Package: CoSMoS (via r-universe)

May 28, 2026

Type Package

Title Complete Stochastic Modelling Solution

Version 2.2.0

Date 2026-05-07

Description Makes univariate, multivariate, or random fields simulations precise and simple. Just select the desired time series or random fields' properties and it will do the rest. CoSMoS is based on the framework described in Papalexiou (2018, <doi:10.1016/j.advwatres.2018.02.013>), extended for random fields in Papalexiou and Serinaldi (2020, <doi:10.1029/2019WR026331>), and further advanced in Papalexiou et al. (2021, <doi:10.1029/2020WR029466>) to allow fine-scale space-time simulation of storms (or even cyclone-mimicking fields).

Depends R (>= 3.5.0), ggplot2, data.table

Imports utils, methods, stats, grDevices, nloptr, MBA, Matrix, mAr, matrixcalc, mvtnorm, patchwork, animation, ggquiver, pracma, plot3D, Rcpp

LinkingTo Rcpp, RcppEigen, RcppNumerical, BH

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

Author Simon Michael Papalexiou [aut], Francesco Serinaldi [aut], Filip Strnad [aut], Yannis Markonis [aut], Kevin Shook [ctb, cre]

Maintainer Kevin Shook <kshook@kshook.ca>

License GPL-3

URL https://github.com/TycheLab/CoSMoS

NeedsCompilation yes

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

Date/Publication 2026-05-07 20:42:33 UTC

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

RemoteRef HEAD

RemoteSha 138e9874bd8523f9816e801870233438f905d929

Contents

`CoSMoS-package` *CoSMoS: Complete Stochastic Modelling Solution*

Description

CoSMoS is an R package that makes time series generation with desired properties easy. Just choose the characteristics of the time series you want to generate, and it will do the rest.

Details

The generated time series preserve any probability distribution and any linear autocorrelation structure. Users can generate as many and as long time series from processes such as precipitation, wind, temperature, relative humidity etc. It is based on a framework that unified, extended, and improved a modelling strategy that generates time series by transforming "parent" Gaussian time series having specific characteristics (Papalexiou, 2018).

Funding

The package was partly funded by the Global Institute for Water Security (GIWS; https://water.usask.ca/) and the Global Water Futures (GWF; https://gwf.usask.ca/) program.

Author(s)

Coded by: Filip Strnad <strnadf@fzp.czu.cz> and Francesco Serinaldi <francesco.serinaldi@ncl.ac.uk>

Conceptual design by: Simon Michael Papalexiou <sm.papalexiou@usask.ca>

Tested and documented by: Yannis Markonis <markonis@fzp.czu.cz>

Maintained by: Kevin Shook <kevin.shook@usask.ca>

References

Papalexiou, S.M. (2018). Unified theory for stochastic modelling of hydroclimatic processes: Preserving marginal distributions, correlation structures, and intermittency. Advances in Water Resources 115, 234-252, doi:10.1016/j.advwatres.2018.02.013

Papalexiou, S.M., Markonis, Y., Lombardo, F., AghaKouchak, A., Foufoula-Georgiou, E. (2018). Precise Temporal Disaggregation Preserving Marginals and Correlations (DiPMaC) for Stationary and Nonstationary Processes. Water Resources Research, 54(10), 7435-7458, doi:10.1029/2018WR022726

Papalexiou, S.M., Serinaldi, F. (2020). Random Fields Simplified: Preserving Marginal Distributions, Correlations, and Intermittency, With Applications From Rainfall to Humidity. Water Resources Research, 56(2), e2019WR026331, doi:10.1029/2019WR026331

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi:10.1029/2020WR029466

See Also

Useful links:

- https://github.com/TycheLab/CoSMoS

`acs` *AutoCorrelation Structure*

Description

Provides a parametric function that describes the values of the linear autocorrelation up to desired lags. For more details on the parametric autocorrelation structures see section 3.2 in Papalexiou (2018).

Usage

```
acs(id, ...)
```

Arguments

`id`	autocorrelation structure id.
`...`	other arguments (`t` as lag and ACS parameters).

Value

A numeric vector of autocorrelation values at the supplied lags.

References

Papalexiou, S.M. (2018). Unified theory for stochastic modelling of hydroclimatic processes: Preserving marginal distributions, correlation structures, and intermittency. Advances in Water Resources, 115, 234-252, doi:10.1016/j.advwatres.2018.02.013

See Also

actpnts, fitACS, fitactf

Examples

```
library(CoSMoS)

## specify lag
t <- 0:10

## get the ACS
f <- acs("fgn",     t = t, H = .75)
b <- acs("burrXII", t = t, scale = 1, shape1 = .6, shape2 = .4)
w <- acs("weibull", t = t, scale = 2, shape = 0.8)
p <- acs("paretoII", t = t, scale = 3, shape = 0.3)

## visualize the ACS
dta   <- data.table(t, f, b, w, p)
m.dta <- melt(dta, id.vars = "t")

ggplot(m.dta,
       aes(x      = t,
           y      = value,
           group  = variable,
           colour = variable)) +
  geom_point(size = 2.5) +
  geom_line(lwd = 1) +
  scale_color_manual(values = c("steelblue4", "red4", "green4", "darkorange"),
                     labels = c("FGN", "Burr XII", "Weibull", "Pareto II"),
                     name   = "") +
  labs(x = bquote(lag ~ tau),
       y = "Acf") +
  scale_x_continuous(breaks = t) +
  theme_classic()
```

---

actpnts *AutoCorrelation Transformed Points*

Description

Evaluates the (rho_x, rho_z) mapping between the target marginal autocorrelation and the underlying Gaussian autocorrelation, using a double numerical integral.

Usage

```
actpnts(margdist, margarg, p0 = 0, distbounds = c(-Inf, Inf))
```

Arguments

`margdist`	target marginal distribution
`margarg`	list of marginal distribution arguments
`p0`	probability zero
`distbounds`	numeric vector of length 2; distribution bounds (default `c(-Inf, Inf)`)

Details

When the package is compiled with Rcpp support (i.e., `actpnts_cpp` is available), the double integral is evaluated in C++ via the Cubature algorithm, which is substantially faster than the nested base-R `integrate()` fallback. The C++ path supports the following distributions natively: `ggamma`, `paretoII`, `burrXII`, `burrIII`, `gev`, `norm`, `beta`, `gamma`, `exp`, `weibull`, `lnorm`, `unif`. Any other distribution falls back to the R quantile function automatically, so correctness is always preserved.

Value

A data frame with columns `rhoz` (Gaussian correlations) and `rhox` (corresponding target marginal correlations).

References

Papalexiou, S.M. (2018). Unified theory for stochastic modelling of hydroclimatic processes: Preserving marginal distributions, correlation structures, and intermittency. Advances in Water Resources, 115, 234-252, doi:10.1016/j.advwatres.2018.02.013

See Also

`fitactf`, `acti`, `generateTS`

Examples

```
library(CoSMoS)

## Pareto type II marginal
x <- actpnts(margdist = "paretoII",
             margarg  = list(scale = 1, shape = .3),
             p0 = 0)
x
```

actpntsB6 *AutoCorrelation Transformed Points for Bardossy dependence structure*

Description

Evaluates the (rho_x, rho_z) mapping using Monte Carlo integration for the Bardossy copula dependence structure.

Usage

```
actpntsB6(margdist, margarg, m, p0 = 0)
```

Arguments

margdist	target marginal distribution
margarg	list of marginal distribution arguments
m	mean of the parent Gaussian processes controlling asymmetry
p0	probability of zero values

Value

A data frame with columns rhoz and rhox.

References

Bardossy, A. (2006). Copula-based geostatistical models for groundwater quality parameters. Water Resources Research, 42(11), doi:10.1029/2005WR004754

See Also

actpnts, generateMTSFast

Examples

```
library(CoSMoS)

x <- actpntsB6(margdist = "paretoII",
               margarg  = list(scale = 1, shape = .3),
               m = 1,
               p0 = 0)
x
```

advectionF *Advection fields*

Description

Provides parametric functions that describe different types of advection fields.

Usage

```
advectionF(id, ...)
```

Arguments

id	advection type id (uniform, rotation, spiral, spiralCE, radial, and hyperbolic)
...	other arguments (vector of coordinates and parameters of advection field functions)

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi:10.1029/2020WR029466

Examples

```
library(ggquiver)
library(ggplot2)

## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

## get the advection field
af <- advectionF("spiral",
                 spacepoints = coord,
                 x0 = floor(m / 2),
                 y0 = floor(m / 2),
                 a = 3,
                 b = 2,
                 rotation = 1)

## visualize advection field
dta <- data.frame(lon = coord[ ,1], lat = coord[ ,2], u = af[ ,1], v = af[ ,2])
ggplot(dta, aes(x = lon, y = lat, u = u, v = v)) +
geom_quiver() +
theme_light()
```

`advectionFhyperbolic` *Hyperbolic advection field*

Description

Provides an advection field with hyperbolic trajectories.

Usage

```
advectionFhyperbolic(spacepoints, x0, y0, a, b)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`x0`	x coordinate of the center of hyperbola
`y0`	y coordinate of the center of hyperbola
`a`	parameter controlling the x component of rotational velocity
`b`	parameter controlling the y component of rotational velocity

Note

- if a > 0, b > 0: toward bottom-left and top-right corner
- if a < 0, b < 0: toward top-left and bottom-right corner

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi:10.1029/2020WR029466

Examples

```
library(ggquiver)
library(ggplot2)
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

af <- advectionFhyperbolic(spacepoints = coord,
                           x0 = floor(m / 2),
                           y0 = floor(m / 2),
                           a = 3,
                           b = 2)

## visualize advection field
dta <- data.frame(lon = coord[ ,1], lat = coord[ ,2], u = af[ ,1], v = af[ ,2])
```

```
ggplot(dta, aes(x = lon, y = lat, u = u, v = v)) +
geom_quiver() +
theme_light()
```

`advectionFradial` *Radial advection field*

Description

Provides an advection field corresponding to radial motion from or towards a specified reference point.

Usage

```
advectionFradial(spacepoints, x0, y0, a, b)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`x0`	x coordinate of the center of radial motion
`y0`	y coordinate of the center of radial motion
`a`	parameter controlling the x component of radial velocity
`b`	parameter controlling the y component of radial velocity

Note

- if a > 0, b > 0: divergence from (x0, y0) (source point effect)
- if a < 0, b < 0: convergence to (x0, y0) (sink effect)

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi:10.1029/2020WR029466

Examples

```
library(ggquiver)
library(ggplot2)

## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

af <- advectionFradial(spacepoints = coord,
```

```
                        x0 = floor(m / 2),
                        y0 = floor(m / 2),
                        a = 3,
                        b = 2)

## visualize advection field
dta <- data.frame(lon = coord[ ,1], lat = coord[ ,2], u = af[ ,1], v = af[ ,2])
ggplot(dta, aes(x = lon, y = lat, u = u, v = v)) +
geom_quiver() +
theme_light()
```

`advectionFrotation` *Rotational advection field*

Description

Provides an advection field corresponding to rotation around a specified center.

Usage

```
advectionFrotation(spacepoints, x0, y0, a, b)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`x0`	x coordinate of the center of rotation
`y0`	y coordinate of the center of rotation
`a`	parameter controlling the x component of rotational velocity
`b`	parameter controlling the y component of rotational velocity

Note

- if a > 0, b > 0: clockwise rotation around (x0, y0)
- if a < 0, b < 0: counter-clockwise rotation around (x0, y0)

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi:10.1029/2020WR029466

Examples

```
library(ggquiver)
library(ggplot2)
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

af <- advectionFrotation(spacepoints = coord,
                         x0 = floor(m / 2),
                         y0 = floor(m / 2),
                         a = 3,
                         b = 2)

## visualize advection field
dta <- data.frame(lon = coord[ ,1], lat = coord[ ,2], u = af[ ,1], v = af[ ,2])
ggplot(dta, aes(x = lon, y = lat, u = u, v = v)) +
geom_quiver() +
theme_light()
```

---

advectionFspiral *Spiraling advection field*

Description

Provides an advection field corresponding to a spiral motion to/from a specified reference point (sink).

Usage

```
advectionFspiral(spacepoints, x0, y0, a, b, rotation = 1)
```

Arguments

spacepoints	vector of coordinates (2 x d), where d is the number of locations/grid points
x0	x coordinate of reference point (sink)
y0	y coordinate of reference point (sink)
a	parameter controlling the x component of rotational velocity
b	parameter controlling the y component of rotational velocity
rotation	parameter controlling the rotational direction. The following combinations hold:

- if a > 0, b > 0, and direction = 1: spiraling CLOCKWISE TO (x0, y0)
- if a < 0, b < 0, and direction = 1: spiraling COUNTER-CLOCKWISE FROM (x0, y0)
- if a > 0, b > 0, and direction = 2: spiraling COUNTER-CLOCKWISE TO (x0, y0)
- if a < 0, b < 0, and direction = 2: spiraling CLOCKWISE FROM (x0, y0)

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi:10.1029/2020WR029466

Examples

```
library(ggquiver)
library(ggplot2)
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

af <- advectionFspiral(spacepoints = coord,
                       x0 = floor(m / 2),
                       y0 = floor(m / 2),
                       a = 3,
                       b = 2,
                       rotation = 1)

## visualize advection field
dta <- data.frame(lon = coord[ ,1], lat = coord[ ,2], u = af[ ,1], v = af[ ,2])
ggplot(dta, aes(x = lon, y = lat, u = u, v = v)) +
geom_quiver() +
theme_light()
```

`advectionFspiralCE` *Spiraling advection field satisfying continuity equation*

Description

Provides an advection field corresponding to a spiral motion to/from a specified reference point (sink) satisfying continuity equation (from Git Mirror of John Burkardt's collection of FORTRAN 90 Software).

Usage

```
advectionFspiralCE(spacepoints, a, C)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`a`	parameter controlling the intensity of rotational velocity (a > 0 clokwise; a < 0 conter-clockwise)
`C`	parameter ranging in (0, 2*pi)

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi:10.1029/2020WR029466

Examples

```
library(ggquiver)
library(ggplot2)
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

af <- advectionFspiralCE(spacepoints = coord,
                         a = 5,
                         C = 1)

## visualize advection field
dta <- data.frame(lon = coord[ ,1], lat = coord[ ,2], u = af[ ,1], v = af[ ,2])
ggplot(dta, aes(x = lon, y = lat, u = u, v = v)) +
geom_quiver() +
theme_light()
```

---

advectionFuniform *Uniform advection field*

Description

Provides an advection field with constant orthogonal (u and v) components at each grid point. This mimics rigid translation in a given direction according to the components u and v of the velocity vector.

Usage

```
advectionFuniform(spacepoints, u, v)
```

Arguments

spacepoints	vector of coordinates (2 x d), where d is the number of locations/grid points
u	velocity component along the x axis
v	velocity component along the y axis

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi:10.1029/2020WR029466

Examples

```
library(ggquiver)
library(ggplot2)
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

af <- advectionFuniform(spacepoints = coord,
                        u = 2,
                        v = 6)

## visualize advection field
dta <- data.frame(lon = coord[ ,1], lat = coord[ ,2], u = af[ ,1], v = af[ ,2])
ggplot(dta, aes(x = lon, y = lat, u = u, v = v)) +
geom_quiver() +
theme_light()
```

---

analyzeTS *Analyse, report, and simulate seasonal time series*

Description

analyzeTS automatically performs seasonal analysis, fits distributions and correlation structures. reportTS visualises the fitted distributions and correlation structures, or returns a table of fitted parameters and descriptive statistics. simulateTS takes the result of analyzeTS and generates synthetic realisations.

Usage

```
analyzeTS(
  TS,
  season = "month",
  dist = "ggamma",
  acsID = "weibull",
  norm = "N1",
  n.points = 30,
  lag.max = 30,
  constrain = FALSE,
  opts = NULL
)

reportTS(aTS, method = "dist")

simulateTS(aTS, from = NULL, to = NULL)
```

Arguments

TS	data frame or data table with columns date and value
season	character; name of a date-component function (e.g. "month", "week")
dist	character; name of the distribution to fit (e.g. "norm", "ggamma")
acsID	character; ACS identifier passed to fitACS
norm	character; norm identifier — one of "N1", "N2", "N3", "N4"
n.points	integer; number of ECDF points used in the norm computation
lag.max	integer; maximum lag for the empirical ACF
constrain	logical; if TRUE, constrains shape2 parameters to (0, 0.48) to enforce finite upper tails
opts	list of nloptr minimisation options
aTS	an analyzeTS result object
method	character; report type — "dist" for distribution fits, "acs" for ACS fits, "stat" for descriptive statistics table
from	POSIXct; start of simulation period (defaults to start of observed series)
to	POSIXct; end of simulation period (defaults to end of observed series)

Details

In practice, we typically want to simulate a natural process from observed data. analyzeTS fits a marginal distribution and autocorrelation structure for each season; reportTS lets you inspect the fit; simulateTS generates synthetic time series with the same seasonal statistical properties.

Recommended distributions by variable type:

- *precipitation / streamflow*: ggamma, burrXII, burrIII
- *relative humidity*: beta
- *temperature*: norm

Value

- analyzeTS: a list with elements data, dfits, afits, and attributes season, dist, acsID, date
- reportTS: a ggplot object ("dist" or "acs" method) or a data.frame ("stat" method)
- simulateTS: a data.table with columns date and value

See Also

fitDist, fitACS, generateTS

Examples

```
library(CoSMoS)

## Load data included in the package
data("precip")

## Fit seasonal ACSs and distributions to the data
a <- analyzeTS(precip)

reportTS(a, "dist")  ## seasonal distribution fits
reportTS(a, "acs")   ## seasonal ACS fits
reportTS(a, "stat")  ## descriptive statistics

## Simulate a time series of the same length
sim <- simulateTS(a)

precip[, id := "observed"]
sim[, id := "simulated"]
dta <- rbind(precip, sim)

ggplot(dta) +
  geom_line(aes(x = date, y = value)) +
  facet_wrap(~id, ncol = 1) +
  theme_classic()

## Simulate a time series of different length
sim <- simulateTS(a,
                  from = as.POSIXct("1978-12-01 00:00:00"),
                  to   = as.POSIXct("2008-12-01 00:00:00"))
```

`anisotropyT` *Anisotropy transformation*

Description

Provides parametric functions that describe different types of planar deformation fields, including affine (rotation and stretching), and swirl-like deformation. For more details see Papalexiou et al.(2021) and references therein.

Usage

```
anisotropyT(id, ...)
```

Arguments

id	anisotropy type id (affine, swirl, and wave)
...	additional arguments (vector of coordinates and parameters of the anisotropy transformations)

References

Papalexiou, S. M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond, Water Resources Research, doi:10.1029/2020WR029466

Examples

```
library(CoSMoS)

## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

## get the anisotropy field
at1 <- anisotropyT("affine",
                   spacepoints = coord,
                   phi1 = 0.5,
                   phi2 = 2,
                   phi12 = 0,
                   theta = -pi/3)
at2 <- anisotropyT("swirl",
                   spacepoints = coord,
                   x0 = floor(m / 2),
                   y0 = floor(m / 2),
                   b = 10,
                   alpha = 1.5 * pi)
at3 <- anisotropyT("wave",
                   spacepoints = coord,
                   phi1 = 0.5,
                   phi2 = 2,
                   beta = 3,
                   theta = 0)

## visualize anisotropy field
aux = data.frame(lon = at2[ ,1], lat = at2[ ,2], id1 = rep(1:m, each = m), id2 = rep(1:m, m))
ggplot(aux, aes(x = lon, y = lat)) +
geom_path(aes(group = id1)) +
geom_path(aes(group = id2)) +
geom_point(col = 2) +
theme_light()
```

`anisotropyTaffine` *Affine anisotropy transformation*

Description

Affine anisotropy transformation.

Usage

```
anisotropyTaffine(spacepoints, phi1, phi2, phi12, theta)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`phi1`	stretching parameter along the x axis
`phi2`	stretching parameter along the y axis
`phi12`	shear effect
`theta`	rotation angle

References

Allard, D., Senoussi, R., Porcu, E. (2016). Anisotropy Models for Spatial Data. Mathematical Geosciences, 48(3), 305-328, doi:10.1007/s110040159594x

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi:10.1029/2020WR029466

Examples

```
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

at <- anisotropyTaffine(spacepoints = coord,
                        phi1 = 0.5,
                        phi2 = 2,
                        phi12 = 0,
                        theta = -pi/3)

## visualize transformed coordinate system
aux = data.frame(lon = at[ ,1], lat = at[ ,2], id1 = rep(1:m, each = m), id2 = rep(1:m, m))
ggplot(aux, aes(x = lon, y = lat)) +
geom_path(aes(group = id1)) +
geom_path(aes(group = id2)) +
geom_point(col = 2) +
theme_light()
```

`anisotropyTswirl` *Swirl anisotropy transformation*

Description

Swirl anisotropy transformation.

Usage

```
anisotropyTswirl(spacepoints, x0, y0, b, alpha)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`x0`	x coordinate of the center of the swirl deformation
`y0`	y coordinate of the center of the swirl deformation
`b`	scaling parameter controlling the swirl deformation
`alpha`	rotation angle

References

Ligas, M., Banas, M., Szafarczyk, A. (2019). A method for local approximation of a planar deformation field. Reports on Geodesy and Geoinformatics, 108(1), 1-8, doi:10.2478/rgg20190007

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi:10.1029/2020WR029466

Examples

```
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

at <- anisotropyTswirl(spacepoints = coord,
                       x0 = floor(m / 2),
                       y0 = floor(m / 2),
                       b = 10,
                       alpha = 1.5 * pi)

## visualize transformed coordinate system
aux = data.frame(lon = at[ ,1], lat = at[ ,2], id1 = rep(1:m, each = m), id2 = rep(1:m, m))
ggplot(aux, aes(x = lon, y = lat)) +
geom_path(aes(group = id1)) +
geom_path(aes(group = id2)) +
geom_point(col = 2) +
theme_light()
```

`anisotropyTwave` *Wave anisotropy transformation*

Description

Wave anisotropy transformation.

Usage

```
anisotropyTwave(spacepoints, phi1, phi2, beta, theta)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`phi1`	stretching parameter along the x axis
`phi2`	stretching parameter along the y axis
`beta`	amplitude of sinusoidal wave
`theta`	rotation angle

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi:10.1029/2020WR029466

Examples

```
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

at <- anisotropyTwave(spacepoints = coord,
                      phi1 = 0.5,
                      phi2 = 2,
                      beta = 3,
                      theta = 0)

## visualize transformed coordinate system
aux = data.frame(lon = at[ ,1], lat = at[ ,2], id1 = rep(1:m, each = m), id2 = rep(1:m, m))
ggplot(aux, aes(x = lon, y = lat)) +
geom_path(aes(group = id1)) +
geom_path(aes(group = id2)) +
geom_point(col = 2) +
theme_light()
```

BurrIII *Burr Type III distribution*

Description

Provides density, distribution function, quantile function, random value generation, and raw moments of order *r* for the Burr Type III distribution.

Usage

```
dburrIII(x, scale, shape1, shape2, log = FALSE)

pburrIII(q, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

qburrIII(p, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

rburrIII(n, scale, shape1, shape2)

mburrIII(r, scale, shape1, shape2)
```

Arguments

x, q	vector of quantiles.
scale, shape1, shape2	scale and shape parameters; the shape arguments cannot be vectors (must have length one).
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise $P[X > x]$.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
r	raw moment order.

Value

dburrIII returns a numeric vector of density values. pburrIII returns a numeric vector of cumulative probabilities. qburrIII returns a numeric vector of quantiles. rburrIII returns a numeric vector of random deviates. mburrIII returns the raw moment of order r.

References

Papalexiou, S.M. (2018). Unified theory for stochastic modelling of hydroclimatic processes: Preserving marginal distributions, correlation structures, and intermittency. Advances in Water Resources, 115, 234-252, doi:10.1016/j.advwatres.2018.02.013

See Also

fitDist, moments

Examples

```
## plot the density

ggplot(data.frame(x = c(1, 15)),
       aes(x)) +
  stat_function(fun = dburrIII,
                args = list(scale = 5,
                            shape1 = .25,
                            shape2 = .75),
                colour = "royalblue4") +
  labs(x = "",
       y = "Density") +
  theme_classic()
```

BurrXII *Burr Type XII distribution*

Description

Provides density, distribution function, quantile function, random value generation, and raw moments of order *r* for the Burr Type XII distribution.

Usage

```
dburrXII(x, scale, shape1, shape2, log = FALSE)

pburrXII(q, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

qburrXII(p, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

rburrXII(n, scale, shape1, shape2)

mburrXII(r, scale, shape1, shape2)
```

Arguments

x, q	vector of quantiles.
scale, shape1, shape2	scale and shape parameters; the shape arguments cannot be vectors (must have length one).
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise $P[X > x]$.
p	vector of probabilities.

n	number of observations. If length(n) > 1, the length is taken to be the number required.
r	raw moment order.

Value

dburrXII returns a numeric vector of density values. pburrXII returns a numeric vector of cumulative probabilities. qburrXII returns a numeric vector of quantiles. rburrXII returns a numeric vector of random deviates. mburrXII returns the raw moment of order r.

References

Papalexiou, S.M. (2018). Unified theory for stochastic modelling of hydroclimatic processes: Preserving marginal distributions, correlation structures, and intermittency. Advances in Water Resources, 115, 234-252, doi:10.1016/j.advwatres.2018.02.013

See Also

fitDist, moments

Examples

```
## plot the density

ggplot(data.frame(x = c(0, 10)),
       aes(x)) +
  stat_function(fun = dburrXII,
                args = list(scale = 5,
                            shape1 = .25,
                            shape2 = .75),
                colour = "royalblue4") +
  labs(x = "",
       y = "Density") +
  theme_classic()
```

---

checkRF *Numerical and visual check of generated random fields*

Description

Compares generated random fields sample statistics with the theoretically expected values (similar to checkTS). It also returns graphical output for visual check.

Usage

```
checkRF(RF, lags = 30, nfields = 49, method = "stat")
```

Arguments

RF	output of generateRF
lags	number of lags of empirical STCF to be considered in the graphical output (default set to 30)
nfields	number of fields to be used in the numerical and graphical output (default set to 49). As the plots are arranged in a matrix with nrows as close as possible to ncol, we suggest using values such as 3x3, 3x4, 7x8, etc.
method	report method - "stat" for basic statistical report, "statplot" for graphical check of lagged SCS, target STCS, and marginal distribution, "field" for plotting a matrix of the first nfields, and "movie" to save the first nfields as a GIF file named "movieRF.gif" in the current working directory

Examples

```
## The example below refers to the fitting and simulation of 10 random fields
## of size 10x10 with AR(1) temporal correlation. As the fitting algorithm has
## O((mxm)^3) complexity for a mxm field, this setting allows for quick fitting
## and simulation (short CPU time). However, for a more effective visualization
## and reliable performance assessment, we suggest to generate a larger number
## of fields (e.g. 100 or more) of size about 30X30. This setting needs more
## CPU time but enables more effective comparison of theoretical and
## empirical statistics. Sizes larger than about 50x50 can be unpractical
##  on standard machines.

fit <- fitVAR(
  spacepoints = 10,
  p = 1,
  margdist ="burrXII",
  margarg = list(scale = 3, shape1 = .9, shape2 = .2),
  p0 = 0.8,
  stcsid = "clayton",
  stcsarg = list(scfid = "weibull", tcfid = "weibull",
                 copulaarg = 2,
                 scfarg = list(scale = 20, shape = 0.7),
                tcfarg = list(scale = 1.1, shape = 0.8))
)

sim <- generateRF(n = 12,
                  STmodel = fit)
checkRF(RF = sim,
          lags = 10,
          nfields = 12)
```

---

checkTS *Check generated time series*

Description

Compares sample statistics of generated time series against theoretically expected values.

Usage

```
checkTS(TS, distbounds = c(-Inf, Inf))
```

Arguments

TS	a cosmosts object from generateTS, or a list of numeric vectors, or a single numeric vector
distbounds	numeric vector of length 2; distribution bounds (default c(-Inf, Inf))

Value

An object of class c("checkTS", "matrix") with rows "expected" and one row per simulated series, and columns for mean, sd, skew, p0, acf_t1, acf_t2, acf_t3. Attributes margdist, margarg, and p0 are attached for use by plot.checkTS.

See Also

generateTS, plot.checkTS, moments

Examples

```
library(CoSMoS)

x <- generateTS(margdist = "burrXII",
                margarg = list(scale = 1,
                               shape1 = .75,
                               shape2 = .25),
                acsvalue = acs(id = "weibull",
                               t = 0:30,
                               scale = 10,
                               shape = .75),
                n = 1000, p = 30, p0 = .5, TSn = 5)

checkTS(x)
```

---

disch *Daily streamflow data data*

Description

Station details

- Name: Nassawango Creek near Snow Hill, Worcester County, Maryland, Hydrologic Unit 02080111
- Network Id: , USGS 01485500
- Latitude/Longitude: 38°13'44.1", 75°28'17.2"
- Elevation: 11.49 ft above North American Vertical Datum of 1988.
- Measurement unit: cubic feet per second

Usage

```
disch
```

Format

A data.table with 23315 rows and 2 variables:

date POSIXct format date/time

value daily avarage values

Details

more details can be found here.

Source

The United States Geological Survey (USGS) National Water Information System (NWIS)

`fitACS` *Autocorrelation structure fitting*

Description

Fits a parametric autocorrelation structure (ACS) to empirical ACF values using Nelder-Mead optimisation with MSE criterion.

Usage

```
fitACS(acf, ID, start = NULL, lag = NULL)
```

Arguments

`acf`	numeric vector of autocorrelation function values from lag 0
`ID`	character; ACS identifier (e.g. `"weibull"`, `"paretoII"`)
`start`	numeric vector of starting parameter values; if `NULL`, all parameters start at 1
`lag`	integer; number of lags to use; if `NULL`, lags up to the first value ≤ 0.01 are used (or all lags if none drops below 0.01)

Value

An object of class "fitACS": a named list of fitted ACS parameters with attributes ID and eACS (empirical ACS used for fitting).

See Also

fitDist, plot.fitACS, acs

Examples

```
x <- arima.sim(model = list(ar = 0.8), n = 1000)

acsfit <- fitACS(acf(x, plot = FALSE)$acf, "weibull", c(1, 1))
```

---

fitactf *Fit the AutoCorrelation Transformation Function*

Description

Fits the ACTF to the estimated (rho_x, rho_z) points using nls.

Usage

```
fitactf(actpnts, discrete = FALSE)
```

Arguments

actpnts	estimated ACT points (output of actpnts)
discrete	logical — is the marginal distribution discrete?

Value

An object of class "acti" with components:

actfcoef fitted ACTF coefficients b and c

actfpoints the input ACT points data frame

See Also

actpnts, actf

Examples

```
library(CoSMoS)

p   <- actpnts(margdist = "paretoII",
               margarg  = list(scale = 1, shape = .3),
               p0 = 0)
fit <- fitactf(p)
plot(fit)
```

`fitDist` *Distribution fitting*

Description

Fits a parametric distribution to data using the Nelder-Mead simplex algorithm to minimise one of four fitting norms.

Usage

```
fitDist(
  data,
  dist,
  n.points,
  norm,
  constrain,
 opts = list(algorithm = "NLOPT_LN_NELDERMEAD", xtol_rel = 1e-08, maxeval = 10000)
)
```

Arguments

`data`	numeric vector of values to fit
`dist`	character; distribution name (e.g. `"norm"`, `"ggamma"`)
`n.points`	integer; number of ECDF points used in norm computation
`norm`	character; norm identifier — one of `"N1"` (ratio RMSE on quantiles), `"N2"` (MSE on quantiles), `"N3"` (ratio RMSE on probabilities), `"N4"` (MSE on probabilities)
`constrain`	logical; if `TRUE`, constrains `shape2` parameters to `(0, 0.48)` to enforce finite upper tails
`opts`	list of `nloptr` minimisation options

Value

An object of class `"fitDist"`: a named list of fitted distribution parameters with attributes `dist`, `edf` (empirical CDF), and `nfo` (full `nloptr` output).

See Also

fitACS, plot.fitDist

Examples

```
x <- fitDist(rnorm(1000), "norm", 30, "N1", FALSE)
x
```

`fitVAR` *VAR model parameters to simulate correlated parent Gaussian random vectors and fields*

Description

Compute VAR model parameters to simulate parent Gaussian random vectors with specified spatiotemporal correlation structure using the method described by Biller and Nelson (2003).

Usage

```
fitVAR(
  spacepoints,
  p,
  margdist,
  margarg,
  p0,
  distbounds = c(-Inf, Inf),
  stcsid,
  stcsarg,
  scalefactor = 1,
  anisotropyid = "affine",
  anisotropyarg = list(phi1 = 1, phi2 = 1, phi12 = 0, theta = 0),
  advectionid = "uniform",
  advectionarg = list(u = 0, v = 0),
  dsid = "gauss",
  dsarg = NULL
)
```

Arguments

`spacepoints`	it can be a numeric integer, which is interpreted as the side length m of the square field (m x m), or a matrix (d x 2) of coordinates (e.g. longitude and latitude) of d spatial locations (e.g. d gauge stations)
`p`	order of VAR(p) model
`margdist`	target marginal distribution of the field

margarg	list of marginal distribution arguments. Please consult the documentation of the selected marginal distribution indicated in the argument margdist for the list of required parameters
p0	probability zero
distbounds	distribution bounds (default set to c(-Inf, Inf))
stcsid	spatiotemporal correlation structure ID
stcsarg	list of spatiotemporal correlation structure arguments. Please consult the documentation of the selected spatiotemporal correlation structure indicated in the argument stcsid for the list of required parameters
scalefactor	factor specifying the distance between the centers of two pixels (default set to 1)
anisotropyid	spatial anisotropy ID (affine by default, swirl or wave)
anisotropyarg	list of arguments characterizing the spatial anisotropy according to the syntax of the function anisotropyT. Isotropic fields by default
advectionid	advection field ID (uniform by default, rotation, spiral, spiralCE, radial, or hyperbolic)
advectionarg	list of arguments characterizing the advection field according to the syntax of advectionF. No advection by default
dsid	dependence structure ID (gauss by default, student, bardossy, and bardossyF)
dsarg	argument characterizing the dependence structure: NULL for gauss by default, number of degrees of freedom for student or parameter m in (-Inf, Inf) for bardossy (see Note section for more details)

Details

The fitting algorithm has $O(m*m)^3$ complexity for a $(m*m)$ field or equivalently $O(d^3)$ complexity for a d-dimensional vector. Very large values of $(m*m)$ (or d) and high order AR correlation structures can be unpractical on standard machines.

Here, we give indicative CPU times for some settings, referring to a Windows 10 Pro x64 laptop with Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, 4-core, 8 logical processors, and 32GB RAM.
: CPU time:
d = 100 or m = 10, p = 1: ~ 0.4s
d = 900 or m = 30, p = 1: ~ 6.0s
d = 900 or m = 30, p = 5: ~ 47.0s
d = 2500 or m = 50, p = 1: ~100.0s

Note

While all the advection types can be applied to isotropic random fields, anisotropic random fields require more care. We suggest combining affine anysotropy with uniform advection, and swirl anisotropy with rotation or spiral advection with the same rotation center..
Concerning the Bardossy model, the increase of the parameter m leads to a more and more symmetrical copula, and if m tends to Inf, then the copula converges to the Gaussian copula. The bardossy model is characterized by lower tail dependence weaker than the upper tail dependence, while the flipped Bárdossy dependence structure, denoted as bardossyF, has lower tail dependence stronger than the upper tail dependence. See Bárdossy (2006) for more details about the properties and parametrization of the multivariate Bardossy distribution

References

Bárdossy, A. (2006), Copula-based geostatistical models for groundwater quality parameters, Water Resour. Res., 42, W11416, doi:10.1029/2005WR004754

Biller, B., Nelson, B.L. (2003). Modeling and generating multivariate time-series input processes using a vector autoregressive technique. ACM Trans. Model. Comput. Simul. 13(3), 211-237, doi:10.1145/937332.937333

Papalexiou, S.M. (2018). Unified theory for stochastic modelling of hydroclimatic processes: Preserving marginal distributions, correlation structures, and intermittency. Advances in Water Resources, 115, 234-252, doi:10.1016/j.advwatres.2018.02.013

Papalexiou, S.M., Serinaldi, F. (2020). Random Fields Simplified: Preserving Marginal Distributions, Correlations, and Intermittency, With Applications From Rainfall to Humidity. Water Resources Research, 56(2), e2019WR026331, doi:10.1029/2019WR026331

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi:10.1029/2020WR029466

Examples

```
## for multivariate simulation
coord <- cbind(runif(4)*30, runif(4)*30)

fit <- fitVAR(
  spacepoints = coord,
  p = 1,
  margdist ='burrXII',
  margarg = list(scale = 3,
                 shape1 = .9,
                 shape2 = .2),
  p0 = 0.8,
  stcsid = "clayton",
  stcsarg = list(scfid = "weibull",
                 tcfid = "weibull",
                 copulaarg = 2,
                 scfarg = list(scale = 20,
                               shape = 0.7),
                 tcfarg = list(scale = 1.1,
                               shape = 0.8))
)

dim(fit$alpha)
dim(fit$res.cov)

fit$m
fit$margarg
fit$margdist

## for random fields simulation
fit <- fitVAR(
  spacepoints = 10,
  p = 1,
```

```
  margdist ='burrXII',
  margarg = list(scale = 3, shape1 = .9, shape2 = .2),
  p0 = 0.8,
  stcsid = "clayton",
  stcsarg = list(scfid = "weibull", tcfid = "weibull",
                 copulaarg = 2,
                 scfarg = list(scale = 20, shape = 0.7),
                 tcfarg = list(scale = 1.1, shape = 0.8))
)

dim(fit$alpha)
dim(fit$res.cov)

fit$m
fit$margarg
fit$margdist
```

`generateMTS` *Simulation of multiple time series with given marginals and spatiotemporal properties*

Description

Generates multiple time series with given marginals and spatiotemporal properties. Provide (1) the output of `fitVAR` and (2) the number of time steps to simulate.

Usage

```
generateMTS(n, STmodel)
```

Arguments

`n`	number of time steps to simulate
`STmodel`	list of arguments from `fitVAR`

Details

Referring to the documentation of `fitVAR` for details on computational complexity, here we report indicative simulation CPU times, assuming model parameters are already evaluated. CPU times refer to a Windows 10 Pro x64 laptop with Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, 4-core, 8 logical processors, and 32 GB RAM.
CPU time:
d = 900, p = 1, n = 1000: ~17s
d = 900, p = 1, n = 10000: ~75s
d = 900, p = 5, n = 100: ~280s
d = 900, p = 5, n = 1000: ~302s
d = 2500, p = 1, n = 1000: ~160s
d = 2500, p = 1, n = 10000: ~570s
where d denotes the number of spatial locations.

Value

A matrix of class "matrix" with attribute STmodel. Rows correspond to time steps and columns to spatial locations.

See Also

fitVAR, generateRF, generateMTSFast

Examples

```
## Simulation of a 4-dimensional vector with VAR(1) correlation structure
coord <- cbind(runif(4) * 30, runif(4) * 30)

fit <- fitVAR(
  spacepoints = coord,
  p = 1,
  margdist = "burrXII",
  margarg = list(scale = 3,
                 shape1 = .9,
                 shape2 = .2),
  p0 = 0.8,
  stcsid = "clayton",
  stcsarg = list(scfid = "weibull",
                 tcfid = "weibull",
                 copulaarg = 2,
                 scfarg = list(scale = 20,
                               shape = 0.7),
                 tcfarg = list(scale = 1.1,
                               shape = 0.8))
)

sim <- generateMTS(n = 100, STmodel = fit)
```

---

generateMTSFast *Faster simulation of multiple time series with approximately separable spatiotemporal correlation structure*

Description

For more details see section 6 in Serinaldi and Kilsby (2018) and section 2.4 in Papalexiou and Serinaldi (2020).

Usage

```
generateMTSFast(
  n,
  spacepoints,
  margdist,
```

```
  margarg,
  p0,
  distbounds = c(-Inf, Inf),
  stcsarg,
  scalefactor = 1,
  anisotropyid = "affine",
  anisotropyarg = list(phi1 = 1, phi2 = 1, phi12 = 0, theta = 0),
  dsid = "gauss",
  dsarg = NULL
)
```

Arguments

`n`	number of time steps to simulate
`spacepoints`	matrix (d x 2) of coordinates (e.g. longitude and latitude) for d spatial locations (e.g. gauge stations)
`margdist`	target marginal distribution
`margarg`	list of marginal distribution arguments; consult the documentation of the selected distribution for the required parameters
`p0`	probability zero
`distbounds`	distribution bounds (default `c(-Inf, Inf)`)
`stcsarg`	list of spatiotemporal correlation structure arguments; consult the documentation of the selected structure for required parameters
`scalefactor`	factor specifying the distance between pixel centres (default 1)
`anisotropyid`	spatial anisotropy ID (`"affine"` by default; `"swirl"` or `"wave"` also available)
`anisotropyarg`	list of arguments for `anisotropyT`; isotropic fields by default
`dsid`	dependence structure ID (`"gauss"` by default; `"student"`, `"bardossy"`, or `"bardossyF"`)
`dsarg`	argument for the dependence structure: `NULL` for `"gauss"`, degrees of freedom for `"student"`, or parameter `m` in $(-\infty, \infty)$ for `"bardossy"`

Details

`generateMTSFast` provides faster multivariate simulation than `generateMTS` by exploiting circulant-embedding fast Fourier transformation. This approach is feasible only for approximately separable target spatiotemporal correlation functions. `generateMTSFast` combines fitting and simulation in a single call. Indicative CPU times (Windows 10 Pro x64, Intel Core i7-6700HQ, 32 GB RAM):
d = 2500, n = 1000: ~58s
d = 2500, n = 10000: ~160s
d = 10000, n = 1000: ~2955s (~50 min)
where d denotes the number of spatial locations.

Value

A matrix of class `c("matrix", "cosmosts")` with attribute `STmodel` containing the fitted model components.

References

Serinaldi, F., Kilsby, C.G. (2018). Unsurprising Surprises: The Frequency of Record-breaking and Overthreshold Hydrological Extremes Under Spatial and Temporal Dependence. Water Resources Research, 54(9), 6460-6487, doi:10.1029/2018WR023055

Papalexiou, S.M., Serinaldi, F. (2020). Random Fields Simplified: Preserving Marginal Distributions, Correlations, and Intermittency, With Applications From Rainfall to Humidity. Water Resources Research, 56(2), e2019WR026331, doi:10.1029/2019WR026331

See Also

generateMTS, generateRFFast, fitVAR

Examples

```
coord <- cbind(runif(4) * 30, runif(4) * 30)

sim <- generateMTSFast(
    n = 50,
    spacepoints = coord,
    p0 = 0.7,
    margdist = "paretoII",
    margarg = list(scale = 1,
                   shape = .3),
    stcsarg = list(scfid = "weibull",
                   tcfid = "weibull",
                   scfarg = list(scale = 20,
                                 shape = 0.7),
                   tcfarg = list(scale = 1.1,
                                 shape = 0.8))
)
```

`generateRF` *Simulation of random fields with given marginals and spatiotemporal properties*

Description

Generates a random field with given marginals and spatiotemporal properties. Provide (1) the output of fitVAR and (2) the number of time steps to simulate.

Usage

```
generateRF(n, STmodel)
```

Arguments

n	number of fields (time steps) to simulate
STmodel	list of arguments from fitVAR

Details

Referring to the documentation of fitVAR for details on computational complexity, here we report indicative simulation CPU times, assuming model parameters are already evaluated. CPU times refer to a Windows 10 Pro x64 laptop with Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, 4-core, 8 logical processors, and 32 GB RAM.
CPU time:
m = 30, p = 1, n = 1000: ~17s
m = 30, p = 1, n = 10000: ~75s
m = 30, p = 5, n = 100: ~280s
m = 30, p = 5, n = 1000: ~302s
m = 50, p = 1, n = 1000: ~160s
m = 50, p = 1, n = 10000: ~570s
where m denotes the side length of a square field (m x m).

Value

A matrix of class "matrix" with attribute STmodel. Rows correspond to spatial locations and columns to time steps.

See Also

fitVAR, checkRF, generateMTS

Examples

```
## The example below simulates few random fields of size 10x10 with AR(1)
## temporal correlation for illustration. For reliable performance assessment
## generate a larger number of fields (e.g. 100 or more) of size ~30x30.
## See 'Details' for running times with different settings.

fit <- fitVAR(
  spacepoints = 10,
  p = 1,
  margdist = "burrXII",
  margarg = list(scale = 3, shape1 = .9, shape2 = .2),
  p0 = 0.8,
  stcsid = "clayton",
  stcsarg = list(scfid = "weibull", tcfid = "weibull",
                 copulaarg = 2,
                 scfarg = list(scale = 20, shape = 0.7),
                 tcfarg = list(scale = 1.1, shape = 0.8))
)

sim <- generateRF(n = 12, STmodel = fit)
checkRF(sim, lags = 10, nfields = 12)
```

`generateRFFast` *Faster simulation of random fields with approximately separable spatiotemporal correlation structure*

Description

For more details see section 6 in Serinaldi and Kilsby (2018) and section 2.4 in Papalexiou and Serinaldi (2020).

Usage

```
generateRFFast(
  n,
  spacepoints,
  margdist,
  margarg,
  p0,
  distbounds = c(-Inf, Inf),
  stcsarg,
  scalefactor = 1,
  anisotropyid = "affine",
  anisotropyarg = list(phi1 = 1, phi2 = 1, phi12 = 0, theta = 0),
  dsid = "gauss",
  dsarg = NULL
)
```

Arguments

`n`	number of fields (time steps) to simulate
`spacepoints`	side length m of the square field (m x m)
`margdist`	target marginal distribution of the field
`margarg`	list of marginal distribution arguments; consult the documentation of the selected distribution for the required parameters
`p0`	probability zero
`distbounds`	distribution bounds (default `c(-Inf, Inf)`)
`stcsarg`	list of spatiotemporal correlation structure arguments; consult the documentation of the selected structure for required parameters
`scalefactor`	factor specifying the distance between pixel centres (default 1)
`anisotropyid`	spatial anisotropy ID (`"affine"` by default; `"swirl"` or `"wave"` also available)
`anisotropyarg`	list of arguments for `anisotropyT`; isotropic fields by default
`dsid`	dependence structure ID (`"gauss"` by default; `"student"`, `"bardossy"`, or `"bardossyF"`)
`dsarg`	argument for the dependence structure: `NULL` for `"gauss"`, degrees of freedom for `"student"`, or parameter `m` in $(-\infty, \infty)$ for `"bardossy"`

Details

generateRFFast provides faster RF simulation than generateRF by exploiting circulant-embedding fast Fourier transformation. This approach is feasible only for approximately separable target spatiotemporal correlation functions. generateRFFast combines fitting and simulation in a single call. Indicative CPU times (Windows 10 Pro x64, Intel Core i7-6700HQ, 32 GB RAM):
m = 50, n = 1000: ~58s
m = 50, n = 10000: ~160s
m = 100, n = 1000: ~2955s (~50 min)

Value

A matrix of class `c("matrix", "cosmosts")` with attribute `STmodel` containing the fitted model components.

References

Serinaldi, F., Kilsby, C.G. (2018). Unsurprising Surprises: The Frequency of Record-breaking and Overthreshold Hydrological Extremes Under Spatial and Temporal Dependence. Water Resources Research, 54(9), 6460-6487, doi:10.1029/2018WR023055

Papalexiou, S.M., Serinaldi, F. (2020). Random Fields Simplified: Preserving Marginal Distributions, Correlations, and Intermittency, With Applications From Rainfall to Humidity. Water Resources Research, 56(2), e2019WR026331, doi:10.1029/2019WR026331

See Also

generateRF, generateMTSFast, checkRF, fitVAR

Examples

```
sim <- generateRFFast(
    n = 50,
    spacepoints = 3,
    p0 = 0.7,
    margdist = "paretoII",
    margarg = list(scale = 1,
                   shape = .3),
    stcsarg = list(scfid = "weibull",
                   tcfid = "weibull",
                   scfarg = list(scale = 20,
                                 shape = 0.7),
                   tcfarg = list(scale = 1.1,
                                 shape = 0.8))
)

checkRF(sim, lags = 10, nfields = 49)
```

`generateTS` *Generate time series*

Description

Generates time series with given properties. Provide (1) the target marginal distribution and its parameters, (2) the target autocorrelation structure or individual autocorrelation values up to a desired lag, and (3) the probability zero if you wish to simulate an intermittent process.

Usage

```
generateTS(
  n,
  margdist,
  margarg,
  p = NULL,
  p0 = 0,
  TSn = 1,
  distbounds = c(-Inf, Inf),
  acsvalue = NULL
)
```

Arguments

`n`	Positive integer. Length of the generated time series.
`margdist`	target marginal distribution
`margarg`	list of marginal distribution arguments
`p`	Positive integer or `NULL`. AR model order. When `NULL` (default), the order is chosen automatically as the number of lags where the transformed ACS exceeds 0.01, capped at 1000.
`p0`	probability zero
`TSn`	number of time series to generate
`distbounds`	numeric vector of length 2; distribution bounds (default `c(-Inf, Inf)`)
`acsvalue`	Numeric vector. Target autocorrelation structure starting from lag 0 (i.e. `acsvalue[1] = 1`).

Details

A step-by-step guide:

- First define the target marginal (`margdist`), that is, the probability distribution of the generated data. For example set `margdist = 'ggamma'` for the Generalised Gamma distribution, `margdist = 'burrXII'` for Burr type XII etc. For a full list of supported distributions see the help vignette. In general, the package supports all built-in distribution functions of R and of other packages.

- Define the parameters (`margarg`) of the selected distribution. For example the Generalised Gamma has one scale and two shape parameters, e.g. `margarg = list(scale = 2, shape1 = 0.9, shape2 = 0.8)`. See the help vignette for details on each distribution's parameters.
- If you wish your time series to be intermittent (e.g. precipitation), define the probability zero. For example `p0 = 0.9` produces 90\
- Define your linear autocorrelations.
  - Supply specific lag autocorrelations starting from lag 0 up to a desired lag, e.g. `acsvalue = c(1, 0.9, 0.8, 0.7)`; this preserves lag-1, lag-2 and lag-3 autocorrelations equal to 0.9, 0.8 and 0.7.
  - Alternatively, use a parametric autocorrelation structure (see section 3.2 in Papalexiou (2018)). Supported structures: `weibull`, `paretoII`, `fgn` and `burrXII`. See also `acs`.
- Define the AR model order `p`. For example if you aim to preserve the first 10 lag autocorrelations then set `p = 10`. Set `p = NULL` and the model will choose `p` to preserve the whole autocorrelation structure.
- Set the time series length, e.g. `n = 1000`, and the number of time series to generate, e.g. `TSn = 10`.

Value

An object of class `'cosmosts'`: a list of `TSn` numeric vectors, each of length `n`, with per-series attributes recording the fitted model parameters.

References

Papalexiou, S.M. (2018). Unified theory for stochastic modelling of hydroclimatic processes: Preserving marginal distributions, correlation structures, and intermittency. Advances in Water Resources, 115, 234-252, doi:10.1016/j.advwatres.2018.02.013

See Also

`regenerateTS`, `ARp`, `actpnts`

Examples

```
library(CoSMoS)

## Case 1:
## Generate 3 time series of length 1000 following the Generalised Gamma
## distribution with scale = 1, shape1 = 0.8, shape2 = 0.8 and ParetoII
## autocorrelation structure with scale = 1 and shape = 0.75.
x <- generateTS(margdist = "ggamma",
                margarg = list(scale = 1,
                               shape1 = .8,
                               shape2 = .8),
                acsvalue = acs(id = "paretoII",
                               t = 0:30,
                               scale = 1,
                               shape = .75),
                n = 1000,
```

```
                p = 30,
                TSn = 3)

## see the results
plot(x)



## Case 2:
## Same as Case 1 but intermittent with probability zero equal to 90%.
y <- generateTS(margdist = "ggamma",
                margarg = list(scale = 1,
                               shape1 = .8,
                               shape2 = .8),
                acsvalue = acs(id = "paretoII",
                               t = 0:30,
                               scale = 1,
                               shape = .75),
                p0 = .9,
                n = 1000,
                p = 30,
                TSn = 3)

## see the results
plot(y)

## Case 3:
## Generate a time series of length 1000 following the Beta distribution
## (e.g. relative humidity in [0, 1]) with shape1 = 0.6, shape2 = 0.8
## and ParetoII autocorrelation structure.
z <- generateTS(margdist = "beta",
                margarg = list(shape1 = .6,
                               shape2 = .8),
                distbounds = c(0, 1),
                acsvalue = acs(id = "paretoII",
                               t = 0:30,
                               scale = 1,
                               shape = .75),
                n = 1000,
                p = 20)

## see the results
plot(z)

## Case 4:
## Same as Case 3 but providing specific autocorrelation values for the
## first three lags (lag 1 to 3 equal to 0.9, 0.8, 0.7).
z <- generateTS(margdist = "beta",
                margarg = list(shape1 = .6,
                               shape2 = .8),
                distbounds = c(0, 1),
                acsvalue = c(1, .9, .8, .7),
                n = 1000,
```

```
                p = NULL)

## see the results
plot(z)
```

GEV *Generalized Extreme Value distribution*

Description

Provides density, distribution function, quantile function, random value generation, and raw moments of order *r* for the generalized extreme value distribution.

Usage

```
dgev(x, loc, scale, shape, log = FALSE)

pgev(q, loc, scale, shape, lower.tail = TRUE, log.p = FALSE)

qgev(p, loc, scale, shape, lower.tail = TRUE, log.p = FALSE)

rgev(n, loc, scale, shape)

mgev(r, loc, scale, shape)
```

Arguments

x, q	vector of quantiles.
loc, scale, shape	location, scale, and shape parameters.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise $P[X > x]$.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
r	raw moment order.

Value

dgev returns a numeric vector of density values. pgev returns a numeric vector of cumulative probabilities. qgev returns a numeric vector of quantiles. rgev returns a numeric vector of random deviates. mgev returns the raw moment of order r (via numerical integration).

See Also

fitDist, moments

Examples

```
## plot the density

ggplot(data.frame(x = c(0, 20)),
       aes(x)) +
  stat_function(fun = dgev,
                args = list(loc = 1,
                            scale = .5,
                            shape = .15),
                colour = "royalblue4") +
  labs(x = "",
       y = "Density") +
  theme_classic()
```

GGamma *Generalized Gamma distribution*

Description

Provides density, distribution function, quantile function, random value generation, and raw moments of order *r* for the generalized gamma distribution.

Usage

```
dggamma(x, scale, shape1, shape2, log = FALSE)

pggamma(q, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

qggamma(p, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

rggamma(n, scale, shape1, shape2)

mggamma(r, scale, shape1, shape2)
```

Arguments

x, q	vector of quantiles.
scale, shape1, shape2	scale and shape parameters; the shape arguments cannot be vectors (must have length one).
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise $P[X > x]$.
p	vector of probabilities.

n	number of observations. If length(n) > 1, the length is taken to be the number required.
r	raw moment order.

Value

dggamma returns a numeric vector of density values. pggamma returns a numeric vector of cumulative probabilities. qggamma returns a numeric vector of quantiles. rggamma returns a numeric vector of random deviates. mggamma returns the raw moment of order r.

References

Papalexiou, S.M., Koutsoyiannis, D. (2012). Entropy based derivation of probability distributions: A case study to daily rainfall. Advances in Water Resources, 45, 51-57, doi:10.1016/j.advwatres.2011.11.007

See Also

fitDist, moments

Examples

```
## plot the density

ggplot(data.frame(x = c(0, 20)),
       aes(x)) +
  stat_function(fun = dggamma,
                args = list(scale = 5,
                            shape1 = .25,
                            shape2 = .75),
                colour = "royalblue4") +
  labs(x = "",
       y = "Density") +
  theme_classic()
```

---

moments *Numerical estimation of moments*

Description

Uses numerical integration to compute the theoretical raw or central moments of the specified distribution.

Usage

```
moments(
  dist,
  distarg,
  p0 = 0,
  raw = TRUE,
  central = TRUE,
  coef = TRUE,
  distbounds = c(-Inf, Inf),
  order = 1:4
)
```

Arguments

`dist`	character; distribution name (e.g. `"norm"`, `"paretoII"`)
`distarg`	list of distribution arguments
`p0`	numeric; probability zero (default 0)
`raw`	logical; compute raw moments?
`central`	logical; compute central moments?
`coef`	logical; compute standardised coefficients (CV, skewness, kurtosis)?
`distbounds`	numeric vector of length 2; distribution bounds (default `c(-Inf, Inf)`)
`order`	integer vector; raw moment orders (default `1:4`)

Value

a named list with zero or more of:

`m` raw moments

`mu` central moments

`coefficients` CV, skewness, kurtosis

See Also

`sample.moments`, `populationstat`

Examples

```
library(CoSMoS)

## Normal distribution
moments("norm", list(mean = 2, sd = 1))

## Pareto type II
moments(dist    = "paretoII",
        distarg = list(shape = 0.2, scale = 1))
```

ParetoII *Pareto Type II distribution*

Description

Provides density, distribution function, quantile function, random value generation, and raw moments of order *r* for the Pareto type II distribution.

Usage

```
dparetoII(x, scale, shape, log = FALSE)

pparetoII(q, scale, shape, lower.tail = TRUE, log.p = FALSE)

qparetoII(p, scale, shape, lower.tail = TRUE, log.p = FALSE)

rparetoII(n, scale, shape)

mparetoII(r, scale, shape)
```

Arguments

x, q	vector of quantiles.
scale, shape	scale and shape parameters; the shape argument cannot be a vector (must have length one).
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise $P[X > x]$.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
r	raw moment order.

Value

dparetoII returns a numeric vector of density values. pparetoII returns a numeric vector of cumulative probabilities. qparetoII returns a numeric vector of quantiles. rparetoII returns a numeric vector of random deviates. mparetoII returns the raw moment of order r.

See Also

fitDist, moments

Examples

```
## plot the density

ggplot(data.frame(x = c(0, 20)),
       aes(x)) +
  stat_function(fun = dparetoII,
                args = list(scale = 1,
                            shape = .3),
                colour = "royalblue4") +
  labs(x = "",
       y = "Density") +
  theme_classic()
```

`plot.acti` *Plot method for* `acti` *objects*

Description

Visualises the autocorrelation transformation function (ACTF) fitted by `fitactf`.

Usage

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

Arguments

`x`	an `acti` object returned by `fitactf`
`...`	optional arguments; `main` sets the plot title

Value

a `ggplot` object (invisibly returned; also printed)

See Also

`fitactf`, `actpnts`

Examples

```
library(CoSMoS)

p   <- actpnts(margdist = "paretoII",
               margarg  = list(scale = 1, shape = .3),
               p0 = 0)
fit <- fitactf(p)

plot(fit)
plot(fit, main = "Pareto type II\nautocorrelation transformation")
```

`plot.checkTS` *Plot method for* `checkTS` *objects*

Description

Displays boxplots of simulated statistics against theoretical expected values for each statistic tracked by checkTS.

Usage

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

Arguments

x	a checkTS object returned by checkTS
...	currently unused

Value

a ggplot object (invisibly returned; also printed)

See Also

checkTS

Examples

```
library(CoSMoS)

x <- generateTS(margdist = "burrXII",
                margarg = list(scale = 1,
                               shape1 = .75,
                               shape2 = .15),
                acsvalue = acs(id = "weibull",
                               t = 0:30,
                               scale = 10,
                               shape = .75),
                n = 1000, p = 30, p0 = .25, TSn = 100)

chck <- checkTS(x)
plot(chck)
```

plot.cosmosts *Plot method for* cosmosts *objects*

Description

Visualises time series generated by generateTS as bar charts, one panel per series.

Usage

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

Arguments

x	a cosmosts object returned by generateTS
...	currently unused

Value

a ggplot object (invisibly returned; also printed)

See Also

generateTS, regenerateTS

Examples

```
library(CoSMoS)

ts <- generateTS(margdist = "ggamma",
                 margarg  = list(scale = 1, shape1 = .8, shape2 = .8),
                 acsvalue = acs(id = "paretoII", t = 0:30,
                                scale = 1, shape = .75),
                 n = 1000, p = 30, TSn = 2)
plot(ts)
```

plot.fitACS *Plot method for* fitACS *objects*

Description

Displays the empirical ACF alongside the fitted theoretical autocorrelation structure.

Usage

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

Arguments

x	a fitACS object returned by fitACS
...	currently unused

Value

a ggplot object (invisibly returned; also printed)

See Also

fitACS

Examples

```
x <- arima.sim(model = list(ar = 0.8), n = 1000)
acsfit <- fitACS(acf(x, plot = FALSE)$acf, "weibull", c(1, 1))
plot(acsfit)
```

---

plot.fitDist *Plot method for* fitDist *objects*

Description

Displays the empirical CDF against the fitted theoretical CDF on a log-exceedance-probability scale.

Usage

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

Arguments

x	a fitDist object returned by fitDist
...	currently unused

Value

a ggplot object (invisibly returned; also printed)

See Also

fitDist

Examples

```
x <- fitDist(rnorm(1000), "norm", 30, "N1", FALSE)
plot(x)
```

---

precip *Hourly station precipitation data*

Description

Station details

- Name: Philadelphia International Airport
- Network ID: COOP:366889
- Latitude/Longitude: 39.87327°, -75.22678°
- Elevation: 3m

Usage

```
precip
```

Format

A data.table with 79633 rows and 2 variables:

date POSIXct format date/time

value precipitation totals

Details

more details can be found here.

Source

The National Oceanic and Atmospheric Administration (NOAA)

quickTSPlot *Quick visualisation of basic time series properties*

Description

Returns a composite figure showing the time series, empirical density function, and empirical autocorrelation function.

Usage

```
quickTSPlot(TS, ci = 0.95)
```

Arguments

TS	numeric vector (or data.frame/data.table) of time series values
ci	numeric; confidence level for the zero-autocorrelation band (default 0.95)

Value

a ggdraw object (printed as a side effect)

See Also

generateTS, plot.cosmosts

Examples

```
ggamma_sim <- rggamma(n = 1000, scale = 1, shape1 = 1, shape2 = .5)
quickTSPlot(ggamma_sim)
```

regenerateTS *Bulk time series generation*

Description

Generates additional time series using parameters already fitted by generateTS, avoiding recomputation of the ACTF.

Usage

```
regenerateTS(ts, TSn = 1)
```

Arguments

ts	a cosmosts object returned by generateTS
TSn	number of time series to generate

Details

After calling generateTS, use regenerateTS to generate more time series with the same fitted parameters. This is faster than re-running generateTS because the ACTF fitting step is skipped.

Value

An object of class 'cosmosts': a list of TSn numeric vectors of the same length as those in ts.

See Also

generateTS, ARp

Examples

```
library(CoSMoS)

## Fit once
x <- generateTS(margdist = "burrXII",
                margarg = list(scale = 1,
                               shape1 = .75,
                               shape2 = .25),
                acsvalue = acs(id = "weibull",
                               t = 0:30,
                               scale = 10,
                               shape = .75),
                n = 1000, p = 30, p0 = .5, TSn = 3)

## Generate more realisations with the same parameters
r <- regenerateTS(x)

plot(r)
```

---

sample.moments *Sample moments*

Description

Computes raw moments, central moments, and standardised coefficients (CV, skewness, kurtosis) from a numeric sample.

Usage

```
sample.moments(
  x,
  na.rm = FALSE,
  raw = TRUE,
  central = TRUE,
```

```
  coef = TRUE,
  order = 1:4
)
```

Arguments

x	numeric vector of values
na.rm	logical; strip NA values before computation?
raw	logical; compute raw moments?
central	logical; compute central moments?
coef	logical; compute standardised coefficients (CV, skewness, kurtosis)?
order	integer vector; raw moment orders (default 1:4)

Value

a named list with zero or more of:

m raw moments

mu central moments

coefficients CV, skewness, kurtosis

See Also

moments, checkTS

Examples

```
library(CoSMoS)

x <- rnorm(1000)
sample.moments(x)

y <- rparetoII(1000, 10, .1)
sample.moments(y)
```

---

stcfclayton *Clayton SpatioTemporal Correlation Structure*

Description

Provides spatiotemporal correlation structure function based on Clayton copula. For more details on the parametric spatiotemporal correlation structures see section 2.3 and 2.4 in Papalexiou and Serinaldi (2020).

Usage

```
stcfclayton(t, s, scfid, tcfid, copulaarg, scfarg, tcfarg)
```

Arguments

t	time lag
s	spatial lag (distance)
scfid	ID of the spatial (marginal) correlation structure (e.g. weibull)
tcfid	ID of the temporal (marginal) correlation structure (e.g. weibull)
copulaarg	parameter of the Clayton copula linking the marginal correlation structures
scfarg	parameters of spatial (marginal) correlation structure
tcfarg	parameters of temporal (marginal) correlation structure

References

Papalexiou, S.M., Serinaldi, F. (2020). Random Fields Simplified: Preserving Marginal Distributions, Correlations, and Intermittency, With Applications From Rainfall to Humidity. Water Resources Research, 56(2), e2019WR026331, doi:10.1029/2019WR026331

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi:10.1029/2020WR029466

Examples

```
library(plot3D)

## specify grid of spatial and temporal lags
d <- 31
st <- expand.grid(0:(d - 1),
                  0:(d - 1))

## get the STCS
wc <- stcfclayton(t = st[, 1],
                  s = st[, 2],
                  scfid = "weibull",
                  tcfid = "weibull",
                  copulaarg = 2,
                  scfarg = list(scale = 20,
                                shape = 0.7),
                  tcfarg = list(scale = 1.1,
                                shape = 0.8))

## visualize the STCS
wc.m <- matrix(wc,
               nrow = d)

persp3D(z = wc.m, x = 1: nrow(wc.m), y = 1:ncol(wc.m),
        expand = 1, main = "", scale = TRUE, facets = TRUE,
        xlab="Time lag", ylab = "Distance", zlab = "STCF",
```

```
colkey = list(side = 4, length = 0.5), phi = 20, theta = 120,
resfac = 5,  col= gg2.col(100))
```

`stcfgneiting14` *Gneiting-14 SpatioTemporal Correlation Structure*

Description

Provides spatiotemporal correlation structure function proposed by Gneiting (2002; Eq.14 at p. 593).

Usage

```
stcfgneiting14(t, s, a, c, alpha, beta, gamma, tau)
```

Arguments

`t`	time lag
`s`	spatial lag (distance)
`a`	nonnegative scaling parameter of time
`c`	nonnegative scaling parameter of space
`alpha`	smoothness parameter of time. Valid range: $(0, 1]$
`beta`	space-time interaction parameter. Valid range: $[0, 1]$
`gamma`	smoothness parameter of space. Valid range: $(0, 1]$
`tau`	space-time interaction parameter. Valid range: ≥ 1 (for 2-dimensional fields)

References

Gneiting, T. (2002). Nonseparable, Stationary Covariance Functions for Space-Time Data, Journal of the American Statistical Association, 97:458, 590-600, doi:10.1198/016214502760047113

Examples

```
library(plot3D)

## specify grid of spatial and temporal lags
d <- 31
st <- expand.grid(0:(d - 1),
                  0:(d - 1))

## get the STCS
g14 <- stcfgneiting14(t = st[, 1],
                      s = st[, 2],
                      a = 1/50,
                      c = 1/10,
                      alpha = 1,
```

```
                      beta = 1,
                      gamma = 0.5,
                      tau = 1)

## visualize the STCS

g14.m <- matrix(g14,
                nrow = d)

persp3D(z = g14.m, x = 1: nrow(g14.m), y = 1:ncol(g14.m),
        expand = 1, main = "", scale = TRUE, facets = TRUE,
        xlab="Time lag", ylab = "Distance", zlab = "STCF",
        colkey = list(side = 4, length = 0.5), phi = 20, theta = 120,
        resfac = 5,  col= gg2.col(100))
```

`stcfgneiting16` *Gneiting-16 SpatioTemporal Correlation Structure*

Description

Provides spatiotemporal correlation structure function proposed by Gneiting (2002; Eq.16 at p. 594).

Usage

```
stcfgneiting16(t, s, a, c, alpha, beta, nu, tau)
```

Arguments

`t`	time lag
`s`	spatial lag (distance)
`a`	nonnegative scaling parameter of time
`c`	nonnegative scaling parameter of space
`alpha`	smoothness parameter of time. Valid range: $(0, 1]$
`beta`	space-time interaction parameter. Valid range: $[0, 1]$
`nu`	smoothness parameter of space. Valid range: > 0
`tau`	space-time interaction parameter. Valid range: ≥ 1 (for 2-dimensional fields)

References

Gneiting, T. (2002). Nonseparable, Stationary Covariance Functions for Space-Time Data, Journal of the American Statistical Association, 97:458, 590-600, doi:10.1198/016214502760047113

Examples

```
library(plot3D)

## specify grid of spatial and temporal lags
d <- 31
st <- expand.grid(0:(d - 1),
                  0:(d - 1))

## get the STCS
g16 <- stcfgneiting16(t = st[, 1],
                      s = st[, 2],
                      a = 1/50,
                      c = 1/10,
                      alpha = 1,
                      beta = 1,
                      nu = 0.5, tau = 1)

## visualize the STCS

g16.m <- matrix(g16,
                nrow = d)

persp3D(z = g16.m, x = 1: nrow(g16.m), y = 1:ncol(g16.m),
        expand = 1, main = "", scale = TRUE, facets = TRUE,
        xlab="Time lag", ylab = "Distance", zlab = "STCF",
        colkey = list(side = 4, length = 0.5), phi = 20, theta = 120,
        resfac = 5,  col= gg2.col(100))
```

---

stcs *SpatioTemporal Correlation Structure*

Description

Provides a parametric function that describes the values of the linear spatiotemporal autocorrelation up to desired lags. For more details on the parametric spatiotemporal correlation structures see section 2.3 and 2.4 in Papalexiou and Serinaldi (2020).

Usage

```
stcs(id, ...)
```

Arguments

id	spatiotemporal correlation structure ID
...	additional arguments (t as time lag, s as spatial lag (distance), and stcs parameters)

References

Papalexiou, S.M., Serinaldi, F. (2020). Random Fields Simplified: Preserving Marginal Distributions, Correlations, and Intermittency, With Applications From Rainfall to Humidity. Water Resources Research, 56(2), e2019WR026331, doi:10.1029/2019WR026331

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi:10.1029/2020WR029466

Examples

```
library(plot3D)

## specify grid of spatial and temporal lags
d <- 31
st <- expand.grid(0:(d-1),
                  0:(d-1))

## get the STCS
wc <- stcs("clayton",
           t = st[, 1],
           s = st[, 2],
           scfid = "weibull",
           tcfid = "weibull",
           copulaarg = 2,
           scfarg = list(scale = 20,
                         shape = 0.7),
           tcfarg = list(scale = 1.1,
                         shape = 0.8))

g14 <- stcs("gneiting14",
            t = st[, 1],
            s = st[, 2],
            a = 1/50,
            c = 1/10,
            alpha = 1,
            beta = 1,
            gamma = 0.5,
            tau = 1)

g16 <- stcs("gneiting16",
            t = st[, 1],
            s = st[, 2],
            a = 1/50,
            c = 1/10,
            alpha = 1,
            beta = 1,
            nu = 0.5,
            tau = 1)

## note: for nu = 0.5 stcfgneiting16 is equivalent to
## stcfgneiting14 with gamma = 0.5
```

```
## visualize the STCS

wc.m <- matrix(wc,
               nrow = d)

persp3D(z = wc.m, x = 1: nrow(wc.m), y = 1:ncol(wc.m),
        expand = 1, main = "", scale = TRUE, facets = TRUE,
        xlab="Time lag", ylab = "Distance", zlab = "STCF",
        colkey = list(side = 4, length = 0.5), phi = 20, theta = 120,
        resfac = 5,  col= gg2.col(100))

g14.m <- matrix(g14,
                nrow = d)

persp3D(z = g14.m, x = 1: nrow(wc.m), y = 1:ncol(wc.m),
        expand = 1, main = "", scale = TRUE, facets = TRUE,
        xlab="Time lag", ylab = "Distance", zlab = "STCF",
        colkey = list(side = 4, length = 0.5), phi = 20, theta = 120,
        resfac = 5,  col= gg2.col(100))
```

Index