Package 'mcgf'

June 29, 2024

Title Markov Chain Gaussian Fields Simulation and Parameter Estimation

Version 1.1.1

Description Simulating and estimating (regime-switching) Markov chain Gaussian fields with covariance functions of the Gneiting class (Gneiting 2002) <doi:10.1198/016214502760047113>. It supports parameter estimation by weighted least squares and maximum likelihood methods, and produces Kriging forecasts and intervals for existing and new locations.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0), doParallel (>= 1.0.17), foreach (>= 1.5.2), parallel (>= 4.3.1), knitr, rmarkdown, lubridate, dplyr, Rsolnp

Config/testthat/edition 3

Imports MASS, sp

Depends R (>= 4.0.0)

LazyData true

URL https://github.com/tianxia-jia/mcgf, https://tianxia-jia.github.io/mcgf/

BugReports https://github.com/tianxia-jia/mcgf/issues

VignetteBuilder knitr

NeedsCompilation no

Author Tianxia Jia [aut, cre, cph] (<https://orcid.org/0000-0001-5430-5019>)

Maintainer Tianxia Jia <tianxia.jia@ucalgary.ca>

Repository CRAN

Date/Publication 2024-06-29 06:40:17 UTC

Contents

`acfs` *Generic function for calculating autocorrelation*

Description

Generic function for calculating autocorrelation

Usage

```
acfs(x, ...)
```

Arguments

`x`	An **R** object.
`...`	Additional parameters or attributes.

Details

Refer to `acfs.mcgf()` and `acfs.mcgf_rs()` for more details.

Value

A vector of mean auto-correlations for `mcgf` objects, or that plus a list of regime-switching mean auto-correlations for `mcgf_rs` objects.

acfs.mcgf *Extract, calculate, or assign mean auto-correlations for an* mcgf *or* mcgf_rs *object*

Description

Extract, calculate, or assign mean auto-correlations for an mcgf or mcgf_rs object

Usage

```
## S3 method for class 'mcgf'
acfs(x, lag_max, replace = FALSE, ...)

## S3 method for class 'mcgf_rs'
acfs(x, lag_max, replace = FALSE, ...)

acfs(x) <- value

add_acfs(x, lag_max, ...)
```

Arguments

x	An mcgf or mcgf_rs object.
lag_max	Maximum lag at which to calculate the acf.
replace	Logical; if TRUE, acfs are recalculated.
...	Additional parameters or attributes.
value	A Vector of mean of auto-correlations for time lags starting from 0.

Details

For mcgf objects, acfs() computes mean auto-correlations for each time lag across locations. The output is a vector of acfs.

For mcgf_rs objects, acfs() computes regime-switching mean auto-correlations for each time lag across locations. The output is a list of vectors of acfs, where each vector in the list corresponds to the acfs for a regime.

acfs<- assigns acfs to x.

add_acfs() adds acfs to x.

Value

acfs() returns (regime-switching) mean auto-correlations. add_acfs() returns the same object with additional attributes of (regime-switching) mean auto-correlations.

Examples

```
# Calculate acfs for 'sim1'
data(sim1)
sim1_mcgf <- mcgf(sim1$data, dists = sim1$dists)
acfs(sim1_mcgf, lag_max = 5)

# Add acfs to 'sim1_mcgf'
sim1_mcgf <- add_acfs(sim1_mcgf, lag_max = 5)
print(sim1_mcgf, "acfs")

# Calculate acfs for 'sim2'
data(sim2)
sim2_mcgf <- mcgf_rs(sim2$data, dists = sim2$dists, label = sim2$label)
acfs(sim2_mcgf, lag_max = 5)

# Add acfs to 'sim2_mcgf'
sim2_mcgf <- add_acfs(sim2_mcgf, lag_max = 5)
print(sim2_mcgf, "acfs")
```

---

acf_rs   *Calculate regime-switching auto-correlation*

Description

Calculate regime-switching auto-correlation

Usage

```
acf_rs(x, label, lag_max, demean = TRUE)
```

Arguments

`x`	A univariate numeric time series.
`label`	A factor of regime labels.
`lag_max`	Maximum lag at which to calculate the acf.
`demean`	Logical. Should the covariances be about the sample means?

Value

Mean auto-correlations for each group in `label`.

Examples

```
set.seed(123)
x <- rnorm(100)
label <- sample(1:2, 100, replace = TRUE)
acf_rs(x, label = factor(label), lag_max = 3)
```

add_base — *Generic function for adding a base model*

Description

Generic function for adding a base model

Usage

```
add_base(x, ...)
```

Arguments

x	An **R** object.
...	Additional parameters or attributes.

Details

Refer to `add_base.mcgf()` and `add_base.mcgf_rs()` for more details.

Value

`x` with the newly added base model.

add_base.mcgf — *Add base model outputted from `fit_base()` to an `mcgf` object.*

Description

Add base model outputted from `fit_base()` to an `mcgf` object.

Usage

```
## S3 method for class 'mcgf'
add_base(x, fit_base, fit_s, fit_t, sep = FALSE, old = FALSE, ...)

base(x) <- value
```

Arguments

`x`	An `mcgf` object.
`fit_base`	Output from the `fit_base()` function.
`fit_s`	Pure spatial model outputted from the `fit_base()` function. Used only when `sep = TRUE`.
`fit_t`	Pure temporal model outputted from the `fit_base()` function. Used only when `sep = TRUE`.
`sep`	Logical; TRUE if spatial and temporal models are fitted separately.
`old`	Logical; TRUE if the old base model needs to be kept.
`...`	Additional arguments. Not in use.
`value`	A list containing the base model as well as its parameters. It must contains the same output as `add_base.mcgf()` or `add_base.mcgf_rs()`.

Details

After fitting the base model by `fit_base()`, the results can be added to `x` by `add_base()`. To supply the base model directly, use `base<-` to add the base model; the value must contain `model`, `par_base`, `cor_base`, `lag`, and `horizon`.

Value

`x` with newly added attributes of the base model.

See Also

Other functions on fitting an mcgf: `add_lagr.mcgf()`, `fit_base.mcgf()`, `fit_lagr.mcgf()`, `krige.mcgf()`, `krige_new.mcgf()`

Examples

```
data(sim1)
sim1_mcgf <- mcgf(sim1$data, dists = sim1$dists)
sim1_mcgf <- add_acfs(sim1_mcgf, lag_max = 5)
sim1_mcgf <- add_ccfs(sim1_mcgf, lag_max = 5)

# Fit a pure spatial model
fit_spatial <- fit_base(
    sim1_mcgf,
    model = "spatial",
    lag = 5,
    par_init = c(c = 0.001, gamma = 0.5),
    par_fixed = c(nugget = 0)
)
# Fit a pure temporal model
fit_temporal <- fit_base(
    sim1_mcgf,
    model = "temporal",
    lag = 5,
```

```
    par_init = c(a = 0.3, alpha = 0.5)
)

# Store the fitted models to 'sim1_mcgf'
sim1_mcgf <-
    add_base(sim1_mcgf,
        fit_s = fit_spatial,
        fit_t = fit_temporal,
        sep = TRUE
    )

# Fit a separable model
fit_sep <- fit_base(
    sim1_mcgf,
    model = "sep",
    lag = 5,
    par_init = c(
        c = 0.001,
        gamma = 0.5,
        a = 0.3,
        alpha = 0.5
    ),
    par_fixed = c(nugget = 0)
)
# Store the newly fitted model, and keep the old fit
sim1_mcgf <- add_base(sim1_mcgf, fit_base = fit_sep, old = TRUE)
model(sim1_mcgf, model = "base", old = TRUE)
```

add_base.mcgf_rs — *Add base model outputted from* `fit_base()` *to an* `mcgf_rs` *object.*

Description

Add base model outputted from `fit_base()` to an `mcgf_rs` object.

Usage

```
## S3 method for class 'mcgf_rs'
add_base(x, fit_base_ls, fit_s_ls, fit_t_ls, sep = FALSE, old = FALSE, ...)
```

Arguments

`x`	An mcgf_rs‘ object.
`fit_base_ls`	Output from the `fit_base()` function.
`fit_s_ls`	Pure spatial model outputted from the `fit_base()` function. Used only when `sep = TRUE`.
`fit_t_ls`	Pure temporal model outputted from the `fit_base()` function. Used only when `sep = TRUE`.

`sep`	Logical; TRUE if spatial and temporal models are fitted separately.
`old`	Logical; TRUE if the old base model needs to be kept. The lag and horizon of the new model are assumed to be the same as that of the old model.
`...`	Additional arguments. Not in use.

Details

After fitting the base model by `fit_base()`, the results can be added to `x` by `add_base()`. To supply the base model directly, use `base<-` to add the base model; the value must contain the same output as `add_base.mcgf()` or `add_base.mcgf_rs()`.

Value

`x` with newly added attributes of the base model.

See Also

Other functions on fitting an mcgf_rs: `add_lagr.mcgf_rs()`, `fit_base.mcgf_rs()`, `fit_lagr.mcgf_rs()`, `krige.mcgf_rs()`, `krige_new.mcgf_rs()`

Examples

```
data(sim2)
sim2_mcgf <- mcgf_rs(sim2$data, dists = sim2$dists, label = sim2$label)
sim2_mcgf <- add_acfs(sim2_mcgf, lag_max = 5)
sim2_mcgf <- add_ccfs(sim2_mcgf, lag_max = 5)

# Fit a regime-switching pure spatial model
fit_spatial <-
    fit_base(
        sim2_mcgf,
        lag_ls = 5,
        model_ls = "spatial",
        par_init_ls = list(c(c = 0.00005, gamma = 0.5)),
        par_fixed_ls = list(c(nugget = 0))
    )

# Fit a regime-switching pure temporal model
fit_temporal <-
    fit_base(
        sim2_mcgf,
        lag_ls = 5,
        model_ls = "temporal",
        par_init_ls = list(
            list(a = 0.8, alpha = 0.8),
            list(a = 0.1, alpha = 0.1)
        )
    )

# Store the fitted models to 'sim2_mcgf'
sim2_mcgf <- add_base(sim2_mcgf,
```

```
    fit_s_ls = fit_spatial,
    fit_t_ls = fit_temporal,
    sep = TRUE
)

# Fit a regime-switching separable model
fit_sep <- fit_base(
    sim2_mcgf,
    lag_ls = 5,
    model_ls = "sep",
    par_init_ls = list(list(
        c = 0.00005,
        gamma = 0.5,
        a = 0.5,
        alpha = 0.5
    )),
    par_fixed_ls = list(c(nugget = 0))
)

# Store the newly fitted model, and keep the old fit
sim2_mcgf <- add_base(sim2_mcgf, fit_base_ls = fit_sep, old = TRUE)
model(sim2_mcgf, model = "base", old = TRUE)
```

`add_lagr` *Generic function for adding a Lagrangian model*

Description

Generic function for adding a Lagrangian model

Usage

```
add_lagr(x, ...)
```

Arguments

x	An **R** object.
...	Additional parameters or attributes.

Details

Refer to `add_lagr.mcgf()` and `add_lagr.mcgf_rs()` for more details.

Value

x with the newly added Lagrangian model.

add_lagr.mcgf *Add lagr model outputted from `fit_lagr()` to a `mcgf` object.*

Description

Add lagr model outputted from `fit_lagr()` to a `mcgf` object.

Usage

```
## S3 method for class 'mcgf'
add_lagr(x, fit_lagr, ...)

lagr(x) <- value
```

Arguments

`x`	An `mcgf` object.
`fit_lagr`	Output from the `fit_lagr()` function.
`...`	Additional arguments. Not in use.
`value`	A list containing the lagr model as well as its parameters. It must contains `model`, `par_lagr`, and `cor_lagr`.

Value

`x` with newly added attributes of the Lagrangian model.

See Also

Other functions on fitting an mcgf: `add_base.mcgf()`, `fit_base.mcgf()`, `fit_lagr.mcgf()`, `krige.mcgf()`, `krige_new.mcgf()`

Examples

```
data(sim1)
sim1_mcgf <- mcgf(sim1$data, dists = sim1$dists)
sim1_mcgf <- add_acfs(sim1_mcgf, lag_max = 5)
sim1_mcgf <- add_ccfs(sim1_mcgf, lag_max = 5)

# Fit a separable model and store it to 'sim1_mcgf'
fit_sep <- fit_base(
    sim1_mcgf,
    model = "sep",
    lag = 5,
    par_init = c(
        c = 0.001,
        gamma = 0.5,
        a = 0.3,
        alpha = 0.5
```

```
    ),
    par_fixed = c(nugget = 0)
)
sim1_mcgf <- add_base(sim1_mcgf, fit_base = fit_sep)

# Fit a Lagrangian model
fit_lagr <- fit_lagr(
    sim1_mcgf,
    model = "lagr_tri",
    par_init = c(v1 = 300, v2 = 300, lambda = 0.15),
    par_fixed = c(k = 2)
)

# Store the fitted Lagrangian model to 'sim1_mcgf'
sim1_mcgf <- add_lagr(sim1_mcgf, fit_lagr = fit_lagr)
model(sim1_mcgf, old = TRUE)
```

---

add_lagr.mcgf_rs   *Add lagr model outputted from* fit_lagr() *to a* mcgf_rs *object.*

Description

Add lagr model outputted from fit_lagr() to a mcgf_rs object.

Usage

```
## S3 method for class 'mcgf_rs'
add_lagr(x, fit_lagr_ls, ...)
```

Arguments

x	An mcgf_rs object.
fit_lagr_ls	Output from the fit_lagr() function.
...	Additional arguments. Not in use.

Details

After fitting the Lagrangian model by fit_lagr(), the results can be added to x by add_base(). To supply the Lagrangian model directly, use lagr<- to add the Lagrangian model; the value must contain the same output as add_lagr.mcgf() or add_lagr.mcgf_rs().

Value

x with newly added attributes of the Lagrangian model.

See Also

Other functions on fitting an mcgf_rs: add_base.mcgf_rs(), fit_base.mcgf_rs(), fit_lagr.mcgf_rs(), krige.mcgf_rs(), krige_new.mcgf_rs()

Examples

```
data(sim3)
sim3_mcgf <- mcgf_rs(sim3$data, dists = sim3$dists, label = sim3$label)
sim3_mcgf <- add_acfs(sim3_mcgf, lag_max = 5)
sim3_mcgf <- add_ccfs(sim3_mcgf, lag_max = 5)

# Fit a fully symmetric model with known variables
fit_fs <- fit_base(
    sim3_mcgf,
    lag_ls = 5,
    model_ls = "fs",
    rs = FALSE,
    par_init_ls = list(list(beta = 0)),
    par_fixed_ls = list(list(
        nugget = 0,
        c = 0.05,
        gamma = 0.5,
        a = 0.5,
        alpha = 0.2
    ))
)

# Set beta to 0 to fit a separable model with known variables
fit_fs[[1]]$fit$par <- 0

# Store the fitted separable model to 'sim3_mcgf'
sim3_mcgf <- add_base(sim3_mcgf, fit_base_ls = fit_fs)

# Fit a regime-switching Lagrangian model.
fit_lagr_rs <- fit_lagr(
    sim3_mcgf,
    model_ls = list("lagr_tri"),
    par_init_ls = list(
        list(v1 = -50, v2 = 50),
        list(v1 = 100, v2 = 100)
    ),
    par_fixed_ls = list(list(lambda = 0.2, k = 2))
)

# Store the fitted Lagrangian model to 'sim3_mcgf'
sim3_mcgf <- add_lagr(sim3_mcgf, fit_lagr_ls = fit_lagr_rs)
model(sim3_mcgf)
```

`ccfs` *Generic function for calculating cross-correlation*

Description

Generic function for calculating cross-correlation

Usage

```
ccfs(x, ...)
```

Arguments

x	An **R** object.
...	Additional parameters or attributes.

Details

Refer to `ccfs.mcgf()` and `ccfs.mcgf_rs()` for more details.

Value

An array of cross-correlations for `mcgf` objects, or that plus a list of regime-switching cross-correlations for `mcgf_rs` objects.

---

ccfs.mcgf — *Extract, calculate, or assign cross-correlations for an* `mcgf` *or* `mcgf_rs` *object*

Description

Extract, calculate, or assign cross-correlations for an `mcgf` or `mcgf_rs` object

Usage

```
## S3 method for class 'mcgf'
ccfs(x, lag_max, ncores = 1, replace = FALSE, ...)

## S3 method for class 'mcgf_rs'
ccfs(x, lag_max, ncores = 1, replace = FALSE, ...)

ccfs(x) <- value

add_ccfs(x, lag_max, ncores = 1, ...)
```

Arguments

x	An `mcgf` or `mcgf_rs` object.
lag_max	Maximum lag at which to calculate the ccfs.
ncores	Number of cpu cores used for computing. The `doParallel` package is required when ncores > 1.
replace	Logical; if TRUE, `acfs` are recalculated.
...	Additional parameters or attributes. Not in use.
value	Cross-correlations.

Details

For mcgf objects, ccfs() computes cross-correlations for each time lag. The output is an array of matrices where each matrix corresponds to the cross-correlation for a time lag.

For mcgf_rs objects, ccfs() computes regime-switching cross-correlations for each time lag. The output is a list of array of matrices where each array in the list corresponds to the cross-correlation for a regime.

ccfs<- assigns ccfs to x.

add_ccfs() adds ccfs and sds to x.

Value

ccfs() returns (regime-switching) cross-correlations. add_ccfs() returns the same object with additional attributes of (regime-switching) cross-correlations and (regime-switching) empirical standard deviations.

Examples

```
# Calculate ccfs for 'sim1'
data(sim1)
sim1_mcgf <- mcgf(sim1$data, dists = sim1$dists)
ccfs(sim1_mcgf, lag_max = 5)

# To use multiple cores, use the `ncores` argument
ccfs(sim1_mcgf, lag_max = 5, ncores = 2)

# Add ccfs and sds to 'sim1_mcgf'
sim1_mcgf <- add_ccfs(sim1_mcgf, lag_max = 5)
print(sim1_mcgf, "ccfs")
print(sim1_mcgf, "sds")

# Calculate ccfs for 'sim2'
data(sim2)
sim2_mcgf <- mcgf_rs(sim2$data, dists = sim2$dists, label = sim2$label)
ccfs(sim2_mcgf, lag_max = 5)

# Add ccfs and sds to 'sim2_mcgf'
sim2_mcgf <- add_ccfs(sim2_mcgf, lag_max = 5)
print(sim2_mcgf, "ccfs")
print(sim2_mcgf, "sds")
```

---

ccf_rs — *Calculate regime-switching cross-correlation*

Description

Calculate regime-switching cross-correlation

Usage

```
ccf_rs(x, y, label, lag_max)
```

Arguments

x, y	A univariate numeric time series.
label	A factor of regime labels.
lag_max	Maximum lag at which to calculate the ccf.

Value

Cross-correlations for each group in `label`.

Examples

```
set.seed(123)
x <- rnorm(100)
y <- rnorm(100)
label <- sample(1:2, 100, replace = TRUE)
ccf_rs(x, y, label = factor(label), lag_max = 3)
```

---

ccov *Generic functions for calculating joint covariance/correlation matrix for mcgf objects*

Description

Generic functions for calculating joint covariance/correlation matrix for mcgf objects

Usage

```
ccov(x, ...)
```

Arguments

x	An **R** object.
...	Additional parameters or attributes.

Details

Refer to `ccov.mcgf()` and `ccov.mcgf_rs()` for more details.

Value

Joint correlation/covariance matrix.

ccov.mcgf — *Covariance/correlation for joint distribution of an* `mcgf` *object*

Description

Covariance/correlation for joint distribution of an `mcgf` object

Usage

```
## S3 method for class 'mcgf'
ccov(x, model = c("all", "base", "empirical"), cor = FALSE, ...)
```

Arguments

`x`	An `mcgf` object.
`model`	Which model to use. One of `all`, `base`, and `empirical`.
`cor`	Logical; if TRUE, correlation is outputted.
`...`	Additional arguments. Not in use.

Value

Joint covariance/correlation matrix.

Examples

```
data(sim1)
sim1_mcgf <- mcgf(sim1$data, dists = sim1$dists)
sim1_mcgf <- add_acfs(sim1_mcgf, lag_max = 5)
sim1_mcgf <- add_ccfs(sim1_mcgf, lag_max = 5)

# Fit a separable model and store it to 'sim1_mcgf'
fit_sep <- fit_base(
    sim1_mcgf,
    model = "sep",
    lag = 5,
    par_init = c(
        c = 0.001,
        gamma = 0.5,
        a = 0.3,
        alpha = 0.5
    ),
    par_fixed = c(nugget = 0)
)
sim1_mcgf <- add_base(sim1_mcgf, fit_base = fit_sep)

ccov(sim1_mcgf, model = "base")
```

ccov.mcgf_rs *Covariance/correlation for joint distribution of an* mcgf_rs*object*

Description

Covariance/correlation for joint distribution of an mcgf_rsobject

Usage

```
## S3 method for class 'mcgf_rs'
ccov(x, model = c("all", "base", "empirical"), cor = FALSE, ...)
```

Arguments

x	An mcgf object.
model	Which model to use. One of all, base, and empirical.
cor	Logical; if TRUE, correlation is returned
...	Additional arguments. Not in use.

Value

A list of joint covariance/correlation matrix.

Examples

```
data(sim2)
sim2_mcgf <- mcgf_rs(sim2$data, dists = sim2$dists, label = sim2$label)
sim2_mcgf <- add_acfs(sim2_mcgf, lag_max = 5)
sim2_mcgf <- add_ccfs(sim2_mcgf, lag_max = 5)

# Fit a regime-switching separable model
fit_sep <- fit_base(
    sim2_mcgf,
    lag_ls = 5,
    model_ls = "sep",
    par_init_ls = list(list(
        c = 0.000001,
        gamma = 0.5,
        a = 0.5,
        alpha = 0.5
    )),
    par_fixed_ls = list(c(nugget = 0))
)
sim2_mcgf <- add_base(sim2_mcgf, fit_base_ls = fit_sep)

ccov(sim2_mcgf, model = "base")
```

cor2cov — *Convert correlation to covariance*

Description

Convert correlation to covariance

Usage

```
cor2cov(V, sd, empirical = FALSE)

cor2cov_ar(V, sd, empirical = FALSE)
```

Arguments

V	A correlation matrix, usually positive semi-definite.
sd	A vector of standard deviations.
empirical	Logical; TRUE if V is empirical correlation.

Details

`cor2cov` converts a matrix. `cor2cov_ar` converts an 3-D array.

Value

A correlation matrix.

Examples

```
V <- matrix(c(1, 0.5, 0.5, 1), ncol = 2)
sd <- 1:2
cor2cov(V, sd)

V_ar <- array(c(1, 0.5, 0.5, 1), dim = c(2, 2, 2))
cor2cov_ar(V_ar, sd)
```

cor_cauchy — *Calculate Cauchy correlation*

Description

Calculate Cauchy correlation

Usage

```
cor_cauchy(x, a, alpha, nu = 1, nugget = 0, is.dist = FALSE)
```

Arguments

`x`	A numeric vector, matrix, or array.
`a`	Smooth parameter, $a > 0$.
`alpha`	Scale parameter, $\alpha \in (0, 1]$.
`nu`	Power parameter, $\nu > 0$. Default is 1.
`nugget`	The nugget effect $\in [0, 1]$.
`is.dist`	Logical; if TRUE, `x` is a distance matrix or an array of distance matrices.

Details

The Cauchy correlation function with scale parameter a and smooth parameter α has the form

$$C(x) = (1 - \text{nugget})(a|x|^{2\alpha} + 1)^{-\nu} + \text{nugget} \cdot \delta_{x=0},$$

where $\delta_{x=0}$ is 1 when $x = 0$ and 0 otherwise.

Value

Correlations of the same dimension as `x`.

References

Gneiting, T., and Schlather, M. (2004). Stochastic Models That Separate Fractal Dimension and the Hurst Effect. SIAM Review, 46(2), 269–282.

See Also

Other correlation functions: `cor_exp()`, `cor_fs()`, `cor_lagr_askey()`, `cor_lagr_exp()`, `cor_lagr_tri()`, `cor_sep()`, `cor_stat()`, `cor_stat_rs()`

Examples

```
x <- matrix(c(0, 5, 5, 0), nrow = 2)
cor_cauchy(x, a = 1, alpha = 0.5)

x <- matrix(c(0, 5, 5, 0), nrow = 2)
cor_cauchy(x, a = 1, alpha = 0.5, nugget = 0.3, is.dist = TRUE)
```

cor_exp — *Calculate exponential correlation*

Description

Calculate exponential correlation

Usage

```
cor_exp(x, c, gamma = 1/2, nugget = 0, is.dist = FALSE)
```

Arguments

x	A numeric vector, matrix, or array.
c	Smooth parameter, $c > 0$.
gamma	Scale parameter, $\gamma \in (0, 1/2]$. Default is 1/2.
nugget	The nugget effect $\in [0, 1]$.
is.dist	Logical; if TRUE, x is a distance matrix or an array of distance matrices.

Details

The exponential correlation function with scale parameter c and smooth parameter γ has the form

$$C(x) = (1 - \text{nugget}) \exp(-c|x|^{2\gamma}) + \text{nugget} \cdot \delta_{x=0},$$

where $\delta_{x=0}$ is 1 when $x = 0$ and 0 otherwise.

Value

Correlations of the same dimension as x.

References

Diggle, P. J., Tawn, J. A., & Moyeed, R. A. (1998). Model-Based Geostatistics. Journal of the Royal Statistical Society. Series C (Applied Statistics), 47(3), 299–350.

See Also

Other correlation functions: cor_cauchy(), cor_fs(), cor_lagr_askey(), cor_lagr_exp(), cor_lagr_tri(), cor_sep(), cor_stat(), cor_stat_rs()

Examples

```
x <- matrix(c(0, 5, 5, 0), nrow = 2)
cor_exp(x, c = 0.01, gamma = 0.5)

x <- matrix(c(0, 5, 5, 0), nrow = 2)
cor_exp(x, c = 0.01, gamma = 0.5, nugget = 0.3, is.dist = TRUE)
```

`cor_fs` *Calculate correlation for fully symmetric model*

Description

Calculate correlation for fully symmetric model

Usage

```
cor_fs(nugget = 0, c, gamma = 1/2, a, alpha, beta = 0, h, u)
```

Arguments

`nugget`	The nugget effect $\in [0, 1]$.
`c`	Scale parameter of `cor_exp`, $c > 0$.
`gamma`	Smooth parameter of `cor_exp`, $\gamma \in (0, 1/2]$.
`a`	Scale parameter of `cor_cauchy`, $a > 0$.
`alpha`	Smooth parameter of `cor_cauchy`, $\alpha \in (0, 1]$.
`beta`	Interaction parameter, $\beta \in [0, 1]$.
`h`	Euclidean distance matrix or array.
`u`	Time lag, same dimension as `h`.

Details

The fully symmetric correlation function with interaction parameter β has the form

$$C(\mathbf{h}, u) = \frac{1}{(a|u|^{2\alpha} + 1)} \left((1 - \text{nugget}) \exp\left(\frac{-c\|\mathbf{h}\|^{2\gamma}}{(a|u|^{2\alpha} + 1)^{\beta\gamma}} \right) + \text{nugget} \cdot \delta_{\mathbf{h}=\mathbf{0}} \right),$$

where $\|\cdot\|$ is the Euclidean distance, and $\delta_{x=0}$ is 1 when $x = 0$ and 0 otherwise. Here $\mathbf{h} \in \mathbb{R}^2$ and $u \in \mathbb{R}$. By default `beta = 0` and it reduces to the separable model.

Value

Correlations of the same dimension as `h` and `u`.

References

Gneiting, T. (2002). Nonseparable, Stationary Covariance Functions for Space–Time Data, Journal of the American Statistical Association, 97:458, 590-600.

See Also

Other correlation functions: `cor_cauchy()`, `cor_exp()`, `cor_lagr_askey()`, `cor_lagr_exp()`, `cor_lagr_tri()`, `cor_sep()`, `cor_stat()`, `cor_stat_rs()`

Examples

```
h <- matrix(c(0, 5, 5, 0), nrow = 2)
u <- matrix(0, nrow = 2, ncol = 2)
cor_fs(c = 0.01, gamma = 0.5, a = 1, alpha = 0.5, beta = 0.5, h = h, u = u)

h <- array(c(0, 5, 5, 0), dim = c(2, 2, 3))
u <- array(rep(0:2, each = 4), dim = c(2, 2, 3))
cor_fs(c = 0.01, gamma = 0.5, a = 1, alpha = 0.5, beta = 0.5, h = h, u = u)
```

`cor_lagr_askey` *Calculate Lagrangian correlation of the Askey form*

Description

Calculate Lagrangian correlation of the Askey form

Usage

```
cor_lagr_askey(v1, v2, k = 2, h1, h2, u)
```

Arguments

`v1`	Prevailing wind, u-component.
`v2`	Prevailing wind, v-component.
`k`	Scale parameter of $\|\boldsymbol{v}\|$, $k > 0$. Default is 2.
`h1`	Horizontal distance matrix or array.
`h2`	Vertical distance matrix or array, same dimension as `h1`.
`u`	Time lag, same dimension as `h1`.

Details

The Lagrangian correlation function of the Askey form with parameters $\boldsymbol{v} = (v_1, v_2)^\top \in \mathbb{R}^2$ has the form

$$C(\mathbf{h}, u) = \left(1 - \frac{1}{k\|\boldsymbol{v}\|} \|\mathbf{h} - u\boldsymbol{v}\|\right)_+^{3/2},$$

where $\| \cdot \|$ is the Euclidean distance, $x_+ = \max(x, 0)$, $\mathbf{h} = (\mathrm{h}_1, \mathrm{h}_2)^\top \in \mathbb{R}^2$, and $k > 0$ is the scale parameter controlling the magnitude of asymmetry in correlation.

Value

Correlations of the same dimension as `h1`.

References

Askey, R. (1973). Radial characteristic functions, Tech. Report No. 1262, Math. Research Center, University of Wisconsin-Madison.

See Also

Other correlation functions: cor_cauchy(), cor_exp(), cor_fs(), cor_lagr_exp(), cor_lagr_tri(), cor_sep(), cor_stat(), cor_stat_rs()

Examples

```
h1 <- matrix(c(0, -5, 5, 0), nrow = 2)
h2 <- matrix(c(0, -8, 8, 0), nrow = 2)
u <- matrix(0.1, nrow = 2, ncol = 2)
cor_lagr_askey(v1 = 5, v2 = 10, h1 = h1, h2 = h2, u = u)

h1 <- array(c(0, -10, 10, 0), dim = c(2, 2, 3))
h2 <- array(c(0, -10, 10, 0), dim = c(2, 2, 3))
u <- array(rep(-c(1, 2, 3), each = 4), dim = c(2, 2, 3))
cor_lagr_askey(v1 = 10, v2 = 10, h1 = h1, h2 = h2, u = u)
```

---

cor_lagr_exp — *Calculate Lagrangian correlation of the exponential form*

Description

Calculate Lagrangian correlation of the exponential form

Usage

```
cor_lagr_exp(v1, v2, k = 2, h1, h2, u)
```

Arguments

v1	Prevailing wind, u-component.
v2	Prevailing wind, v-component.
k	Scale parameter of $\|\boldsymbol{v}\|$, $k > 0$. Default is 2.
h1	Horizontal distance matrix or array.
h2	Vertical distance matrix or array, same dimension as h1.
u	Time lag, same dimension as h1.

Details

The Lagrangian correlation function of the exponential form with parameters $\boldsymbol{v} = (v_1, v_2)^\top \in \mathbb{R}^2$ has the form

$$C(\mathbf{h}, u) = \exp\left(-\frac{1}{k\|\boldsymbol{v}\|}\|\mathbf{h} - u\boldsymbol{v}\|\right),$$

where $\|\cdot\|$ is the Euclidean distance, $\mathbf{h} = (\mathrm{h}_1, \mathrm{h}_2)^\top \in \mathbb{R}^2$, and $k > 0$ is the scale parameter controlling the magnitude of asymmetry in correlation.

Value

Correlations of the same dimension as h1.

References

Diggle, P. J., Tawn, J. A., & Moyeed, R. A. (1998). Model-Based Geostatistics. Journal of the Royal Statistical Society. Series C (Applied Statistics), 47(3), 299–350.

See Also

Other correlation functions: cor_cauchy(), cor_exp(), cor_fs(), cor_lagr_askey(), cor_lagr_tri(), cor_sep(), cor_stat(), cor_stat_rs()

Examples

```
h1 <- matrix(c(0, -5, 5, 0), nrow = 2)
h2 <- matrix(c(0, -8, 8, 0), nrow = 2)
u <- matrix(0.1, nrow = 2, ncol = 2)
cor_lagr_exp(v1 = 5, v2 = 10, h1 = h1, h2 = h2, u = u)

h1 <- array(c(0, -10, 10, 0), dim = c(2, 2, 3))
h2 <- array(c(0, -10, 10, 0), dim = c(2, 2, 3))
u <- array(rep(-c(1, 2, 3), each = 4), dim = c(2, 2, 3))
cor_lagr_exp(v1 = 10, v2 = 10, h1 = h1, h2 = h2, u = u)
```

---

cor_lagr_tri *Calculate Lagrangian correlation of the triangular form*

Description

Calculate Lagrangian correlation of the triangular form

Usage

```
cor_lagr_tri(v1, v2, k = 2, h1, h2, u)
```

Arguments

v1	Prevailing wind, u-component.
v2	Prevailing wind, v-component.
k	Scale parameter of $\|\boldsymbol{v}\|$, $k > 0$. Default is 2.
h1	Horizontal distance matrix or array.
h2	Vertical distance matrix or array, same dimension as h1.
u	Time lag, same dimension as h1.

Details

The Lagrangian correlation function of the triangular form with parameters $\boldsymbol{v} = (v_1, v_2)^\top \in \mathbb{R}^2$ has the form

$$C(\mathbf{h}, u) = \left(1 - \frac{1}{k\|\boldsymbol{v}\|}\left|\frac{\mathbf{h}^\top \boldsymbol{v}}{\|\boldsymbol{v}\|} - u\|\boldsymbol{v}\|\right|\right)_+,$$

where $\|\cdot\|$ is the Euclidean distance, $x_+ = \max(x, 0)$, $\mathbf{h} = (\mathrm{h}_1, \mathrm{h}_2)^\top \in \mathbb{R}^2$, and $k > 0$ is the scale parameter controlling the magnitude of asymmetry in correlation.

Value

Correlations of the same dimension as `h1`.

See Also

Other correlation functions: `cor_cauchy()`, `cor_exp()`, `cor_fs()`, `cor_lagr_askey()`, `cor_lagr_exp()`, `cor_sep()`, `cor_stat()`, `cor_stat_rs()`

Examples

```
h1 <- matrix(c(0, -5, 5, 0), nrow = 2)
h2 <- matrix(c(0, -8, 8, 0), nrow = 2)
u <- matrix(0.1, nrow = 2, ncol = 2)
cor_lagr_tri(v1 = 5, v2 = 10, h1 = h1, h2 = h2, u = u)

h1 <- array(c(0, -10, 10, 0), dim = c(2, 2, 3))
h2 <- array(c(0, -10, 10, 0), dim = c(2, 2, 3))
u <- array(rep(-c(1, 2, 3), each = 4), dim = c(2, 2, 3))
cor_lagr_tri(v1 = 10, v2 = 10, h1 = h1, h2 = h2, u = u)
```

`cor_sep` *Calculate correlation for separable model*

Description

Calculate correlation for separable model

Usage

```
cor_sep(
  spatial = c("exp", "cauchy"),
  temporal = c("exp", "cauchy"),
  par_s,
  par_t,
  h,
  u
)
```

Arguments

spatial	Pure spatial model, exp or cauchy for now.
temporal	Pure temporal model, exp or cauchy for now.
par_s	Parameters for the pure spatial model. Nugget effect supported.
par_t	Parameters for the pure temporal model.
h	Euclidean distance matrix or array.
u	Time lag, same dimension as h.

Details

The separable model is the product of a pure temporal model, $C_T(u)$, and a pure spatial model, $C_S(\mathbf{h})$. It is of the form

$$C(\mathbf{h}, u) = C_T(u)\left[(1 - \text{nugget})C_S(\mathbf{h}) + \text{nugget}\delta_{\mathbf{h}=0}\right],$$

where $\delta_{x=0}$ is 1 when $x = 0$ and 0 otherwise. Here $\mathbf{h} \in \mathbb{R}^2$ and $u \in \mathbb{R}$. Now only exponential and Cauchy correlation models are available.

Value

Correlations of the same dimension as h and u.

References

Gneiting, T. (2002). Nonseparable, Stationary Covariance Functions for Space–Time Data, Journal of the American Statistical Association, 97:458, 590-600.

See Also

Other correlation functions: cor_cauchy(), cor_exp(), cor_fs(), cor_lagr_askey(), cor_lagr_exp(), cor_lagr_tri(), cor_stat(), cor_stat_rs()

Examples

```
h <- matrix(c(0, 5, 5, 0), nrow = 2)
par_s <- list(nugget = 0.5, c = 0.01, gamma = 0.5)
u <- matrix(0, nrow = 2, ncol = 2)
par_t <- list(a = 1, alpha = 0.5)
cor_sep(
    spatial = "exp", temporal = "cauchy", par_s = par_s, par_t = par_t,
    h = h, u = u
)

h <- array(c(0, 5, 5, 0), dim = c(2, 2, 3))
par_s <- list(nugget = 0.5, c = 0.01, gamma = 0.5)
u <- array(rep(0:2, each = 4), dim = c(2, 2, 3))
par_t <- list(a = 1, alpha = 0.5)
cor_sep(
    spatial = "exp", temporal = "cauchy", par_s = par_s, par_t = par_t,
```

```
    h = h, u = u
)
```

cor_stat *Calculate general stationary correlation.*

Description

Calculate general stationary correlation.

Usage

```
cor_stat(
  base = c("sep", "fs"),
  lagrangian = c("none", "lagr_tri", "lagr_askey"),
  par_base,
  par_lagr,
  lambda,
  h,
  h1,
  h2,
  u,
  base_fixed = FALSE
)
```

Arguments

base	Base model, `sep` or `fs` for now. Or correlation matrix/array.
lagrangian	Lagrangian model, `none`, `lagr_tri`, or `lagr_askey`.
par_base	Parameters for the base model (symmetric), used only when `base_fixed = FALSE`.
par_lagr	Parameters for the Lagrangian model. Used only when `lagrangian` is not `none`.
lambda	Weight of the Lagrangian term, $\lambda \in [0, 1]$.
h	Euclidean distance matrix or array, used only when `base_fixed = FALSE`.
h1	Horizontal distance matrix or array, same dimension as `h`. Used only when `lagrangian` is not `none`.
h2	Vertical distance matrix or array, same dimension as `h`. Used only when `lagrangian` is not `none`.
u	Time lag, same dimension as `h`.
base_fixed	Logical; if TRUE, `base` is the correlation.

Details

The general station model, a convex combination of a base model and a Lagrangian model, has the form
$$C(\mathbf{h}, u) = (1 - \lambda)C_{\text{Base}}(\mathbf{h}, u) + \lambda C_{\text{Lagr}}(\mathbf{h}, u),$$
where λ is the weight of the Lagrangian term.

If `base_fixed = TRUE`, the correlation is of the form

$$C(\mathbf{h}, u) = (1 - \lambda)C_{\text{Base}} + \lambda C_{\text{Lagr}}(\mathbf{h}, u),$$

where `base` is a correlation matrix/array and `par_base` and `h` are not used.

When `lagrangian = "none"`, `lambda` must be 0.

Value

Correlations for the general stationary model. Same dimension of `base` if `base_fixed = FALSE`.

See Also

Other correlation functions: `cor_cauchy()`, `cor_exp()`, `cor_fs()`, `cor_lagr_askey()`, `cor_lagr_exp()`, `cor_lagr_tri()`, `cor_sep()`, `cor_stat_rs()`

Examples

```
par_s <- list(nugget = 0.5, c = 0.01, gamma = 0.5)
par_t <- list(a = 1, alpha = 0.5)
par_base <- list(par_s = par_s, par_t = par_t)
par_lagr <- list(v1 = 5, v2 = 10)
h1 <- matrix(c(0, 5, -5, 0), nrow = 2)
h2 <- matrix(c(0, 8, -8, 0), nrow = 2)
h <- sqrt(h1^2 + h2^2)
u <- matrix(0.1, nrow = 2, ncol = 2)
cor_stat(
    base = "sep", lagrangian = "lagr_tri", par_base = par_base,
    par_lagr = par_lagr, lambda = 0.8, h = h, h1 = h1, h2 = h2, u = u
)

h1 <- array(c(0, 5, -5, 0), dim = c(2, 2, 3))
h2 <- array(c(0, 8, -8, 0), dim = c(2, 2, 3))
h <- sqrt(h1^2 + h2^2)
u <- array(rep(c(0.1, 0.2, 0.3), each = 4), dim = c(2, 2, 3))
fit_base <- cor_fs(
    nugget = 0.5, c = 0.01, gamma = 0.5, a = 1, alpha = 0.5,
    beta = 0.0, h = h, u = u
)
par_lagr <- list(v1 = 5, v2 = 10)
cor_stat(
    base = fit_base, lagrangian = "lagr_askey", par_lagr = par_lagr,
    h1 = h1, h2 = h2, u = u, lambda = 0.8, base_fixed = TRUE
)
```

cor_stat_rs *Calculate general stationary correlation.*

Description

Calculate general stationary correlation.

Usage

```
cor_stat_rs(
  n_regime,
  base_ls,
  lagrangian_ls,
  par_base_ls,
  par_lagr_ls,
  lambda_ls,
  h_ls,
  h1_ls,
  h2_ls,
  u_ls,
  base_fixed = FALSE
)
```

Arguments

`n_regime`	Integer, number of regimes.
`base_ls`	List of base model, `sep` or `fs` for now. Or list of correlation matrices/arrays.
`lagrangian_ls`	List of Lagrangian model, `lagr_tri` or `lagr_askey` for now.
`par_base_ls`	List of parameters for the base model, used only when `base_fixed = FALSE`.
`par_lagr_ls`	List of parameters for the Lagrangian model. Used only when `lagrangian_ls` is not none.
`lambda_ls`	List of weight of the Lagrangian term, $\lambda \in [0, 1]$.
`h_ls`	List of Euclidean distance matrix or array, used only when `base_fixed = FALSE`.
`h1_ls`	List of horizontal distance matrix or array, same dimension as `h_ls`. Used only when `lagrangian_ls` is not none.
`h2_ls`	List of vertical distance matrix or array, same dimension as `h_ls`. Used only when `lagrangian_ls` is not none.
`u_ls`	List of time lag, same dimension as `h_ls`.
`base_fixed`	Logical; if TRUE, `base_ls` is the list of correlation.

Details

It gives a list of general stationary correlation for `n_regime` regimes. See cor_stat for the model details. Model parameters are lists of length 1 or `n_regime`. When length is 1, same values are used for all regimes. If `base_fixed = TRUE`, the base is a list of correlation and `par_base_ls` and `h_ls` are not used.

Value

Correlations for the general stationary model. Same dimension of `base_ls` if `base_fixed` = `TRUE`.

See Also

Other correlation functions: `cor_cauchy()`, `cor_exp()`, `cor_fs()`, `cor_lagr_askey()`, `cor_lagr_exp()`, `cor_lagr_tri()`, `cor_sep()`, `cor_stat()`

Examples

```
# Fit general stationary model with sep base.
par_s <- list(nugget = 0.5, c = 0.01, gamma = 0.5)
par_t <- list(a = 1, alpha = 0.5)
par_base <- list(par_s = par_s, par_t = par_t)
h1 <- array(c(0, 5, -5, 0), dim = c(2, 2, 3))
h2 <- array(c(0, 8, -8, 0), dim = c(2, 2, 3))
h <- sqrt(h1^2 + h2^2)
u <- array(rep(c(1, 2, 3), each = 4), dim = c(2, 2, 3))
cor_stat_rs(
    n_regime = 2,
    base_ls = list("sep"),
    lagrangian_ls = list("none", "lagr_tri"),
    par_base_ls = list(par_base),
    par_lagr_ls = list(NULL, list(v1 = 10, v2 = 20)),
    lambda_ls = list(0, 0.2),
    h_ls = list(h),
    h1_ls = list(NULL, h1),
    h2_ls = list(NULL, h2),
    u_ls = list(u, u + 1)
)

# Fit general stationary model given fs as the base model.
h1 <- array(c(0, 5, -5, 0), dim = c(2, 2, 3))
h2 <- array(c(0, 8, -8, 0), dim = c(2, 2, 3))
h <- sqrt(h1^2 + h2^2)
u <- array(rep(c(0.1, 0.2, 0.3), each = 4), dim = c(2, 2, 3))
fit_base <- cor_fs(
    nugget = 0.5, c = 0.01, gamma = 0.5, a = 1, alpha = 0.5,
    beta = 0.0, h = h, u = u
)
par_lagr <- list(v1 = 5, v2 = 10)
cor_stat_rs(
    n_regime = 2,
    par_lagr_ls = list(par_lagr),
    h1_ls = list(h1),
    h2_ls = list(h2),
    u_ls = list(u, u + 1),
    lambda_ls = list(0, 0.8),
    base_ls = list(fit_base),
    lagrangian = list("lagr_tri", "lagr_askey"),
    base_fixed = TRUE
)
```

dists — *Generic function for calculating distance matrices*

Description

Generic function for calculating distance matrices

Usage

```
dists(x, ...)
```

Arguments

x	An **R** object.
...	Additional parameters or attributes.

Value

A list of signed distance matrices: `h` (Euclidean), `h1` (horizontal), and `h2` (vertical) with the same dimensions.

dists.mcgf — *Calculating distance matrices for an `mcgf` object*

Description

Calculating distance matrices for an `mcgf` object

Usage

```
## S3 method for class 'mcgf'
dists(x, return_grid = FALSE, ...)

dists(x) <- value
```

Arguments

x	An `mcgf` object.
return_grid	Logical; used when `locations` in `x` are longitudes and latitudes.
...	Additional parameters or attributes.
value	List of signed distance matrices, outputted from `dists()`.

Details

If the dists attribute is available in x, it will be printed. Otherwise dists will be calculated based on the locations attribute.

Value

A list of signed distance matrices: h (Euclidean), h1 (horizontal), and h2 (vertical).

Examples

```
data <- cbind(S1 = 1:5, S2 = 4:8, S3 = 5:9)
lon <- c(110, 120, 130)
lat <- c(50, 55, 60)
locations <- cbind(lon, lat)

# if locations are longitudes and latitudes
obj <- mcgf(data = data, locations = locations)
obj
dists(obj)
dists(obj) <- dists(obj)
obj

# if locations are just coordinates in a 2D plane:
obj <- mcgf(data = data, locations = locations, longlat = FALSE)
obj

# calculate distances
dists(obj)

# add distances to the `mcgf` object
dists(obj) <- dists(obj)
obj
```

---

find_dists *Calculate (signed) distances between coordinates*

Description

Calculate (signed) distances between coordinates

Usage

```
find_dists(locations, longlat = TRUE, origin = 1L, return_grid = FALSE, ...)
```

Arguments

`locations`	A matrix or data.frame of 2D points, the first column is x/longitude, and the second column is y/latitude.
`longlat`	Logical, if TURE Great Circle (WGS84 ellipsoid) distance; if FALSE, Euclidean distance.
`origin`	Optional; used when `longlat` is TRUE. An integer index indicating the reference location which will be used as the origin.
`return_grid`	Logical; used when `longlat` is TRUE. If TRUE the mapped coordinates on a 2D plane is returned.
`...`	Optional arguments passed to `.find_dists()`.

Details

`locations` must be a matrix or data.frame containing 2 columns, first column x/longitude, and second column y/latitude. The row names of `locations` are used as the names of the locations.

If `longlat` is TRUE, the original coordinates are mapped to a 2D Euclidean plane given the reference location. First, the Great Circle (WGS84 ellipsoid) signed distance matrices are calculated, where the original latitudes are replaced by the the mean of them to find the signed longitudinal distances and the original longitudes are replaced by the the mean of them to find the signed latitudinal distances. Then given the index of a reference location `origin`, a new set of coordinates in a 2D plane is generated where the coordinates are determined by the signed distances between the locations and the reference location. Finally distance matrices of the new coordinates are outputted.

Value

A list of distance matrices. If `return_grid` is TRUE, a list consists of a list of distance matrices, the mapped 2D grid, and the origin is returned.

Examples

```
lon <- c(110, 120, 130)
lat <- c(50, 55, 60)
locations <- cbind(lon, lat)
rownames(locations) <- paste("Site", 1:3)
find_dists(locations)
```

`find_dists_new` *Calculate (signed) distances between coordinates*

Description

Calculate (signed) distances between coordinates

Usage

```
find_dists_new(
  locations,
  locations_new,
  longlat = TRUE,
  origin = 1L,
  return_grid = FALSE,
  ...
)
```

Arguments

`locations`	A matrix or data.frame of 2D points, the first column is x/longitude, and the second column is y/latitude.
`locations_new`	A matrix or data.frame of 2D points, the first column is x/longitude, and the second column is y/latitude.
`longlat`	Logical, if TURE Great Circle (WGS84 ellipsoid) distance; if FALSE, Euclidean distance.
`origin`	Optional; used when `longlat` is TRUE. An integer index indicating the reference location from `locations` which will be used as the origin. Same `origin` from `find_dists` must be used to ensure consistancy between outputs from `find_dists` and `find_dists_new`.
`return_grid`	Logical; used when `longlat` is TRUE. If TRUE the mapped coordinates on a 2D plane for all locations is returned.
`...`	Optional arguments passed to `.find_dists_new()`.

Details

`locations` and `locations_new` must be matrices or data.frames containing 2 columns, first column x/longitude, and second column y/latitude. The row names of `locations` and `locations_new` are used as the names of the locations.

If `longlat` is TRUE, the original coordinates are mapped to a 2D Euclidean plane given the reference location from `locations`. First, the Great Circle (WGS84 ellipsoid) signed distance matrices are calculated, where the original latitudes are replaced by the the mean of latitudes in `locations` to find the signed longitudinal distances and the original longitudes are replaced by the the mean of longitudes in `locations` to find the signed latitudinal distances. Then given the index of a reference location `origin`, a new set of coordinates in a 2D plane is generated where the coordinates are determined by the signed distances between the locations and the reference location. Finally distance matrices of the new coordinates for all stations are outputted.

Value

A list of distance matrices for all locations. If `return_grid` is TRUE, a list consists of a list of distance matrices for all locations, the mapped 2D grid for all locations, and the origin is returned.

Examples

```
lon <- c(110, 120, 130)
lat <- c(50, 55, 60)
locations <- cbind(lon, lat)
rownames(locations) <- paste("Site", 1:3)
find_dists(locations)

locations_new <- c(115, 55)
find_dists_new(locations, locations_new)
```

fit_base *Fit correlation base models*

Description

Fit correlation base models

Usage

```
fit_base(x, ...)
```

Arguments

x	An **R** object.
...	Additional parameters or attributes.

Details

Refer to `fit_base.mcgf()` and `fit_base.mcgf_rs()` for more details.

Value

A vector of estimated parameters.

fit_base.mcgf *Parameter estimation for symmetric correlation functions for an* `mcgf` *object.*

Description

Parameter estimation for symmetric correlation functions for an `mcgf` object.

Usage

```
## S3 method for class 'mcgf'
fit_base(
  x,
  lag,
  horizon = 1,
  model = c("spatial", "temporal", "sep", "fs", "none"),
  method = c("wls", "mle"),
  optim_fn = c("nlminb", "optim", "other"),
  par_fixed = NULL,
  par_init,
  lower = NULL,
  upper = NULL,
  other_optim_fn = NULL,
  dists_base = NULL,
  scale_time = 1,
  ...
)
```

Arguments

x	An mcgf object containing attributes dists, acfs, ccfs, and sds.
lag	Integer time lag.
horizon	Integer forecast horizon.
model	Base model, one of spatial, temporal, sep, fs, none. Only sep and fs are supported when method = mle. If none, NULLs are returned.
method	Parameter estimation methods, weighted least square (wls) or maximum likelihood estimation (mle).
optim_fn	Optimization functions, one of nlminb, optim, other. When optim_fn = other, supply other_optim_fn.
par_fixed	Fixed parameters.
par_init	Initial values for parameters to be optimized.
lower	Optional; lower bounds of parameters.
upper	Optional: upper bounds of parameters.
other_optim_fn	Optional, other optimization functions. The first two arguments must be initial values for the parameters and a function to be minimized respectively (same as that of optim and nlminb).
dists_base	List of distance matrices. If NULL, dists(x) is used. Must be a matrix or an array of distance matrices.
scale_time	Scale of time unit, default is 1. lag is divided by scale_time for parameter estimation.
...	Additional arguments passed to optim_fn.

Details

This function fits the separable and fully symmetric models using weighted least squares and maximum likelihood estimation. Optimization functions such as `nlminb` and `optim` are supported. Other functions are also supported by setting `optim_fn = "other"` and supplying `other_optim_fn`. `lower` and `upper` are lower and upper bounds of parameters in `par_init` and default bounds are used if they are not specified.

Note that both `wls` and `mle` are heuristic approaches when `x` contains observations from a subset of the discrete spatial domain, though estimation results are close to that using the full spatial domain for large sample sizes.

Value

A list containing outputs from optimization functions of `optim_fn`.

See Also

Other functions on fitting an mcgf: `add_base.mcgf()`, `add_lagr.mcgf()`, `fit_lagr.mcgf()`, `krige.mcgf()`, `krige_new.mcgf()`

Examples

```
data(sim1)
sim1_mcgf <- mcgf(sim1$data, dists = sim1$dists)
sim1_mcgf <- add_acfs(sim1_mcgf, lag_max = 5)
sim1_mcgf <- add_ccfs(sim1_mcgf, lag_max = 5)

# Fit a pure spatial model
fit_spatial <- fit_base(
    sim1_mcgf,
    model = "spatial",
    lag = 5,
    par_init = c(c = 0.001, gamma = 0.5),
    par_fixed = c(nugget = 0)
)
fit_spatial$fit

# Fit a pure temporal model
fit_temporal <- fit_base(
    sim1_mcgf,
    model = "temporal",
    lag = 5,
    par_init = c(a = 0.3, alpha = 0.5)
)
fit_temporal$fit

# Fit a separable model
fit_sep <- fit_base(
    sim1_mcgf,
    model = "sep",
    lag = 5,
    par_init = c(
```

```
        c = 0.001,
        gamma = 0.5,
        a = 0.3,
        alpha = 0.5
    ),
    par_fixed = c(nugget = 0)
)
fit_sep$fit
```

`fit_base.mcgf_rs` *Parameter estimation for symmetric correlation functions for an* `mcgf_rs` *object.*

Description

Parameter estimation for symmetric correlation functions for an `mcgf_rs` object.

Usage

```
## S3 method for class 'mcgf_rs'
fit_base(
  x,
  lag_ls,
  horizon = 1,
  model_ls,
  method_ls = "wls",
  optim_fn_ls = "nlminb",
  par_fixed_ls = list(NULL),
  par_init_ls,
  lower_ls = list(NULL),
  upper_ls = list(NULL),
  other_optim_fn_ls = list(NULL),
  dists_base_ls = list(NULL),
  scale_time = 1,
  rs = TRUE,
  ...
)
```

Arguments

`x`	An `mcgf_rs` object containing attributes `dists`, `acfs`, `ccfs`, and `sds`.
`lag_ls`	List of integer time lags.
`horizon`	Integer forecast horizon.
`model_ls`	List of base models, each element must be one of `spatial`, `temporal`, `sep`, `fs`. Only `sep` and `fs` are supported when `mle` is used in `model_ls`.
`method_ls`	List of parameter estimation methods, weighted least square (`wls`) or maximum likelihood estimation (`mle`).

`optim_fn_ls`	List of optimization functions, each element must be one of `nlminb`, `optim`, `other`. When use `other`, supply `other_optim_fn_ls`.
`par_fixed_ls`	List of fixed parameters.
`par_init_ls`	List of initial values for parameters to be optimized.
`lower_ls`	Optional; list of lower bounds of parameters.
`upper_ls`	Optional: list of upper bounds of parameters.
`other_optim_fn_ls`	Optional, list of other optimization functions. The first two arguments must be initial values for the parameters and a function to be minimized respectively (same as that of `optim` and `nlminb`).
`dists_base_ls`	List of lists of distance matrices. If NULL, `dists(x)` is used. Each element must be a matrix or an array of distance matrices.
`scale_time`	Scale of time unit, default is 1. `lag` is divided by `scale_time` for parameter estimation.
`rs`	Logical; if TRUE `x` is treated as a regime-switching, FALSE if the parameters need to be estimated in a non-regime-switching setting.
`...`	Additional arguments passed to all `optim_fn_ls`.

Details

This functions is the regime-switching variant of `fit_base.mcgf()`. Arguments are in lists. The length of arguments that end in `_ls` must be 1 or the same as the number of regimes in `x`. If the length of an argument is 1, then it is set the same for all regimes. Refer to `fit_base.mcgf()` for more details of the arguments.

Note that both `wls` and `mle` are heuristic approaches when `x` contains observations from a subset of the discrete spatial domain, though estimation results are close to that using the full spatial domain for large sample sizes.

Value

A list containing outputs from optimization functions of `optim_fn` for each regime.

See Also

Other functions on fitting an mcgf_rs: `add_base.mcgf_rs()`, `add_lagr.mcgf_rs()`, `fit_lagr.mcgf_rs()`, `krige.mcgf_rs()`, `krige_new.mcgf_rs()`

Examples

```
data(sim2)
sim2_mcgf <- mcgf_rs(sim2$data, dists = sim2$dists, label = sim2$label)
sim2_mcgf <- add_acfs(sim2_mcgf, lag_max = 5)
sim2_mcgf <- add_ccfs(sim2_mcgf, lag_max = 5)

# Fit a regime-switching pure spatial model
fit_spatial <-
    fit_base(
```

```
        sim2_mcgf,
        lag_ls = 5,
        model_ls = "spatial",
        par_init_ls = list(c(c = 0.00005, gamma = 0.5)),
        par_fixed_ls = list(c(nugget = 0))
    )
lapply(fit_spatial[1:2], function(x) x$fit)

# Fit a regime-switching pure temporal model
fit_temporal <-
    fit_base(
        sim2_mcgf,
        lag_ls = 5,
        model_ls = "temporal",
        par_init_ls = list(
            list(a = 0.8, alpha = 0.8),
            list(a = 0.1, alpha = 0.1)
        )
    )
lapply(fit_temporal[1:2], function(x) x$fit)

# Fit a regime-switching separable model
fit_sep <- fit_base(
    sim2_mcgf,
    lag_ls = 5,
    model_ls = "sep",
    par_init_ls = list(list(
        c = 0.00005,
        gamma = 0.5,
        a = 0.5,
        alpha = 0.5
    )),
    par_fixed_ls = list(c(nugget = 0))
)
lapply(fit_sep[1:2], function(x) x$fit)
```

`fit_lagr` *Fit correlation Lagrangian models*

Description

Fit correlation Lagrangian models

Usage

```
fit_lagr(x, ...)
```

Arguments

x	An **R** object.
...	Additional parameters or attributes.

Details

Refer to fit_lagr.mcgf() and fit_lagr.mcgf_rs() for more details.

Value

A vector of estimated parameters.

---

fit_lagr.mcgf — *Parameter estimation for Lagrangian correlation functions for an* mcgf *object.*

Description

Parameter estimation for Lagrangian correlation functions for an mcgf object.

Usage

```
## S3 method for class 'mcgf'
fit_lagr(
  x,
  model = c("lagr_tri", "lagr_askey", "lagr_exp", "none"),
  method = c("wls", "mle"),
  optim_fn = c("nlminb", "optim", "other"),
  par_fixed = NULL,
  par_init,
  lower = NULL,
  upper = NULL,
  other_optim_fn = NULL,
  dists_base = FALSE,
  dists_lagr = NULL,
  ...
)
```

Arguments

x	An mcgf object containing attributes dists, acfs, ccfs, and sds. x must have been passed to add_base() or base<-
model	Base model, one of lagr_tri, lagr_askey, lagr_exp, or none. If none, NULLs are returned.
method	Parameter estimation methods, weighted least square (wls) or maximum likelihood estimation (mle).
optim_fn	Optimization functions, one of nlminb, optim, other. When optim_fn = other, supply other_optim_fn.
par_fixed	Fixed parameters.
par_init	Initial values for parameters to be optimized.

lower	Optional; lower bounds of parameters lambda, v1, v2, and k.
upper	Optional: upper bounds of parameters lambda, v1, v2, and k.
other_optim_fn	Optional, other optimization functions. The first two arguments must be initial values for the parameters and a function to be minimized respectively (same as that of optim and nlminb).
dists_base	Logical; if TRUE dists_base from the base model is used as the distance.
dists_lagr	List of distance matrices/arrays. Used when dists_base is FALSE. If NULL, dists(x) is used.
...	Additional arguments passed to optim_fn.

Details

This function fits the Lagrangian models using weighted least squares and maximum likelihood estimation. The base model must be fitted first using add_base() or base<-. Optimization functions such as nlminb and optim are supported. Other functions are also supported by setting optim_fn = "other" and supplying other_optim_fn. lower and upper are lower and upper bounds of parameters in par_init and default bounds are used if they are not specified.

Note that both wls and mle are heuristic approaches when x contains observations from a subset of the discrete spatial domain, though estimation results are close to that using the full spatial domain for large sample sizes.

Since parameters for the base model and the Lagrangian model are estimated sequentially, more accurate estimation may be obtained if the full model is fitted all at once.

Value

A list containing outputs from optimization functions of optim_fn.

See Also

Other functions on fitting an mcgf: add_base.mcgf(), add_lagr.mcgf(), fit_base.mcgf(), krige.mcgf(), krige_new.mcgf()

Examples

```
data(sim1)
sim1_mcgf <- mcgf(sim1$data, dists = sim1$dists)
sim1_mcgf <- add_acfs(sim1_mcgf, lag_max = 5)
sim1_mcgf <- add_ccfs(sim1_mcgf, lag_max = 5)

# Fit a separable model and store it to 'sim1_mcgf'
fit_sep <- fit_base(
    sim1_mcgf,
    model = "sep",
    lag = 5,
    par_init = c(
        c = 0.001,
        gamma = 0.5,
        a = 0.3,
```

```
        alpha = 0.5
    ),
    par_fixed = c(nugget = 0)
)
sim1_mcgf <- add_base(sim1_mcgf, fit_base = fit_sep)

# Fit a Lagrangian model
fit_lagr <- fit_lagr(
    sim1_mcgf,
    model = "lagr_tri",
    par_init = c(v1 = 300, v2 = 300, lambda = 0.15),
    par_fixed = c(k = 2)
)
fit_lagr$fit
```

`fit_lagr.mcgf_rs` *Parameter estimation for Lagrangian correlation functions for an* `mcgf_rs` *object.*

Description

Parameter estimation for Lagrangian correlation functions for an `mcgf_rs` object.

Usage

```
## S3 method for class 'mcgf_rs'
fit_lagr(
  x,
  model_ls,
  method_ls = "wls",
  optim_fn_ls = "nlminb",
  par_fixed_ls = list(NULL),
  par_init_ls,
  lower_ls = list(NULL),
  upper_ls = list(NULL),
  other_optim_fn_ls = list(NULL),
  dists_base_ls,
  dists_lagr_ls = list(NULL),
  rs = TRUE,
  ...
)
```

Arguments

`x`	An `mcgf_rs` object containing attributes `dists`, `acfs`, `ccfs`, and `sds`. `x` must have been passed to `add_base()` or `base<-`
`model_ls`	List of base models, each element must be one of `lagr_tri`, `lagr_askey`, `lagr_exp`, or none. If none, NULLs are returned.

`method_ls` List of parameter estimation methods, weighted least square (`wls`) or maximum likelihood estimation (`mle`).

`optim_fn_ls` List of optimization functions, each element must be one of `nlminb`, `optim`, `other`. When use `other`, supply `other_optim_fn_ls`

`par_fixed_ls` List of fixed parameters.

`par_init_ls` List of initial values for parameters to be optimized.

`lower_ls` Optional; list of lower bounds of parameters.

`upper_ls` Optional: list of upper bounds of parameters.

`other_optim_fn_ls`
Optional, list of other optimization functions. The first two arguments must be initial values for the parameters and a function to be minimized respectively (same as that of `optim` and `nlminb`).

`dists_base_ls` List of lists of distance matrices. If NULL, `dists(x)` is used. Each element must be a matrix or an array of distance matrices.

`dists_lagr_ls` List of distance matrices/arrays. Used when `dists_base` is FALSE. If NULL, `dists(x)` is used.

`rs` Logical; if TRUE `x` is treated as a regime-switching, FALSE if the parameters need to be estimated in a non-regime-switching setting.

`...` Additional arguments passed to `optim_fn`.

Details

This functions is the regime-switching variant of `fit_lagr.mcgf()`. Arguments are in lists. The length of arguments that end in `_ls` must be 1 or the same as the number of regimes in `x`. If the length of an argument is 1, then it is set the same for all regimes. Refer to `fit_lagr.mcgf()` for more details of the arguments.

Note that both `wls` and `mle` are heuristic approaches when `x` contains observations from a subset of the discrete spatial domain, though estimation results are close to that using the full spatial domain for large sample sizes.

Since parameters for the base model and the Lagrangian model are estimated sequentially, more accurate estimation may be obtained if the full model is fitted all at once.

Value

A list containing outputs from optimization functions of `optim_fn`.

See Also

Other functions on fitting an mcgf_rs: `add_base.mcgf_rs()`, `add_lagr.mcgf_rs()`, `fit_base.mcgf_rs()`, `krige.mcgf_rs()`, `krige_new.mcgf_rs()`

Examples

```
data(sim3)
sim3_mcgf <- mcgf_rs(sim3$data, dists = sim3$dists, label = sim3$label)
sim3_mcgf <- add_acfs(sim3_mcgf, lag_max = 5)
```

```
sim3_mcgf <- add_ccfs(sim3_mcgf, lag_max = 5)

# Fit a fully symmetric model with known variables
fit_fs <- fit_base(
    sim3_mcgf,
    lag_ls = 5,
    model_ls = "fs",
    rs = FALSE,
    par_init_ls = list(list(beta = 0)),
    par_fixed_ls = list(list(
        nugget = 0,
        c = 0.05,
        gamma = 0.5,
        a = 0.5,
        alpha = 0.2
    ))
)

# Set beta to 0 to fit a separable model with known variables
fit_fs[[1]]$fit$par <- 0

# Store the fitted separable model to 'sim3_mcgf'
sim3_mcgf <- add_base(sim3_mcgf, fit_base_ls = fit_fs)

# Fit a regime-switching Lagrangian model.
fit_lagr_rs <- fit_lagr(
    sim3_mcgf,
    model_ls = list("lagr_tri"),
    par_init_ls = list(
        list(v1 = -50, v2 = 50),
        list(v1 = 100, v2 = 100)
    ),
    par_fixed_ls = list(list(lambda = 0.2, k = 2))
)
lapply(fit_lagr_rs[1:2], function(x) x$fit)
```

---

is.mcgf *Check if an object is an* mcgf *object.*

Description

Check if an object is an mcgf object.

Usage

```
is.mcgf(x)
```

Arguments

x	An Object.

Value

Logical; TRUE if x is of the mcgf class

Examples

```
data(sim1)
is.mcgf(sim1)

sim1_mcgf <- mcgf(sim1$data, dists = sim1$dists)
is.mcgf(sim1_mcgf)
```

---

is.mcgf_rs — *Check if an object is an* mcgf_rs *object..*

Description

Check if an object is an mcgf_rs object..

Usage

```
is.mcgf_rs(x)

as.mcgf_rs(x, label, ncores = 1)
```

Arguments

x	An Object.
label	A vector of regime labels. Its length must be the same as the number rows in data.
ncores	Number of cpu cores used for computing in [ccfs()].

Value

is.mcgf_rs returns a logical valud; TRUE if x is of the mcgf_rs class. as.mcgf_rs coerces an mcgf object to an mcgf_rs object by adding regime labels. Fitted base or Lagrangian models in x are kept.

Examples

```
data(sim2)
is.mcgf_rs(sim2)

sim2_mcgf <- mcgf(sim2$data, dists = sim2$dists)
is.mcgf_rs(sim2_mcgf)

sim2_mcgf <- mcgf_rs(sim2$data, dists = sim2$dists, label = sim2$label)
is.mcgf_rs(sim2_mcgf)
data(sim2)
```

```
sim2_mcgf <- mcgf(sim2$data, dists = sim2$dists)
sim2_mcgf <- as.mcgf_rs(sim2_mcgf, label = sim2$label)
```

---

krige *Generic function for computing kriging forecasts*

Description

Generic function for computing kriging forecasts

Usage

```
krige(x, ...)
```

Arguments

x	An **R** object.
...	Additional parameters or attributes.

Details

Refer to `krige.mcgf()` and `krige.mcgf_rs()` for more details.

Value

Kriging results of `x`.

---

krige.mcgf *Obtain kriging forecasts for an* `mcgf` *object.*

Description

Obtain kriging forecasts for an `mcgf` object.

Usage

```
## S3 method for class 'mcgf'
krige(
  x,
  newdata = NULL,
  model = c("all", "base", "empirical"),
  interval = FALSE,
  level = 0.95,
  ...
)
```

Arguments

x	An mcgf object.
newdata	A data.frame with the same column names as x. If newdata is missing the forecasts at the original data points are returned.
model	Which model to use. One of all, base, or empirical.
interval	Logical; if TRUE, prediction intervals are computed.
level	A numeric scalar between 0 and 1 giving the confidence level for the intervals (if any) to be calculated. Used when interval = TRUE
...	Additional arguments. Give lag and horizon if they are not defined in x for the empirical model.

Details

It produces simple kriging forecasts for a zero-mean mcgf. It supports kriging for the empirical model, the base model, and the all model which is the general stationary model with the base and Lagrangian model from x.

When interval = TRUE, confidence interval for each forecasts and each horizon is given. Note that it does not compute confidence regions.

Value

A list of kriging forecasts (and prediction intervals).

See Also

Other functions on fitting an mcgf: add_base.mcgf(), add_lagr.mcgf(), fit_base.mcgf(), fit_lagr.mcgf(), krige_new.mcgf()

Examples

```
data(sim1)
sim1_mcgf <- mcgf(sim1$data, dists = sim1$dists)
sim1_mcgf <- add_acfs(sim1_mcgf, lag_max = 5)
sim1_mcgf <- add_ccfs(sim1_mcgf, lag_max = 5)

# Fit a separable model and store it to 'sim1_mcgf'
fit_sep <- fit_base(
    sim1_mcgf,
    model = "sep",
    lag = 5,
    par_init = c(
        c = 0.001,
        gamma = 0.5,
        a = 0.3,
        alpha = 0.5
    ),
    par_fixed = c(nugget = 0)
)
sim1_mcgf <- add_base(sim1_mcgf, fit_base = fit_sep)
```

```
# Fit a Lagrangian model
fit_lagr <- fit_lagr(
    sim1_mcgf,
    model = "lagr_tri",
    par_init = c(v1 = 300, v2 = 300, lambda = 0.15),
    par_fixed = c(k = 2)
)

# Store the fitted Lagrangian model to 'sim1_mcgf'
sim1_mcgf <- add_lagr(sim1_mcgf, fit_lagr = fit_lagr)

# Calculate the simple kriging predictions and intervals
sim1_krige <- krige(sim1_mcgf, interval = TRUE)

# Calculate RMSE for each location
rmse <- sqrt(colMeans((sim1_mcgf - sim1_krige$fit)^2, na.rm = TRUE))
rmse

# Calculate MAE for each location
mae <- colMeans(abs(sim1_mcgf - sim1_krige$fit), na.rm = TRUE)
mae

# Calculate POPI for each location
popi <- colMeans(
    sim1_mcgf < sim1_krige$lower | sim1_mcgf > sim1_krige$upper,
    na.rm = TRUE
)
popi
```

`krige.mcgf_rs` *Obtain kriging forecasts for an* `mcgf_rs` *object.*

Description

Obtain kriging forecasts for an `mcgf_rs` object.

Usage

```
## S3 method for class 'mcgf_rs'
krige(
  x,
  newdata = NULL,
  newlabel = NULL,
  soft = FALSE,
  prob,
  model = c("all", "base", "empirical"),
  interval = FALSE,
  level = 0.95,
  ...
)
```

Arguments

x	An mcgf_rs object.
newdata	A data.frame with the same column names as x. If newdata is missing the forecasts at the original data points are returned.
newlabel	A vector of new regime labels.
soft	Logical; if true, soft forecasts (and bounds) are produced.
prob	Matrix with simplex rows. Number of columns must be the same as unique labels in x.
model	Which model to use. One of all, base, or empirical.
interval	Logical; if TRUE, prediction intervals are computed.
level	A numeric scalar between 0 and 1 giving the confidence level for the intervals (if any) to be calculated. Used when interval = TRUE
...	Additional arguments. Give lag and horizon if they are not defined in x for the empirical model.

Details

It produces simple kriging forecasts for a zero-mean mcgf. It supports kriging for the empirical model, the base model, and the all model which is the general stationary model with the base and Lagrangian model from x.

When soft = TRUE, prob will be used to compute the soft forecasts (weighted forecasts). The number of columns must match the number of unique levels in x. The column order must be the same as the order of regimes as in levels(attr(x, "label", exact = TRUE)). If not all regimes are seen in newlabel, then only relevant columns in prob are used.

When interval = TRUE, confidence interval for each forecasts and each horizon is given. Note that it does not compute confidence regions.

Value

A list of kriging forecasts (and prediction intervals).

See Also

Other functions on fitting an mcgf_rs: add_base.mcgf_rs(), add_lagr.mcgf_rs(), fit_base.mcgf_rs(), fit_lagr.mcgf_rs(), krige_new.mcgf_rs()

Examples

```
data(sim2)
sim2_mcgf <- mcgf_rs(sim2$data, dists = sim2$dists, label = sim2$label)
sim2_mcgf <- add_acfs(sim2_mcgf, lag_max = 5)
sim2_mcgf <- add_ccfs(sim2_mcgf, lag_max = 5)

# Fit a regime-switching separable model
fit_sep <- fit_base(
    sim2_mcgf,
```

```
    lag_ls = 5,
    model_ls = "sep",
    par_init_ls = list(list(
        c = 0.00005,
        gamma = 0.5,
        a = 0.5,
        alpha = 0.5
    )),
    par_fixed_ls = list(c(nugget = 0))
)

# Store the fitted separable models to 'sim2_mcgf'
sim2_mcgf <- add_base(sim2_mcgf, fit_base_ls = fit_sep)

# Calculate the simple kriging predictions and intervals
sim2_krige <- krige(sim2_mcgf, model = "base", interval = TRUE)

# Calculate RMSE for each location
rmse <- sqrt(colMeans((sim2_mcgf - sim2_krige$fit)^2, na.rm = TRUE))
rmse

# Calculate MAE for each location
mae <- colMeans(abs(sim2_mcgf - sim2_krige$fit), na.rm = TRUE)
mae

# Calculate POPI for each location
popi <- colMeans(
    sim2_mcgf < sim2_krige$lower | sim2_mcgf > sim2_krige$upper,
    na.rm = TRUE
)
popi
```

`krige_new` *Generic function for computing kriging forecasts for new locations*

Description

Generic function for computing kriging forecasts for new locations

Usage

```
krige_new(x, ...)
```

Arguments

x	An **R** object.
...	Additional parameters or attributes.

Details

Refer to krige_new.mcgf() and krige_new.mcgf_rs() for more details.

Value

Kriging results of x.

---

krige_new.mcgf *Obtain kriging forecasts for new locations for an mcgf object.*

Description

Obtain kriging forecasts for new locations for an mcgf object.

Usage

```
## S3 method for class 'mcgf'
krige_new(
  x,
  newdata = NULL,
  locations_new = NULL,
  dists_new = NULL,
  newdata_new = NULL,
  sds_new = 1,
  model = c("all", "base"),
  interval = FALSE,
  level = 0.95,
  dists_new_base,
  ...
)
```

Arguments

x	An mcgf object.
newdata	A data.frame with the same column names as x. If newdata is missing the forecasts at the original data points are returned.
locations_new	A matrix of data.frame of 2D points of new locations, first column longitude, second column latitude, both in decimal degrees. Supply only if x contains locations. Required when dists_new is not supplied.
dists_new	List of signed distance matrices (vectors) with names h, h1, and 'h2' for all locations, with new locations in the end. Each matrix must have the same number of columns. Required when locations_new is not supplied.
newdata_new	Optional; a data.frame with the same number of rows as newdata. It contains the data of the new locations.
sds_new	The standard deviations of the new locations. Default is 1.

model	Which model to use. One of all or base.
interval	Logical; if TRUE, prediction intervals are computed.
level	A numeric scalar between 0 and 1 giving the confidence level for the intervals (if any) to be calculated. Used when interval = TRUE.
dists_new_base	Optional; a distance array for all locations for the base model, with new locations in the end. Used for the base model.
...	Additional arguments. Not in use.

Details

It produces simple kriging forecasts for a zero-mean mcgf for new locations given their coordinates or relative distances. It supports kriging for the base model and the all model which is the general stationary model with the base and Lagrangian model from x.

Users can either supply the coordinates via locations_new, or a list of distance for all locations via dists_new, with new locations at the end. dists_new will be used to calculate the new covariance matrices. When locations_new is used, make sure x contains the attribute locations of the coordinates of the old locations. When dists_new is used, it should be a list of signed distance matrices of the same dimension, where each row corresponds to the relative distances between a new location and old locations in the same order as they appear in x.

If data for the new locations are available, use newdata_new to include them and they will be used to calculate the kriging forecasts for the new locations; otherwise only data of the old locations will be used via newdata.

When interval = TRUE, confidence interval for each forecasts and each horizon is given. Note that it does not compute confidence regions.

Value

A list of kriging forecasts (and prediction intervals) for all locations.

See Also

Other functions on fitting an mcgf: add_base.mcgf(), add_lagr.mcgf(), fit_base.mcgf(), fit_lagr.mcgf(), krige.mcgf()

Examples

```
data(sim1)
sim1_mcgf <- mcgf(sim1$data, locations = sim1$locations)
sim1_mcgf <- add_acfs(sim1_mcgf, lag_max = 5)
sim1_mcgf <- add_ccfs(sim1_mcgf, lag_max = 5)

# Fit a separable model and store it to 'sim1_mcgf'
fit_sep <- fit_base(
    sim1_mcgf,
    model = "sep",
    lag = 5,
    par_init = c(
        c = 0.001,
```

```
        gamma = 0.5,
        a = 0.3,
        alpha = 0.5
    ),
    par_fixed = c(nugget = 0)
)
sim1_mcgf <- add_base(sim1_mcgf, fit_base = fit_sep)

# Fit a Lagrangian model
fit_lagr <- fit_lagr(
    sim1_mcgf,
    model = "lagr_tri",
    par_init = c(v1 = 300, v2 = 300, lambda = 0.15),
    par_fixed = c(k = 2)
)

# Store the fitted Lagrangian model to 'sim1_mcgf'
sim1_mcgf <- add_lagr(sim1_mcgf, fit_lagr = fit_lagr)

# Calculate the simple kriging predictions and intervals for all locations
locations_new <- rbind(c(-110, 55), c(-109, 54))
sim1_krige <- krige_new(sim1_mcgf,
    locations_new = locations_new,
    interval = TRUE
)
```

`krige_new.mcgf_rs` *Obtain kriging forecasts for new locations for an* `mcgf_rs` *object.*

Description

Obtain kriging forecasts for new locations for an `mcgf_rs` object.

Usage

```
## S3 method for class 'mcgf_rs'
krige_new(
  x,
  newdata = NULL,
  locations_new = NULL,
  dists_new_ls = NULL,
  newdata_new = NULL,
  sds_new_ls = 1,
  newlabel,
  soft = FALSE,
  prob,
  dists_new_base,
  model = c("all", "base"),
  interval = FALSE,
```

```
  level = 0.95,
  ...
)
```

Arguments

x	An mcgf_rs object.
newdata	A data.frame with the same column names as x. If newdata is missing the forecasts at the original data points are returned.
locations_new	A matrix of data.frame of 2D points of new locations, first column longitude, second column latitude, both in decimal degrees. Supply only if x contains locations. Required when dists_new_ls is not supplied.
dists_new_ls	List of signed distance matrices (vectors) with names h, h1, and 'h2' for all locations(and for each regime), with new locations in the end. Each matrix must have the same number of columns. Required when locations_new is not supplied.
newdata_new	Optional; a data.frame with the same number of rows as newdata. It contains the data of the new locations.
sds_new_ls	List of the standard deviations of the new locations for each regime. Format must be the same as the output from sds.mcgf_rs(). Default is 1 for all regimes.
newlabel	A vector of new regime labels.
soft	Logical; if true, soft forecasts (and bounds) are produced.
prob	Matrix with simplex rows. Number of columns must be the same as unique labels in x.
dists_new_base	Optional, list of distance matrices for the base model. Used when the base model is non-regime switching. Default is h from the first list of dists_new_ls.
model	Which model to use. One of all, base, or empirical.
interval	Logical; if TRUE, prediction intervals are computed.
level	A numeric scalar between 0 and 1 giving the confidence level for the intervals (if any) to be calculated. Used when interval = TRUE
...	Additional arguments.

Details

It produces simple kriging forecasts for a zero-mean mcgf for new locations given theri coordinates or relative distances. It supports kriging for the base model and the all model which is the general stationary model with the base and Lagrangian model from x.

Users can either supply the coordinates via locations_new, or a list of distance for all locations via dists_new_ls, with new locations at the end. dists_new_ls will be used to calculate the new covariance matrices. When locations_new is used, make sure x contains the attribute locations of the coordinates of the old locations. When dists_new_ls is used, it should be a list of a list of signed distance matrices of the same dimension, where each row corresponds to the relative distances between a new location and old locations in the same order as they appear in x. If only one list is provided, it will be used for all regimes.

When soft = TRUE, prob will be used to compute the soft forecasts (weighted forecasts). The number of columns must match the number of unique levels in x. The column order must be the same as the order of regimes as in levels(attr(x, "label", exact = TRUE)). If not all regimes are seen in newlabel, then only relevant columns in prob are used.

When interval = TRUE, confidence interval for each forecasts and each horizon is given. Note that it does not compute confidence regions.

Value

A list of kriging forecasts (and prediction intervals) for all locations.

See Also

Other functions on fitting an mcgf_rs: add_base.mcgf_rs(), add_lagr.mcgf_rs(), fit_base.mcgf_rs(), fit_lagr.mcgf_rs(), krige.mcgf_rs()

Examples

```
data(sim2)
sim2_mcgf <- mcgf_rs(sim2$data,
    locations = sim2$locations,
    label = sim2$label
)
sim2_mcgf <- add_acfs(sim2_mcgf, lag_max = 5)
sim2_mcgf <- add_ccfs(sim2_mcgf, lag_max = 5)

# Fit a regime-switching separable model
fit_sep <- fit_base(
    sim2_mcgf,
    lag_ls = 5,
    model_ls = "sep",
    par_init_ls = list(list(
        c = 0.00005,
        gamma = 0.5,
        a = 0.5,
        alpha = 0.5
    )),
    par_fixed_ls = list(c(nugget = 0))
)

# Store the fitted separable models to 'sim2_mcgf'
sim2_mcgf <- add_base(sim2_mcgf, fit_base_ls = fit_sep)

# Calculate the simple kriging predictions and intervals for all locations
locations_new <- rbind(c(-110, 55), c(-109, 54))
sim2_krige <- krige_new(sim2_mcgf,
    locations_new = locations_new,
    model = "base", interval = TRUE
)
```

`mcgf` *Create mcgf object*

Description

Create mcgf object

Usage

```
mcgf(data, locations, dists, time, longlat = TRUE, origin = 1L)
```

Arguments

`data`	Time series data set in space-wide format.
`locations`	A matrix of data.frame of 2D points, first column x/longitude, second column y/latitude. Required when `dists` is not supplied. If longitudes and latitudes are provided, they are mapped to a 2D Euclidean. See `find_dists()` for more details.
`dists`	List of signed distance matrices on a 2D Euclidean Plane. Required when `locations` is not supplied.
`time`	Optional, a vector of equally spaced time stamps.
`longlat`	Logical, if TURE `locations` contains longitudes and latitudes.
`origin`	Optional; used when `longlat` is TRUE. An integer index indicating the reference location which well be used as the origin.

Details

An `mcgf` object extends the S3 class `data.frame`.

For inputs, `data` must be in space-wide format where rows correspond to different time stamps and columns refer to spatial locations. Supply either `locations` or `dists`. `locations` is a matrix or data.frame of 2D points with first column x/longitude and second column y/latitude. By default it is treated as a matrix of Earth's coordinates in decimal degrees. Number of rows in `locations` must be the same as the number of columns of `data`. `dists` must be a list of signed distance matrices with names `h1`, `h2`, and `h`. If `h` is not given, it will be calculated as the Euclidean distance of `h1` and `h2`. `time` is a vector of equally spaced time stamps. If it is not supplied then `data` is assumed to be temporally equally spaced.

An `mcgf` object extends the S3 class `data.frame`, all methods remain valid to the `data` part of the object.

Value

An S3 object of class `mcgf`. As it inherits and extends the `data.frame` class, all methods remain valid to the `data` part of the object. Additional attributes may be assigned and extracted.

Examples

```
data <- cbind(S1 = 1:5, S2 = 4:8, S3 = 5:9)
lon <- c(110, 120, 130)
lat <- c(50, 55, 60)
locations <- cbind(lon, lat)
obj <- mcgf(data, locations = locations)
print(obj, "locations")
```

`mcgf_rs` *Create mcgf_rs object*

Description

Create mcgf_rs object

Usage

```
mcgf_rs(data, locations, dists, label, time, longlat = TRUE, origin = 1L)
```

Arguments

`data`	Time series data set in space-wide format.
`locations`	A matrix of data.frame of 2D points, first column longitude, second column latitude, both in decimal degrees. Required when `dists` is not supplied.
`dists`	List of signed distance matrices. Required when `locations` is not supplied.
`label`	A vector of regime labels. Its length must be the same as the number rows in `data`.
`time`	Optional, a vector of equally spaced time stamps.
`longlat`	Logical, if TURE `locations` contains longitudes and latitudes.
`origin`	Optional; used when `longlat` is TRUE. An integer index indicating the reference location which well be used as the origin.

Details

An `mcgf_rs` object extends the S3 classes `mcgf` and `data.frame`.

For inputs, `data` must be in space-wide format where rows correspond to different time stamps and columns refer to spatial locations. Supply either `locations` or `dists`. `locations` is a matrix or data.frame of 2D points with first column x/longitude and second column y/latitude. By default it is treated as a matrix of Earth's coordinates in decimal degrees. Number of rows in `locations` must be the same as the number of columns of `data`. `dists` must be a list of signed distance matrices with names `h1`, `h2`, and `h`. If `h` is not given, it will be calculated as the Euclidean distance of `h1` and `h2`. `time` is a vector of equally spaced time stamps. If it is not supplied then `data` is assumed to be temporally equally spaced. `label` must be a vector containing regime labels, and its length must be the same as the number of rows in `x`.

An `mcgf_rs` object extends the S3 classes `mcgf` and `data.frame`, all methods remain valid to the `data` part of the object.

Value

An S3 object of class `mcgf_rs`. As it inherits and extends the `mcgf` and then the`data.frame` class, all methods remain valid to the `data` part of the object. Additional attributes may be assigned and extracted.

Examples

```
data <- cbind(S1 = 1:5, S2 = 4:8, S3 = 5:9)
lon <- c(110, 120, 130)
lat <- c(50, 55, 60)
locations <- cbind(lon, lat)
label <- c(1, 1, 2, 2, 2)
obj <- mcgf_rs(data, locations = locations, label = label)
print(obj, "locations")
print(obj, "label")
```

`mcgf_rs_sim` *Simulate regime-switching Markov chain Gaussian field*

Description

Simulate regime-switching Markov chain Gaussian field

Usage

```
mcgf_rs_sim(
  N,
  label,
  base_ls,
  lagrangian_ls,
  par_base_ls,
  par_lagr_ls,
  lambda_ls,
  dists_ls,
  sd_ls,
  lag_ls,
  scale_time = 1,
  init = 0,
  mu_c_ls = list(0),
  mu_p_ls = list(0),
  return_all = FALSE
)
```

Arguments

`N`	Sample size.
`label`	Vector of regime labels of the same length as `N`.

base_ls	List of base model, sep or fs for now.
lagrangian_ls	List of Lagrangian model, "none" or lagr_tri for now.
par_base_ls	List of parameters for the base model.
par_lagr_ls	List of parameters for the Lagrangian model.
lambda_ls	List of weight of the Lagrangian term, $\lambda \in [0, 1]$.
dists_ls	List of distance matrices or arrays.
sd_ls	List of standard deviation for each location.
lag_ls	List of time lags.
scale_time	Scale of time unit, default is 1. Elements in lag_ls are divided by scale_time.
init	Initial samples, default is 0.
mu_c_ls, mu_p_ls	List of means of current and past.
return_all	Logical; if TRUE the joint covariance matrix, arrays of distances and time lag are returned.

Value

Simulated regime-switching Markov chain Gaussian field with user-specified covariance structures. The simulation is done by kriging. The output data is in space-wide format. Each element in dists_ls must contain h for symmetric models, and h1 and h2 for general stationary models. init can be a scalar or a vector of appropriate size. List elements in sd_ls, mu_c_ls, and mu_p_ls must be vectors of appropriate sizes.

See Also

Other simulations of Markov chain Gaussian fields: mcgf_sim()

Examples

```
par_s <- list(nugget = 0.5, c = 0.01, gamma = 0.5)
par_t <- list(a = 1, alpha = 0.5)
par_base <- list(par_s = par_s, par_t = par_t)
par_lagr <- list(v1 = 5, v2 = 10)
h1 <- matrix(c(0, 5, -5, 0), nrow = 2)
h2 <- matrix(c(0, 8, -8, 0), nrow = 2)
h <- sqrt(h1^2 + h2^2)
dists <- list(h = h, h1 = h1, h2 = h2)

set.seed(123)
label <- sample(1:2, 1000, replace = TRUE)
X <- mcgf_rs_sim(
    N = 1000,
    label = label,
    base_ls = list("sep"),
    lagrangian_ls = list("none", "lagr_tri"),
    lambda_ls = list(0, 0.5),
    par_base_ls = list(par_base),
```

```
    par_lagr_ls = list(NULL, par_lagr),
    dists_ls = list(dists, dists)
)
# plot.ts(X[, -1])
```

`mcgf_sim` *Simulate Markov chain Gaussian field*

Description

Simulate Markov chain Gaussian field

Usage

```
mcgf_sim(
  N,
  base = c("sep", "fs"),
  lagrangian = c("none", "lagr_tri", "lagr_askey"),
  par_base,
  par_lagr,
  lambda,
  dists,
  sd = 1,
  lag = 1,
  scale_time = 1,
  horizon = 1,
  init = 0,
  mu_c = 0,
  mu_p = 0,
  return_all = FALSE
)
```

Arguments

`N`	Sample size.
`base`	Base model, `sep` or `fs` for now.
`lagrangian`	Lagrangian model, "none" or `lagr_tri` for now.
`par_base`	Parameters for the base model (symmetric).
`par_lagr`	Parameters for the Lagrangian model.
`lambda`	Weight of the Lagrangian term, $\lambda \in [0, 1]$.
`dists`	Distance matrices or arrays.
`sd`	Standard deviation for each location.
`lag`	Time lag.
`scale_time`	Scale of time unit, default is 1. `lag` is divided by `scale_time`.

horizon	Forecast horizon, default is 1.
init	Initial samples, default is 0.
mu_c, mu_p	Means of current and past.
return_all	Logical; if TRUE the joint covariance matrix, arrays of distances and time lag are returned.

Value

Simulated Markov chain Gaussian field with user-specified covariance structure. The simulation is done by kriging. The output data is in space-wide format. dists must contain h for symmetric models, and h1 and h2 for general stationary models. horizon controls forecasting horizon. sd, mu_c, mu_p, and init must be vectors of appropriate sizes.

See Also

Other simulations of Markov chain Gaussian fields: mcgf_rs_sim()

Examples

```
par_s <- list(nugget = 0.5, c = 0.01, gamma = 0.5)
par_t <- list(a = 1, alpha = 0.5)
par_base <- list(par_s = par_s, par_t = par_t)
par_lagr <- list(v1 = 5, v2 = 10)
h1 <- matrix(c(0, 5, -5, 0), nrow = 2)
h2 <- matrix(c(0, 8, -8, 0), nrow = 2)
h <- sqrt(h1^2 + h2^2)
dists <- list(h = h, h1 = h1, h2 = h2)

set.seed(123)
X <- mcgf_sim(
    N = 1000, base = "sep", lagrangian = "lagr_tri", lambda = 0.5,
    par_base = par_base, par_lagr = par_lagr, dists = dists
)
plot.ts(X)
```

---

model *Generic function for displaying fitted models for* mcgf *objects*

Description

Generic function for displaying fitted models for mcgf objects

Usage

```
model(x, ...)
```

Arguments

`x`	An **R** object.
`...`	Additional parameters or attributes.

Details

Refer to `model.mcgf()` and `model.mcgf_rs()` for more details.

Value

Details of the fitted models.

`model.mcgf` *Display fitted models for an* `mcgf` *or* `mcgf_rs` *object*

Description

Display fitted models for an `mcgf` or `mcgf_rs` object

Usage

```
## S3 method for class 'mcgf'
model(
  x,
  model = c("all", "base", "lagrangian"),
  old = FALSE,
  print_model = TRUE,
  ...
)

## S3 method for class 'mcgf_rs'
model(
  x,
  model = c("all", "base", "lagrangian"),
  old = FALSE,
  print_model = TRUE,
  ...
)
```

Arguments

`x`	An mcgf object.
`model`	Which model to display.
`old`	Logical; TRUE if the old model needs to be printed.
`print_model`	Logical; TRUE if time lag and forecast horizon need to be printed.
`...`	Additional arguments. Not in use.

Details

For mcgf and mcgf_rs objects, model() displays the fitted models and their parameters. When old = TRUE, the old model is printed as well. Note that the old model is not used for parameter estimation or for kriging.

Value

None (invisible NULL).

---

print.mcgf *Print an mcgf object.*

Description

Print an mcgf object.

Usage

```
## S3 method for class 'mcgf'
print(x, attr = ".Data", ...)
```

Arguments

x	An mcgf object.
attr	Attribute to be printed.
...	Optional arguments to print methods.

Value

No return value, called for side effects.

Examples

```
data(sim1)
sim1_mcgf <- mcgf(sim1$data, dists = sim1$dists)
print(sim1_mcgf, "dists")
```

rdists *Generate random distance matrices*

Description

Generate random distance matrices

Usage

```
rdists(N, names, scale = 100)
```

Arguments

N	Number of locations.
names	Names of locations.
scale	Scale of the distance matrices. Default is 100.

Details

This function generates random distance matrices using `rnorm`. `scale` controls the scale of the distance matrices.

Value

List of signed distances.

Examples

```
set.seed(123)
rdists(3)
rdists(3, scale = 1)
rdists(3, names = LETTERS[1:3])
```

sds *Generic function for standard deviations for each column*

Description

Generic function for standard deviations for each column

Usage

```
sds(x, ...)
```

Arguments

`x`	An **R** object.
`...`	Additional parameters or attributes.

Details

Refer to `sds.mcgf()` and `sds.mcgf_rs()` for more details.

Value

A vector of standard deviations for `mcgf` objects, or that plus a list of regime-switching standard deviations for `mcgf_rs` objects.

`sds.mcgf` *Extract, calculate, or assign standard deviations for an* `mcgf` *or* `mcgf_rs` *object.*

Description

Extract, calculate, or assign standard deviations for an `mcgf` or `mcgf_rs` object.

Usage

```
## S3 method for class 'mcgf'
sds(x, ...)

## S3 method for class 'mcgf_rs'
sds(x, replace = FALSE, ...)

sds(x) <- value
```

Arguments

`x`	An `mcgf` or `mcgf_rs` object.
`...`	Additional parameters or attributes. Not in use.
`replace`	Logical; if TRUE, `sds` are recalculated.
`value`	A vector (or list of vectors) of standard deviations for all stations (under each regime and combined).

Details

For `mcgf` objects, `sds()` extracts or computes the empirical standard deviations. The output is a vector of sds.

For `mcgf_rs` objects, `sds()` extracts or computes the regime-switching empirical standard deviations. The output is a list of vectors of sds. Each element in the list corresponds to the sds for a regime.

`sds<-` assigns `sds` to `x`. Use `add_ccfs()` to add both `ccfs` and `sds` to `x`.

Value

sds() returns empirical (regime-switching) standard deviations.

Examples

```
data(sim1)
sim1_mcgf <- mcgf(sim1$data, dists = sim1$dists)
sds(sim1_mcgf)

data(sim2)
sim2_mcgf <- mcgf_rs(sim2$data, dists = sim2$dists, label = sim2$label)
sds(sim2_mcgf)
data(sim1)
sim1_mcgf <- mcgf(sim1$data, dists = sim1$dists)
sim1_sds <- sds(sim1_mcgf)
sds(sim1_mcgf) <- sim1_sds

data(sim2)
sim2_mcgf <- mcgf_rs(sim2$data, dists = sim2$dists, label = sim2$label)
sim2_sds <- sds(sim2_mcgf)
sds(sim2_mcgf) <- sim2_sds
```

---

sd_rs *Calculate standard deviation for each location under each regime.*

Description

Calculate standard deviation for each location under each regime.

Usage

```
sd_rs(x, label)
```

Arguments

x	A data.frame or matrix.
label	A vector of regime labels. Its length must be the same as the number rows in x.

Value

A list of standard deviations for each regime.

Examples

```
set.seed(123)
x <- matrix(rnorm(200), nrow = 100)
label <- sample(1:2, 100, replace = TRUE)
sd_rs(x, label = factor(label))
```

sim1 — *Simulated Markov chain Gaussian field*

Description

Simulated MCGF for 10 locations.

Usage

```
sim1
```

Format

`sim1`: a list containing a data.frame with 1000 rows and 10 columns and a list of distances

Details

`sim1` contains a simulated MCGF for 10 locations. It is simulated with a separable base model and a triangular Lagrangian model. The true parameters for the base model are: nugget $= 0, c = 0.001, \gamma = 0.5, a = 0.5, \alpha = 0.8$, and those for the Lagrangian model are: $v1 = 200, v2 = 200, k = 2, \lambda = 0.2$

See Also

Other (simulated) datasets: `sim2`, `sim3`, `wind`

Examples

```
# Code used to generate `sim1`

library(mcgf)
set.seed(123)
x <- stats::rnorm(10, -110)
y <- stats::rnorm(10, 50)
locations <- cbind(x, y)
h <- find_dists(locations, longlat = TRUE)

N <- 1000
lag <- 5

par_base <- list(
    par_s = list(nugget = 0, c = 0.001, gamma = 0.5),
    par_t = list(a = 0.5, alpha = 0.8)
)
par_lagr <- list(v1 = 200, v2 = 200, k = 2)

sim1 <- mcgf_sim(
    N = N, base = "sep", lagrangian = "lagr_tri",
    par_base = par_base, par_lagr = par_lagr, lambda = 0.2,
```

```
    dists = h, lag = lag
)
sim1 <- sim1[-c(1:(lag + 1)), ]
rownames(sim1) <- 1:nrow(sim1)

sim1 <- list(data = sim1, locations = locations, dists = h)
```

---

sim2 *Simulated regime-switching Markov chain Gaussian field*

Description

Simulated RS-MCGF for 10 locations.

Usage

```
sim2
```

Format

`sim2`: a list containing a data.frame with 1000 rows and 10 columns, a list of distances, and a vector of regime labels.

Details

`sim2` contains a simulated RS-MCGF for 10 locations. It is simulated with a regime-switching separable base model. The true parameters for the base model are:

$$\text{Regime 1} : \text{nugget} = 0, c = 0.01, \gamma = 0.5, a = 0.5, \alpha = 0.2,$$

$$\text{Regime 2} : \text{nugget} = 0, c = 0.04, \gamma = 0.5, a = 0.3, \alpha = 0.9.$$

See Also

Other (simulated) datasets: sim1, sim3, wind

Examples

```
# Code used to generate `sim2`

library(mcgf)
set.seed(123)
x <- stats::rnorm(10, -110)
y <- stats::rnorm(10, 50)
locations <- cbind(x, y)
h <- find_dists(locations, longlat = TRUE)

# simulate regimes
K <- 2
```

```
N <- 1000
lag <- 5

tran_mat <- matrix(rnorm(K^2, mean = 0.06 / (K - 1), sd = 0.01), nrow = K)
diag(tran_mat) <- rnorm(K, mean = 0.94, sd = 0.1)
tran_mat <- sweep(abs(tran_mat), 1, rowSums(tran_mat), `/`)
tran_mat
#            [,1]       [,2]
# [1,] 0.94635675 0.05364325
# [2,] 0.06973429 0.93026571

regime <- rep(NA, N)
regime[1] <- 1

for (n in 2:(N)) {
    regime[n] <- sample(1:K, 1, prob = tran_mat[regime[n - 1], ])
}
table(regime)
# regime
#   1   2
# 567 433

# simulate RS MCGF
par_base1 <- list(
    par_s = list(nugget = 0, c = 0.001, gamma = 0.5),
    par_t = list(a = 0.5, alpha = 0.2)
)

par_base2 <- list(
    par_s = list(nugget = 0, c = 0.004, gamma = 0.5),
    par_t = list(a = 0.3, alpha = 0.9)
)

sim2 <- mcgf_rs_sim(
    N = N, label = regime,
    base_ls = list("sep"), lagrangian_ls = list("none"),
    par_base_ls = list(par_base1, par_base2),
    lambda_ls = list(0.1, 0.3),
    lag_ls = list(lag, lag),
    dists_ls = list(h, h)
)
sim2 <- sim2[-c(1:(lag + 1)), ]
rownames(sim2) <- 1:nrow(sim2)

sim2 <- list(
    data = sim2[, -1], locations = locations, dists = h,
    label = sim2[, 1]
)
```

---

sim3 *Simulated regime-switching Markov chain Gaussian field*

Description

Simulated RS-MCGF for 20 locations.

Usage

```
sim3
```

Format

`sim3`: a list containing a data.frame with 5000 rows and 20 columns and a list of locations.

Details

`sim3` contains a simulated RS-MCGF for 20 locations. It is simulated with the same base model and a regime-switching Lagrangian model. The true parameters for the base model are: nugget $=0, c = 0.05, \gamma = 0.5, a = 0.5, \alpha = 0.2$, and the true parameters for the Lagrangian model are

$$\text{Regime 1} : \lambda = 0.2, v_1 = -100, v_2 = 100, k = 2,$$

$$\text{Regime 1} : \lambda = 0.2, v_1 = 200, v_2 = 200, k = 2.$$

For parameter estimation, the base model is assumed known and is used to estimate the regime-switching prevailing winds.

See Also

Other (simulated) datasets: `sim1`, `sim2`, `wind`

Examples

```
# Code used to generate `sim3`

library(mcgf)
set.seed(123)
x <- stats::rnorm(10, -110)
y <- stats::rnorm(10, 50)
locations <- cbind(x, y)
h <- find_dists(locations, longlat = TRUE)

# simulate regimes
K <- 2
N <- 1000
lag <- 5

tran_mat <- matrix(rnorm(K^2, mean = 0.06 / (K - 1), sd = 0.01), nrow = K)
diag(tran_mat) <- rnorm(K, mean = 0.94, sd = 0.1)
tran_mat <- sweep(abs(tran_mat), 1, rowSums(tran_mat), `/`)
tran_mat
# [,1]       [,2]
# [1,] 0.94635675 0.05364325
# [2,] 0.06973429 0.93026571
```

```
regime <- rep(NA, N)
regime[1] <- 1

for (n in 2:(N)) {
    regime[n] <- sample(1:K, 1, prob = tran_mat[regime[n - 1], ])
}
table(regime)
# regime
#   1   2
# 567 433

# simulate RS MCGF
par_base <- list(
    par_s = list(nugget = 0, c = 0.05, gamma = 0.5),
    par_t = list(a = 0.5, alpha = 0.2)
)

par_lagr1 <- list(v1 = -100, v2 = 100, k = 2)
par_lagr2 <- list(v1 = 200, v2 = 200, k = 2)

sim3 <- mcgf_rs_sim(
    N = N, label = regime,
    base_ls = list("sep"), lagrangian_ls = list("lagr_tri"),
    par_base_ls = list(par_base),
    par_lagr_ls = list(par_lagr1, par_lagr2),
    lambda_ls = list(0.2, 0.2),
    lag_ls = list(lag, lag),
    dists_ls = list(h, h)
)
sim3 <- sim3[-c(1:(lag + 1)), ]
rownames(sim3) <- 1:nrow(sim3)

sim3 <- list(
    data = sim3[, -1], locations = locations, dists = h,
    label = sim3[, 1]
)
```

`wind` *Ireland wind data, 1961-1978*

Description

Daily average wind speeds for 1961-1978 at 11 synoptic meteorological stations in the Republic of Ireland (Haslett and raftery 1989). Wind speeds are in m/s. De-trended data sets are also provided.

Usage

```
wind
```

Format

`wind`: a list containing a data.frame with 6574 rows and 12 columns, and a list of locations.

Details

The data were obtained from the **gstat** package, and were modified so that the first column is the time stamps. Locations of the 11 stations are given in `wind_loc`. `wind_train` and `wind_test` contain de-trended and square-root transformed train (1961-1970) and test (1971-1978) data sets. See Gneiting et al. (2006) for de-trending details. `wind_trend` contains the estimated annual trend and station-wise mean from the training dataset.

References

Haslett, J. and Raftery, A. E. (1989). Space-time Modelling with Long-memory Dependence: Assessing Ireland's Wind Power Resource (with Discussion). Applied Statistics 38, 1-50.

Gneiting, T., Genton, M., & Guttorp, P. (2006). Geostatistical Space-Time Models, Stationarity, Separability, and Full Symmetry. In C&H/CRC Monographs on Statistics & Applied Probability (pp. 151–175). Chapman and Hall/CRC.

See Also

Other (simulated) datasets: `sim1`, `sim2`, `sim3`

Index