areas unbiased but can break thin structures. Combine a high priority with touches = TRUE to keep a thin structure continuous above the layers it runs through. Points, lines and polygons can be mixed in the same habitat object.

Note that a raster holds one value per pixel: a structure narrower than a pixel is either widened to a full pixel or dropped, and two structures sharing a pixel cannot both be kept. When this matters, the only real solution is a finer ground_raster.

Value

A `terra::SpatRaster` containing habitat weights, with `NA` values in explicit non-habitat zones.

Examples

```
library(terra)
library(sf)

# Create a dummy reference raster
ref_grid <- rast(nrows=10, ncols=10, xmin=0, xmax=10, ymin=0, ymax=10)

# Create habitat zones
good_hab <- st_sf(geometry = st_sfc(st_polygon(list(cbind(c(1, 9, 9, 1, 1), c(1, 1, 9, 9, 1))))))
bad_hab  <- st_sf(geometry = st_sfc(st_polygon(list(cbind(c(4, 6, 6, 4, 4), c(4, 4, 6, 6, 4))))))

# Build habitat object
my_habitat <- habitat() |>
  add_habitat(good_hab, weight = 0.8) |>
  add_nohabitat(bad_hab)

# Rasterize
hab_rast <- habitat_raster(ref_grid, my_habitat)
plot(hab_rast)

# A hedgerow kept continuous across the exclusion zone, whatever the order
# in which the layers are added
hedge <- st_sf(geometry = st_sfc(st_linestring(cbind(c(0, 10), c(0.2, 9.7)))))
my_habitat <- habitat() |>
  add_habitat(hedge, weight = 1, priority = 10) |>
  add_habitat(good_hab, weight = 0.8, priority = 1, touches = FALSE) |>
  add_nohabitat(bad_hab, priority = 2)
plot(habitat_raster(ref_grid, my_habitat))
```

`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)
```

---

join_ocsge_species *Join OCSGE Spatial Data with Species Traits*

Description

This function merges a spatial 'sf' object containing OCSGE (Occupation du Sol à Grande Échelle) polygons with a dictionary of species-specific traits (e.g., habitat suitability, resistance). Polygons are matched on their cover code (`CS`) and, when the object also carries the land use (`US`), on the (cover, use) pair.

Usage

```
join_ocsge_species(
  sf_obj,
  species_pattern,
  code_col = "code_cs",
  species_dict = spacemodR::ocsge_species_dict,
  ref_dict = spacemodR::ref_ocsge,
  us_col = "code_us"
)
```

Arguments

`sf_obj`	A spatial `sf` object containing the landscape polygons.
`species_pattern`	Character. A regex pattern or exact name to search for in the species dictionary.
`code_col`	Character. The name of the column in `sf_obj` that contains the OCSGE cover codes. Default is `"code_cs"`.
`species_dict`	A data.frame containing the species traits. Defaults to `spacemodR::ocsge_species_dict`.
`ref_dict`	An optional reference data.frame giving the label and color of each cover code, added to the result. Defaults to `spacemodR::ref_ocsge`.
`us_col`	Character. The name of the column in `sf_obj` that contains the OCSGE use codes. Default is `"code_us"`. Ignored when the column is absent; set to `NULL` to score polygons on their cover only.

Details

Cover codes are compared without spaces, so `"CS2.2.1"` and `"CS 2.2.1"` are the same class.

In `ocsge_species_dict`, rows with `code_us = NA` hold the scores of a cover whatever its use, and rows with a `code_us` hold the scores of a (cover, use) pair. A polygon takes the scores of its (cover, use) pair when the dictionary has one, and the scores of its cover otherwise. The logical column `use_matched` of the result tells which of the two was used.

Value

An `sf` object enriched with the species traits. Returns `NULL` if the species is not found.

`kernel_smooth_soil` *Gaussian kernel smoothing of soil concentrations*

Description

A simple, assumption-light alternative to kriging: the value of each cell is the weighted mean of the sampled values, the weights being a Gaussian function of the distance (Nadaraya-Watson estimator):

$$\hat{z}(s) = \frac{\sum_i K(s - s_i) z_i}{\sum_i K(s - s_i)}$$

It re-uses [compute_kernel()] (the dispersal kernel of the package): the points are first rasterised (sum of values and number of points per cell), both rasters are convolved with the kernel using [terra::focal()], and the ratio gives the smoothed map.

Unlike kriging, the result does not honour the data and the smoothing strength (‘radius‘) must be chosen by the user, e.g. by cross-validation with [cv_soil()].

Usage

```
kernel_smooth_soil(
  points,
  value,
  radius,
  template = NULL,
  log10 = TRUE,
  res = 25,
  margin = 1000,
  size_std = 3,
  mask = NULL
)
```

Arguments

points	An ‘sf‘ object of POINT geometries in a projected CRS (metres).
value	Character. Name of the concentration column (e.g. ‘"cd"‘).
radius	Numeric. Standard deviation (m) of the Gaussian kernel.
template	Optional ‘SpatRaster‘ defining the output grid. If ‘NULL‘, a grid is built from the points with [soil_grid()] using ‘res‘ and ‘margin‘. If ‘covariates‘ is given and ‘template‘ is ‘NULL‘, the covariate grid is used.
log10	Logical. Krige ‘log10(value)‘ (default ‘TRUE‘).
res, margin	Resolution and margin (m) of the grid built when ‘template‘ and ‘covariates‘ are ‘NULL‘.
size_std	Numeric. Half-size of the kernel window, in number of ‘radius‘ (passed to [compute_kernel()]). Cells with no point within the window are ‘NA‘.
mask	Optional ‘sf‘ polygon or ‘SpatVector‘; cells outside are set to ‘NA‘.

Value

A ‘SpatRaster‘ with the layer ‘<value>_log10‘ (or ‘<value>‘ if ‘log10 = FALSE‘) and, if ‘log10 = TRUE‘, the back-transformed layer ‘<value>‘.

See Also

[compute_kernel()], [krige_soil()], [cv_soil()]

Examples

```
data(soil_metaleurop)
r <- kernel_smooth_soil(soil_metaleurop, "cd", radius = 200, res = 50)
terra::plot(r)
```

---

krige_soil *Krige soil concentrations into a raster*

Description

Interpolates a soil concentration measured at sampling points onto a regular raster grid by kriging. Two flavours are available:

- **Ordinary kriging (OK)** when ‘covariates = NULL‘: the mean is unknown but constant over the area.
- **Kriging with external drift (KED)** when ‘covariates‘ is a ‘SpatRaster‘: the mean varies linearly with the covariate layers (e.g. the log-distance to a contamination source, see [source_covariates()]) and the kriging interpolates the residuals.

By default the interpolation is done on ‘log10(value)‘, which is the natural scale for metal concentrations (log-normal distribution).

Usage

```
krige_soil(
  points,
  value,
  template = NULL,
  covariates = NULL,
  model = "Exp",
  vgm = NULL,
  log10 = TRUE,
  res = 25,
  margin = 1000,
  mask = NULL,
  cutoff = NULL,
  width = NULL,
  kappa = 0.5,
  clamp = TRUE,
  verbose = TRUE
)
```

Arguments

points	An ‘sf‘ object of POINT geometries in a projected CRS (metres).
value	Character. Name of the concentration column (e.g. ‘"cd"‘).
template	Optional ‘SpatRaster‘ defining the output grid. If ‘NULL‘, a grid is built from the points with [soil_grid()] using ‘res‘ and ‘margin‘. If ‘covariates‘ is given and ‘template‘ is ‘NULL‘, the covariate grid is used.
covariates	Optional ‘SpatRaster‘ of drift covariates (must cover the points and the output grid, same CRS).
model	Character vector of candidate variogram models (see [fit_soil_variogram()]); the best fitting one is used. Ignored when ‘vgm‘ is given.
vgm	Optional ‘soil_vgm‘ object (from [fit_soil_variogram()] or [soil_vgm()]) to use instead of fitting one.
log10	Logical. Krige ‘log10(value)‘ (default ‘TRUE‘).
res, margin	Resolution and margin (m) of the grid built when ‘template‘ and ‘covariates‘ are ‘NULL‘.
mask	Optional ‘sf‘ polygon or ‘SpatVector‘; cells outside are set to ‘NA‘.
cutoff, width	Passed to [soil_variogram()].
kappa	Matérn smoothness, passed to [fit_soil_variogram()].
clamp	Logical. Restrict the covariate values of the grid cells to the range observed at the sampling points (default ‘TRUE‘). This prevents the linear drift from being extrapolated outside the conditions actually sampled (e.g. right next to a source where no sample was taken).
verbose	Logical. Print messages?

Value

A 'SpatRaster' with three layers:

'<value>_log10' the kriging prediction on the log10 scale (named '<value>' if 'log10 = FALSE');

'<value>_var' the kriging variance (same scale as the prediction);

'<value>' (only if 'log10 = TRUE') the back-transformed concentration '10^prediction', i.e. the median concentration.

The fitted variogram is returned as the attribute '"vgm"' when available.

References

Pebesma, E. J. (2004). Multivariable geostatistics in S: the gstat package. Computers & Geosciences, 30(7), 683-691.

Hengl, T., Heuvelink, G. B. M., & Rossiter, D. G. (2007). About regression-kriging: From equations to case studies. Computers & Geosciences, 33(10), 1301-1315.

See Also

[soil_variogram()], [fit_soil_variogram()], [cv_soil()], [source_covariates()], [kernel_smooth_soil()]

Examples

```
data(soil_metaleurop)
# Ordinary kriging of log10(Cd) at 100 m
r <- krige_soil(soil_metaleurop, "cd", res = 100, margin = 500)
terra::plot(r)
```

---

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` *Species habitat dictionary for OCS-GE land cover and land use*

Description

Habitat suitability, breeding, foraging and movement scores of 935 European terrestrial vertebrates (amphibians, reptiles, birds, mammals) for the French OCS-GE nomenclature: the 14 cover classes (`CS`), and 47 (cover, use) pairs for which the land use (`US`) changes what the cover stands for. It has the same columns as `eunis_species_dict`, plus `code_us`.

Usage

```
data(ocsge_species_dict)
```

Format

A data frame with 57,035 rows (935 species x 61 classes) and 14 columns:

code_cs Character. OCS-GE cover code, as in `ref_ocsge$code_cs` (e.g. `"CS2.2.1"`).

code_us Character. OCS-GE use code (e.g. `"US1.1"`), or `NA` on the rows giving the scores of the cover whatever its use.

species_name Character. Scientific name, genus and species separated by `"_"`.

sppID Character. Species identifier from the EEA source table; the first letter gives the group (A, R, B, M).

taxon_group Character. `"amphibian"`, `"reptile"`, `"bird"` or `"mammal"`.

suitability Integer (0-10). Overall habitat suitability: 0 unused or avoided, 2 marginal, 5 secondary, 8 main, 10 core habitat.

weight_breeding Integer (0-10). Value of the class as a breeding, nesting, denning, roosting or larval site.

weight_foraging Integer (0-10). Value of the class for feeding.

permeability Integer (0-10). Ease of movement through the class: 10 moves freely, 7 readily, 4 reluctantly, 1 strong barrier, 0 impassable. Accounts for flight, swimming or fossorial habits.

resistance Numeric (1-100). Derived from permeability as `10^(2 * (1 - permeability / 10))`: 10 gives 1, 5 gives 10, 0 gives 100.

habitat_breadth Integer. Species-level: number of cover classes (out of 14) with `suitability >= 5`.

specialization Numeric (0-1). Species-level: 1 minus the normalised Shannon evenness of the suitability profile over the 14 cover classes (0 generalist, 1 specialist).

confidence Integer (1-3). Species-level: 1 vague or missing source description, 2 medium, 3 detailed.

in_france Integer (0/1). Species-level: 1 if the species occurs natively and regularly in metropolitan France (expert knowledge). The dictionary covers Europe while OCS-GE covers France: filter on this column first.

Details

Cover rows (code_us = NA, 935 x 14). Scores were assigned by a large language model (Claude Opus 5.5) reading the EEA habitat and avoided-habitat descriptions of each species against an ecological description of each cover class, with the anchored rubric given above.

Cover x use rows (935 x 47). They are derived, not scored again: each (cover, use) pair is linked to one or several EUNIS Level 2 classes by ref_ocsge_eunis, and takes the mean of their eunis_species_dict scores weighted by 1 / rank. For instance herbaceous cover under agricultural use (CS2.2.1, US1.1) is mostly arable land (I1, rank 1) but may be permanent grassland (E2, rank 2). Only the pairs listed in ref_ocsge_eunis have a row; for any other pair use the cover row, which is what join_ocsge_species does.

These are expert-judgement scores and have not been validated against occurrence data. They were checked against the keywords of the EEA habitat, avoided-habitat and threat descriptions (raw_data/check_ocsge_species_dict.R of the source repository): species naming a habitat score higher on the matching class than the others, with a probability of 0.57 to 0.93 depending on the habitat for cover rows, and 0.68 to 0.93 for cover x use rows. The use helps most for gardens and parks (0.57 to 0.80), quarries (0.69 to 0.93) and arable land (0.74 to 0.77).

Change from spacemodR 0.3.0. The dictionary used to hold scores from another model for the cover classes only, in the columns weight_global, weight_movement, weight_foraging and resistance (1-10), with nom_espece and cover codes written with a space ("CS 2.2.1"). suitability replaces weight_global and permeability replaces weight_movement; values differ (Spearman correlation 0.72 for suitability, 0.22 for movement, mostly because flying species were scored as if they walked) and are lower on classes that are not habitat.

References

The species list and the habitat descriptions come from:

O'Connor, L. M. J., Renaud, J., Dou, Y., Karger, D. N., Maiorano, L., Verburg, P. H., & Thuiller, W. (2024). Habitat Suitability of European Land Systems for Terrestrial Vertebrates. *Global Ecology and Biogeography*, 33(11), e13903.

See Also

join_ocsge_species, ref_ocsge, ref_ocsge_eunis, eunis_species_dict

Examples

```
data(ocsge_species_dict)
apsy <- subset(ocsge_species_dict, species_name == "Apodemus_sylvaticus")

# scores of each cover, whatever its use
subset(apsy, is.na(code_us), c(code_cs, suitability, permeability, resistance))

# herbaceous cover, by use
subset(apsy, code_cs == "CS2.2.1", c(code_us, suitability, weight_foraging))
```

ocsge_us_metaleurop *OCS-GE land cover and land use of the Metaleurop site*

Description

OCS-GE polygons of `roi_metaleurop` with both the cover (`CS`) and the use (`US`), dissolved into one multipolygon per (cover, use) pair. `ocsge_metaleurop` is dissolved by cover only and cannot be used to study the land use.

Usage

```
data(ocsge_us_metaleurop)
```

Format

An `sf` object (EPSG:2154) with 68 multipolygons and 3 columns:

code_cs Character. OCS-GE cover code (e.g. `"CS2.2.1"`).

code_us Character. OCS-GE use code (e.g. `"US1.1"`).

area_ha Numeric. Area of the (cover, use) pair on the site, in hectares.

Details

Built from the 2021 OCS-GE of the Nord and Pas-de-Calais departments, as returned by `get_ocsge_data_fgb`. Outlines are simplified with a 3 m tolerance to keep the dataset small.

See Also

`join_ocsge_species`, `ocsge_species_dict`

Examples

```
data(ocsge_us_metaleurop)
head(ocsge_us_metaleurop)
```

plot.habitat *Plot a Habitat object*

Description

Visually inspect a `habitat` object. Favorable habitat zones are colored in green, and absolute barriers (non-habitat) are colored in red.

Usage

```
## S3 method for class 'habitat'
plot(x, ...)
```

Arguments

x	A habitat object.
...	Additional arguments passed to plot (e.g., main, border).

Value

x, invisibly. Called for its side effect: a plot of the habitat zones.

Examples

```
library(sf)
square <- function(a, b) st_polygon(list(cbind(c(a, b, b, a, a), c(a, a, b, b, a))))
good_polygons <- st_sf(geometry = st_sfc(square(1, 9)))
roads <- st_sf(geometry = st_sfc(square(4, 6)))
hab <- habitat() |>
  add_habitat(good_polygons, weight = 1) |>
  add_nohabitat(roads)
plot(hab)
```

---

plot.soil_vgm *Plot an empirical variogram and its fitted model(s)*

Description

Plot an empirical variogram and its fitted model(s)

Usage

```
## S3 method for class 'soil_vgm'
plot(x, all_models = TRUE, ...)
```

Arguments

x	A ‘soil_vgm‘ object (output of [fit_soil_variogram()]).
all_models	Logical. Draw every model of ‘x$comparison‘? (default ‘TRUE‘)
...	Not used.

Value

A ‘ggplot‘ object.

`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,
  colors = NULL,
  edge_width = c("none", "raw", "norm_target", "norm_source"),
  ...
)
```

Arguments

`x`	A `trophic_tbl` object.
`shift`	To shift x_axis between trophic level and avoid the potential overlapping of arrows.
`colors`	A named character vector of colors (hexadecimal or standard R color names) where the names match the node names in the trophic table. If NULL, default ggplot2 colors are used.
`edge_width`	A character string indicating how to scale the width of the arrows. Must be one of "none" (default), "raw" (uses the raw weights), "norm_target" (normalizes weights to sum to 1 for each target/to node), or "norm_source" (normalizes weights to sum to 1 for each source/from node).
`...`	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 with default colors
plot(net)

# Plot with custom colors
my_pal <- c("a" = "brown", "b" = "green", "c" = "blue")
plot(net, colors = my_pal)
```

---

plot_species_habitat *Plot Habitat Suitability Maps from OCSGE Data*

Description

Generates a composite plot of three spatial maps (suitability, foraging weight and resistance) from an ‘sf‘ object previously enriched with species traits.

Usage

```
plot_species_habitat(sf_merged, title_species = NULL)
```

Arguments

sf_merged	An ‘sf‘ object enriched with species traits (e.g., output of join_ocsge_species).
title_species	Character (optional). The name of the species to display in the title. If NULL, it will try to retrieve the name from the object attributes.

Details

Suitability and foraging weight are drawn on their full 0-10 scale, and resistance on its 1-100 logarithmic scale, so that maps of different species can be compared.

Value

A patchwork object containing the combined ggplot2 maps.

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_eunis *Nomenclature and render colors of EUNIS land cover classes*

Description

Code / label / render-color lookup table for the EEA "Ecosystem Type Map v3.1, terrestrial part" ArcGIS service, at both EUNIS Level 1 (9 broad classes) and Level 2 (46 detailed classes). It is used internally by get_eunis_data to decode the service's exported image back into EUNIS codes, since that service publishes no raw-value raster endpoint.

Usage

```
data(ref_eunis)
```

Format

A data frame with the following columns:

level Character. `"L1"` or `"L2"`.

code Character. The EUNIS code (e.g. `"J1"` at L2, `"J"` at L1).

nomenclature Character. The class label.

couleur Character. The hex render color used by the source ArcGIS service.

r, g, b Integer (0-255). The render color, split into channels for exact matching.

Details

At `level == "L2"`, two pairs of classes share an identical render color and cannot be distinguished from the exported image alone: `D5/F6` and `D6/F7`. `level == "L1"` has no such collision. See `get_eunis_data` for how this is handled.

Source

European Environment Agency, "Ecosystem Type Map v3.1, terrestrial part": https://bio.discomap.eea.europa.eu/arcgis/rest/services/Ecosystem/EcosystemTypeMap_v3_1_Terrestrial/MapServer

Examples

```
data(ref_eunis)
```

---

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)
```

`ref_ocsge_eunis` *Crosswalk between OCS-GE (cover and use) and EUNIS Level 2*

Description

Correspondence table between the French OCS-GE nomenclature and the 45 EUNIS Level 2 classes of `ref_eunis`. The two nomenclatures do not nest: OCS-GE cover (`CS`) describes what physically covers the ground in 14 classes, while EUNIS describes habitats. The relation is therefore many-to-many, and each OCS-GE class is given a ranked list of EUNIS classes. Rows with a land use code (`US`) refine the match for a given cover: for instance herbaceous cover (`CS2.2.1`) is arable land (`I1`) under agricultural use, but gardens and parks (`I2`) under residential use.

Usage

```
data(ref_ocsge_eunis)
```

Format

A data frame with 152 rows and 7 columns:

code_cs Character. OCS-GE cover code, as in `ref_ocsge$code_cs` (e.g. `"CS2.2.1"`).

code_us Character. OCS-GE use code (e.g. `"US1.1"`), or `NA` for the cover-only match, to be used when the use is unknown or has no dedicated row.

code_eunis Character. EUNIS Level 2 code, as in `ref_eunis$code` and `eunis_species_dict$code_eunis`.

rank Integer. Order of the EUNIS classes within a (`code_cs`, `code_us`) group; `1` is the default match.

main_cs Logical. `TRUE` on the single row giving the main OCS-GE cover of each EUNIS class (EUNIS to OCS-GE direction, 45 rows).

pct_metaleurop Numeric. Measured share (%) of the OCS-GE group area falling in this EUNIS class on the Metaleurop site; `NA` when the group covers less than 10 ha there. See Details.

note Character. What the match stands for, and what extra data would be needed to resolve it.

Details

`rank` is an expert reading of the two nomenclatures (CNIG OCS-GE specification and EUNIS habitat definitions). `pct_metaleurop` is a measurement: OCS-GE polygons of `roi_metaleurop` were overlaid on the EUNIS raster returned by `get_eunis_data`. The two columns are not expected to agree closely, because the EUNIS raster has a 100 m pixel carrying one dominant class, while OCS-GE maps objects a few meters wide: a road or a garden usually falls in a pixel classified as buildings (`J1`, `J2`). The measurement also only covers the classes present on this site.

Some EUNIS distinctions cannot be recovered from OCS-GE, with or without the use: wetlands (`D`), grassland types (`E1` to `E4`), standing versus running water (`C1`, `C2`), deciduous versus evergreen broadleaved woodland (`G1`, `G2`), and crops versus permanent grassland, which share the same cover and the same agricultural use.

The source table and the build script are `raw_data/ocsge_eunis_crosswalk.csv` and `raw_data/build_ref_ocsge_eunis` of the source repository.

See Also

ref_ocsge, ref_eunis, eunis_species_dict, get_ocsge_data_fgb

Examples

```
data(ref_ocsge_eunis)

# default EUNIS class of each OCS-GE cover
subset(ref_ocsge_eunis, is.na(code_us) & rank == 1)

# main OCS-GE cover of each EUNIS class
subset(ref_ocsge_eunis, main_cs, c(code_eunis, code_cs))

# EUNIS class of OCS-GE polygons from cover and use, falling back on cover alone
ocs <- data.frame(code_cs = c("CS2.2.1", "CS2.2.1", "CS2.1.1.1"),
                  code_us = c("US1.1", "US5", "US1.2"))
best <- subset(ref_ocsge_eunis, rank == 1)
with_us <- best$code_eunis[match(paste(ocs$code_cs, ocs$code_us),
                                 paste(best$code_cs, best$code_us))]
cs_only <- best$code_eunis[match(paste(ocs$code_cs, NA),
                                 paste(best$code_cs, best$code_us))]
ocs$code_eunis <- ifelse(is.na(with_us), cs_only, with_us)
ocs
```

---

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)
```

soil_grid *Build a regular raster grid around sampling points*

Description

Creates an empty ‘SpatRaster‘ covering the bounding box of ‘points‘, enlarged by ‘margin‘ on each side, with square cells of size ‘res‘. The extent is aligned on multiples of ‘res‘.

Usage

```
soil_grid(points, res = 25, margin = 1000)
```

Arguments

`points`	An ‘sf‘ / ‘sfc‘ object (or any object accepted by [sf::st_bbox()]) in a projected CRS (metres).
`res`	Numeric. Cell size in metres (ground sampling distance).
`margin`	Numeric. Margin in metres added around the points.

Value

An empty ‘SpatRaster‘.

Examples

```
data(soil_metaleurop)
soil_grid(soil_metaleurop, res = 25, margin = 1000)
```

`soil_metaleurop` *Soil metal concentrations sampled around the former Metaleurop smelter*

Description

Topsoil concentrations of cadmium (Cd), lead (Pb) and zinc (Zn) measured at 595 sampling points around the former Metaleurop Nord lead and zinc smelter (Noyelles-Godault, northern France). This is the data set used to build the soil rasters of the package (‘ground_concentration_*.tif‘; the cadmium raster is shipped in ‘inst/extdata‘), see the kriging article.

Simple feature collection (POINT), projected CRS: RGF93 v1 / Lambert-93 (EPSG:2154).

Usage

```
data(soil_metaleurop)
```

Format

An ‘sf‘ object with 595 rows and 9 columns:

id Sample identifier.

usage Land-use class of the sampled plot, as coded in the original survey (‘"A"‘, ‘"L"‘, ‘"U"‘).

dist Distance (m) to the smelter chimney.

angle Direction of the sample seen from the smelter (degrees clockwise from the North).

wind Wind-rose frequency (in %) of the 20-degree sector containing ‘angle‘ (18 sectors centred on 0, 20, ..., 340 degrees, summing to 100). The highest values lie north-east of the smelter, i.e. downwind of the dominant south-westerly winds.

cd, pb, zn Total concentrations in mg/kg of dry soil.

geometry Sampling location.

Examples

```
data(soil_metaleurop)
summary(soil_metaleurop[, c("cd", "pb", "zn")])
```

`soil_variogram` *Empirical semi-variogram of soil concentrations*

Description

Computes the experimental (empirical) semi-variogram of a soil concentration measured at sampling points: for every pair of points separated by a distance falling in a given distance class (a *lag*), half of the squared difference of their values is averaged.

When ‘covariates‘ are given, the variogram is computed on the residuals of the linear regression of the values on the covariates (this is the variogram needed for kriging with external drift, see [krige_soil()]).

Usage

```
soil_variogram(
  points,
  value,
  log10 = TRUE,
  covariates = NULL,
  cutoff = NULL,
  width = NULL,
  estimator = c("classical", "robust")
)
```

Arguments

`points`	An ‘sf‘ object of POINT geometries in a projected CRS (metres).
`value`	Character. Name of the column holding the concentration.
`log10`	Logical. Work on ‘log10(value)‘? (default ‘TRUE‘, recommended for skewed concentration data).
`covariates`	Optional ‘SpatRaster‘ whose layers are used as linear drift (trend) covariates.
`cutoff`	Numeric. Maximal distance (m) considered. Default: one third of the diagonal of the bounding box of the points.
`width`	Numeric. Width (m) of the distance classes. Default ‘cutoff / 15‘.
`estimator`	Character. ‘"classical"‘ (Matheron) or ‘"robust"‘ (Cressie-Hawkins, less sensitive to outliers).

Value

A ‘data.frame‘ of class ‘soil_variogram‘ with columns ‘np‘ (number of pairs), ‘dist‘ (mean distance of the pairs) and ‘gamma‘ (semi-variance).

See Also

[fit_soil_variogram()], [krige_soil()]

Examples

```
data(soil_metaleurop)
v <- soil_variogram(soil_metaleurop, "cd")
v
```

`soil_vgm` *Build a variogram model by hand*

Description

Creates a ‘soil_vgm‘ object from known parameters, e.g. to re-use a variogram fitted elsewhere (gstat, gstools, ...).

Usage

```
soil_vgm(model, psill, range, nugget = 0, kappa = 0.5)
```

Arguments

`model`	Character. One of ‘"Exp"‘, ‘"Sph"‘, ‘"Gau"‘, ‘"Mat"‘.
`psill`	Numeric. Partial sill.
`range`	Numeric. Range parameter (m).
`nugget`	Numeric. Nugget effect (default 0).
`kappa`	Numeric. Matérn smoothness (default 0.5).

Value

A ‘soil_vgm‘ object.

Examples

```
soil_vgm("Exp", psill = 0.1, range = 500, nugget = 0.02)
```

`source_covariates` *Distance and wind covariates around a contamination source*

Description

Builds covariate rasters describing the position of every cell relative to a point source of contamination (e.g. a smelter chimney). They can be used as external drift in [krige_soil()]:

‘log10_dist‘ the log10 of the distance (m) to the nearest source. Atmospheric deposition typically decreases as a power law of the distance, i.e. linearly in log-log scale.

‘wind‘ (only if ‘wind_rose‘ is given) the frequency of the wind sector in which the cell lies, seen from the source. Cells lying downwind of the dominant winds receive more deposition.

Usage

```
source_covariates(template, source, wind_rose = NULL, min_dist = NULL)
```

Arguments

`template`	A 'SpatRaster' defining the grid.
`source`	An 'sf' / 'sfc' POINT object: location(s) of the source(s). Must be a single point when 'wind_rose' is used.
`wind_rose`	Optional 'data.frame' with columns 'direction' (centre of the sector, in degrees clockwise from the North, as seen from the source) and 'freq' (wind frequency, any unit). Each cell gets the 'freq' of the closest sector direction.
`min_dist`	Numeric. Distances below this value are set to 'min_dist' (avoids 'log10(0)' at the source). Default: the cell size.

Value

A 'SpatRaster' with layers 'log10_dist' and, optionally, 'wind'.

Examples

```
data(soil_metaleurop)
grid <- soil_grid(soil_metaleurop, res = 100)
smelter <- sf::st_sfc(sf::st_point(c(701156, 7036799)), crs = 2154)
cov <- source_covariates(grid, smelter)
terra::plot(cov)
```

`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`

`SSD_ecoSSL_all` *Eco-SSL Toxicity Data for Multiple Taxonomic Groups*

Description

A comprehensive dataset combining toxicity values used for the derivation of Ecological Soil Screening Levels (Eco-SSL). It includes ecotoxicological data (NOAEL and LOAEL) for mammals, birds, invertebrates, and plants exposed to various chemical compounds.

Usage

```
SSD_ecoSSL_all
```

Format

A tibble (data frame) with 9 variables:

ERE Character. Abbreviation for the Ecological Receptor Endpoint or effect category (e.g., "BIO" for Biochemical, "BEH" for Behavior).

order_tox_value Integer. The sequential order or index of the toxicity value within its specific group.

tox_value_NOAEL Numeric. The No Observed Adverse Effect Level (NOAEL), typically expressed in mg/kg/day or mg/kg soil.

tox_value_LOAEL Numeric. The Lowest Observed Adverse Effect Level (LOAEL), typically expressed in mg/kg/day or mg/kg soil.

compound Character. The chemical compound or trace element evaluated (e.g., "Antimony", "Cadmium").

species_group Character. The broad taxonomic or functional group of the tested species (e.g., "Mammalian Wildlife").

test_organism Character. The common and/or scientific name of the specific organism tested (e.g., "Rat (Rattus norvegicus)").

tox_value Numeric. The primary toxicity value retained for the assessment (usually corresponds to the NOAEL or a derived threshold).

ERE_full Character. The full, unabbreviated name of the endpoint or effect category (e.g., "Biochemical", "Behavior").

Source

Derived from the United States Environmental Protection Agency (US EPA) Ecological Soil Screening Levels (Eco-SSL) database and documentation.

Examples

```
data(SSD_ecoSSL_all)
head(SSD_ecoSSL_all)
table(SSD_ecoSSL_all$species_group)
```

---

summary.soil_cv *Summarise a cross-validation*

Description

Summarise a cross-validation

Usage

```
## S3 method for class 'soil_cv'
summary(object, ...)
```

Arguments

`object`	A ‘soil_cv‘ object from [cv_soil()].
`...`	Not used.

Value

A one-row ‘data.frame‘ with ‘RMSE‘ (root mean squared error), ‘MAE‘ (mean absolute error), ‘bias‘ (mean error), ‘R2‘ (proportion of variance explained, $1 - SSE/SST$) and ‘n‘ (number of predicted points).

`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 flux functions.

Usage

```
transfer(
  spacemodel,
  kernels,
  fluxes = 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.
`fluxes`	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. Flux is computed using `apply_flux()`.
4. Contributions from all resources are summed.

The function assumes that flux 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")
```

Index

∗ **datasets**
  bappet_cd, 8
  DBFunc_BirdFuncDat, 13
  DBFunc_MamFuncDat, 14
  earthworm_cd, 17
  eunis_species_dict, 17
  FmrBT, 20
  load_raster_extdata, 37
  ocsge_metaleurop, 39
  ocsge_species_dict, 40
  ocsge_us_metaleurop, 42
  ref_eunis, 46
  ref_ocsge, 47
  ref_ocsge_eunis, 48
  roi_metaleurop, 49
  sf_micromammals, 50
  soil_metaleurop, 51
  SSD_ecoSSL_all, 55
[.trophic_tbl, 4

add_flux, 4
add_habitat, 5, *7*, *30, 31*
add_link, 6
add_nohabitat, *5*, 7
apply_flux, 8

bappet_cd, 8

calibrate_direct, 9
calibrate_simple, 9
calibrate_trophic, 10
compute_dispersal, 10, *16*
compute_kernel, 11
cv_soil, 11

DBFunc_BirdFuncDat, 13
DBFunc_MamFuncDat, 14
dispersal, 16

earthworm_cd, 17
eunis_species_dict, 17, *24*, *40, 41*, *49*

fit_soil_variogram, 19
flux, 20
FmrBT, 20

get_departements_for_roi, 21, *28*
get_eunis_data, *17, 18*, 22, *25, 26*, *46–48*
get_gbif_data, 23
get_ocsge, 25, *25*, *27*, *29*
get_ocsge_data_fgb, *22, 23*, *25*, 26, *29*, *42*, *49*
get_ocsge_data_ign, *25*, *27*, 28

habitat, 29
habitat_raster, *5*, *7*, *29, 30*, 31

is_cyclic, 32

join_ocsge_species, *26*, 33, *41, 42*, *45*

kernel_smooth_soil, 34
krige_soil, 35

levels, *23*
load_raster_extdata, 37
lower_neighbors, 38

normalize_weights, 38

ocsge_metaleurop, 39, *42*
ocsge_species_dict, *17, 18*, *24*, *34*, 40, *42*
ocsge_us_metaleurop, 42

plot.habitat, 42
plot.soil_vgm, 43
plot.trophic_tbl, 44
plot_species_habitat, 45

rast, *38*, *55*
raster_stack, 46
ref_eunis, *18*, *22, 23*, 46, *48, 49*
ref_ocsge, *41*, 47, *49*

ref_ocsge_eunis, *41*, 48
roi_metaleurop, *42*, *48*, 49

sampling, *9, 10*
sf, *22*, *24–29*
sf_micromammals, 50
soil_grid, 50
soil_metaleurop, 51
soil_variogram, 52
soil_vgm, 53
source_covariates, 53
spacemodel, 54
spacemodR *(*spacemodR-package*)*, 3
spacemodR-package, 3
SpatRaster, *11*, *23*, *38*, *46*, *54, 55*
SSD_ecoSSL_all, 55
summary.soil_cv, 56

transfer, 57
trophic, 58