

Package ‘r4pde’

September 7, 2026

Type Package

Title Tools for Quantitative Plant Disease Epidemiology

Version 0.2.2

Description Tools for quantitative plant disease epidemiology, including functional analysis of disease progress curves, spatial epidemiology, disease quantification, and weather-driven epidemic analysis. The package also serves as an educational resource and companion to the book “R for Plant Disease Epidemiology” (R4PDE). Several functions are based on classical and contemporary methods, including those discussed in Laurence V. Madden, Gareth Hughes, and Frank van den Bosch (2007) <[doi:10.1094/9780890545058](https://doi.org/10.1094/9780890545058)>.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Imports boot, car, cluster, dplyr, ggplot2, igraph, lubridate, mgcv, magrittr, nasapower, progress, purrr, rlang, stats, survival, tidyr, tibble, httr, jsonlite, terra

Suggests interval, testthat, knitr, rmarkdown, ggdendro, patchwork, cowplot, ncd4, ggrepel

Config/testthat/edition 3

Encoding UTF-8

LazyData true

VignetteBuilder knitr

RoxygenNote 7.3.3

BugReports <https://github.com/emdelponte/r4pde/issues>

URL <https://emdelponte.github.io/r4pde/>,
<https://github.com/emdelponte/r4pde>

NeedsCompilation no

Author Emerson Del Ponte [aut, cre] (ORCID:
<<https://orcid.org/0000-0003-4398-409X>>)

Maintainer Emerson Del Ponte <delponte@ufv.br>

Repository CRAN

Date/Publication 2026-09-07 17:50:02 UTC

Contents

AFSD	3
augment	4
augment.functional_rate	5
augment_functional_pca	5
augment_functional_rate	6
BlastWheat	6
BPL	7
BudBlightSoybean	8
CompMuCens	9
count_subareas	10
count_subareas_random	11
diagnose_curves	12
DidymellaWatermelon	12
DSI	13
DSI2	14
FHBWheat	14
FHBWheatBrazil	15
fit_gradients	16
functional_contrast	17
functional_curves	18
functional_distances	21
functional_instability	22
functional_pca	25
functional_pca_clusters	27
functional_rate	27
functional_resistance	30
functional_summary	32
functional_suppression_profiles	33
FusariumBanana	36
get_brdwgd	37
get_era5	38
get_fpca_eigenfunctions	39
get_fpca_scores	40
get_fpca_variance	40
get_nasapower	41
join_count	42
oruns_test	43
oruns_test_boustrophedon	44
oruns_test_byrowcol	44
plot.functional_curves	45
plot.functional_rate	46
plot.functional_suppression_profiles	47
plot.functional_suppression_profiles_list	48
plot.r4pde_functional_pca	48
plot_AFSD	49
plot_curves	50

plot_dendrogram	51
plot_diagnostics	51
plot_dsp	52
plot_dsp_rank_heatmap	53
plot_functional_instability	53
print.functional_curves	54
print.functional_distances	54
print.functional_rate	55
print.functional_resistance	55
print.functional_suppression_profiles	56
print.functional_suppression_profiles_list	56
print.r4pde_curve_diagnostics	57
print.suggest_k	57
print.summary.functional_suppression_profiles	58
print.summary.functional_suppression_profiles_list	58
rank_dsp	59
reconstruct_curves	59
RustSoybean	60
simulate_dsp_data	60
SpatialAggregated	61
SpatialRandom	61
suggest_k	62
summary.functional_rate	64
summary.functional_suppression_profiles	64
summary.functional_suppression_profiles_list	65
theme_r4pde	65
WhiteMoldSoybean	66
windowpane	67
windowpane_tests	68
Index	70

AFSD

*Analysis of foci structure and dynamics (AFSD)***Description**

This function performs the analysis of a simple method introduced by Nelson (1996) and expanded by Laranjeira et al. (1998). The function assumes the dataframe supplied as input has columns 'x', 'y', and 'i', where 'x' and 'y' are spatial coordinates and 'i' is a disease indicator variable (1 if diseased, otherwise 0). The function performs several steps including filtering rows where 'i' is 1, converting to an adjacency matrix, and creating foci using igraph. It then calculates various statistics about the foci and returns these in a list.

Usage

AFSD(df)

Arguments

`df` A dataframe containing at least three columns: 'x', 'y', and 'i'. 'x' and 'y' represent spatial coordinates and 'i' is a disease indicator (1 if diseased, otherwise 0).

Value

A list containing: `cluster_summary2`: a dataframe summarizing the number and size of foci, and proportions of diseased plants. `cluster_df`: a dataframe containing foci information, including size and number of rows and columns in each foci. `df_clustered`: the original dataframe with an added 'focus_id' column, showing which foci each row belongs to.

See Also

Other Spatial analysis: [BPL\(\)](#), [count_subareas\(\)](#), [count_subareas_random\(\)](#), [fit_gradients\(\)](#), [join_count\(\)](#), [oruns_test\(\)](#), [oruns_test_boustrophedon\(\)](#), [oruns_test_byrowcol\(\)](#), [plot_AFSD\(\)](#)

Examples

```
# Generate a sample dataframe
set.seed(123)
df <- data.frame(x = sample(1:100, 500, replace = TRUE),
                 y = sample(1:100, 500, replace = TRUE),
                 i = sample(0:1, 500, replace = TRUE, prob = c(0.7, 0.3)))

# Perform the AFSD
result <- AFSD(df)
```

 augment

Augment generic for functional objects

Description

Augment generic for functional objects

Usage

```
augment(x, ...)
```

Arguments

`x` An object to augment.
`...` Additional arguments.

Value

An augmented data frame or tibble.

```
augment.functional_rate
```

Augment functional_rate

Description

Augment functional_rate

Usage

```
## S3 method for class 'functional_rate'  
augment(x, ...)
```

Arguments

x	An object of class "functional_rate".
...	Additional arguments.

Value

A tibble of the full rate evaluation grid with predictions, derivatives, and confidence intervals.

```
augment_functional_pca
```

Augment functional PCA

Description

Augment functional PCA

Usage

```
augment_functional_pca(x)
```

Arguments

x	An object of class "r4pde_functional_pca".
---	--

```
augment_functional_rate
```

Augment functional_rate (alias)

Description

Augment functional_rate (alias)

Usage

```
augment_functional_rate(x, ...)
```

Arguments

x An object of class "functional_rate".
... Additional arguments.

Value

A tibble of the full rate evaluation grid.

BlastWheat	<i>BlastWheat dataset</i>
------------	---------------------------

Description

Wheat blast dataset with severity and weather covariates.

Usage

```
BlastWheat
```

Format

A data frame with the following columns:

heading Date of heading
inc_mean Mean incidence
index_mean FHB index mean
latitude Latitude coordinate
location Experimental site name
longitude Longitude coordinate
state Brazilian state
study Study ID or code
year Crop year
yld_mean Mean yield

Source

Del Ponte Lab internal data

BPL

Binary Power Law Analysis for Spatial Disease Patterns

Description

This function calculates the Binary Power Law (BPL) parameters for spatial disease patterns, fits a linear model, and performs a hypothesis test for the slope.

Usage

```
BPL(data)
```

Arguments

data A data frame containing the following columns:

- **field**: The field identifier.
- **n**: The number of observations in each quadrat.
- **i**: The incidence count in each quadrat.

Details

The function performs the following steps:

1. Summarizes the data by field to calculate the total number of observations (`n_total`), mean incidence (`incidence_mean`), observed variance (`V`), and binomial variance (`Vbin`).
2. Log-transforms the variances.
3. Fits a linear model to the log-transformed variances.
4. Tests the hypothesis that the slope of the linear model is equal to 1.

Value

A list containing the following elements:

- **summary**: A data frame summarizing the input data by field, including total observations (`n_total`), mean incidence (`incidence_mean`), observed variance (`V`), and binomial variance (`Vbin`).
- **model_summary**: A summary of the linear model fitted to the log-transformed variances.
- **hypothesis_test**: The result of the hypothesis test for the slope being equal to 1.
- **ln_Ap**: The intercept of the linear model, representing the natural logarithm of the parameter $\lambda(A_p)$.
- **slope**: The slope of the linear model.

See Also

Other Spatial analysis: [AFSD\(\)](#), [count_subareas\(\)](#), [count_subareas_random\(\)](#), [fit_gradients\(\)](#), [join_count\(\)](#), [oruns_test\(\)](#), [oruns_test_boustrophedon\(\)](#), [oruns_test_byrowcol\(\)](#), [plot_AFSD\(\)](#)

Examples

```
# Example usage with a sample data frame
result <- BPL(FHBWheat)
print(result$summary)
print(result$model_summary)
print(result$hypothesis_test)
print(paste("ln(Ap):", result$ln_Ap))
print(paste("Slope (b):", result$slope))
```

BudBlightSoybean	<i>BudBlightSoybean dataset</i>
------------------	---------------------------------

Description

Soybean bud blight incidence in experimental blocks.

Usage

BudBlightSoybean

Format

A data frame with the following columns:

- block** Block number
- time** Time point of assessment
- treat** Treatment name
- y** Incidence or severity value

Source

Del Ponte Lab internal data

CompMuCens	<i>Survival analysis for quantitative ordinal scale data</i>
------------	--

Description

Survival analysis for quantitative ordinal scale data

Usage

CompMuCens(dat, scale, grade = TRUE, ckData = FALSE)

Arguments

dat	Data frame containing the data to be processed.
scale	A numeric vector indicating the scale or order of classes.
grade	Logical. If TRUE, uses the class value. If FALSE, uses the NPE (Non-Parametric Estimate).
ckData	Logical. If TRUE, returns the input data along with the results. If FALSE, returns only the results.

Details

This function supports analysis of quantitative ordinal scale data via interval-censored methods. The approach follows the workflow described by Chiang et al. (2023) and the reference implementation provided in the CompMuCens repository.

Value

A list containing the score statistic, hypothesis tests, adjusted significance level, and conclusion based on pairwise comparisons.

References

Chiang, K.S., Chang, Y.M., Liu, H.I., Lee, J.Y., El Jarroudi, M. and Bock, C. (2023). Survival Analysis as a Basis to Test Hypotheses When Using Quantitative Ordinal Scale Disease Severity Data. Phytopathology. <https://apsjournals.apsnet.org/doi/abs/10.1094/PHYTO-02-23-0055-R>

See Also

Other Disease quantification: [DSI\(\)](#), [DSI2\(\)](#)

Examples

```
if (requireNamespace("interval", quietly = TRUE)) {
  trAs <- c(5,4,2,5,5,4,4,2,5,2,2,3,4,3,2,2,6,2,2,4,2,4,2,4,5,3,4,2,2,3)
  trBs <- c(5,3,2,4,4,5,4,5,4,4,6,4,5,5,5,2,6,2,3,5,2,6,4,3,2,5,3,5,4,5)
  trCs <- c(2,3,1,4,1,1,4,1,1,3,2,1,4,1,1,2,5,2,1,3,1,4,2,2,2,4,2,3,2,2)
  trDs <- c(5,5,4,5,5,6,6,4,6,4,3,5,5,6,4,6,5,6,5,4,5,5,5,3,5,6,5,5,5,6)
  inputData <- data.frame(
    treatment = c(rep("A",30), rep("B",30), rep("C",30), rep("D",30)),
    x = c(trAs, trBs, trCs, trDs)
  )
  CompMuCens(dat = inputData,
    scale = c(0,3,6,12,25,50,75,88,94,97,100,100),
    ckData = TRUE)
}
```

count_subareas

Count the Number of Ones in Subareas of a Matrix

Description

This function takes a binary matrix (0s and 1s) and divides it into rectangular subareas, counting the number of ones in each. Subareas are defined by the number of rows and columns specified by the user. If the matrix dimensions are not perfectly divisible by the subarea size, edge subareas may be smaller.

Usage

```
count_subareas(matrix_data, sub_rows, sub_cols)
```

Arguments

matrix_data	A matrix of 0s and 1s to analyze.
sub_rows	Number of rows in each subarea.
sub_cols	Number of columns in each subarea.

Value

A matrix where each cell corresponds to a subarea and contains the count of ones.

See Also

Other Spatial analysis: [AFSD\(\)](#), [BPL\(\)](#), [count_subareas_random\(\)](#), [fit_gradients\(\)](#), [join_count\(\)](#), [oruns_test\(\)](#), [oruns_test_boustrophedon\(\)](#), [oruns_test_byrowcol\(\)](#), [plot_AFSD\(\)](#)

Examples

```
set.seed(123)
mat <- matrix(sample(c(0, 1), 12 * 16, replace = TRUE), nrow = 16, ncol = 12)
count_matrix <- count_subareas(mat, sub_rows = 3, sub_cols = 3)
print(count_matrix)
```

count_subareas_random *Random Subgrid Sampling of a Binary Matrix*

Description

Randomly samples submatrices (quadrats) of specified size from a binary matrix, and returns the positions, submatrices, and count of 1s in each sampled quadrat.

Usage

```
count_subareas_random(matrix_data, sub_rows = 3, sub_cols = 3, n_samples = 100)
```

Arguments

matrix_data	A binary matrix of 0s and 1s.
sub_rows	Number of rows in each subgrid sample.
sub_cols	Number of columns in each subgrid sample.
n_samples	Number of subgrid samples to draw.

Value

A list of sampled subgrids. Each element is a list with:

position	Row and column start position of the sample.
submatrix	The sampled subgrid matrix.
count	Number of 1s in the sampled submatrix.

See Also

Other Spatial analysis: [AFSD\(\)](#), [BPL\(\)](#), [count_subareas\(\)](#), [fit_gradients\(\)](#), [join_count\(\)](#), [oruns_test\(\)](#), [oruns_test_boustrophedon\(\)](#), [oruns_test_byrowcol\(\)](#), [plot_AFSD\(\)](#)

diagnose_curves	<i>Diagnostic tools for functional epidemic curve models</i>
-----------------	--

Description

Diagnostic tools for functional epidemic curve models fitted with `functional_curves()`. Produces diagnostic plots without invoking base graphics.

Usage

```
diagnose_curves(x, grid_n = 200)
```

Arguments

<code>x</code>	An object of class "functional_curves".
<code>grid_n</code>	Number of points used for the diagnostic smooth curve.

Value

An object of class "r4pde_curve_diagnostics" containing:

- diagnostic data,
- k-index checks,
- concurvity estimates,
- ggplot-based diagnostic panels.

DidymellaWatermelon	<i>DidymellaWatermelon dataset</i>
---------------------	------------------------------------

Description

Assessment of Didymella symptoms in watermelon plots.

Usage

```
DidymellaWatermelon
```

Format

A data frame with:

EW_row	Row position (east–west)
NS_col	Column position (north–south)
dap	Days after planting
severity	Disease severity

Source

Del Ponte Lab internal data

DSI

Calculate the Disease Severity Index (DSI) (class for each unit)

Description

This function calculates the Disease Severity Index (DSI) based on the provided unit, class, and maximum class value. The DSI is computed by aggregating the classes, calculating weights by multiplying the frequency of each class by the class itself, and then dividing the sum of these weights by the product of the total number of entries and the maximum class value, then multiplying by 100.

Usage

```
DSI(unit, class, max)
```

Arguments

unit	A vector representing the units.
class	A vector representing the classes corresponding to the units.
max	A numeric value representing the maximum possible class value.

Value

Returns a single numeric value representing the DSI.

See Also

Other Disease quantification: [CompMuCens\(\)](#), [DSI2\(\)](#)

Examples

```
# Example usage:
unit <- c(1, 2, 3, 4, 5, 6)
class <- c(1, 2, 1, 2, 3, 1)
max <- 3
DSI(unit, class, max)
```

DSI2	<i>Calculate the Disease severity Index (DSI) (frequency of each class)</i>
------	---

Description

This function calculates the Disease Severity Index (DSI) given a vector of classes, a vector of frequencies, and a maximum possible class value. The DSI is calculated as a weighted sum of class values, where each class is multiplied by its corresponding frequency, then divided by the product of the total frequency and maximum class value, and finally multiplied by 100 to get a percentage.

Usage

```
DSI2(class, freq, max)
```

Arguments

class	A numeric vector representing the classes.
freq	A numeric vector representing the frequency of each class. Must be the same length as 'class'.
max	A numeric value representing the maximum possible class value.

Value

Returns a single numeric value representing the DSI.

See Also

Other Disease quantification: [CompMuCens\(\)](#), [DSI\(\)](#)

Examples

```
DSI2(c(0, 1, 2, 3, 4), c(2, 0, 5, 0, 5), 4)
```

FHBWheat	<i>FHBWheat dataset</i>
----------	-------------------------

Description

Fusarium head blight quadrat assessments in wheat.

Usage

```
FHBWheat
```

Format

A data frame with:

field Field identifier
i Row position
n Column position
quadrat Quadrat ID
season Crop season

Source

Del Ponte Lab internal data

FHBWheatBrazil	<i>FHBWheatBrazil dataset</i>
----------------	-------------------------------

Description

Fusarium head blight (FHB) index in wheat in Brazil.

Usage

FHBWheatBrazil

Format

A data frame with the following columns:

study Study ID or code
year Crop year
location Experimental site name
state Brazilian state
lat Latitude coordinate
lon Longitude coordinate
cultivar Wheat cultivar
planting_date Planting date
flowering Flowering date
index FHB index

Source

Del Ponte Lab internal data

fit_gradients

Fit Gradient Models to Data

Description

This function fits three gradient models (exponential, power, and modified power) to given data. It then ranks the models based on their R-squared values and returns diagnostic plots for each model.

Usage

```
fit_gradients(data, C = 1)
```

Arguments

data	A dataframe containing the data, with columns "x" representing distances and "Y" representing the corresponding measurements or counts.
C	A constant to be used in the modified power model. Defaults to 1.

Value

A list containing:

data	The input data, which will include an additional column 'mod_x'.
results_table	A table of the model parameters and R-squared values.
plot_exponential	Diagnostic plot for the exponential model.
plot_power	Diagnostic plot for the power model.
plot_modified_power	Diagnostic plot for the modified power model.
plot_exponential_original	Plot of the original data with the exponential model fit.
plot_power_original	Plot of the original data with the power model fit.
plot_modified_power_original	Plot of the original data with the modified power model fit.

See Also

Other Spatial analysis: [AFSD\(\)](#), [BPL\(\)](#), [count_subareas\(\)](#), [count_subareas_random\(\)](#), [join_count\(\)](#), [oruns_test\(\)](#), [oruns_test_boustrophedon\(\)](#), [oruns_test_byrowcol\(\)](#), [plot_AFSD\(\)](#)

Examples

```
x <- c(0.8, 1.6, 2.4, 3.2, 4, 7.2, 12, 15.2, 21.6, 28.8)
Y <- c(184.9, 113.3, 113.3, 64.1, 25, 8, 4.3, 2.5, 1, 0.8)
grad1 <- data.frame(x = x, Y = Y)
library(ggplot2)
mg <- fit_gradients(grad1, C = 0.4)
mg$plot_power_original +
  labs(title = "", x = "Distance from focus (m)", y = "Count of lesions")
```

functional_contrast	<i>Functional Contrast (Disease Suppression Profiles)</i>
---------------------	---

Description

Calculates functional contrast curves (Disease Suppression Profiles - DSP) between a reference treatment (e.g., untreated control) and other fungicide treatments over time. The DSP is defined by default as:

$$DSP(t) = y_{control}(t) - y_{treatment}(t)$$

where $y(t)$ is the disease intensity at time t . The goal is to evaluate the intensity, timing, and persistence of the fungicide protection.

Usage

```
functional_contrast(
  data,
  time = "time",
  response = "response",
  treatment = "treatment",
  reference = "Control",
  group = NULL,
  smooth = FALSE,
  grid = NULL,
  contrast = c("difference", "relative"),
  keep_reference = FALSE,
  ...
)
```

Arguments

data	A data frame containing disease progress observations, or an object of class "functional_curves".
time	Character string naming the time variable.
response	Character string naming the response variable.
treatment	Character string naming the treatment variable.
reference	Character string specifying the reference treatment level. Default is "Control".

group	Optional character vector of grouping variables (e.g., <code>c("experiment", "location")</code>).
smooth	Logical; if TRUE, the function could apply smoothing (currently placeholder).
grid	Optional numeric vector of common time points for interpolation. If provided, curves are interpolated.
contrast	Character specifying the contrast type. Currently "difference" is implemented.
keep_reference	Logical; if TRUE, includes the reference treatment in the output (with DSP = 0). Default is FALSE.
...	Additional arguments.

Value

An object of class "functional_dsp" (inheriting from "tbl_df"), containing:

- grouping variables (if any)
- treatment variable
- time variable
- y_reference: disease intensity of the reference
- y_treatment: disease intensity of the treatment
- DSP: the computed contrast
- contrast_type: type of contrast used
- reference: name of the reference treatment

See Also

[functional_summary](#), [plot_dsp](#)

functional_curves	<i>Fit genotype-specific epidemic trajectories using GAM</i>
-------------------	--

Description

Fits a generalized additive model (GAM) to replicated disease progress curves, returning adjusted treatment mean curves evaluated on a common time grid.

Usage

```
functional_curves(
  data,
  time,
  response,
  treatment,
  environment = NULL,
  block = NULL,
  unit = NULL,
```

```

response_scale = c("proportion", "percent"),
eps = 1e-04,
min_points = 5,
grid_n = 140,
env_ref = NULL,
k_smooth = 10,
k_env = 4,
k_trt = 6,
gamma = 1.4,
discrete = TRUE,
family_try = c("betar", "quasibinomial"),
show_progress = TRUE,
covariates = NULL,
include_covariates = FALSE,
covariate_smooths = FALSE,
global_smooth = FALSE,
...
)

```

Arguments

<code>data</code>	A data.frame containing disease progress observations.
<code>time</code>	Character string naming the time variable.
<code>response</code>	Character string naming the response variable.
<code>treatment</code>	Character string naming the treatment/cultivar factor.
<code>environment</code>	Optional character string naming an environment factor.
<code>block</code>	Optional character string naming a blocking factor.
<code>unit</code>	Optional character string naming a unique experimental unit identifier.
<code>response_scale</code>	Character string specifying the response scale: "proportion" or "percent".
<code>eps</code>	Numeric small constant retained for API compatibility.
<code>min_points</code>	Minimum number of observations required per curve.
<code>grid_n</code>	Number of time points for evaluating predicted curves.
<code>env_ref</code>	Reference environment level used for adjusted predictions.
<code>k_smooth</code>	Basis dimension for the global smooth of time.
<code>k_env</code>	Basis dimension for the environment-specific smooth.
<code>k_trt</code>	Basis dimension for the treatment-specific smooth.
<code>gamma</code>	Penalization parameter passed to <code>mgcv::bam()</code> .
<code>discrete</code>	Logical; whether to use discrete (approximate) fitting.
<code>family_try</code>	Character string specifying the GAM family to try.
<code>show_progress</code>	Logical; whether to show progress.
<code>covariates</code>	Optional character vector of genotype-level covariates (e.g., <code>c("heading_group")</code>).
<code>include_covariates</code>	Logical; whether to include covariates as fixed effects in the model.

covariate_smooths	Logical; if TRUE and include_covariates = TRUE, adds smooth interactions <code>s(time, by=covariate)</code> for factor covariates.
global_smooth	Logical; if TRUE, fits a shared global curve plus treatment deviations. If FALSE (default), fits independent smooths per treatment.
...	Additional arguments.

Details

Genotype-level covariates are descriptors that do not vary within a genotype, such as phenological groups. For example, in wheat blast studies, a cultivar's `heading_group` (early, intermediate, late) can be supplied. This helps distinguish between true genetic resistance and phenological escape, as cultivars with different heading dates may encounter different infection-risk windows in the same environment. When `include_covariates = TRUE`, the model accounts for these covariates, yielding heading-adjusted functional resistance.

Value

An object of class "functional_curves" containing:

- `gam`: fitted GAM object and `family_used`;
- `curves`: environment-adjusted treatment mean curves on the common grid;
- `grid`: the time grid used;
- `observed_data`: the processed input data;
- `genotype_info`: a tibble with one row per genotype containing covariates;
- `vars`: variable names used;
- `settings`: model settings;
- `warnings_betar`: any warnings caught during beta fitting;
- `plot_mean`: ggplot object of the mean curves.

Examples

```
## Not run:
fc <- functional_curves(
  data = my_data,
  time = "time_var",
  response = "severity",
  treatment = "cultivar",
  environment = "env",
  covariates = c("heading_group"),
  include_covariates = TRUE,
  covariate_smooths = TRUE
)
plot(fc)

## End(Not run)
```

functional_distances *Compute pairwise functional distances and cluster curves*

Description

Computes L2 functional distances among treatment mean trajectories evaluated on a common time grid from a functional_curves object. Also computes curve-level distances, performs hierarchical clustering, and optionally performs permutation testing.

Usage

```
functional_distances(
  object,
  cluster_k = 4,
  hc_method = "ward.D2",
  test_factor = NULL,
  n_perm = 999,
  test_mode = c("auto", "none", "global", "global_pairwise"),
  min_strata_for_pairwise = 6,
  perm_unit = NULL,
  perm_strata = NULL,
  bootstrap = FALSE,
  boot_B = 399,
  boot_seed = 1,
  boot_ci = c(0.025, 0.975),
  show_progress = TRUE,
  curve_level = NULL,
  ...
)
```

Arguments

object	An object of class functional_curves.
cluster_k	Number of clusters used to cut the hierarchical tree.
hc_method	Clustering linkage method passed to stats::hclust().
test_factor	Optional a priori grouping used for a distance-based permutation test.
n_perm	Number of permutations for the distance-based test.
test_mode	Character; permutation test mode.
min_strata_for_pairwise	Minimum number of strata levels required for pairwise tests.
perm_unit	Optional character naming the unit at which group labels are permuted.
perm_strata	Optional character naming a factor defining restricted permutation strata.
bootstrap	Logical; if TRUE, compute bootstrap envelopes and distance summaries.
boot_B	Integer number of bootstrap replicates.

boot_seed	Integer seed for bootstrap resampling.
boot_ci	Length-2 numeric vector of quantiles for bootstrap intervals.
show_progress	Logical; whether to show progress.
curve_level	Logical; whether to compute curve-level distance matrix.
...	Additional arguments.

Value

An object of class "functional_distances" containing:

- distance: treatment-by-treatment functional distance matrix;
- distance_table: pairwise distances as a data frame;
- hc: hclust object;
- clusters: treatment cluster assignments;
- curve_distance: curve-by-curve distance matrix;
- test: permutation test results;
- bootstrap: bootstrap results if requested.

Examples

```
## Not run:
fd <- functional_distances(fc, cluster_k = 4)
print(fd)
plot_dendrogram(fd)
plot_curves(fd)

## End(Not run)
```

functional_instability

Normalized functional instability from fitted functional disease curves

Description

Computes normalized functional instability (NFI) for each treatment based on genotype-by-environment predicted curves extracted from a [functional_curves](#) object. Optionally, instability can be decomposed into spatial and temporal components if the environment identifier can be split into location and year.

Usage

```
functional_instability(x, ...)

## S3 method for class 'functional_curves'
functional_instability(
  x,
  n_time = 200,
  env_sep = NULL,
  env_names = c("location", "year"),
  return_curves = FALSE,
  ...
)

## S3 method for class 'functional_dsp'
functional_instability(
  x,
  env_sep = NULL,
  env_names = c("location", "year"),
  return_curves = FALSE,
  ...
)
```

Arguments

x	An object returned by functional_curves or a functional_dsp object.
...	Additional arguments.
n_time	Number of points in the prediction grid over the time domain.
env_sep	Optional separator used to split env into spatial and temporal components, for example "_" or "-". If NULL, only the overall NFI is returned.
env_names	A character vector of length 2 giving the names of the components obtained after splitting env. Default is c("location", "year").
return_curves	Logical. If TRUE, the predicted genotype-by-environment curves are also returned.

Details

The overall instability metric is defined as the mean integrated squared deviation of each genotype-by-environment curve from the genotype-specific mean curve across environments, normalized by the integrated squared mean curve.

Let $f_{ge}(t)$ denote the predicted epidemic trajectory of genotype g in environment e , and let $\bar{f}_g(t)$ denote the mean trajectory of genotype g across environments. Functional instability is computed as:

$$FI_g = \frac{1}{E_g} \sum_{e=1}^{E_g} \int_T (f_{ge}(t) - \bar{f}_g(t))^2 dt$$

and normalized as:

$$nFI_g = \frac{FI_g}{\int_T \bar{f}_g(t)^2 dt}$$

Numerical integration is performed with the trapezoidal rule on a regular prediction grid over the observed time domain.

Value

If `return_curves = FALSE`, a tibble with one row per genotype and columns:

geno Treatment or genotype identifier.

n_env Number of environments used in the calculation.

FI Absolute functional instability.

mean_energy Integrated squared mean epidemic curve.

nFI Normalized functional instability. Lower values indicate greater stability.

If `env_sep` is provided, the returned tibble also includes:

FI_space, mean_energy_space, nFI_space Spatial component of instability.

FI_time, mean_energy_time, nFI_time Temporal component of instability.

If `return_curves = TRUE`, a list is returned with two elements:

metrics The tibble described above.

curves A tibble of predicted genotype-by-environment curves with columns `geno`, `env`, `time`, `mu`, and `eta`.

See Also

[functional_curves](#)

Examples

```
## Not run:
m1 <- r4pde::functional_curves(
  data = dat_ready,
  time = "time",
  response = "y",
  treatment = "geno",
  environment = "env"
)

# Overall instability
res <- functional_instability(m1)
res

# Overall + spatial and temporal components
# Assuming env has values such as "PF_2021"
```

```

res2 <- functional_instability(m1, env_sep = "_")
res2

# Return curves used in the calculations
out <- functional_instability(m1, env_sep = "_", return_curves = TRUE)
out$metrics
head(out$curves)

## End(Not run)

```

functional_pca	<i>Functional principal component analysis of disease progress curves</i>
----------------	---

Description

Performs functional principal component analysis on fitted disease progress curves returned by [functional_curves](#). The function decomposes variation among epidemic trajectories into orthogonal temporal components and returns curve-level scores, eigenfunctions, variance explained, and reconstructed curves.

Usage

```

functional_pca(object, ...)

## S3 method for class 'functional_curves'
functional_pca(
  object,
  n_components = NULL,
  var_explained = 0.95,
  center = TRUE,
  scale = FALSE,
  method = c("pca_on_grid"),
  ...
)

## S3 method for class 'functional_dsp'
functional_pca(
  object,
  n_components = NULL,
  var_explained = 0.95,
  center = TRUE,
  scale = FALSE,
  method = c("pca_on_grid"),
  ...
)

```

Arguments

object	An object returned by <code>functional_curves</code> .
...	Additional arguments for future extensions.
n_components	Optional integer number of functional principal components to retain.
var_explained	Cumulative variance threshold used when n_components = NULL.
center	Logical; whether to center curves before PCA. Default TRUE.
scale	Logical; whether to scale grid columns before PCA. Default FALSE.
method	Character; method for FPCA, currently only "pca_on_grid" is supported.

Details

The function uses the fitted curves from `functional_curves()` and does not refit the disease progress model. The first implementation uses PCA on a common prediction grid. FPC scores can be analyzed as functional epidemiological traits in downstream models.

Interpretation:

- FPC1 often captures the largest mode of variation, commonly overall epidemic intensity or speed.
- Later FPCs may capture timing of disease onset, curve crossing, late acceleration, or other shape-related deviations.
- Interpretation must be data-driven and should be based on eigenfunction plots and mean +/- perturbation plots.

Value

An object of class "r4pde_functional_pca" containing:

- scores: Tibble of curve-level FPC scores.
- eigenfunctions: Tibble of eigenfunction values across the time grid.
- variance: Tibble of eigenvalues, variance explained, and cumulative variance.
- mean_curve: Tibble of the mean curve.
- reconstructed: Tibble of reconstructed curves using retained components.
- input_curves: Tibble of original fitted curves.
- pca: The underlying prcomp object.
- settings: List of settings used.
- call: The matched call.

Examples

```
## Not run:
curves <- functional_curves(...)
fpca <- functional_pca(curves, var_explained = 0.95)

print(fpca)
plot(fpca, type = "scree")
```

```

plot(fpca, type = "components")
plot(fpca, type = "scores", components = c(1, 2))

scores <- get_fpca_scores(fpca)

## End(Not run)

```

functional_pca_clusters

Cluster curves based on FPCA scores

Description

Cluster curves based on FPCA scores

Usage

```

functional_pca_clusters(
  x,
  k = NULL,
  components = NULL,
  method = c("kmeans", "hclust"),
  choose_k = c("none", "silhouette", "elbow"),
  ...
)

```

Arguments

x	An object of class "r4pde_functional_pca".
k	Number of clusters.
components	Integer vector of components to use.
method	Clustering method: "kmeans" or "hclust".
choose_k	Method for suggesting k: "none", "silhouette", or "elbow".
...	Additional arguments passed to clustering functions.

functional_rate

Instantaneous Disease Progress Rates from Functional Curves

Description

Estimates instantaneous rates of plant disease progress ($S'(t) = dS(t)/dt$) and associated uncertainty from epidemic trajectories fitted by [functional_curves](#). Derivatives are computed using the GAM linear predictor matrix (`lpmatrix`) with finite differences, supporting both the original response scale (e.g., severity proportion/percentage points per day) and the link (linear predictor) scale. Also computes epidemic rate phenotypes including maximum rate (r_{max}), time of maximum rate ($t_{r_{max}}$), growth duration, cumulative positive growth, and distinct growth windows.

Usage

```
functional_rate(
  object,
  scale = c("response", "link"),
  method = c("central"),
  n_grid = 200,
  eps = NULL,
  interval = 0.95,
  threshold = 0,
  positive_only = TRUE,
  uncertainty = TRUE,
  growth_criterion = c("rate", "significant"),
  newdata = NULL,
  ...
)
```

Arguments

object	An object of class "functional_curves".
scale	Character specifying the derivative scale: "response" (default; e.g., severity change per unit time) or "link" (linear predictor scale).
method	Character specifying the finite difference method. Currently "central" (central difference in the interior with one-sided differences at temporal boundaries).
n_grid	Integer; number of points in the regular temporal evaluation grid per group. Default is 200.
eps	Numeric step size used in finite differences. If NULL (default), automatically chosen as $10^{-4} \times (t_{max} - t_{min})$.
interval	Numeric confidence level for intervals. Default is 0.95.
threshold	Numeric minimum rate threshold considered biologically relevant. Default is 0.
positive_only	Logical; whether to restrict summarized growth descriptors to positive rates. Default is TRUE. Note that negative rates are always preserved in the detailed rate table (data).
uncertainty	Logical; whether to compute standard errors and confidence intervals using the GAM covariance matrix. Default is TRUE.
growth_criterion	Character specifying how active epidemic growth is defined: "rate" (default; $rate > threshold$) or "significant" ($rate_lower > threshold$, requiring <code>uncertainty = TRUE</code>).
newdata	Optional user-supplied data frame with custom evaluation points. Must contain the time and treatment variables.
...	Additional arguments passed to methods.

Details

Mathematical Formulation: Let $S(t)$ denote the fitted disease trajectory and $\eta(t) = X(t)\beta$ the linear predictor from the GAM model, with $\mu(t) = g^{-1}(\eta(t))$.

The derivative on the link scale is estimated by:

$$D = \frac{X(t + \epsilon) - X(t - \epsilon)}{2\epsilon}$$

$$\eta'(t) = D\beta$$

with variance $\text{Var}(\eta'(t)) = \text{diag}(DV_p D^T)$, where V_p is the Bayesian posterior covariance matrix of the GAM parameters.

On the response scale (scale = "response"), the derivative of the mean trajectory is:

$$S'(t) = \frac{\mu(\eta(t + \epsilon)) - \mu(\eta(t - \epsilon))}{2\epsilon}$$

The approximate gradient vector G with respect to β is:

$$G = \frac{\mu'(\eta(t + \epsilon))X(t + \epsilon) - \mu'(\eta(t - \epsilon))X(t - \epsilon)}{2\epsilon}$$

where $\mu'(\eta) = d\mu/d\eta$ is given by `family$mu.eta()`. The uncertainty is then:

$$\text{Var}(S'(t)) = \text{diag}(GV_p G^T)$$

Boundary Handling: At the observed boundaries t_{min} and t_{max} , one-sided forward and backward differences are used to prevent unintended extrapolation beyond the observed domain.

Time-Invariant Terms and Random Effects: Intercepts, main treatment effects, and terms constant with respect to time cancel analytically in $X(t + \epsilon) - X(t - \epsilon)$. Environmental and experimental unit random effects are excluded by default to yield population-adjusted treatment rates, consistent with [functional_curves](#).

No-Epidemic and Flat Curves: Curves with all observed response values equal to zero (or estimated rates below numerical tolerance 10^{-6}) explicitly return $r_{max} = 0$, $t_{r_{max}} = \text{NA}$, $\text{growth_duration} = 0$, and $\text{cumulative_positive_growth} = 0$.

Value

An object of class "functional_rate" containing:

data A tibble of full evaluation grid with columns including treatment, time, fitted, fitted_se, fitted_lower, fitted_upper, rate, rate_se, rate_lower, rate_upper, significant_increase, above_threshold, and significant_above_threshold.

summary A tibble of curve-level rate phenotypes: r_max, t_r_max, r_mean, r_mean_positive, t_growth_start, t_growth_end, growth_duration, cumulative_positive_growth, and n_growth_windows.

model The underlying GAM model object.

call The matched call.

settings List of analysis settings used.

vars List of variable names identified in the model.

family Character string naming the GAM family.

References

- Madden, L. V., Hughes, G., & van den Bosch, F. (2007). The Study of Plant Disease Epidemics. APS Press, St. Paul, MN.
- Wood, S. N. (2017). Generalized Additive Models: An Introduction with R (2nd ed.). Chapman and Hall/CRC.
- Simpson, G. L. (2018). Modelling Palaeoecological Time Series Using Generalized Additive Models. *Frontiers in Ecology and Evolution*, 6, 149.

Examples

```
## Not run:
set.seed(123)
df <- data.frame(
  time = rep(1:30, 3),
  treatment = rep(c("Early", "Late", "None"), each = 30),
  y = c(
    1 / (1 + exp(-(1:30 - 10) * 0.4)), # Early
    1 / (1 + exp(-(1:30 - 20) * 0.4)), # Late
    rep(0, 30)                        # None
  )
)

curves <- functional_curves(
  data = df, time = "time", response = "y", treatment = "treatment",
  min_points = 3, show_progress = FALSE, family_try = "quasibinomial"
)

rates <- functional_rate(curves)
print(rates)
summary(rates)
plot(rates)

## End(Not run)
```

functional_resistance *Compute functional resistance and stability-adjusted functional resistance*

Description

Computes a functional resistance index (FRI) relative to a reference genotype curve. Optionally adjusts FRI by a functional instability penalty to calculate a stability-adjusted functional resistance index (SAFRI). Supports grouping genotypes into classes based on quantiles, clustering, or bootstrap-supported differences.

Usage

```
functional_resistance(
  object,
  instability = NULL,
  reference,
  lambda = 1,
  method = c("positive_area", "l2_difference"),
  scale_nfi = FALSE,
  n_groups = 4,
  group_method = c("none", "quantile", "bootstrap", "clustering"),
  time_var = NULL,
  genotype_var = NULL,
  n_boot = 1000,
  ci_level = 0.95,
  clustering_method = c("hclust", "kmeans"),
  adjust_by = NULL,
  reference_within_group = FALSE,
  group_within_adjust_by = TRUE,
  ...
)
```

Arguments

<code>object</code>	An object of class <code>functional_curves</code> .
<code>instability</code>	Optional output from <code>functional_instability()</code> .
<code>reference</code>	Character string naming the reference genotype, or a named vector if <code>reference_within_group = TRUE</code> .
<code>lambda</code>	Numeric penalty weight for instability. Default is 1.
<code>method</code>	Character string for distance method.
<code>scale_nfi</code>	Logical; if TRUE, normalizes nFI to a 0-1 scale before applying penalty.
<code>n_groups</code>	Integer number of classes to group into.
<code>group_method</code>	Character string for the grouping method.
<code>time_var</code>	Optional variable name overrides.
<code>genotype_var</code>	Optional variable name overrides.
<code>n_boot</code>	Integer number of bootstrap iterations.
<code>ci_level</code>	Numeric confidence level for bootstrap intervals.
<code>clustering_method</code>	Character string for clustering method if <code>group_method = "clustering"</code> .
<code>adjust_by</code>	Optional character string naming a genotype-level covariate to stratify by.
<code>reference_within_group</code>	Logical; if TRUE, the reference is evaluated within each <code>adjust_by</code> group.
<code>group_within_adjust_by</code>	Logical; if TRUE, resistance classes are assigned within each <code>adjust_by</code> group.
<code>...</code>	Additional arguments.

Value

A list with a table containing genotype, FRI, nFI, SAFRI, rank, and classes, and optionally bootstrap results.

Examples

```
## Not run:
fr <- functional_resistance(
  fc,
  reference = "Susceptible",
  group_method = "bootstrap"
)
print(fr)

## End(Not run)
```

functional_summary	<i>Summarize Disease Suppression Profiles</i>
--------------------	---

Description

Extracts functional descriptors from Disease Suppression Profiles (DSPs).

Usage

```
functional_summary(
  object,
  time = "time",
  response = "DSP",
  treatment = "treatment",
  group = NULL,
  threshold = 0,
  positive_only = TRUE
)
```

Arguments

object	An object of class "functional_dsp" or a data frame with a DSP column.
time	Character string naming the time column.
response	Character string naming the DSP column.
treatment	Character string naming the treatment column.
group	Optional character vector of grouping variables.
threshold	Numeric; minimum suppression threshold for calculating persistence. Default is 0.
positive_only	Logical; if TRUE, negative DSP values are set to 0 before area/centroid calculations. Default is TRUE.

Value

A tibble with one row per group/treatment containing:

- `protected_area`: Trapezoidal integral of $DSP(t)$.
- `mean_suppression`: Temporal mean of $DSP(t)$.
- `max_suppression`: Maximum value of $DSP(t)$.
- `time_max_suppression`: Time at which maximum suppression occurs.
- `persistence`: Total duration where $DSP(t) > \text{threshold}$.
- `centroid`: Temporal centroid defined as $\int t \cdot DSP(t)dt / \int DSP(t)dt$.
- `energy`: Integral of $DSP(t)^2$.
- `early_area`, `mid_area`, `late_area`: Protected area in the first, second, and third tertiles of the time period.
- `late_decline_slope`: Slope of $DSP(t)$ in the late period.
- `n_time_points`: Number of time points evaluated.

Examples

```
## Not run:
# Using a generic data frame with custom column names
dat <- data.frame(
  DAA = c(0, 10, 20),
  fungicide = c("A", "A", "A"),
  contrast = c(0, 10, 0)
)
functional_summary(
  dat,
  time = "DAA",
  response = "contrast",
  treatment = "fungicide"
)

## End(Not run)
```

functional_suppression_profiles

Functional suppression profiles of disease control treatments

Description

Computes disease suppression trajectories relative to a reference treatment, estimates functional distances among suppression curves, clusters treatments according to their temporal suppression profiles, and summarizes each profile using functional suppression metrics.

Usage

```
functional_suppression_profiles(
  data,
  reference,
  time = "time",
  response = "severity",
  treatment = "treatment",
  environment = NULL,
  by_environment = FALSE,
  env_ref = NULL,
  threshold = 5,
  metrics = c("protected_area", "max_suppression", "persistence", "centroid"),
  dist_method = "euclidean",
  hclust_method = "average",
  k = "auto",
  cut_height = NULL,
  ...
)
```

Arguments

<code>data</code>	A data frame or tibble containing the disease progress data.
<code>reference</code>	Character string specifying the reference treatment name.
<code>time</code>	Character string specifying the time column. Default is "time".
<code>response</code>	Character string specifying the response column. Default is "severity".
<code>treatment</code>	Character string specifying the treatment column. Default is "treatment".
<code>environment</code>	Character string specifying the environment column (optional).
<code>by_environment</code>	Logical; if TRUE, the analysis is run separately for each environment. Default is FALSE.
<code>env_ref</code>	Character string specifying the reference environment for joint analysis (optional).
<code>threshold</code>	Numeric threshold for the persistence metric. Default is 5.
<code>metrics</code>	Character vector of metrics to include in the ranking. Default is c("protected_area", "max_suppression", "persistence", "centroid").
<code>dist_method</code>	Character string specifying the distance method. Default is "euclidean".
<code>hclust_method</code>	Character string specifying the hierarchical clustering method. Default is "average".
<code>k</code>	Integer specifying the number of functional profiles to create, or "auto" (default). When "auto", the optimal number of profiles is estimated by maximizing the average silhouette width.
<code>cut_height</code>	Numeric specifying the cut height for clustering. If provided, overrides k.
<code>...</code>	Additional arguments passed to functional_contrast .

Details

A Functional Suppression Profile (FSP) is the temporal trajectory of disease suppression produced by a treatment relative to an untreated or reference control. Functional distances and hierarchical clustering are computed from smoothed disease suppression trajectories rather than raw observed suppression values. Functional summary metrics are subsequently used to interpret the resulting suppression profiles.

The workflow includes contrasting treatments against a reference, fitting functional curves, calculating distances, clustering, and summarizing the suppression using metrics like protected area, maximum suppression, and persistence.

Treatments belonging to the same functional suppression profile are displayed using the same colour throughout all graphical summaries. This visual consistency facilitates interpretation of the relationship between suppression trajectories, functional clustering, and suppression metrics.

The classification table combines functional profile membership, suppression metrics, and mean functional rank. Functional profiles are defined from distances among smoothed suppression trajectories, whereas mean rank summarizes the overall performance across functional suppression metrics.

Value

A list of class "functional_suppression_profiles" containing:

- `call`: The matched call.
- `reference`: The reference treatment name.
- `contrast`: The contrast data from `functional_contrast`.
- `dsp_curves`: The fitted curves from `functional_curves`.
- `distances`: The distances object from `functional_distances`.
- `hclust`: The hierarchical clustering object.
- `clusters`: A tibble with treatment cluster assignments (internal).
- `profiles`: A lightweight tibble mapping treatments to profiles.
- `classification`: A tibble with the final classification combining profiles, mean rank, and functional metrics, ordered by mean rank.
- `summary`: A tibble with functional summary metrics.
- `ranking`: A tibble with metric rankings.
- `cluster_summary`: A tibble combining metrics, ranks, and cluster assignments.
- `parameters`: A list of input parameters.

See Also

[functional_contrast](#), [functional_curves](#), [functional_distances](#), [functional_summary](#), [rank_dsp](#), [plot_dendrogram](#)

Examples

```
## Not run:
sim_dat <- tibble::tibble(
  treatment = rep(c("Control", "A", "B", "C"), each = 6),
  time = rep(seq(0, 25, by = 5), times = 4),
  severity = c(
    c(5, 10, 20, 35, 50, 65),
    c(3, 5, 10, 18, 30, 40),
    c(4, 6, 12, 22, 35, 45),
    c(2, 4, 8, 14, 22, 32)
  )
)

fsp <- functional_suppression_profiles(
  data = sim_dat,
  reference = "Control",
  time = "time",
  response = "severity",
  treatment = "treatment",
  threshold = 5,
  k = 2
)

fsp
summary(fsp)
plot(fsp, type = "dendrogram")
plot(fsp, type = "profiles")
plot(fsp, type = "heatmap")
plot(fsp, type = "rank")

## End(Not run)
```

FusariumBanana	<i>FusariumBanana dataset</i>
----------------	-------------------------------

Description

Observations of Fusarium symptoms in banana fields.

Usage

FusariumBanana

Format

- A data frame with:
- field** Field ID
 - lat** Latitude
 - lon** Longitude
 - marker** Infection marker presence

Source

Del Ponte Lab internal data

get_brdwgd

Fetch BR-DWGD Data for Multiple Locations

Description

This function extracts daily weather data from the Brazilian Daily Weather Gridded Data (BR-DWGD) NetCDF files for specified locations and dates. It mimics the functionality of `get_nasapower` and `get_era5` by returning a daily-aggregated data frame with columns named according to `nasapower` conventions.

Usage

```
get_brdwgd(
  data,
  days_around,
  date_col,
  study_col = "study",
  path = "netcdf_files",
  vars = c("pr", "Tmax", "Tmin", "Rs", "RH"),
  direction = "both"
)
```

Arguments

data	A data frame containing the input data, including columns for latitude, longitude, and the date column specified by <code>date_col</code> .
days_around	An integer specifying the number of days before and after the date in the date column to download data.
date_col	A character string specifying the name of the date column in the data frame.
study_col	A character string specifying the name of the column containing the study identifier in the input data frame (default: "study").
path	A character string specifying the directory where the BR-DWGD NetCDF files are stored.
vars	A character vector specifying the weather variables to fetch. These are expected to be the prefix of the NetCDF files (e.g., "pr", "Tmax"). Default: <code>c("pr", "Tmax", "Tmin", "Rs", "RH")</code> .
direction	Character string specifying the direction of the date range relative to the reference date. Options are "both" (default), "back", or "forth". "back" retrieves data from <code>date - days_around</code> to <code>date</code> . "forth" retrieves data from <code>date</code> to <code>date + days_around</code> . "both" retrieves data from <code>date - days_around</code> to <code>date + days_around</code> .

Details

The function requires the `terra` package to read NetCDF files and the `tidyr` package for data reshaping. It expects NetCDF files to be named starting with the variable name (e.g., `pr_*.nc`) and to contain the variable with the same name.

Value

A data frame (tibble) with daily weather data. Columns are named to match `nasapower` conventions where possible: `date`, `PRECTOTCORR` (from `pr`), `T2M_MAX` (from `Tmax`), `T2M_MIN` (from `Tmin`), `ALLSKY_SFC_SW_DWN` (from `Rs`), `RH2M` (from `RH`), `longitude`, `latitude`, and `study`.

See Also

Other Disease modeling: `get_era5()`, `get_nasapower()`, `windowpane()`

`get_era5`

Fetch ERA5 Data from Open-Meteo for Multiple Locations

Description

This function downloads ERA5 historical weather data from the Open-Meteo API for specified locations and dates. It mimics the functionality of `get_nasapower` by fetching weather variables and returning a daily-aggregated data frame. It uses the Open-Meteo Archive API and aggregates hourly data to daily values to ensure all variables (including relative humidity and dew point) are available.

Usage

```
get_era5(
  data,
  days_around,
  date_col,
  study_col = "study",
  pars = c("temperature_2m", "relative_humidity_2m", "precipitation", "dewpoint_2m"),
  models = NULL,
  direction = "both"
)
```

Arguments

<code>data</code>	A data frame containing the input data, including columns for <code>latitude</code> , <code>longitude</code> , and the date column specified by <code>date_col</code> .
<code>days_around</code>	An integer specifying the number of days before and after the date in the date column to download data.
<code>date_col</code>	A character string specifying the name of the date column in the data frame.

study_col	A character string specifying the name of the column containing the study identifier in the input data frame (default: "study").
pars	A character vector specifying the weather variables to fetch from Open-Meteo. These are mapped to nasapower equivalents: "temperature_2m" (T2M), "relative_humidity_2m" (RH2M), "precipitation" (PRECTOTCORR), etc.
models	A character string specifying the reanalysis model(s) to use. If NULL (default), Open-Meteo uses its "best_match" logic (ERA5, ERA5-Land, etc.).
direction	Character string specifying the direction of the date range relative to the reference date. Options are "both" (default), "back", or "forth". "back" retrieves data from date - days_around to date. "forth" retrieves data from date to date + days_around. "both" retrieves data from date - days_around to date + days_around.

Details

Since the Open-Meteo Archive API daily endpoint does not provide all variables (like mean relative humidity) directly, this function fetches hourly data and performs the daily aggregation manually:

- T2M: Mean of hourly temperature_2m
- T2M_MAX: Max of hourly temperature_2m
- T2M_MIN: Min of hourly temperature_2m
- RH2M: Mean of hourly relative_humidity_2m
- PRECTOTCORR: Sum of hourly precipitation
- T2MDEW: Mean of hourly dewpoint_2m

Value

A data frame (tibble) with daily weather data. Columns are named to match nasapower conventions: date, T2M, T2M_MAX, T2M_MIN, RH2M, PRECTOTCORR, T2MDEW, longitude, latitude, and study.

See Also

Other Disease modeling: [get_brdwgd\(\)](#), [get_nasapower\(\)](#), [windowpane\(\)](#)

```
get_fpca_eigenfunctions
```

Get FPCA eigenfunctions

Description

Get FPCA eigenfunctions

Usage

```
get_fpca_eigenfunctions(x, components = NULL)
```

Arguments

x	An object of class "r4pde_functional_pca".
components	Optional integer vector of components to return.

get_fpca_scores	<i>Get FPCA scores</i>
-----------------	------------------------

Description

Get FPCA scores

Usage

```
get_fpca_scores(x, components = NULL)
```

Arguments

x	An object of class "r4pde_functional_pca".
components	Optional integer vector of components to return.

get_fpca_variance	<i>Get FPCA variance explained</i>
-------------------	------------------------------------

Description

Get FPCA variance explained

Usage

```
get_fpca_variance(x)
```

Arguments

x	An object of class "r4pde_functional_pca".
---	--

get_nasapower

*Fetch NASA POWER Data for Multiple Locations with a Progress Bar***Description**

This function downloads daily NASA POWER data for specified weather variables over a specified number of days around a given date column for multiple locations. It includes a progress bar to show the download progress.

Usage

```
get_nasapower(
  data,
  days_around,
  date_col,
  study_col = "study",
  pars = c("T2M", "RH2M", "PRECTOTCORR", "T2M_MAX", "T2M_MIN", "T2MDEW"),
  direction = "both"
)
```

Arguments

data	A data frame containing the input data, including columns for latitude, longitude, and the date column.
days_around	An integer specifying the number of days before and after the date in the date column to download data.
date_col	A character string specifying the name of the date column in the data frame.
study_col	A character string specifying the name of the column containing the study identifier in the input data frame (default: "study").
pars	A character vector specifying the weather variables to fetch from NASA POWER (default: c("T2M", "RH2M", "PRECTOTCORR", "T2M_MAX", "T2M_MIN", "T2MDEW")).
direction	Character string specifying the direction of the date range relative to the reference date. Options are "both" (default), "back", or "forth". "back" retrieves data from date - days_around to date. "forth" retrieves data from date to date + days_around. "both" retrieves data from date - days_around to date + days_around.

Details

The function uses the `get_power` function from the `nasapower` package to fetch weather data for a range of dates around the specified date column for each location. A progress bar is shown during the data download process, and the results are combined into a single data frame.

Value

A data frame with the downloaded weather data from NASA POWER, combined for all specified locations. Includes a variable study indicating the study identifier from the input data. Returns an empty data frame if no data is retrieved.

See Also

Other Disease modeling: [get_brdwgd\(\)](#), [get_era5\(\)](#), [windowpane\(\)](#)

join_count	<i>Test for Spatial Join Count Statistics</i>
------------	---

Description

The function join_count calculates spatial join count statistics for a binary matrix, identifying patterns of aggregation or randomness.

Usage

```
join_count(matrix_data, verbose = TRUE)
```

Arguments

matrix_data	A binary matrix (with elements 0 and 1) representing the spatial distribution of two types of points: 0 for healthy plants (H) and 1 for diseased plants (D). This matrix reflects the geographical distribution or layout of plants in the studied area.
verbose	Logical. If TRUE (default), prints a formatted message to the console.

Details

The function conducts an analysis by first counting the occurrence of specific sequences ("01 or 10" and "11" - equivalent to HD and DD) in the binary matrix. It then calculates expected values, standard deviations, and Z-scores to determine the spatial randomness or aggregation. The analysis considers both horizontal and vertical adjacency (rook case) in the matrix.

Value

A comprehensive, rich-text formatted string of results that includes:

- Statistical counts of specific binary sequences (e.g., "01 or 10", "11")
- Expected counts under the assumption of Complete Spatial Randomness (CSR)
- Standard deviations and Z-scores (ZHD for "01 or 10" sequences, ZDD for "11" sequences)
- Interpretation of whether the spatial distribution for each sequence type is "Aggregated" or "Not Aggregated" based on Z-scores
- A summary explaining the implications of these statistics and patterns

The return value aims to provide a clear understanding of the spatial arrangement's characteristics, aiding in further spatial analysis or research.

References

Madden, L. V., Hughes, G., & van den Bosch, F. (2007). The Study of Plant Disease Epidemics. The American Phytopathological Society.

See Also

Other Spatial analysis: [AFSD\(\)](#), [BPL\(\)](#), [count_subareas\(\)](#), [count_subareas_random\(\)](#), [fit_gradients\(\)](#), [oruns_test\(\)](#), [oruns_test_boustrophedon\(\)](#), [oruns_test_byrowcol\(\)](#), [plot_AFSD\(\)](#)

oruns_test	<i>Runs Test</i>
------------	------------------

Description

Perform a runs test on the input data to test for clustering or randomness.

Usage

```
oruns_test(x)
```

Arguments

x A numeric vector representing the input data

Value

an `r4pde.oruns` object.

An `r4pde.oruns` object is a list containing:

- U, number of runs,
- EU, expected number of runs,
- sU, standard deviation of the expected number of runs
- Z, Z-score of the observed number of runs,
- pvalue, the p-value of the Z-score, and
- result, the test result of either "aggregation or clustering" or "randomness"

See Also

Other Spatial analysis: [AFSD\(\)](#), [BPL\(\)](#), [count_subareas\(\)](#), [count_subareas_random\(\)](#), [fit_gradients\(\)](#), [join_count\(\)](#), [oruns_test_boustrophedon\(\)](#), [oruns_test_byrowcol\(\)](#), [plot_AFSD\(\)](#)

Examples

```
oruns_test(c(1, 0, 1, 1, 0, 1, 0, 0, 1, 1))
```

oruns_test_boustrophedon

Boustrophedon Run Test for Binary Matrix

Description

Applies the ordinary runs test to a binary matrix using boustrophedon-style traversal. The function supports two modes: row-wise and column-wise boustrophedon. Each traversal flattens the matrix into a 1D sequence which is then tested using `oruns_test`.

Usage

```
oruns_test_boustrophedon(mat)
```

Arguments

`mat` A binary matrix (containing 0s and 1s, and possibly NAs).

Value

A list with two elements:

`rowwise_boustrophedon`

List containing the sequence and result of `oruns_test` for row-wise traversal.

`colwise_boustrophedon`

List containing the sequence and result of `oruns_test` for column-wise traversal.

See Also

[oruns_test](#)

Other Spatial analysis: [AFSD\(\)](#), [BPL\(\)](#), [count_subareas\(\)](#), [count_subareas_random\(\)](#), [fit_gradients\(\)](#), [join_count\(\)](#), [oruns_test\(\)](#), [oruns_test_byrowcol\(\)](#), [plot_AFSD\(\)](#)

oruns_test_byrowcol

Runs Test for Each Row and Column of a Binary Matrix

Description

Applies the ordinary runs test to each row and column of a binary matrix individually.

Usage

```
oruns_test_byrowcol(mat)
```

Arguments

mat A binary matrix (containing 0s and 1s, and possibly NAs).

Value

A list with four elements:

row_results Data frame with test results for each row.
col_results Data frame with test results for each column.
row_summary Percentage summary of interpretation for rows.
col_summary Percentage summary of interpretation for columns.

See Also

[oruns_test](#)

Other Spatial analysis: [AFSD\(\)](#), [BPL\(\)](#), [count_subareas\(\)](#), [count_subareas_random\(\)](#), [fit_gradients\(\)](#), [join_count\(\)](#), [oruns_test\(\)](#), [oruns_test_boustrophedon\(\)](#), [plot_AFSD\(\)](#)

plot.functional_curves

Plot functional_curves

Description

Plot functional_curves

Usage

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

Arguments

x An object of class "functional_curves".
... Additional arguments.

plot.functional_rate *Plot functional_rate curves and rates*

Description

Plot functional_rate curves and rates

Usage

```
## S3 method for class 'functional_rate'
plot(
  x,
  which = NULL,
  type = c("both", "curve", "rate"),
  show_ci = TRUE,
  show_threshold = TRUE,
  ncol = NULL,
  ...
)
```

Arguments

x	An object of class "functional_rate".
which	Optional character vector specifying which treatments/groups to display.
type	Character specifying what to plot: "both" (default; fitted curve in top panel and derivative in bottom panel), "curve" (only trajectory), or "rate" (only instantaneous rate).
show_ci	Logical; whether to show confidence ribbons. Default is TRUE.
show_threshold	Logical; whether to show threshold and zero horizontal lines. Default is TRUE.
ncol	Optional integer specifying number of facet columns.
...	Additional arguments.

Value

A ggplot or combined patchwork object.

`plot.functional_suppression_profiles`*Plot functional suppression profiles*

Description

Plot functional suppression profiles

Usage

```
## S3 method for class 'functional_suppression_profiles'
plot(
  x,
  type = c("dendrogram", "profiles", "heatmap", "rank", "all"),
  show_cut = TRUE,
  show_points = FALSE,
  ...
)
```

Arguments

<code>x</code>	An object of class "functional_suppression_profiles".
<code>type</code>	Character string specifying the plot type. One of "dendrogram", "profiles", "heatmap", "rank", or "all".
<code>show_cut</code>	Logical; whether to display the cluster cut height in dendrogram. Default is TRUE.
<code>show_points</code>	Logical; whether to overlay observed DSP values as points in the profiles plot. Default is FALSE.
<code>...</code>	Additional arguments passed to specific plot functions.

Details

The plot method visualizes the components of a Functional Suppression Profile. The dendrogram defines the Functional Suppression Profiles based on temporal similarity. The heatmap is ordered by the dendrogram to help interpret these functional groups using summary metrics.

Value

A ggplot object or a list of ggplot objects.

```
plot.functional_suppression_profiles_list
```

Plot a list of functional suppression profiles

Description

Plot a list of functional suppression profiles

Usage

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

Arguments

x An object of class "functional_suppression_profiles_list".

... Additional arguments passed to plot.functional_suppression_profiles.

Value

A list of ggplot objects, one for each environment.

```
plot.r4pde_functional_pca
```

Plot functional PCA results

Description

Plot functional PCA results

Usage

```
## S3 method for class 'r4pde_functional_pca'
plot(
  x,
  type = c("scree", "components", "scores", "reconstruction", "mean"),
  components = c(1, 2),
  curve_id = NULL,
  ...
)

autoplot.r4pde_functional_pca(object, ...)
```

Arguments

x	An object of class "r4pde_functional_pca".
type	Type of plot: "scree", "components", "scores", "reconstruction", or "mean".
components	Integer vector of length 2 indicating which components to plot for scores and mean perturbation.
curve_id	Optional vector of curve IDs to include in reconstruction plot.
...	Additional arguments.
object	An object of class "r4pde_functional_pca".

plot_AFSD

*Plot ASFD***Description**

This function creates a tile plot of the foci (cluster) identified by the AFSD function. It colors each cell in a foci and labels the centroid of each cluster with the foci ID. The 'ggplot2' package is used for the plot, and will be automatically installed if not already present.

Usage

```
plot_AFSD(df)
```

Arguments

df	A dataframe containing at least three columns: 'x', 'y', and 'cluster_id'. 'x' and 'y' are spatial coordinates and 'cluster_id' is the cluster identifier to which each cell belongs.
----	---

Value

A ggplot object with the scatter plot of foci (clusters).

See Also

Other Spatial analysis: [AFSD\(\)](#), [BPL\(\)](#), [count_subareas\(\)](#), [count_subareas_random\(\)](#), [fit_gradients\(\)](#), [join_count\(\)](#), [oruns_test\(\)](#), [oruns_test_boustrophedon\(\)](#), [oruns_test_byrowcol\(\)](#)

Examples

```
df <- data.frame(x = sample(1:100, 500, replace = TRUE),
                 y = sample(1:100, 500, replace = TRUE),
                 i = sample(0:1, 500, replace = TRUE, prob = c(0.7, 0.3)))

# Perform the AFSD
result <- AFSD(df)
# Plot the foci
plot_AFSD(result[[3]])
```

plot_curves

Plot environment-adjusted epidemic curves by cluster

Description

Plots environment-adjusted mean epidemic curves for each treatment, colored according to functional cluster membership.

The returned object is a ggplot and can be further modified using standard ggplot2 layers.

Usage

```
plot_curves(x, ...)

## S3 method for class 'functional_distances'
plot_curves(
  x,
  label_fun = NULL,
  palette = NULL,
  alpha = 0.9,
  linewidth = 1.1,
  ...
)

## S3 method for class 'functional_dsp'
plot_curves(x, ...)
```

Arguments

x	An object of class "functional_distances".
...	Additional arguments.
label_fun	Optional function to modify treatment labels.
palette	Optional named vector of colors for clusters.
alpha	Line transparency.
linewidth	Line width.

Value

A ggplot object.

plot_dendrogram	<i>Plot functional dendrogram of epidemic curves</i>
-----------------	--

Description

Visualizes the hierarchical clustering of treatments based on functional distances among epidemic curves.

Usage

```
plot_dendrogram(
  x,
  label_fun = NULL,
  palette = NULL,
  show_cut = TRUE,
  label_size = 2.8
)
```

Arguments

x	An object of class "functional_distances".
label_fun	Optional function to modify treatment labels.
palette	Optional named vector of colors for clusters.
show_cut	Logical; whether to display the cluster cut height.
label_size	Numeric specifying the font size of the labels. Default is 2.8.

Value

A ggplot object.

plot_diagnostics	<i>Plot diagnostics for functional epidemic curve models</i>
------------------	--

Description

Combines multiple diagnostic plots from [diagnose_curves](#) into a multi-panel layout.

Usage

```
plot_diagnostics(
  diag,
  layout = c("2x2", "vertical", "horizontal"),
  show_curve = FALSE,
  rel_heights = c(1, 0.7)
)
```

Arguments

diag	An object of class "r4pde_curve_diagnostics".
layout	Layout of diagnostic panels: "2x2", "vertical", or "horizontal".
show_curve	Logical; whether to include the example adjusted curve.
rel_heights	Relative heights for panels when stacking.

Value

A composite plot object generated using **cowplot**.

plot_dsp	<i>Plot Disease Suppression Profiles</i>
----------	--

Description

Plots functional contrast curves (DSP) over time.

Usage

```
plot_dsp(
  object,
  time = NULL,
  response = NULL,
  treatment = NULL,
  group = NULL,
  facet = NULL,
  show_zero = TRUE,
  linewidth = 1,
  alpha = 1,
  ylab = "Disease suppression profile",
  xlab = "Time"
)
```

Arguments

object	An object of class "functional_dsp".
time	Character string for the time column (auto-detected if object is "functional_dsp").
response	Character string for the DSP column.
treatment	Character string for the treatment column.
group	Optional character vector of grouping variables.
facet	Optional character string specifying a variable for faceting.
show_zero	Logical; whether to draw a horizontal reference line at y = 0. Default is TRUE.
linewidth	Numeric; width of the lines.
alpha	Numeric; transparency of the lines.
ylab	Character string; label for the y-axis.
xlab	Character string; label for the x-axis.

Value

A ggplot object.

plot_dsp_rank_heatmap *Plot DSP Rankings Heatmap*

Description

Creates a heatmap visualization of treatment rankings based on DSP descriptors.

Usage

```
plot_dsp_rank_heatmap(ranks_data, treatment = "treatment", group = NULL)
```

Arguments

ranks_data	Tibble output from rank_dsp .
treatment	Character string naming the treatment column (for the y-axis).
group	Optional character string for faceting by group.

Value

A ggplot object.

plot_functional_instability
Plot method for functional instability results

Description

Plot method for functional instability results

Usage

```
plot_functional_instability(object, type = c("overall", "space", "time"), ...)
```

Arguments

object	Output from functional_instability() .
type	One of "overall", "space", or "time".
...	Further arguments passed to methods.

Value

A ggplot object.

```
print.functional_curves
```

Print functional_curves

Description

Print functional_curves

Usage

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

Arguments

x	An object of class "functional_curves".
...	Additional arguments.

```
print.functional_distances
```

Print functional_distances

Description

Print functional_distances

Usage

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

Arguments

x	An object of class "functional_distances".
...	Additional arguments.

```
print.functional_rate
```

Print a functional_rate object

Description

Print a functional_rate object

Usage

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

Arguments

x	An object of class "functional_rate".
...	Additional arguments.

Value

The object x, invisibly.

```
print.functional_resistance
```

Print functional_resistance

Description

Print functional_resistance

Usage

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

Arguments

x	An object of class "functional_resistance".
...	Additional arguments.

```
print.functional_suppression_profiles
```

Print functional suppression profiles

Description

Print functional suppression profiles

Usage

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

Arguments

x	An object of class "functional_suppression_profiles".
...	Additional arguments.

```
print.functional_suppression_profiles_list
```

Print a list of functional suppression profiles

Description

Print a list of functional suppression profiles

Usage

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

Arguments

x	An object of class "functional_suppression_profiles_list".
...	Additional arguments.

```
print.r4pde_curve_diagnostics
```

Print method for curve diagnostics

Description

Print method for curve diagnostics

Usage

```
## S3 method for class 'r4pde_curve_diagnostics'  
print(x, ...)
```

Arguments

x An object of class "r4pde_curve_diagnostics".
... Additional arguments (ignored).

Value

Invisibly returns x.

```
print.suggest_k
```

Print method for suggest_k

Description

Print method for suggest_k

Usage

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

Arguments

x An object of class "suggest_k".
... Additional arguments (ignored).

Value

Invisibly returns x.

```
print.summary.functional_suppression_profiles
```

Print summary of functional suppression profiles

Description

Print summary of functional suppression profiles

Usage

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

Arguments

x	An object of class "summary.functional_suppression_profiles".
...	Additional arguments.

```
print.summary.functional_suppression_profiles_list
```

Print summary of a list of functional suppression profiles

Description

Print summary of a list of functional suppression profiles

Usage

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

Arguments

x	An object of class "summary.functional_suppression_profiles_list".
...	Additional arguments.

rank_dsp

*Rank Treatments based on Functional DSP Descriptors***Description**

Generates rankings of treatments based on extracted DSP functional descriptors.

Usage

```
rank_dsp(
  summary_data,
  metrics = c("protected_area", "max_suppression", "persistence", "centroid"),
  higher_is_better = TRUE,
  group = NULL,
  ties.method = "average"
)
```

Arguments

summary_data	The tibble output from functional_summary .
metrics	Character vector of metrics to rank.
higher_is_better	Logical vector or named logical vector indicating if higher values are better for each metric. If a single logical, applies to all.
group	Optional character vector of grouping variables.
ties.method	Character string specifying how to handle ties. Default is "average".

Value

A tibble with ranks for each specified metric, and an average_rank.

reconstruct_curves

*Reconstruct curves using specified FPCA components***Description**

Reconstruct curves using specified FPCA components

Usage

```
reconstruct_curves(x, components = NULL)
```

Arguments

x	An object of class "r4pde_functional_pca".
components	Integer vector of components to use for reconstruction. If NULL, uses all retained components.

RustSoybean	<i>RustSoybean dataset</i>
-------------	----------------------------

Description

Soybean rust severity and field metadata.

Usage

RustSoybean

Format

A data frame with:

detection Detection score or date

epidemia Epidemic phase

latitude Latitude

local Location name

longitude Longitude

planting Planting date or stage

severity Disease severity

Source

Del Ponte Lab internal data

simulate_dsp_data	<i>#' Simulate Disease Suppression Profile Data</i>
-------------------	---

Description

Simulates a dataset for demonstrating functional analysis with Disease Suppression Profiles. Returns a data frame with varying treatments and replicates over multiple time points.

Usage

simulate_dsp_data()

Value

A tibble with columns treatment, rep, time, y_control, dsp_true, severity_true, and severity.

SpatialAggregated	<i>SpatialAggregated dataset</i>
-------------------	----------------------------------

Description

Simulated aggregated spatial binary disease pattern.

Usage

SpatialAggregated

Format

A data frame with:

x x-coordinate

y y-coordinate

Source

Simulated example

SpatialRandom	<i>SpatialRandom dataset</i>
---------------	------------------------------

Description

Simulated random spatial binary disease pattern.

Usage

SpatialRandom

Format

A data frame with:

x x-coordinate

y y-coordinate

Source

Simulated example

suggest_k

*Suggest GAM smoothing parameters for epidemic curve models***Description**

A helper function to guide selection of the `k_smooth`, `k_trt`, `k_env`, and `gamma` parameters used by [functional_curves](#). Can infer the effective replication from a data frame or accept scalar values directly when the data are not yet available.

For sparse epidemic data it is strongly recommended to use `rule = "minimum"` so that `k` values are based on the least-replicated treatment-by-environment combination, guarding against over-fitting.

Usage

```
suggest_k(
  data = NULL,
  time = NULL,
  treatment = NULL,
  environment = NULL,
  n_time = NULL,
  n_env = NULL,
  rule = c("minimum", "median"),
  smoothness = c("conservative", "moderate", "flexible")
)
```

Arguments

<code>data</code>	Optional data frame. If supplied, <code>time</code> , <code>treatment</code> , and optionally <code>environment</code> are used to compute summaries of the number of unique time points and environments per treatment.
<code>time</code>	Unquoted column name for the time variable, or a character string naming the column.
<code>treatment</code>	Unquoted column name for the treatment / cultivar variable, or a character string naming the column.
<code>environment</code>	Optional unquoted column name for the environment variable, or a character string naming the column.
<code>n_time</code>	Integer. Number of unique time points to use directly (ignored when data is supplied).
<code>n_env</code>	Integer. Number of unique environments to use directly (ignored when data is supplied, or when there is no environment variable).
<code>rule</code>	Character string. How to summarise the distribution of replication counts across treatment-by-environment combinations: "minimum" (default, conservative) or "median".
<code>smoothness</code>	Character string controlling how liberal the recommendations are: "conservative" (default), "moderate", or "flexible".

Details

The function computes, for every treatment-by-environment combination, the number of unique time points at which observations are available. It then summarises these counts using the chosen rule to obtain `effective_n_time`. Similarly it computes, per treatment, the number of unique environments, and summarises using the same rule to obtain `effective_n_env`.

Recommended `k` values follow these heuristics:

- `k_smooth`: global smooth over time; capped below `effective_n_time - 1`.
- `k_trt`: treatment-specific smooth; capped below `k_smooth`.
- `k_env`: environment random effect; capped below `effective_n_env`.
- `gamma`: penalty multiplier; higher values encourage smoother fits.

Value

A named list with the following elements:

`time_summary` Named numeric vector with minimum, median, and maximum unique time points per treatment-by-environment combination.

`environment_summary` Named numeric vector with minimum, median, and maximum unique environments per treatment (or `NULL` when no environment variable is given).

`effective_n_time` The effective number of time points chosen by rule.

`effective_n_env` The effective number of environments chosen by rule, or `NULL`.

`k_smooth` Recommended basis dimension for the global time smooth.

`k_trt` Recommended basis dimension for the treatment-specific smooth.

`k_env` Recommended basis dimension for the environment random effect.

`gamma` Recommended penalisation multiplier.

`message` A short interpretation message.

Examples

```
# Using explicit values
suggest_k(n_time = 5, n_env = 8)

# Inferring from a data frame (unquoted column names)
df <- data.frame(
  time      = rep(1:6, times = 6),
  cultivar  = rep(c("A", "B", "C"), each = 12),
  env       = rep(rep(c("E1", "E2"), each = 6), times = 3),
  severity  = runif(36, 0, 0.5)
)
suggest_k(
  data      = df,
  time      = time,
  treatment = cultivar,
  environment = env,
  rule      = "minimum",
  smoothness = "conservative"
```

)

summary.functional_rate
<i>Summarize a functional_rate object</i>

Description

Summarize a functional_rate object

Usage

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

Arguments

object An object of class "functional_rate".
... Additional arguments.

Value

A tibble of curve-level rate phenotypes.

summary.functional_suppression_profiles
<i>Summarize functional suppression profiles</i>

Description

Summarize functional suppression profiles

Usage

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

Arguments

object An object of class "functional_suppression_profiles".
... Additional arguments.

Value

A list of class "summary.functional_suppression_profiles".

```
summary.functional_suppression_profiles_list
```

Summarize a list of functional suppression profiles

Description

Summarize a list of functional suppression profiles

Usage

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

Arguments

`object` An object of class "functional_suppression_profiles_list".
`...` Additional arguments.

Value

A list of summaries, one for each environment.

```
theme_r4pde
```

Custom ggplot2 theme based on cowplot::theme_half_open

Description

This function creates a new ggplot2 theme by modifying the cowplot::theme_half_open theme. It sets a custom font size and changes the panel background color to gray96.

Usage

```
theme_r4pde(font_size = 16)
```

Arguments

`font_size` The base font size. Default is 16.

Value

A ggplot2 theme object.

WhiteMoldSoybean	<i>WhiteMoldSoybean dataset</i>
------------------	---------------------------------

Description

National dataset of white mold severity and yield.

Usage

WhiteMoldSoybean

Format

A data frame with:

- country** Country name
- elevation** Field elevation
- elevation_class** Elevation class
- harvest_year** Year of harvest
- inc** Incidence
- inc_check** Check plot incidence
- inc_class** Incidence class
- location** Location name
- region** Geographical region
- scl** Soybean canopy layer
- season** Crop season
- state** State name
- study** Study identifier
- treat** Treatment applied
- yld** Yield
- yld_check** Yield of untreated check
- yld_class** Yield class

Source

Del Ponte Lab internal data

Description

This function calculates summary statistics within specified windows around a given end date in a dataset, facilitating epidemiological analysis. It allows backward, forward, or both directions of window calculations based on a user-defined variable and window lengths.

Usage

```
windowpane(
  data,
  end_date_col,
  date_col,
  variable,
  summary_type,
  threshold = NULL,
  window_lengths,
  direction = "backward",
  group_by_cols = NULL,
  date_format = "%Y-%m-%d"
)
```

Arguments

<code>data</code>	A data frame containing the input data.
<code>end_date_col</code>	A string specifying the name of the column representing the end date.
<code>date_col</code>	A string specifying the name of the column representing the date variable.
<code>variable</code>	A string specifying the name of the column for which summary statistics are calculated.
<code>summary_type</code>	A string specifying the type of summary to calculate. Options are "mean", "sum", "above_threshold", or "below_threshold".
<code>threshold</code>	Optional numeric value used when <code>summary_type</code> is "above_threshold" or "below_threshold".
<code>window_lengths</code>	A numeric vector specifying the window lengths (in days) for the calculations.
<code>direction</code>	A string specifying the direction of the window. Options are "backward" (default), "forward", or "both".
<code>group_by_cols</code>	Optional vector of strings specifying column names for grouping the data.
<code>date_format</code>	A string specifying the format of the date columns. Default is "%Y-%m-%d".

Value

A data frame with the calculated summary values for each window.

See Also

Other Disease modeling: [get_brdwgd\(\)](#), [get_era5\(\)](#), [get_nasapower\(\)](#)

windowpane_tests

Windowpane Tests for Correlation Analysis

Description

This function performs bootstrapped correlation analysis for multiple predictors against a response variable. It applies the Simes method for global significance testing and calculates individual correlations, p-values, and bootstrap statistics.

Usage

```
windowpane_tests(
  data,
  response_var,
  corr_type = "spearman",
  R = 1000,
  global_alpha = 0.05,
  individual_alpha = 0.005
)
```

Arguments

<code>data</code>	A data frame containing the predictors and the response variable.
<code>response_var</code>	A string representing the name of the response variable in the data frame.
<code>corr_type</code>	A string specifying the correlation method to use; options are "spearman" (default), "pearson", or "kendall".
<code>R</code>	An integer indicating the number of bootstrap replications. Default is 1000.
<code>global_alpha</code>	A numeric value representing the global alpha level for the Simes correction. Default is 0.05.
<code>individual_alpha</code>	A numeric value for the individual alpha threshold for testing individual predictors. Default is 0.005.

Details

The function calculates correlations between the response variable and each predictor in the data frame, using bootstrapping to generate mean, standard deviation, and median estimates of the correlation. The Simes correction is applied to control for multiple testing, providing a global p-value (P_g). The function also returns the maximum observed correlation.

Value

A list containing the following elements:

- | | |
|--------------------|---|
| results | A data frame with columns: variable, correlation, p_value, mean_corr, sd_corr, median_corr, rank, simes_threshold, significant_simes, and individual_significant. |
| summary_table | A data frame summarizing the global p-value (P_g) and maximum correlation. |
| global_significant | A logical value indicating whether the global test is significant. |

Index

- * **Disease modeling**
 - get_brdwgd, 37
 - get_era5, 38
 - get_nasapower, 41
 - windowpane, 67
 - * **Disease quantification**
 - CompMuCens, 9
 - DSI, 13
 - DSI2, 14
 - * **Miscellaneous**
 - theme_r4pde, 65
 - * **Spatial analysis**
 - AFSD, 3
 - BPL, 7
 - count_subareas, 10
 - count_subareas_random, 11
 - fit_gradients, 16
 - join_count, 42
 - oruns_test, 43
 - oruns_test_boustrophedon, 44
 - oruns_test_byrowcol, 44
 - plot_AFSD, 49
 - * **datasets**
 - BlastWheat, 6
 - BudBlightSoybean, 8
 - DidymellaWatermelon, 12
 - FHBWheat, 14
 - FHBWheatBrazil, 15
 - FusariumBanana, 36
 - RustSoybean, 60
 - SpatialAggregated, 61
 - SpatialRandom, 61
 - WhiteMoldSoybean, 66
- AFSD, 3, 8, 10, 11, 16, 43–45, 49
- augment, 4
- augment.functional_rate, 5
- augment_functional_pca, 5
- augment_functional_rate, 6
- autoplot.r4pde_functional_pca
(plot.r4pde_functional_pca), 48
- BlastWheat, 6
- BPL, 4, 7, 10, 11, 16, 43–45, 49
- BudBlightSoybean, 8
- CompMuCens, 9, 13, 14
- count_subareas, 4, 8, 10, 11, 16, 43–45, 49
- count_subareas_random, 4, 8, 10, 11, 16, 43–45, 49
- diagnose_curves, 12, 51
- DidymellaWatermelon, 12
- DSI, 9, 13, 14
- DSI2, 9, 13, 14
- FHBWheat, 14
- FHBWheatBrazil, 15
- fit_gradients, 4, 8, 10, 11, 16, 43–45, 49
- functional_contrast, 17, 34, 35
- functional_curves, 12, 18, 22–27, 29, 35, 62
- functional_distances, 21, 35
- functional_instability, 22
- functional_instability(), 53
- functional_pca, 25
- functional_pca_clusters, 27
- functional_rate, 27
- functional_resistance, 30
- functional_summary, 18, 32, 35, 59
- functional_suppression_profiles, 33
- FusariumBanana, 36
- get_brdwgd, 37, 39, 42, 68
- get_era5, 38, 38, 42, 68
- get_fpca_eigenfunctions, 39
- get_fpca_scores, 40
- get_fpca_variance, 40
- get_nasapower, 38, 39, 41, 68
- join_count, 4, 8, 10, 11, 16, 42, 43–45, 49

oruns_test, [4](#), [8](#), [10](#), [11](#), [16](#), [43](#), [43](#), [44](#), [45](#), [49](#)
oruns_test_boustrophedon, [4](#), [8](#), [10](#), [11](#), [16](#),
[43](#), [44](#), [45](#), [49](#)
oruns_test_byrowcol, [4](#), [8](#), [10](#), [11](#), [16](#), [43](#),
[44](#), [44](#), [49](#)

plot.functional_curves, [45](#)
plot.functional_rate, [46](#)
plot.functional_suppression_profiles,
[47](#)
plot.functional_suppression_profiles_list,
[48](#)
plot.r4pde_functional_pca, [48](#)
plot_AFSD, [4](#), [8](#), [10](#), [11](#), [16](#), [43–45](#), [49](#)
plot_curves, [50](#)
plot_dendrogram, [35](#), [51](#)
plot_diagnostics, [51](#)
plot_dsp, [18](#), [52](#)
plot_dsp_rank_heatmap, [53](#)
plot_functional_instability, [53](#)
print.functional_curves, [54](#)
print.functional_distances, [54](#)
print.functional_rate, [55](#)
print.functional_resistance, [55](#)
print.functional_suppression_profiles,
[56](#)
print.functional_suppression_profiles_list,
[56](#)
print.r4pde_curve_diagnostics, [57](#)
print.suggest_k, [57](#)
print.summary.functional_suppression_profiles,
[58](#)
print.summary.functional_suppression_profiles_list,
[58](#)

rank_dsp, [35](#), [53](#), [59](#)
reconstruct_curves, [59](#)
RustSoybean, [60](#)

simulate_dsp_data, [60](#)
SpatialAggregated, [61](#)
SpatialRandom, [61](#)
suggest_k, [62](#)
summary.functional_rate, [64](#)
summary.functional_suppression_profiles,
[64](#)
summary.functional_suppression_profiles_list,
[65](#)

theme_r4pde, [65](#)

WhiteMoldSoybean, [66](#)
windowpane, [38](#), [39](#), [42](#), [67](#)
windowpane_tests, [68](#)